Development and Implementation of Techniques for Ontology Engineering and an Ontology-based Search for Bioinformatics Tools and Methods

I n a u g u r a l - D i s s e r t a t i o n
zur
Erlangung des Doktorgrades der
Mathematisch-Naturwissenschaftlichen Fakultät
der Heinrich-Heine-Universität Düsseldorf
vorgelegt von

Indra Mainz
aus Krefeld

Dezember 2008

Aus dem Institut für Physikalische Biologie
der Heinrich-Heine-Universität Düsseldorf

Gedruckt mit der Genehmigung der
Mathematisch-Naturwissenschaftlichen Fakultät der
Heinrich-Heine-Universität Düsseldorf

Referent: apl. Prof. Dr. G. Steger

Korreferent: Univ.-Prof. Dr. M. Lercher

Tag der mündlichen Prüfung: 21. Januar 2009

Die hier vorgelegte Dissertation habe ich eigenständig und ohne unerlaubte Hilfe angefertigt. Die Dissertation wurde in der vorgelegten oder in ähnlicher Form noch bei keiner anderen Institution eingereicht. Ich habe bisher keine erfolglosen Promotionsversuche unternommen.

Düsseldorf, den 12.12.2008

(Indra Mainz)

"Everything should be made as simple as possible–but no simpler!"

- Albert Einstein

Meinen Eltern.

Acknowledgements

First of all I would like to thank my supervisor apl. Prof. Dr. Gerhard Steger for the freedom in doing research and the confidence he had shown me. Thank you, Gerhard, for facing the world of ontologies.

Additionally, I wish to thank Prof. Dr. Martin Lercher for accepting the task to read this thesis as a second reviewer.

Special thanks are addressed to my colleagues Dominic, Ingo and Katrin for always fruitful and constructive discussions and a close collaboration. Additional thanks to all other ONTOVERSE project partners.

Furthermore, I like to thank all my colleagues in the Institute of Biophysics for a great atmosphere, especially the bioinformatics guys and Natalie. I had a great time.

Of course I whish to thank my dear family and friends just for being you. Becci & Jeremia, Domi, Nahal, Evgin and Timo, this is for you. Special thanks to Deniz, who read and challenged this thesis and beared me in good and bad times.

I am very grateful to my parents for paving this way for me and supporting me all the time. You are wonderful!

Financial support from the German Federal Ministry of Education and Research is gratefully acknowledged.

Contents

List of Figures

List of Tables

1

Introduction

Today in many fields the amount of available information and data increases exponentially. Since this growth does not entail a just as large quality increase of the available knowledge, methods and tools are needed to filter and process this mass of information. For this purpose *ontologies*, which represent data and their interrelations in a computer "interpretable" form, get more and more established. Ontologies are introduced in detail in Section 2.1.

The exponential increase of knowledge in particular is observable in the internet, to which the data explosion mainly can be ascribed. To make the available information easier trackable in the World Wide Web (WWW, Web), ontologies serve as foundation of the *Semantic Web*. That is, a semantic layer is being built on top of the WWW, which adds meaning to Web contents and data. By this means, the information in the internet gets understandable by computers. Machines operating on semantic annotated data are able to intelligently search the Web. Thus, for example the number of results of a google search can be decreased by listing only those hits the user really is interested in. Furthermore, these results can be clustered according to their context.

This thesis deals with the creation and machine-supported extension of ontologies (*ontology engineering*), describes the extension of an ontology in the field of life sciences and introduces an information system using this ontology as knowledge base.

1.1 Data Situation in the Life Sciences

Today, technical improvements give rise to high-throughput screenings (HTS) in experimental biomedical research. That means robotics, sensitive detector and highly specialized computer software for controlling and data processing are applied in laboratories to enable the automatic execution of millions of biomedical tests. This method in particular is adopted in pharmaceutics for example to fastly detect a reagent that modifies or inhibits a pathogenic target in drug discovery. Furthermore, high-throughput sequencing facilitates the fast sequencing even of whole

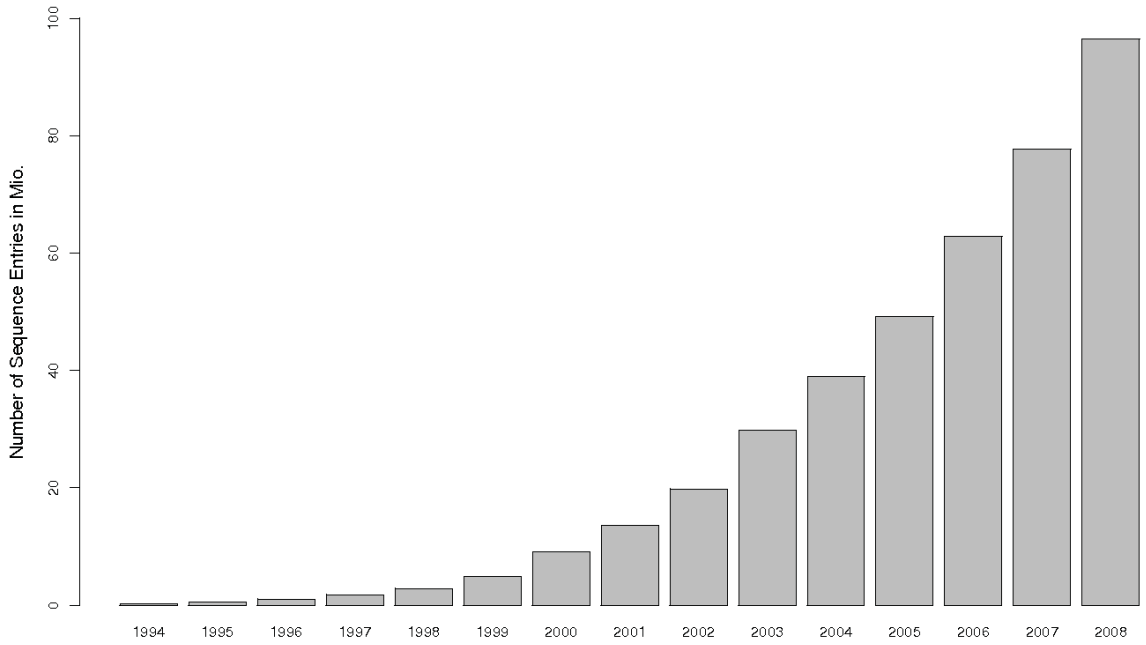

Figure 1.1: GENBANK entry increase over time. The diagram shows the inrease of sequence entries in GENBANK over time (included are October values of each year). The shown data are taken from the release notes of GENBANK from the NCBI (National Center for Biotechnology Information of the United States of America).

genomes. This results in a vast increase in the size of sequence databases, exemplified in Fig. 1.1 by one of the most prominent nucleotide databases, GENBANK® (Benson *et al.*, 2008). This high-throughput methods yield an enormous data increase, which cannot be handled manually, but have to be adequately worked up and made searchable.

Additionally, unlike in physics, in biology knowledge predominantly cannot be described in mathematical formulas. Similarity comparisons of biosequences, for example, can be made automatically based on mathematics, but the interpretation of the results is accomplished by the knowledge of scientists (Bodenreider & Stevens, 2006). This knowledge often is collected in natural language. These data are highly heterogeneous and cannot be interpreted and compared automatically. The problem of heterogeneous data is already described in 1995 by Davidson *et al.* and Karp.

In biology often inconsistent terminologies are utilized. On the one hand these emerge from historical reasons as for example genes were described in different organisms. Based on this genes serving the same function got different names, which extremely complicates the information retrieval about those sequences. On the other hand there exist no definitions even of fundamental biological terms (Stevens *et al.*, 2000). A prominent example is the term “gene”, which can constitute a DNA region that codes for a protein or a segment of genomic information that specifies a trait. Such ambiguous terminologies make discussions and knowledge retrieving tremendously difficult.

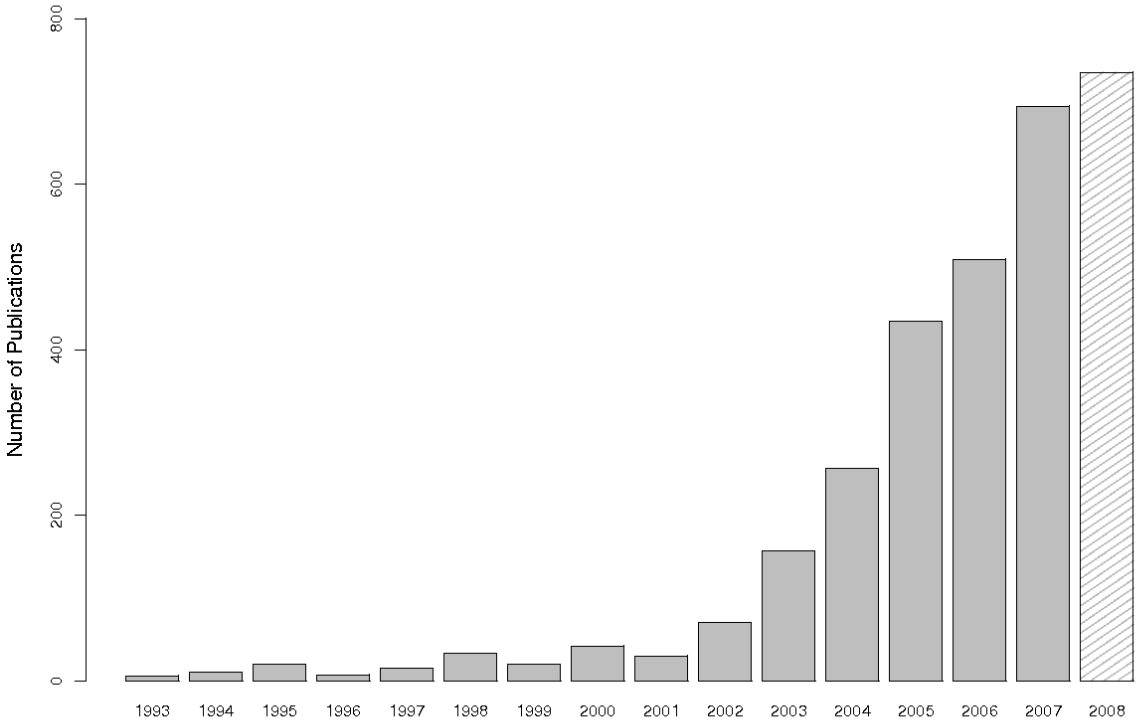

Figure 1.2: Ontology relevant publications in PUBMED. The plot visualizes the increase of publications in PUBMED that deal with ontologies. Presented are the results of the keyword search with "ontology OR ontologies". Data are gathered at December 5th, 2008. Thus the 2008 bar is not complete.

Based on the just described challenges, the inconsistent terminologies, the vast amount and heterogeneity of data, adequate knowledge representations and effective information integration had to be adapted.

1.1.1 Bio-ontologies

The life sciences have proved themselves as particularly interested in the use of ontologies. Fig. 1.2 shows this by plotting the number of ontology relevant publications in PUBMED. In biology observations and organisms were always categorized (McCray, 2006). In extending this classification into complex knowledge representations like ontologies, there exist a chance in overcoming the problems (e. g. Baclawski & Niu, 2006; Bodenreider & Stevens, 2006). Differences between ontologies and classifications are described in Section 2.1.

Fig. 1.3 shows the historical appearance of biological knowledge representations in different complexities. The scale begins in 1992 with ECOCYC (Keseler *et al.*, 2005). This is a database of *Escherichia coli* K-12 genes and metabolism and semantically less complex. The most prominent bio-ontology is the GENE ONTOLOGY (GO) (Ashburner *et al.*, 2000; Consortium, 2001, 2006, 2008), which was started in 1998 by three model organism databases (Flybase, Saccharomyces Genome Database, Mouse Genome Database) and aims at collaboratively describing

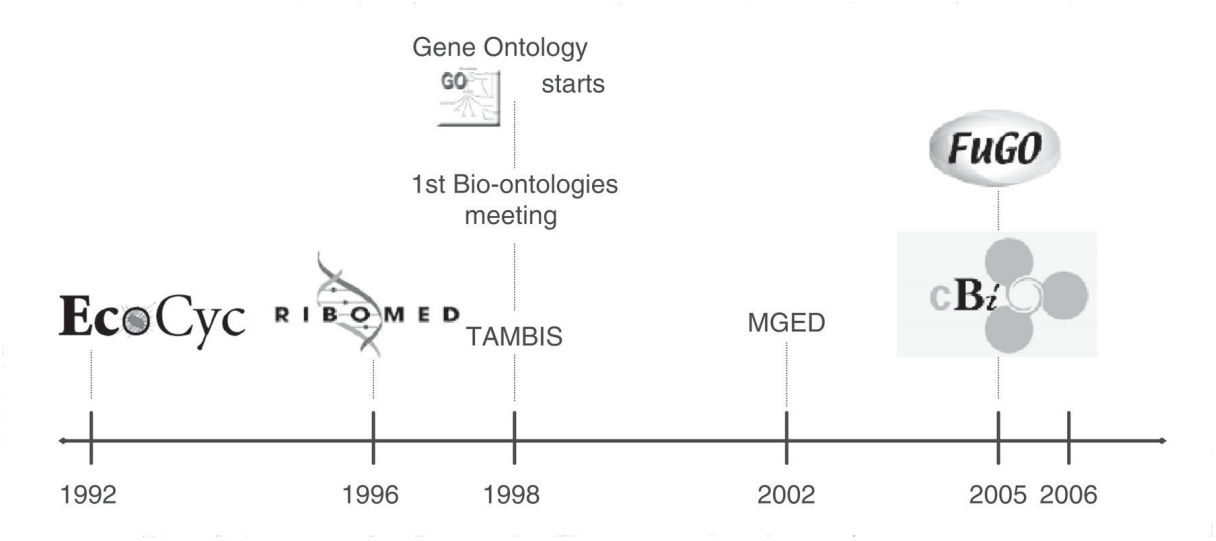

Figure 1.3: Bio-ontology timeline. Early Bio-ontologies show less complexity and semantics than new data models. (*Source:* Bodenreider & Stevens, 2006).

gene products in different databases in a consistent way. Today, the GO consortium increased to 16 members and four associates. Furthermore, the ontology currently[1] includes 27,769 terms, which can be linked by five types of relationships: is_a, part_of, regulates, positively_regulates and negatively_regulates. Actually GO consists of three ontologies that characterize biological processes, cellular components and molecular functions of gene products. These controlled vocabularies are cross-linked with databases. Meanwhile, there exist a variety of tools utilizing GO. GOPubMed for example is a search engine for biomedical texts from PUBMED that structures its results according to the gene ontology. For this reason the millions of publications in PUBMED can be searched much faster.

In the same year of the beginning of GO, the first Bio-ontologies meeting was held and the TAMBIS project (Transparent Access to Multiple Bioinformatics Information Sources; Baker *et al.*, 1998) started its efforts in providing an information retrieval service based on the unification of different data sources. For the reason of unifying, an ontology (TAMBIS Ontology, TaO) of molecular biological and bioinformatics concepts was built.

These days, the number of bio-ontologies has considerably increased. That is why an umbrella community for a range of ontologies designed for biomedical domains has been established, the Open Biomedical Ontologies (OBO, Smith *et al.*, 2007). This community aims at reducing the redundancies in ontology developments. Ontologies of this collection have to meet the following citeria: openness, common representation, independance, identifiers and natural language definitions.

[1] Release of December 1st, 2008

1.2 Thesis outline

The exponential growth of data for example from high-throughput methods has to be handled by a plenty of specific software tools. Thus, there exists a vast amount of bioinformatics tools and methods, which can hardly be surveyed even by bioinformaticians. That is why this thesis continues with the development of an ontology that characterizes such tools and methods and introduces an information system that enables searching of bioinformatics tools and methods for specific tasks.

First, the next chapter provides background information, which helps to follow the results of this thesis. Ontologies will be introduced, which form the backbone of the work in this thesis. Furthermore, some tools and frameworks are presented, which has been utilized.

After that the results of this thesis are presented and discussed. The ontology that is described in this thesis was started in my Bachelor thesis in computer science. Mainz (2006a) summarises the design of the ontology for bioinformatics tools and methods (BIO2ME). Chapter 3 describes this preliminary work and the extension of the ontology. A lot of experiences could be gained and challenges had been discovered, which point out the need of an ontology engineering support. There exist a lot of formal ontology editors (e. g. ONTOSTUDIO[2], SWOOP[3] and PROTÉGÉ; see Section 2.3), but rare support of the preliminary ontology engineering phases, e. g. the informal knowledge aquisition. Furthermore, the need for a semi-automatic ontology extension method was detected. The realization of these new strategies is introduced in Chapters 4 and 5.

To harness the ontology, an information system was implemented to query the ontology and facilitate the insertion of additional knowledge. This application is described in Chapter 6.

The results presented in this thesis emerged from the work within the research project ONTOVERSE (see Section 2.2). Several partners participated in the project, which in part closely collaborated; so in this thesis some work of colleagues was utilized. All those parts that I did not do by myself, are marked in the text.

[2] `http://www.ontoprise.de/index.php?id=34`

[3] `http://www.mindswap.org/2004/SWOOP/`

2

Background

This chapter introduces basic tools and principles that are relevant to follow the results of this thesis. First, ontologies are introduced including their formal shape, engineering and usability. After that, the project ONTOVERSE is described wherein this work emerged. Then sections on utilized ontology tools and some programming utilities follow. Finally, general principles on the wiki and tagging ideas are provided that are introduced in Chapters 4 and 5 as new tools for ontology extension.

2.1 Ontologies

The notion *ontology* is not clearly defined. It originated in philosophy at least in the 18th century and is transfered to knowledge representations at least in the early 20th century in information sciences and since approximately 20 years in computer sciences and artificial intelligence, respectively. Figure 2.1 shows the deployment spectrum of the notion "ontology" as described in Lassila & McGuinness (2001). The complexity axis starts with simple controlled vocabularies like catalogs, which tie terms of equal interpretation by assigning the same identifier to those concepts, and glossaries, which specify the meaning of a notion by attaching a natural language definition. These interpretations of the terms are not processible by computers and do not comprehend any interrelations between the specified notions. Thesauri are the next more complex step for they can be made interpretable by computers and comprise synonym relationships. Thesauri as regarded in the schema and in its general meaning do not imply hierarchical relations, in contrast to the more strict DIN-standardized definition of "thesaurus" in the information sciences that postulates is-a relationships. By simple is-a knowledge representations comprising inheritance, the transfer between informal and formal specification is accomplished. These formal models are then extended by logics and properties of concepts.

This thesis regards ontologies as formal and more complex knowledge representations that are written in specialized, machine-interpretable languages. The language of choice in this thesis is the currently most prominent and widely supported ontology language OWL (see Section 2.1.3).

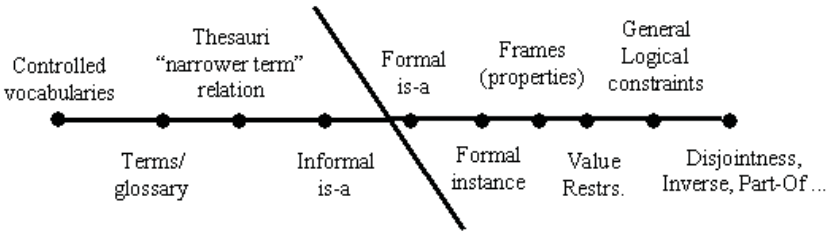

Figure 2.1: Ontology Spectrum. The figure shows the degrees of complexity of ontologies. The complexity increases from left to right. Informal and formal specifications are seperated by the red slash. The notion "ontology" in this dissertation is applied for the more complex models, appearing on the right side. (*Source:* Lassila & McGuinness, 2001)

In the sense of this paper ontologies are knowledge representation systems, which are defined as formal conceptualizations of a domain of knowledge (Gruber, 1993). That is, ontologies model knowledge in an unambiguous and computer "understandable" way. They are therefore suitable for knowledge bases and serve as fundament of a common view on a domain.

2.1.1 Application of Ontologies

Ontologies serve different purposes. They provide knowledge bases for intelligent search with which they particularly facilitate the reuse of knowledge. More than that, the machine-readable form of ontologies enables knowledge inference basing on pre- and self-defined rules. For large models this is difficult to realize manually.

Based on these features ontologies are increasingly applied in a diverse set of knowledge based applications in areas like biology (see Section 1.1.1). They provide global information infrastructures in terms of e-Science (Goble *et al.*, 2006; Hey & Trefethen, 2005), the Semantic Web (Berners-Lee *et al.*, 2001) and the Semantic Grid (De Roure *et al.*, 2003). The Semantic Web aims at making the meaning of information utilizable for computers. The mass of information can then automatically be interpreted, processed and related to each other. In contrast to information retrieval based on computer linguistics, which operate on unstructured, natural language data, the information is provided explicitly in form of ontologies.

Furthermore, ontologies are also often intended to be a shared conceptualization of a domain of interest in terms of a consensual knowledge base for a community (Gruber, 2005, 1993).

2.1.2 Formal Ontology

Ontologies, as considered here, consist of three different constructs: Classes (concepts), individuals and properties (relations). See Fig. 2.2 for an illustration of the following definitions and examples.

Classes define a set of individuals with common properties. In an ontology of bioinformatics tools the class `Program` comprises computer programs, and the class `StrAl` compre-

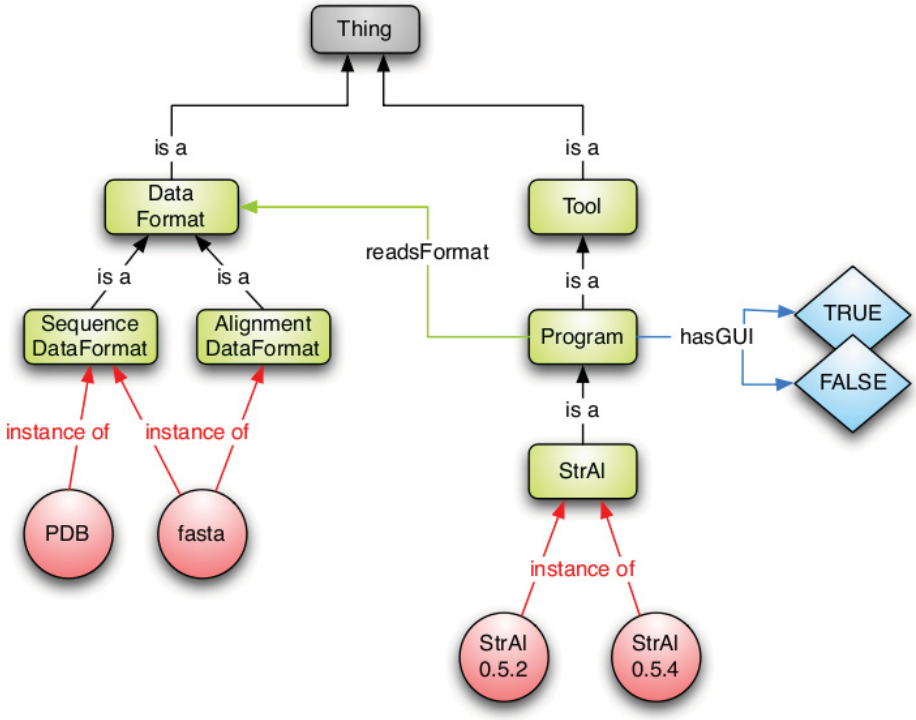

Figure 2.2: Ontology schema. The figure illustrates the various elements of an ontology. Classes are visualized in boxes, individuals in cycles and datatype values are depicted in rhombi. Arrows start from an instance ("instance of" relation), a subclass ("is a" relation) or a property domain (object or datatype property). The arrowhead points to the type class of an instance, a superclass or the property range, respectively. Detailed information can be found in the text.

---

hends versions of the program StrAl. The class `owl:Thing` is the superclass of all classes. `owl:Nothing` is the negation of `owl:Thing`.

Properties relate classes to other classes (*object properties*) or bind a class to a datatype value (*datatype property*). Simple properties are "is a" relations which model hierarchical relations between classes. This means that the subclass inherits all properties of its superclass. In figure 2.2, a self-defined object property is `readsFormat` which relates the class `Program`, called domain of the property, with the range class `DataFormat` and models possible input formats of a computer program. The datatype property `hasGUI` relates a `rdfs:BOOLEAN` (values: `true`, `false`) to the class `Program` and declares if a program has a graphical user interface (GUI).

Individuals are entities that model a specific thing in the world. They are called instance of a class if they are asserted to it. As mentioned above, the class `StrAl` is the set of all program versions of the program StrAl. So `StrAl0.5.2` and `StrAl0.5.4` are instances of `StrAl`.

Additionally, rules can be attached to ontology constructs that help to model the knowledge more accurately. For example "heterogeneous transitivity" (involving different relations) can be modeled for self-defined object properties such as:

```
(program readsFormat format) ∧ (format serializes data)
⇒ (program readsData data)
```

Uniform Resource Identifier

Each ontology element is unambiguously identifiable by a character string called Uniform Resource Identifier[1] (URI). Tim Berners-Lee introduced the *Universal* Resource Identifier in 1994[2] which then became the *Uniform* Resource Identifier. The simplified syntax of an URI is:

```
<namespace>#<local_name>.
```

For the bioinformatics ontology BIO2ME the namespace is:

```
http://www.ontoverse.org/BIO2Me.owl#.
```

The URI for the resource `StrAl` is then:

```
http://www.ontoverse.org/BIO2Me.owl#StrAl.
```

So in simple terms, "StrAl" is the local name of the resource `StrAl`.

Class Level

The level of an ontology class is the distance of the class to the overall superclass `owl:Thing`. In figure 2.2 the class `StrAl` has level 3. In this thesis the term "top level concept" is used which means classes with level 1. Additionally, "direct subclasses" or "direct instances" are classes or instances, respectively, which are directly assigned to a class. In figure 2.2 individual `StrAl0.5.4` is the direct instance of class `StrAl`. Because of the subclass dependency and its inheritance, `StrAl0.5.4` is also instance of `Program`, `Tool` and `owl:Thing`. Consequently, all individuals are instances of `owl:Thing`.

2.1.3 RDF, RDFS & OWL

The formalization of ontologies is carried out by standardized ontology languages.

[1] `http://www.ietf.org/rfc/rfc2396.txt`

[2] `http://tools.ietf.org/html/rfc1630`

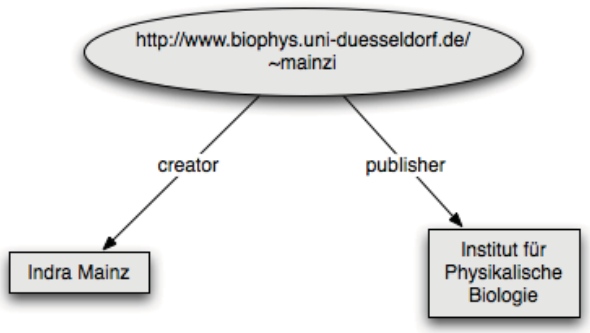

Figure 2.3: RDF graph. RDF graph of two statements.

RDF

The first Semantic Web language was the Resource Description Framework[3] designed by the World Wide Web Consortium[4] (W3C), which was launched in 1994 by the founder of the Web, Tim Berners-Lee, and is a committee for the standardization of WWW techniques, to provide metadata in the World Wide Web (WWW, Web). Thus the properties of Web resources are described in machine-readable form. Those RDF models can be retained in form of a graph or by using RDF syntax in a XML document.

RDF expresses information in form of triples, which make statements about specific resources. Each triple is composed of a subject, a predicate and an object. *Subjects* are resources that are unambiguously identified for example by a URI (see Section 2.1.2). A subject can be a Web page or even a thing, which is not available in the internet. The *predicate* characterizes the subject by relating objects (attributes). So *objects* specify the value of the predicate, which form a statement about the subject. Fig. 2.3 illustrates the two triples

```
<http://www.biophys.uni-duesseldorf.de/~mainzi/>,
    creator, "Indra Mainz".
<http://www.biophys.uni-duesseldorf.de/~mainzi/>,
    publisher, "Institute of Biophysics".
```

in form of a RDF graph.

RDFS

RDF enables expression of simple statements about resources by relating attributes. These assertions are written in form of triples. To use those statements more specifically by defining specific kinds (classes) of resources and properties, the definition of vocabularies that are used in statements is necessary. The RDF Vocabulary Description Language 1.0 (RDF Schema[5], RDFS) defines such vocabularies for RDF by providing information about the interpretation of

[3] http://www.w3.org/RDF/

[4] http://www.w3.org/

[5] http://www.w3.org/TR/rdf-schema/

RDF statements. In contrast, RDFS does not say anything about the syntactical appearance of the RDF description.

RDFS introduces classes (`rdfs:Class`), subclasses (`rdfs:subClassOf`) and instances (`rdf:type`). It allows the representation of classes, class hierarchies and properties and therewith is not very expressive. OWL is a richer vocabulary enabling the representation of more complex statements.

OWL

The Web Ontology Language[6] (OWL) is the currently most widely prevalent ontology language and was used for the formalization in this thesis. It is also standardized by the W3C Web Ontology Working Group[7] (WebOnt) of the W3C. OWL mainly serves the publication, provision and re-use of information in the internet. It extends RDF and RDFS by complex relations and rules to map the semantics of information more expressive. OWL was developed for computer applications that not only display information but also interpret its contents. This enables an extended interpretation of the Web content by machines and is presumed to be a significant technology for the implementation of the Semantic Web.

OWL is organized into three sublanguages of different expressiveness:

- **OWL Lite** is least expressive. It provides class hierarchies and simple contraints. For example, it supports cardinality contraints, but only for values 0 and 1.
- **OWL DL** is the most expressive sublanguage that can be reasoned in finite time. “DL” abbreviates Description Logics[8], which is the subset of first-order logic on that OWL bases. OWL DL for example allows an unrestricted cardinality constraint.
- **OWL Full** has no restrictions and therefore is able to express assertions of higher order predicate logic. For example a class can be instance of another class. From this it follows that this sublanguage has not to be determinable and reasoning might be endless computable. OWL Full extends OWL DL by constructs that intend to provide compatibility with RDF Schema.

These sublanguages are built upon each other, which means that a valid OWL Lite ontology also is a valid OWL DL ontology and a valid OWL DL ontology is a valid OWL Full ontology.

OWL as well as RDF and RDFS makes the open world assumption. This means that the correctness of any statement is independent of whether someone knows it to be true or not. For example if “Indra would like to have her doctor’s degree.” is stated. The question “Does Indra have a doctor’s degree?” would be negated in the closed world assumption (e. g. SQL) and the response of the open world assumption would be “Unknown.”, because no statement about her doctor’s degree was made. Indra possibly could already have one.

[6] `http://www.w3.org/TR/owl-semantics/#ref-overview`

[7] `http://www.w3.org/2001/sw/WebOnt/`

[8] `http://dl.kr.org/`

Serialization

There exist several forms of serialization of OWL, making OWL applications independent from programming languages and operating systems. Here the RDF/XML[9] format, which is based on XML, and the N-triple format[10] is presented by the following example.

N-Triples file:

```
<http://www.ontoverse.org/example.owl>
   <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   <http://www.w3.org/2002/07/owl#Ontology> .
<http://www.ontoverse.org/example.owl#Program>
   <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   <http://www.w3.org/2002/07/owl#Class> .
<http://www.ontoverse.org/example.owl#StrAl>
   <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   <http://www.ontoverse.org/example.owl#Program> .
```

RDF/XML file:

```
<rdf:RDF xmlns="http://www.ontoverse.org/example.owl#"
     xml:base="http://www.ontoverse.org/example.owl"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:example="http://www.ontoverse.org/example.owl#">
    <owl:Ontology rdf:about=""/>

    <owl:Class rdf:about="#Program">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>
    <owl:Class rdf:about="&owl;Thing"/>

    <Program rdf:about="#StrAl"/>
</rdf:RDF>
```

Both files represent a simple ontology, which models a class `Program` (subclass of `owl:Thing`) and its instance `StrAl` in the namespace “http://www.ontoverse.org/example.owl”.

2.1.4 Ontology Engineering Process

There coexist many different published and unpublished methods for ontology building. But there is no standard. Fernández López (2001), for expample, gives an overview of different methods with regard to a formulation of a standard so far. Some of the approaches base on practical experiences and some seem to be resulting from theoretical considerations which ideas are reasonable but to some extend less practical. For the purpose of gaining experiences in

[9] `http://www.w3.org/TR/rdf-syntax-grammar/`

[10] `http://www.w3.org/TR/rdf-testcases/#ntriples`

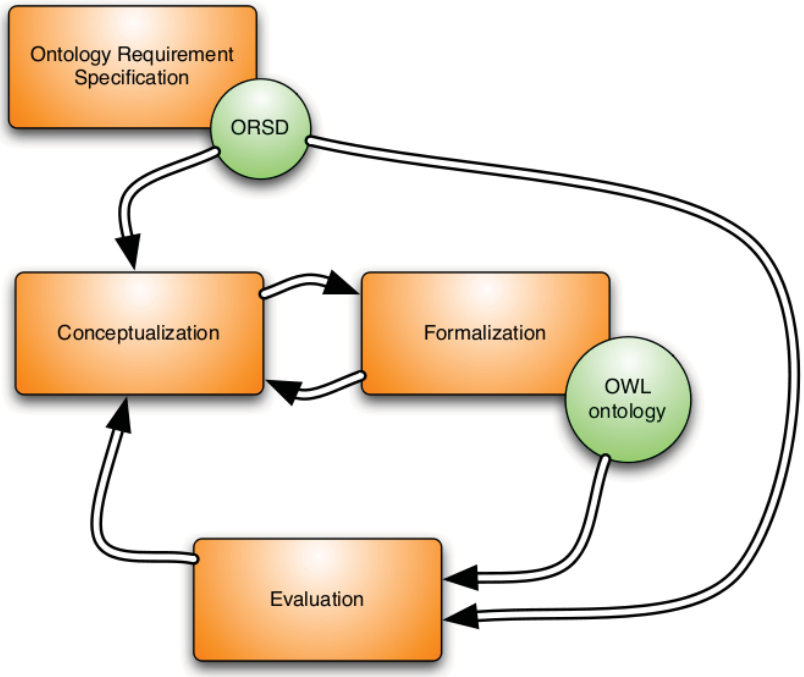

Figure 2.4: Ontology engineering process. The schema visualizes the four different ontology engineering phases. Green colored cycles depict the resulting documents of a design step.

ontology design, which then should influence the realization of the ONTOVERSE intention, I studied some approaches in my Bachelor thesis (Mainz, 2006a) and combined them to a new "ONTOVERSE" approach with special attention on collaborative ontology building.

The resulting engineering process finally comprises in particular elements from the KOWIEN[11] project (Apke & Dittmann, 2004) which was extended and modified by other methods. In the following the different phases of the approach are described (see figure 2.4):

ORSD The Ontology Requirement Specification Document (modified from Sure, 2002) serves as starting point for ontology design. The document captures the motivation, domain and goal of the ontology. Furthermore, it specifies the application and addressed users. The use of the ontology should clearly be defined before the ontology engineering is started because the actual modeling/structure of the ontology is highly depending on this. Very helpful to clarify the intention of the planned knowledge model is a *Competency Questionnaire*, which is suggested in Grüninger & Fox (1995) and which was included into the ORSD. The questionnaire comprises questions which should be answered by the ontology. It is also useful to make a collection of knowledge sources of the domain which then will help in the conceptualization phase of ontology building. Additionally, the design criteria should be defined. Important is the choice of the ontology language, but also other guidelines like naming conventions of classes, properties and individuals are desired. Some of them are depending on the intended application of the ontology.

[11] `http://www.kowien.uni-essen.de/`

Especially for collaborative ontology building it is important that engineers share a common view on the domain and purpose of the ontology. The ORSD should finalize that and particularly helps ontology engineers joining later to become acquainted with the project.

Conceptualization This phase comprises the unformal gathering of domain relevant concepts, finding properties and establishing relations between them. Concept graphs can for example be created, which illustrate relations. Some utilities are taken from (Gómez-Pérez *et al.*, 2004).

Formalization In this step the protoontological data are translated into a formal ontology in a machine-understandable ontology language like OWL. This process is supported by formal ontology editors like PROTÉGÉ (see Section 2.3).

Evaluation The validation (content check) of the ontology can be made on the basis of the ORSD and with the aid of domain experts. The verification like syntax and consistency checking can be made by (e. g.) the OWL reasoner PELLET.

The whole ontology design, however, is rather an iterative than a sequential process.

2.2 ONTOVERSE

This dissertation emerged within the research project ONTOVERSE[12,13] (e. g. Mainz *et al.*, 2008) which aims at building a Web-based platform for collaborative ontology engineering and management (Paulsen *et al.*, 2007). Beneath the formal cooperative ontology editor (Bai & El Jerroudi, 2008), which is built by a research group at the university of Duisburg-Essen led by Professor Jürgen Ziegler, the platform should serve as social networking and communication tool for life scientists, too. Users are able to search co-operation partners with special expertise for their ontology project.

Figure 2.5 not only summarizes the different modules of ONTOVERSE but also illustrates the supported ontology engineering workflow. On the ONTOVERSE platform each ontology is built within an ontology project. At the beginning of each project the initiator becomes *project admin*. This role has specific privileges in the project, among these are rights for activating and deactivating memberships of the project. After defining the domain of the ontology the next step in ontology building is gathering assistants. In the scope of ONTOVERSE two types (roles) of project members in addition to the project admin are introduced which differ in their field of interest and skill: *Domain experts* have expertise in parts or even the entire domain of the ontology. They contribute their knowledge into the conceptualization and evalutation of the ontology. *Ontology designers* in contrast have skills in knowledge representation and the formulization of ontologies for example in OWL. The roles are thus divided according to the protoontological and the actual ontology formalization phase of the ontology building process. An ontology engineer of course may hold both roles. The distinction and denotation of domain experts and

[12] `http://ontoverse.cs.uni-duesseldorf.de`

[13] `http://ontoverse.org/`

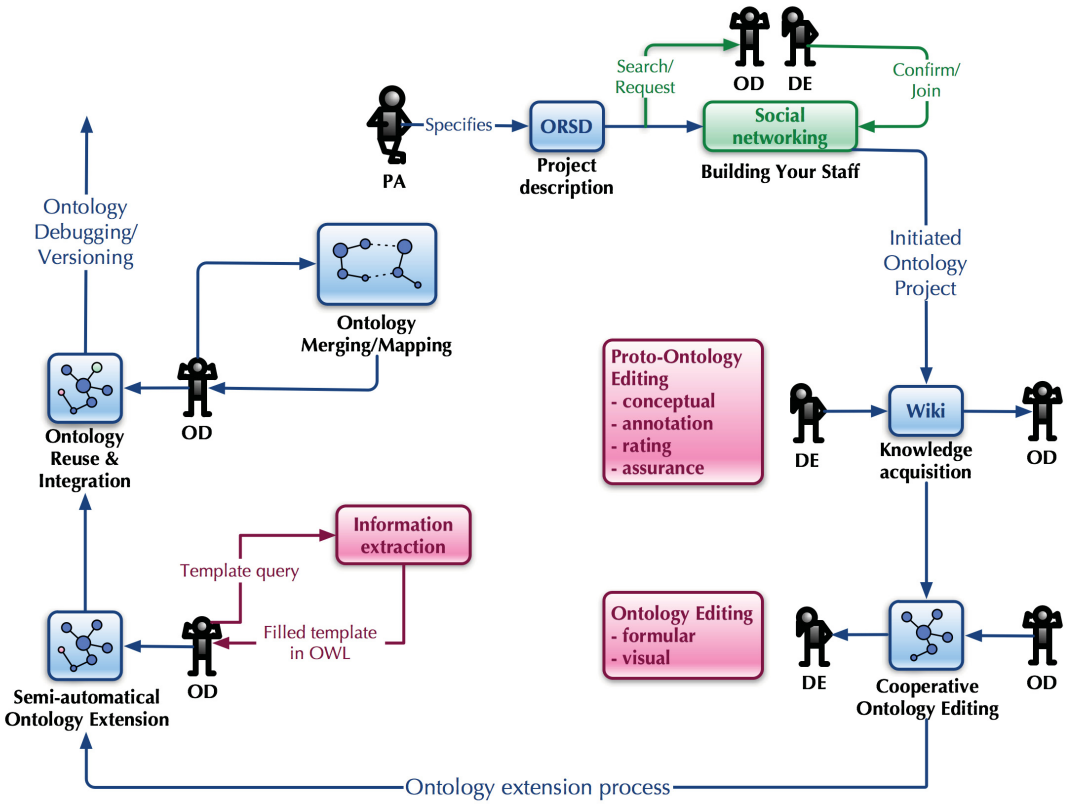

Figure 2.5: ONTOVERSE workflow. The schema shows the collaborative ontology engineering cycle in the ONTOVERSE system. You can see the different functions of domain experts (DE) and ontology designers (OD). The project admin (PA) coordinates the ontology process in one ontology project.

ontology designers are employed throughout this dissertation. With the aid of project members, the entire scope of the ontology is captured in the ORSD (see Section 2.1.4) under the guidance of the admin. By the experiences collected during the BIO2ME creation a special importance is attached to the support of the protoontological phases, for this reason a wiki is integrated (detailed information in Chapter 4) to facilitate the collaboration of project members. Within this wiki the collaborative acquisition of knowledge (conceptualization) follows. Domain experts provide their knowledge in an unstructured as well as semi-structured way. Ontology designers are than able to formalize this information and discuss it with the domain experts. For this formalization each ontology project has its own cooperative ontology editor, which comprises a formular based editor and ontology visualization frame. The editor provides collaboration support by marking modifications, offering locking and highlighting functions. A chat function in the editor window additionally facilitates direct arrangements and discussion. The editor is connected to the wiki and vice versa, respectively. Thus the user is able to jump to the wiki page of a selected ontology element and back. The data management and modification recording is handled by a so called tuple space server (see Section 2.7) developed by the group of Professor Hoppe of the University of Duisburg-Essen.

The extension of the ontology can be accomplished either by the sole brainpower of domain experts or supported by different mechanisms on the basis of PubDB, the ONTOVERSE publication

database. PubDB has an PubMed search interface, which enables the easy loading of biomedical publications into the database. additionally, users are invited to upload other articles. Based on the publications in PubDB, users are able to perform an information extraction (Jurafsky & Martin, 2008), which automatically receives ontology triples by extracting predefined relations from a publication. The triples then have to be surveyed and accepted by domain experts and afterwards inserted into the ontology by ontology designers. The information extraction backend was developed by the department of Computer Linguistics of the Institute for Language and Information (Heinrich-Heine-University Düsseldorf). Another less machine-supported approach is the ontology extension by tagging. The idea, implementation and some case studies are elucidated in Chapter 5.

The ONTOVERSE formal editor additionally supports ontology merging and mapping to facilitate the reuse and integration of already existing ontologies. The evaluation finally is carried out by the semi-structured presentation of the ontology data in the projects wiki. This feature provides domain experts not familiar with formal ontologies to evaluate the ontologies structure and content. Additionally, the modification suggestions can be discussed directly in the wiki article of the ontology element and with the connection between wiki and editor, ontology designers easily can switch to the right places in the wiki and the ontology, respectively.

2.3 PROTÉGÉ

BIO2ME was built and extended with the aid of PROTÉGÉ[14]. PROTÉGÉ is a widely used ontology editor. It is JAVA based and freely available under the Open Source License. PROTÉGÉ is elaborated and highly supports the formal ontology modeling process for one single ontology designer. In this thesis PROTÉGÉ-OWL was utilized in version 3.2.1, 3.4 beta and 4 alpha. The implementation of the OWL specification was inaccurate in the former two versions (see Section 3.2). PROTÉGÉ does not offer any support of the vitally important informal ontology editing (see Section 2.1.4), which was realized during the work on this thesis.

2.4 PELLET & JENA

PELLET[15] is an open-source JAVA OWL DL reasoner, which was used in this thesis to check the consistency (logical correctness) of the ontology and infer implicit information (see Section 6.1.2). It was used in version 1.5.1 with JENA[16].

JENA is an open source Semantic Web framework also implemented in JAVA. It reads ontology models from files, databases and URLs, abstracts it and provides an API for operating on the contained data. Furthermore, JENA supports divers ontology formats like RDF/XML and N-Triple.

[14] http://protege.stanford.edu/index.html

[15] http://pellet.owldl.com/

[16] http://jena.sourceforge.net/ontology/

2.5 RUBY

The ONTOVERSE platform and the BIO2ME application are both implemented in RUBY[17]. This belongs to the class of scripting languages, which are in contrast to pure programming languages not compiled into machine code, but interpreted during runtime.

RUBY unifies parts of the languages PERL[18], SMALLTALK[19], EIFFEL[20], ADA[21] and LISP[22]. Yukihiro Matsumoto, the creator of RUBY, once said in a mailing list: "Ruby is simple in appearance, but is very complex inside, just like our human body."[23]. This for instance becomes clear in the simplification that everything is an object in RUBY. This characteristic is adopted from SMALLTALK. So unlike to other object-oriented programming languages, in which primitive types like boolean are not objects, in RUBY each type has its own attributes (instance variables) and methods.

Due to the freedom in terms of using, copying and particularly modifying the source code of RUBY[24], the language is very flexible to use. Standard objects can easily be extended and adapted at will.

In RUBY one can use metaprogramming, which means that the source code is not written by a human, but by another computer program. In RUBY code can be created, changed and added during runtime.

2.5.1 RUBY ON RAILS

RUBY ON RAILS[25] (short RAILS) is an open source RUBY framework for an easy development of Web applications. RAILS incorporates a MVC (Model View Controller) architecture, which divides the framework up into three main parts:

Models are RUBY classes that defines data types used in the application. They specify the logic to manipulate and access data. Models build the interface between the controller and the data.

Views consist of template and HTML files, which are rendered by Web browsers. View code can also be composed of JAVASCRIPT etc. So called RJS templates (RUBY JAVASCRIPT can be used to simply add AJAX (Asynchronous JavaScript and XML; Garrett, 2005) functionalities, which modify an already rendered page. AJAX requests are called in the background and updates the page the request origined from.

[17] http://www.ruby-lang.org/

[18] http://www.perl.org/

[19] e. g. http://directory.fsf.org/project/smalltalk/

[20] http://www.ecma-international.org/publications/standards/Ecma-367.htm

[21] http://www.open-std.org/jtc1/sc22/wg9/

[22] http://www-formal.stanford.edu/jmc/recursive.html

[23] http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-talk/2773

[24] http://www.ruby-lang.org/de/about/license.txt

[25] http://rubyonrails.org/

Controllers build the logic layer in RAILS applications. They process input data and pass them on to the view. Controllers are able to call methods of models. Controller methods (actions) per default correspond one file in the view.

RUBY ON RAILS is composed of several libraries, *ActionRecord* for example serves as object-oriented mapper (ORM) that ties database tables to models by naming conventions. The fact that RAILS is open source resulted in a plenty of available plugins that serve different purposes like the ACTS_AS_VERSIONED plugin, which facilitates the versioning of database entries (see Chapter 4.1.3).

RAILS provides by default URL mappings of controller actions. This makes URL simple and straightforward. For example:

```
http://localhost/forum/show/1
```

Is the URL of the "forum" controller, action "show" and object "1". This means that the forum object with database id 1 is shown in the web browser.

2.6 DEEP SEMANTICS

The BIO2ME information system uses the Semantic Web framework DEEP SEMANTICS to realize the integration of the knowledge base and operate on it. The framework was developed by Mainz (2008) and converts OWL ontologies in fuctional RUBY code. In the process DEEP SEMANTICS takes advantage of the metaprogramming feature of RUBY. In contrast to other Semantic Web libraries like JENA 2, which provides an API for ontology handling, DEEP SEMANTICS dynamically generates a functional RUBY program that comprehends RUBY classes and objects, which can be used as any other RUBY object. The ontology is mapped to a object-oriented, functional RUBY model including all its logical and structural properties. This process is called *deep integration* and is in this context introduced by Fernandez (2005). In a deep-integrated ontology, OWL classes are modeled as RUBY classes, OWL instances are RUBY class instances and ontology properties build class and instance methods, respectively.

Fig. 2.6 illustrates the architecture of DEEP SEMANTICS with its main components: Source Adapter, Director, Triple Parser and Deep Integration Builder. The deep integration process as implemented in DEEP SEMANTICS consists of five steps (see numeration in the figure):

1. Input of the OWL ontology in N-Triple format (file or database).
2. The Source Adapter reads in the ontology triples, preprocesses them and returns an object of class `TripleSet` to the Director.
3. Then the Director invokes the Triple Parser, which transfers the ontology triples in `TripleSet` to RUBY objects and returns instances of class `TemplateValues`.

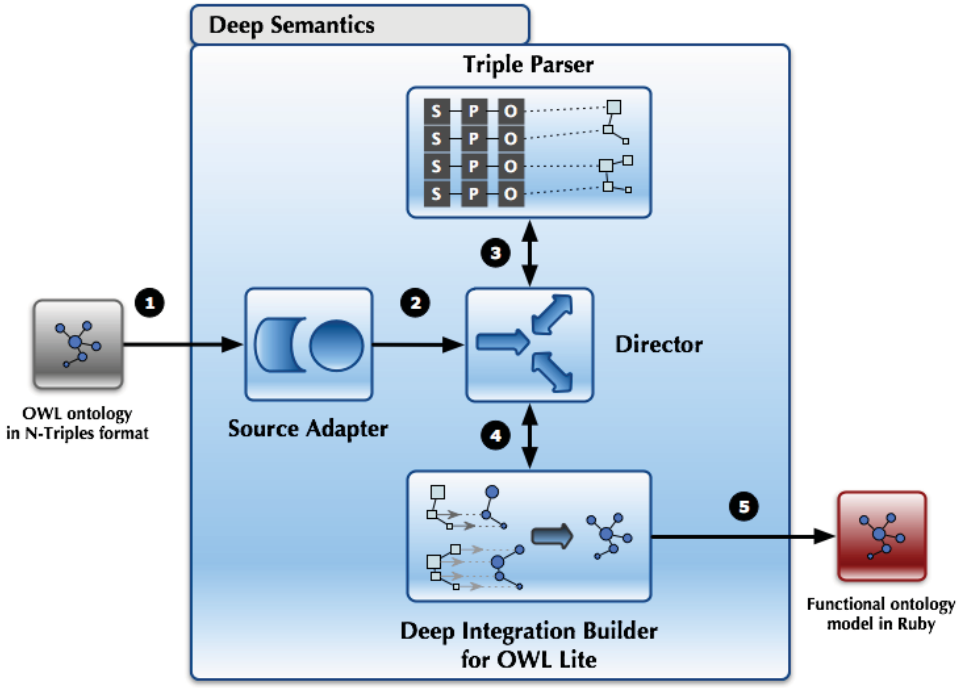

Figure 2.6: Architecture of DEEP SEMANTICS. The director controls the deep integration process, in which ontology triples consisting of subjects (S), predicates (P) and objects (O) are mapped to RUBY code. (*Source:* Mainz, 2008)

4. The template objects are passed to the Deep Integration Builder and the ontology classes, properties and instances are converted resulting the deep-integrated ontology in functional RUBY code.

5. The dynamically built RUBY ontology model can then be applied for handling the ontology in RUBY programs.

The latest version of DEEP SEMANTICS is 0.9. The development of the BIO2ME application was a live test of the framework and provided further impulses of the implementation of DEEP SEMANTICS particularly regarding the access methods to the ontology data.

2.7 Tuple Spaces

In the ONTOVERSE system the management and storage of the ontology data is handled by a so-called tuple space server. The idea of tuple spaces can be ascribed to Prof. David Gelernter and Dr. Nicholas Carriero of the University of Yale (Gelernter, 1985) and describes an architecture (so-called *blackboard architecture*) for distributed systems, in which a tuple space server is the agent and data container. These data persist in form of tuples that are ordered sets of objects or values. Diverse clients exchange data across this central component, the tuple space server, and synchronize processes on the spaces. These clients operate autonomously; so the advantage of this modular and flexible structure is that it easily can be extended and it is robust against failure of single subcomponents. Queries on tuple spaces are not index-oriented, but associatively

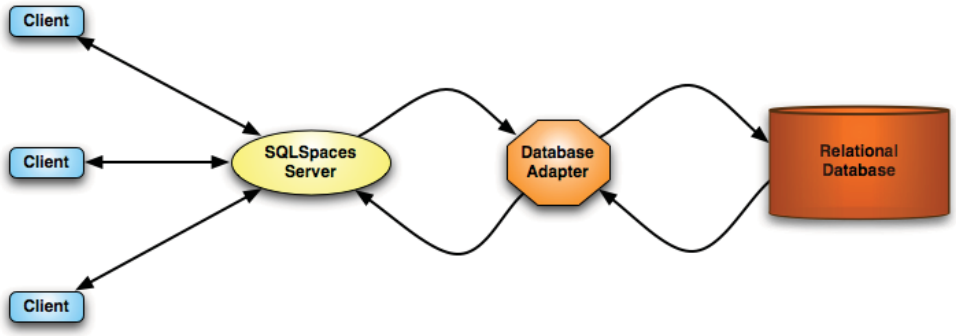

Figure 2.7: Architecture of SQLSpaces. The figure schematizes the position of the SQLSpaces server in the data flow from database to client, *vice versa*.

formulated; this means queries are not expressed by an ID like in databses, but by using the content of a template tuple in which some (tuple) fields are or are not allocated (see Section 4.3 for an example).

There exist diverse implementations of the tuple space idea, well-established realizations are TSpaces by IBM (Wyckoff *et al.*, 1998) and JavaSpaces by Sun (Freeman *et al.*, 1999). ONTOVERSE utilizes the tuple space implementation of Prof. Hoppe's research group from the University of Duisburg-Essen who maintain and extended it by the needs of the project (Malzahn *et al.*, 2007). This implementation is called SQLSpaces and provides versioning of tuples and the administration of users and their rights. Figure 2.7 shows the architecture of SQLSpaces. The Relational databases, like MySQL in ONTOVERSE, are used as persistence layer. The basic function of the SQLSpaces server is the conversion of tuple space operations into SQL queries to the underlying relational database which forms the persistence layer.

Ontological data is stored in form of RDF triples (subject, predicate, object). Each triple with its creation and modification time forms a tuple in a space whereas each ontology has its own space. The mapping of OWL data in simple RDF statements is performed by SWAT (Semantic Web Application Toolkit), which builds on the SQLspaces. SWATClients provide functions to access ontology data. There exist methods to list all classes, instances and properties of an ontology. Additionally to the ontology spaces, the ONTOVERSE SQLSpaces server also holds the so-called *session space*, in which all modifications on the ontology data are recorded.

In this thesis, a RUBY-based SWATClient was used to realize the connection between the ONTOVERSE wiki and the formal ontology data stored in the SQLspaces (see Chapter 4).

2.8 Wiki

A wiki is a collection of internet pages that are interlinked to each other by hyperlinks. The key concept of a wiki is its feature, that these web pages cannot only be read by its visitors, but mostly can be edited directly. Therefore, wikis enable a collaborative collection of the users' knowledge in form of (hyper-) texts by simply using a web browser. Wikis do also provide a simple markup language, which is then translated to html for text display; and by offering wiki

editors writing text is just as easily performed as in text editors like WORD of MICROSOFT OFFICE. More than that, the work on a wiki forms a self regulatory system, because each user is able to add his knowledge which is than in turn checked on correctness by other users. That is why it is hard for vandalism to be inserted in a wiki. This self regulatory process is directly depending on the number of visitors. A further aspect in a wiki is the above mentioned interconnectedness of wiki pages. Users are able to add hyperlinks in their texts in an easy way. This enables unrestricted navigations between web pages. Editors can create wiki articles only for navigation purposes as table of contents (e. g. the wiki main page in the ONTOVERSE wiki, see Section 4).

The first system called wiki was the WikiWikiWeb[26] which was developed by Howard Cunningham and got online in 1995. Leuf & Cunningham (2001) describes this wiki's features as well as the collaborative editing processes in a wiki in principle. Today, the most famous example of a wiki is Wikipedia[27], an internet encyclopedia. But there exist also a lot of wikis, which are utilized in companies' intranets for collaborative work.

The wiki is a software of the Web 2.0 initiative where internet users offer there knowledge to form the world wide web themselves. More than that, the wiki principle realizes the actual idea of Tim Berners-Lee which he had in mind for the www itself. But there exist also wikis that are not open for everyone, that for example have access rights for certain user groups or provide special admin pages.

Key component of the most wiki software is a version management. Each save of a wiki article is recorded under its own version number. So it is possible to revert articles to an older version or to compare different versions of a wiki page to easily retrace modifications. The version management facilitates the elimination of vandalism or the correcting of mistakes, for there is no review before the modification are saved, in general. The philosophy of wikis is the ease of correcting mistakes, instead of tightening input controls.

The disadvantage of wikis is that there are practically no rules, which can and do in practice result to messy article collections. The search function is to counteract this linking chaos. The minimum of restrictions philosophy results in less revision before saving, so content of wiki articles should be handeled with care in particular in wikis with an open community. That is why the german Wikipedia introduced a revision control system[28], the Flagged Revisions, in May 2008 (test status). Users now are able to label single article versions to report the quality of an article.

The information of this section adequately origined, beneath a lot of wiki experience by myself, from the Wikipedia article about "Wiki" and associated articles (as of August 3rd, 2008).

[26] http://c2.com/cgi/wiki

[27] http://www.wikipedia.com

[28] http://de.wikipedia.org/wiki/Hilfe:Gesichtete_und_geprüfte_Versionen

2.9 Tagging

Tagging is the mechanism in which keywords (tags) are assigned to items such as publications or digital images. This is a useful way of categorizing items making it easy to search and browse objects. Tagging provides information about information, thus tags are classified as metadata.

Although tagging is already in use for a long time, it became famous on the internet. In 2003 DELICIOUS[29], a social bookmarking platform, went online making internet bookmarks better searchable by available tags. Each user can assign tags to bookmarks without having to stick to any rules. In early 2004 FLICKR[30] followed by providing a website in which a community is able to publish their images and videos, which can be tagged in a DELICIOUS-like manner (Mathes, 2004; Peters & Stock, 2007). That was the beginning of the Social Web (e. g. Ebersbach *et al.*, 2008). Tag clouds display the most common tags in the largest font size, which form starting points to allow people to discover objects on a website. Another usage for tags is to find related objects that share most of the same tags.

On the one hand uncontrolled vocabularies in these tagging mechanisms do not limit the freedom of users in tagging with their personal terms, but on the other hand has messy side effects for the searching for tagged items. There is no information about the semantics of a keyword. So for example synonyms, which are different words expressing (nearly) identical meanings like "abstract" and "summary", cannot be related. Additionally, homonyms cannot be separated, these are words that have different meanings like "mint" ("coin" and e. g. "spearmint").

Several tagging systems antagonize these problems by using controlled vocabularies. The publication database PUBMED[31], for example, tags its publications with the MEDICAL SUBJECT HEADINGS[32] (MeSH®), a controlled vocabulary of more than 22,000 terms. These terms are linked to each other mostly by simple hierarchical relations. To tag publications with those notions, a lot of indexers at the National Library of Medicine (NLM) are engaged with analyzing the journals' and publications' contents. To ensure the most specific MESH term is used, authors are not allowed to assign their own keywords in PUBMED. Additionally, NLM employs many information specialists, who develop and maintain the retrieval system. All these information is derived from the NLM Frequently Asked Questions about Indexing[33].

[29] http://delicious.com/

[30] http://www.flickr.com/

[31] http://www.ncbi.nlm.nih.gov/pubmed/

[32] http://www.nlm.nih.gov/mesh/

[33] http://www.nlm.nih.gov/bsd/indexfaq.html#keywords

3

Bioinformatics Ontology For Tools and Methods (BIO2ME)

This chapter gives an introduction into the ontology of bioinformatics tools and methods (BIO2ME). First, fundamentals and the preliminary work are described which is outlined in Mainz (2006a). After that, some general ontology extensions in the scope of the investigations for this dissertation are delineated as well as some efforts to encourage external domain experts to offer their knowledge to extend the ontology. Additional modifications on the ontology are commented in the according Chapters 4, 5 and 6. In the discussion and conclusion section of this chapter challenges in building BIO2ME are deduced which motivated the next investigations of this thesis.

3.1 Fundamentals and Preliminary Work

The ontology emerged from the ONTOVERSE project (see section 2.2). It was very important to build an ontology within this project, for this aims at developing a Web-based, collaborative ontology engineering platform. By doing so, a lot of experiences was gained in ontology engineering, advantages and disadvantages of available ontology editors was exposed, and necessary features for the support of the whole ontology engineering process collected. Moreover, the ontology served as a well-known ontology for evaluating and testing purposes on the evolving ONTOVERSE platform. I created BIO2ME, gained relevant concepts, modeled the domain, consulted external domain experts, preprocessed their informal contributions and formalized the ontology in OWL. However, additionally I enlisted the assistance of colleagues with background in bioinformatics, information science and linguistics. We discussed the ontology in form and content and I refined the formal BIO2ME where appropriate regarding their advises (see section 3.2 for an example).

BIO2ME was motivated by earlier research I did during my diplom thesis in biology (Mainz, 2006b) and with some colleagues (Wilm *et al.*, 2006) of the Institute of Biophysics in Düssel-

dorf. Both studies compare several alignment tools[1] with respect to the composition of the input data. This research yielded, amongst others, that the most popular and mostly employed program of this field, CLUSTALW (Thompson *et al.*, 1994), provided not necessarily the best results for each input data set. This pointed out a big challenge in biology and bioinformatics in particular: There is a variety of programs, packages, databases etc. dealing with various problems like the efficient processing of experimental data, sequence analysis and structure prediction and visualization. The major problem is that these resources can currently not be surveyed with reasonable effort. Sometimes even for experts in a specific domain of bioinformatics it is hard to decide which tool fits the given requirements best. Moreover, there are lots of programs dealing with the same problems but using miscellaneous computational, mathematical and biological approaches. The specific challenges of bioinformatics entail the development of new computational methods, some copied from natural processes like genetic algorithms, others are common approaches in mathematics.

In BIO2ME detailed information about bioinformatics tools and methods is collected in a structured way. The practical aim to make these tools easily accessible via a web interface highly influences the actual structure of the ontology. The whole ontology conceptualization was focused on this practical aspect, sometimes dominating over strictly logical representations (e. g. see the modeling of programs in section 3.2). Tools are categorized according to their application ranges (with bioinformatics perspective), supported biological application fields, utilized computational methods, processed data formats and information about the tools. Figure 3.1 exemplifies the filing of a program. BIO2ME is being built to serve as a basis for tool retrieval that meet the user requirements and to examine application ranges of computational methods. Therefore a competency questionnaire was created in Mainz (2006a), which comprehends questions the ontology should be able to answer. Such questions are: “Which tools and methods exist that deal with given problems?” and “Which data output do they provide?”. The realization of the search tool is described in Chapter 6.

A bioinformatician’s motivation to use the BIO2ME ontology is very diverse. It is auxiliary to become acquainted with a new research task in the field of bioinformatics. The search over this ontology quickly reveals relevant references and presents information about tools and how other scientists addressed a certain biological and bioinformatics problem. Moreover, a lot of tools developed in diplom, bachelor, master or PhD theses are not published although they provide good approaches to pursue. The provision of the access to such unpublished tools and their methods is considered as worthwhile for the scientific community. In addition to searching functionalities the web application described in Chapter 6 therefore provides the possibility to easily supply tools to the ontology. Moreover, bioinformaticians benefit by getting an overview of available tools and by having a quick reference to differences between versions and tools, to publications and additional features. The information about input and output formats of a tool facilitates the workflow of tools. In contrast, the additional benefit for experimental biologists is obvious. They can use BIO2ME to find adequate tools for their data analyses and for the planning phase of experiments.

[1] In this context, alignment tools are programs which perform comparative analyses of bio-molecules (RNA, DNA, proteins). The in this way identified similarities of the molecules’ sequences serve as basis for the analysis of the relationship and/or common functions.

At the end of my bachelor thesis mainly the core structure of BIO2ME was designed. Figure 3.1 illustrates the structure of the former ontology by an example tool called "StrAl". For the sake of clarity the values of datatype properties are added into the instance box and not formally correct via arrows. One can see the structure of the ontology which models the tool and its properties. In this figure four top level concepts exist which describe the program STRAL (other branches of the ontological hierarchy are omitted like `DataFormat` and `OperatingSystem`):

`Tool` subsumes all tool classes like `Program` in the figure. `Program` tied all types of bioinformatics programs (here: `StructureAlignmentProgram` and `SequenceAlignmentProgram`). The program class `StrAl` finally was sorted as subclass of some program types. `StrAl0.5.4` is an instance of `StrAl`.

`Task` relates the instance `StrAl0.5.4` to its `BioinformaticsTask` and `BiologicalTask` via the object properties `hasBioinformaticsTask` and `hasBiologicalTask`.

`Data` describes different types of data. There exists the object properties `readsData`, `writesData` and `utilizesScoringMatrix` to relate `Data` instances to the tool's instance.

`ComputationalMethod` classifies algorithms and computational techniques which are used (with object property `usesComputationalMethod`) in programs.

The datatype properties for example assigns information about publications (`isPublishedIn`), help pages (`hasOnlineSupport`), runtime and memory complexity (`hasRuntimeComplexity`, `hasMemoryComplexity`), available user interfaces (`hasGUI`) and download pages (`hasDownloadLocation`).

3.1.1 Scenario

In the following a scenario is described which deals with the protein structure determination of an unknown virus and the development of a therapeutic agent. This scenario illustrates the usability of the ontology and its Web-based application. The procedure of the scenario is yet not fully supported by BIO2ME, for it does not comprise all relevant bioinformatics fields entirely.

See figure 3.2 for an illustration of the following investigative steps:

1. Isolation and sequencing of the nucleic acid of the virus. Based on technical limitations, only short sequence pieces of about 1.000 base pairs are sequenced at once. Viruses can have up to 350.000 base pairs and therefore have to be separated into smaller parts prior to sequencing. Based on overlapping sequences the resulting fragments are finally automatically reassembled with the indispensable aid of *sequence analysis programs*.

2. Characterization of the virus by screening special *virus databases* with *local alignment search tools*. Similar nucleic acid sequences allow conclusions to be drawn to the relationship to known viruses.

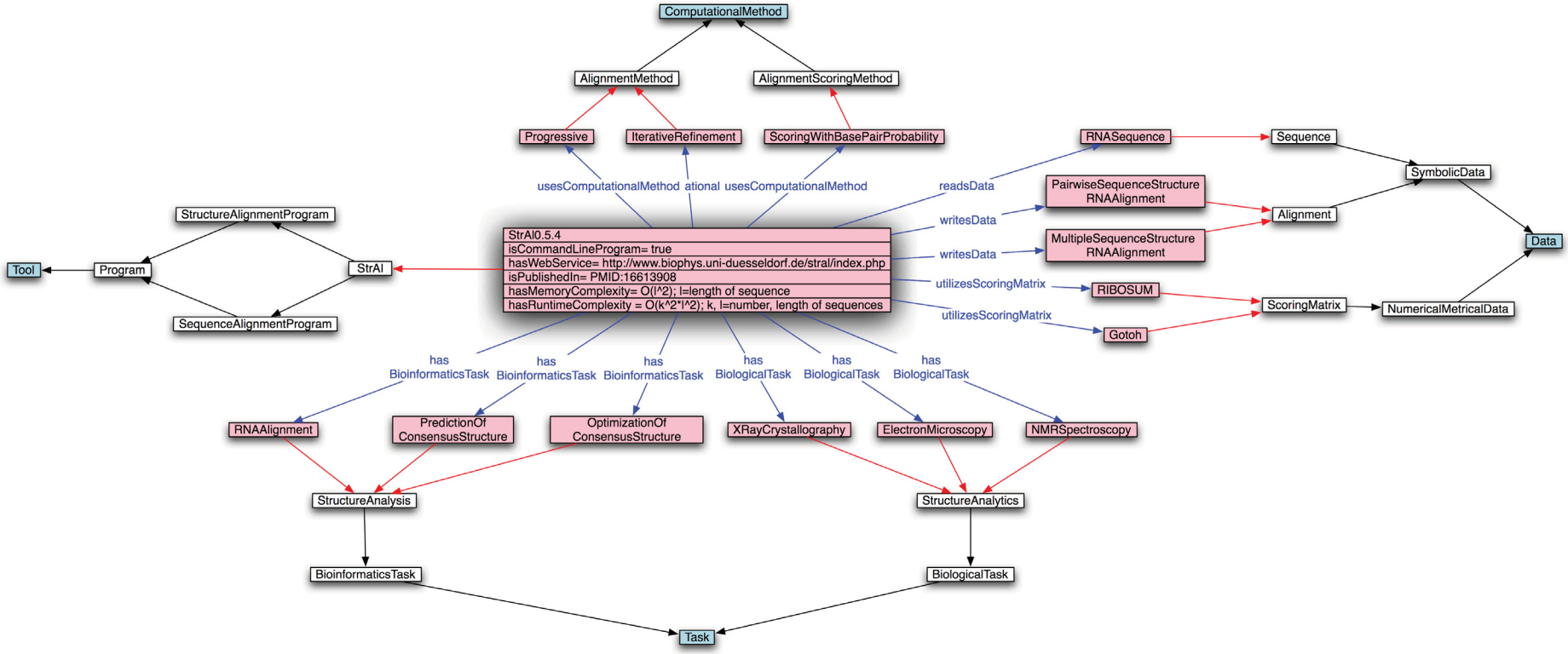

Figure 3.1: Modeling of STRAL 0.5.4. The schema exemplifies the characterization of the alignment program STRAL 0.5.4 in the structure of BIO2ME at the beginning of the research for this thesis. The figure only shows a cutout of the actual modeling of STRAL 0.5.4. Top level concepts are shown in blue boxes and instances in pink. Hierarchical relations are black arrows, object properties blue and "instance of" relations red colored. The box of the instance `StrAl0.5.4` contains a cutout of assigned datatype properties. Some intermediate classes are omitted. (*Source:* Mainz, 2006a)

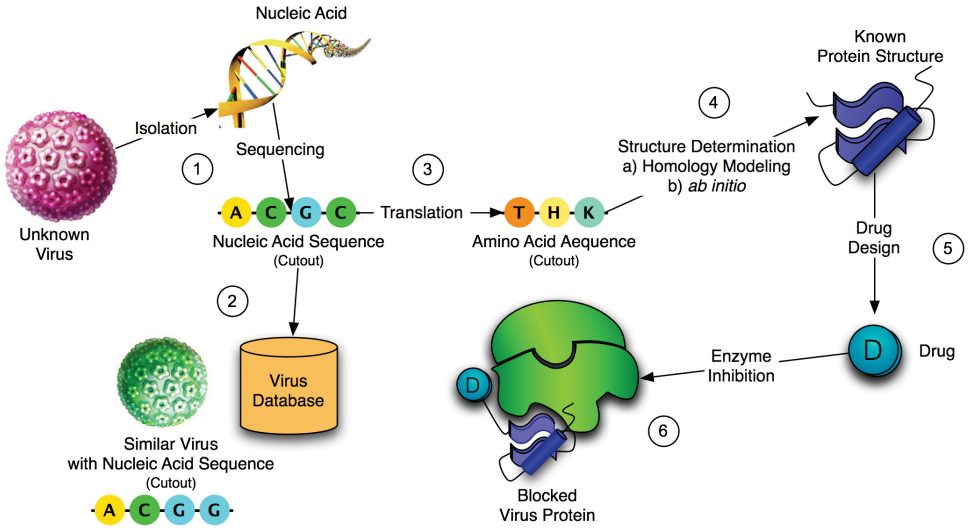

Figure 3.2: BIO2ME scenario. Laboratory techniques and bioinformatics tools which are utilized in the development of a drug against an unknown virus.

3. Translation of the nucleic acid into the amino acid sequence of virus proteins facilitated by *programs using the according genetic code*. This enables the detection of target sequences and structures on the protein's surface for the development of virus-inhibitive agents.

4. Simulation of protein structures by special *protein structure prediction programs*. For that purpose specific *databases* are searched for related proteins with known structure. If a protein with similar sequence is found, *homology modeling programs* can determine the protein structure. Otherwise *ab initio* methods can be utilized to compute the three dimensional structure.

5. Development of a virus-inhibitive agent based on the determined structure.

6. Some regions on the surface of proteins are essential for its function and can be blocked. For example a molecule might be identified which is complementary to such a sequence and therefore inhibits this virus.

This scenario reveals the context-dependent use of certain types of databases and programs, for example the distinction of sequence, structure and organism databases.

3.2 Refinement of BIO2ME

Based on the results of the bachelor thesis, BIO2ME was extented and refined during the research for this thesis. The class structure of the ontology was reorganized to some extent. Figure 3.3 shows the same information of the program STRAL as shown in figure 3.1 but modeled in the current ontology version. It is recognizable that the modeling of the top level concepts `ComputationalMethod` and `Data` did not change. There are additional subclasses and some refinements in their structure, but in this narrow context no changes are visible. The main

Table 3.1: BIO2ME data. The table shows the ontology metrics of the state at the end of the bachelor thesis (BIO2ME BA) and now (BIO2ME NOW). "Object property assignments" are interrelations between individuals and "Datatype roperty assignments" subsume mappings of datatype values to individuals. Relation counts only consider direct relationships.

	BIO2ME BA	**BIO2ME NOW**
Classes	100	212
Individuals	139	348
Object properties	16	42
Datatype properties	12	29
Instance of relations	155	473
Object property assignments	449	1139
Datatype property assignments	78	405
Annotations	99	1100

modification in BIO2ME is the movement of the class `BioinformaticsTask` to `Tool` and its renaming to `BioinformaticsTool`. Although the same information of tools are modeled, the shifting and renaming of the class completely changes the formal logic of the concept. Subclasses of `BioinformaticsTool` are no longer meant as bioinformatics research fields and problem solving strategies, but map specific types of tools. So by now a tool's bioinformatics tasks are no longer modeled in BIO2ME, but the bioinformatics type of a tool. A bioinformatics type of STRAL 0.5.4 for example is sequence alignment tool in contrast to its classification as program. The result in the sense of the information that can be extracted from the ontology is the same, however logically these have totally different meanings. The instances of tools are no longer related to bioinformatics tasks by the object property `hasBioinformaticsTask`, but are now instances of `BioinformaticsTool`, too. `Program` retained, although logically meaningless, because it immensely facilitates the retrieving and insertion of tools (see Chapter 6). The top level concept `Task` was no longer needed and `BiologicalTask` ascended to a direct subclass of `Thing`.

Figure 3.4 illustrates an overview of the current structure of BIO2ME. The figure in particular shows the modeling of the concept program. For the reason of clarity only low level classes are shown and just a subset of properties. For the most object properties inverses are omitted. The actual counts of ontology elements and assignments are summarized in Table 3.1. Figure 3.4 emphasizes the interwoven structure of an ontology. Each class is related to at least one other through subclass relationships and object properties, respectively. In the following the process of BIO2ME refinement is described in more detail for three aspects.

Domains & Ranges

For some datatype and object properties the domain and/or range specification had to be refined, because the initially used ontology editor PROTÉGÉ version 3.2.1 contains a bug in the domain and range implementation. If more than one domain or range is specified, the editor stores the

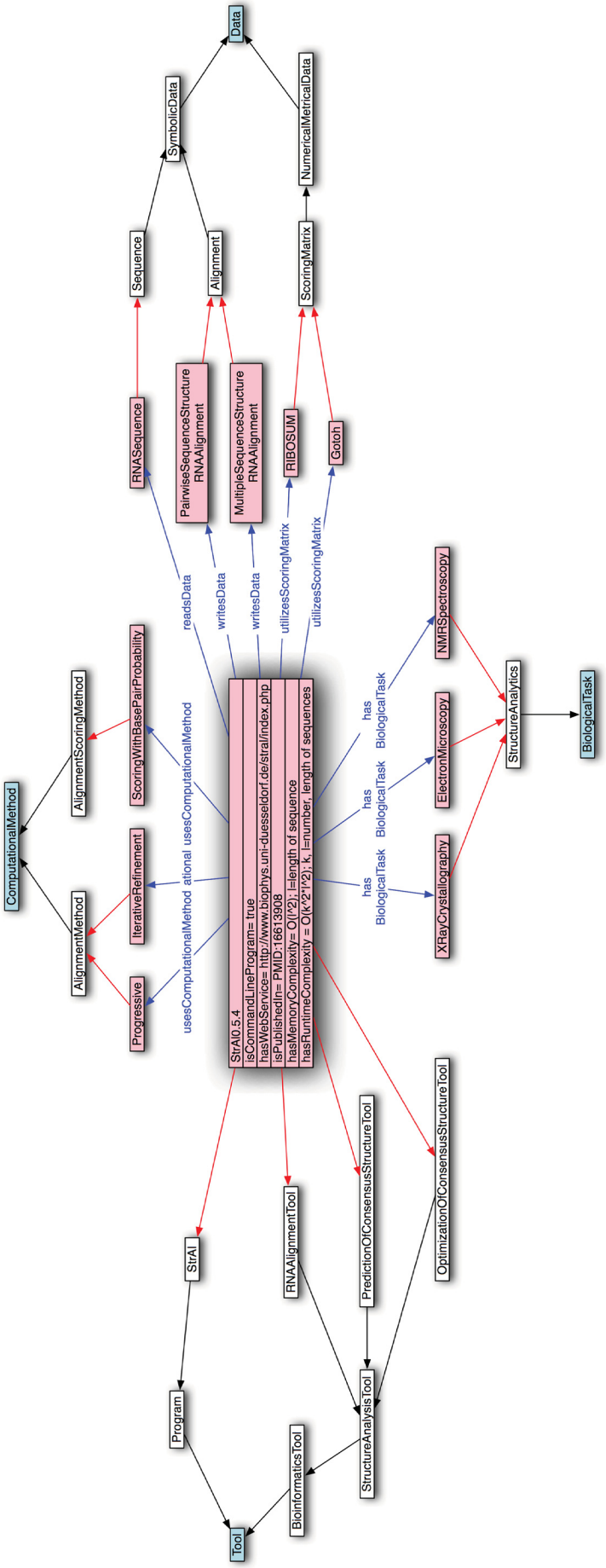

Figure 3.3: New Modeling of STRAL 0.5.4. The schema visualizes the same cutout of the ontology as figure 3.1 does, but modifies it with regard to the refinements of BIO2ME.

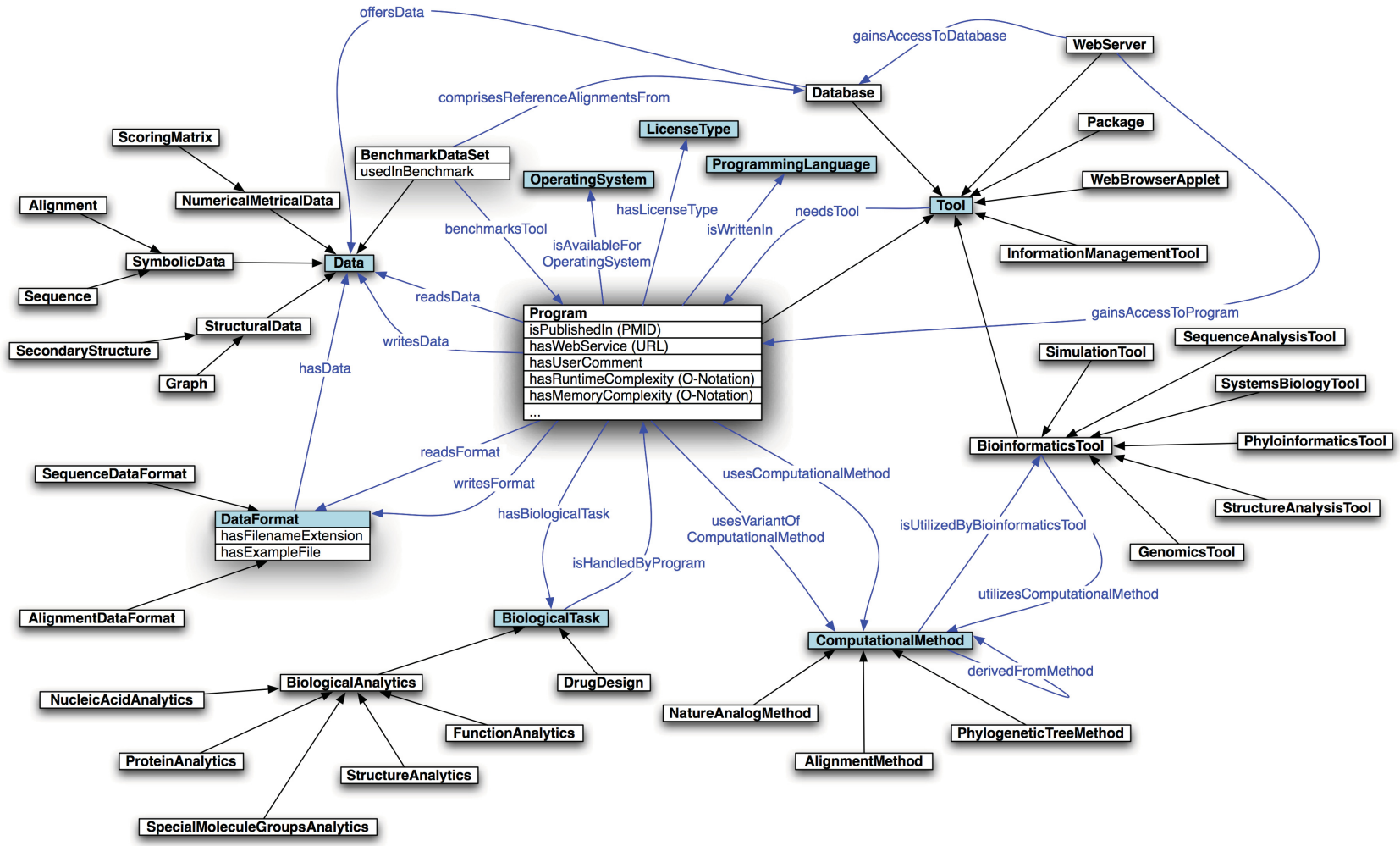

Figure 3.4: BIO2ME Overview. The figure shows extracts of BIO2ME. Black arrows illustrate subclass relationships and blue pointers mean object properties. Some datatype properties are inserted in the concept boxes. Direct subclasses of `owl:Thing` are blue colored. No instances are shown.

union of them, this is not conform to the OWL specification of the W3C. Its section about abstract syntax[2] in paragraph 2.3.1.3. "OWL Lite Property Axioms" says:

> "There can be multiple domains, in which case only individuals that belong to all of the domains are potential subjects. [...] Again, there can be multiple ranges, in which case only individuals or data values that belong to all of the ranges are potential objects."

This means that multiple domains and ranges are the *intersection* of all domain and range classes of a property, respectively. On logical level these are two completely different modelings. PROTÉGÉ version 4 fixes this bug.

For the implementation of the BIO2ME application (see Chapter 6) the DEEP SEMANTICS framework (see Chapter 2.6) was utilized. DEEP SEMANTICS by now handles OWL lite, but is already extended by some additional OWL DL features (see Section 2.1.3). Such features are unions; intersections are not supported yet; so multiple domains and ranges had to be removed from BIO2ME. For example, object property `gainsAccessTo` with domain `WebServer` currently has `Tool` as range. This is not quite correct, because for example web servers do not gain access to the functions of packages. Since `Package` is modeled as subclass of `Tool`, it inherits the range axiom of `Tool`. That is why two subproperties are modeled, `gainsAccesToDatabase` and `gainsAccessToProgram`, which have range `Database` and `Program`, respectively.

Object Property: `usesVariantOfComputationalMethod`

The object property `usesVariantOfComputationalMethod` resulted from a discussion with a linguist. The old version of BIO2ME included an instance of `ComputationalMethod` named `VariationOfSankoffAlgorithm`. The linguist stressed that this is linguistically incorrect, because an instance models a specific entity of the domain, but a variant of something is only a blurred describtion of a thing. From the bioinformatician's perspective, it is yet useful to have the information that a program uses not exactly the Sankoff algorithm, but a variant of it. This method even is called "variant of Sankoff algorithm" in bioinformatics publications (e. g. Hofacker *et al.*, 2004). The alternative way in modeling this information linguistically correct without any loss of facts was the insertion of the new object property `usesVariantOfComputationalMethods`. This clearly defines the relationship between a tool and its computational method.

Language of Datatype Values

OWL allows the declaration of the language of datatype values. Thus one can include several languages in one's ontology to enable its utilization in different language areas. However, PROTÉGÉ stores this information in the RDF/XML file in a not OWL DL conform way. By

[2] `http://www.w3.org/TR/owl-semantics/syntax.html`

reading ontologies including such constructs, the OWL reasoner PELLET warns that an extra OWL description has to be inserted into the OWL file to trnasfer the ontology into OWL DL.

3.2.1 Integration of External Domain Experts

The collection of bioinformatics programs and the guarantee of a common agreement of the mapping of biological and bioinformatics domains brought me to consult external domain experts. For that purpose a document containing a short introduction to ontologies and the motivation of BIO2ME was created, together with a questionnaire which made a survey of bioinformatics tools and their domains. This document was sent to some bioinformaticians. The response was rather poor. Nevertheless I captured some new tools in the ontology and got feedback on the questionnaire itself. Based on that I implemented a Web form, which is much more comfortable in usage. The link was sent to some bioinformatics groups in Germany and Austria, but again the response was not worth to mention. After these attempts I decided to stop the recruiting of external domain experts for the moment and concentrated on semi-automated extension techniques and the implementation of the search application of BIO2ME to show its benefits.

The result of one discussion with domain experts, for example, was the integration of the new datatype property `hasUserComment` (see Fig. 3.4) into the ontology that gathers users' experiences with the tool.

3.3 Discussion

This chapter dealed with the bioinformatics ontology for tools and methods. It points out the permanently unfinished state of an ontology for this application. During the work the class structure and property definition of BIO2ME was reorganized. Additional changes were done due to application purposes. Now it serves as a semantic basis of a search application, moreover the search application was designed around this knowledge base. Further modifications on the ontology would now entail a reimplementation of the application. That is why the class modeling task has to be well thought before the application is built around it. That was the reason to rethink about BIO2ME after the bachelor thesis. The described modifications were discussed and approved by several (internal) domain experts. BIO2ME does not claim to be complete, but the core structure is modeled sufficiently and expanded in some bioinformatics domains. Additional extensions with regard to the ontology design guidelines can now be done without suspecting any trouble with the web application. BIO2ME will always grow with its application.

Table 3.1 shows the data of BIO2ME. Obviously, the number of classes doubled since the final version of the bachelor thesis. The individual count increased even more by two and a half. The most investigations can be ascribed to the BIO2ME extension and the more precisely modeling of the domain by inserting new types of relations (object and datatype properties). The new structure of the ontology is described above and illustrated in figure 3.4 as well. The

triplication of "instance of" relations is caused by insertion of new individuals and the new modeling of the bioinformatics task of a tool. Program versions are now instances of `Program` and the actual program's class, respectively, and of `BioinformaticsTool` which replaced `BioinformaticsTask` and the object property `hasBioinformaticsTask` (more details above). Based on the new modeled properties the connections of instances among each other and instances with values are increased by two and a half and more than five, respectively. The drastically increased number of annotations by 11-times in particular is caused by the insertion of `rdfs:labels` which add natural language identifiers and synonyms to the artificial, rule-based local names of ontology objects.

3.3.1 Integration of External Domain Experts

The less successful integration of external domain experts is ascribed to various reasons. First, the ontology was not published in a biological journal. So the domain experts did not know the benefits it will offer for them. Moreover, for the characterization of tools a lot of information has to be gained. The experts did not have to answer each question but the size of the questionnaire might have alienated them. At the time of consulting external experts the web tool was not available, so the Word document and the simple web form might also not have attracted.

The challenge in recruiting domain experts in combination with the wide domain induced the support of the knowledge acquisition phase by semi-automatic methods (see Chapter 5). Additionally, the feedback, I got from some experts, was considered in the implementation of the BIO2ME application (see Section 6.3). Additionally, Chapter 6 provides a new approach of attracting domain experts by offering more information about the structure of BIO2ME and invloving them into the extension of the ontology.

3.3.2 Conclusions

During the design and extension of BIO2ME a lot of experiences were gained. Obviously the benefit of the BIO2ME application relies on the quantity and quality of the underlying ontology. That is why it has to be constantly enlarged and updated. This is certainly not manageable by a single person, and even a (small) group of knowledge engineers is reliant on the support of the tools' developers and users, the domain experts, in extending and evaluating the ontology. Currently available ontology engineering tools are insufficient for these purposes, that is why a common internet technique, a wiki, has been adapted that supports the collaboration of domain experts. Investigations on this wiki are outlined in Chapter 4.

Furthermore, Section 3.2.1 showed that the acquisition of domain experts is a challenge. Hence, a machine-supported ontology extension approach was introduced, which is described and evaluated in Chapter 5. This approach utilizes tagging of publications basing on the concepts of the ontology and adverts to possible new ontology relevant notions and relations. The user gets hints of text passages that could be relevant for the ontology.

The construction of BIO2ME teached the following lessons and motivated the research described in the next sections:

- The domain of BIO2ME eminently points out the need for collaborative ontology engineering. To represent bioinformatics tools with their applications, the whole bioinformatics research field and biological areas utilizing bioinformatics tools have to be mapped adequately in a structured way. Various kinds of expert knowledge are needed to characterize different functions and application areas of bioinformatics tools. Furthermore, a vital community is needed to add information on the different tools that should be represented.
  ⇒ To support this, a wiki was implemented into the ONTOVERSE platform (see Chapter 4).

- In long term, the major challenge with BIO2ME will be to keep it up-to-date. It will be necessary to keep track of new developments in bioinformatics, e. g. as new tools and new versions of existing tools may be published.
  ⇒ To support this informally, a wiki was implemented into the ONTOVERSE platform (see Chapter 4) and the information system that uses BIO2ME as knowledge base also integrates a simple interface to insert tools and methods directly into the ontology (see Chapter 6).

- The problem with recruiting domain experts who are willing to share their knowledge was detected, so an alternative way of collecting relevant background knowledge (in form of publications) for extending the ontology was developed. Though this approach will never be able to replace the intervention and mental power of a domain expert.
  ⇒ This approach is based on tagging and is described in Chapter 5.

- The fundamental challenge of this particular ontology was to define its basic structure. This is where the highest quality control is needed, because it is most difficult to change underlying structures at a later time point. It is also the part of the ontology engineering process that requires the most discussion and planning. Less fundamental aspects, like adding new instances to existing concepts, can however easily and freely be handled by domain experts.

4

Wiki – Support of the Protoontological Phase and More

The informal phases of the ontology engineering process, the acquisition of domain, motivation and goal in the ORSD (see Section 2.1.4) and the conceptualization are fundamental particularly for collaborative ontology development. As my colleagues and I experienced during the collaborative extension of BIO2ME, these steps are hardly supported in published ontology editors. I am aware of one editor, ONTOEDIT (Sure *et al.*, 2002), that provides support for the creation of competency questionnaires (ONTOKICK plugin) and of mind maps (MIND2ONTO plugin). That is why a locally installed MEDIAWIKI[1] was utilized for BIO2ME design. The wiki idea and its functions (see Background 2.8) are perfectly fitting this informal knowledge capturing process. For these reasons and because its intuitive handling, a wiki was included into the ONTOVERSE system, which facilitates such protoontological phases. Domain experts, who will primarily take part in the informal ontology development steps and who are not necessarily familiar with formal ontology aspects, might be overstrained by a formal editor. Additionally, content-specific evaluation of the formal ontology, periodical checks for consistency, accuracy and correctness by domain experts should also be supported. For this reason the ONTOVERSE wiki has a connection to the formal ontology (see Section 4.3). Due to this wiki-editor conjunction, no traditional software package (like MEDIAWIKI) was integrated into the system, but I implemented a wiki in the programming language RUBY. This implementation was much easier to extend with required ONTOVERSE-specific wiki features. Furthermore, the ONTOVERSE platform itself is realized with RUBY and RUBY ON RAILS, that is why this solution fits much better into the whole system.

Section 2.1.4 describes the ontology engineering process developed in my Bachelor thesis. Figure 4.1 shows modifications of Figure 2.4 by using the wiki as collaborative document editor and repository. Now (meta-)results of informal phases can be cooperatively worked out in wiki

[1] `http://www.mediawiki.org/wiki/MediaWiki`

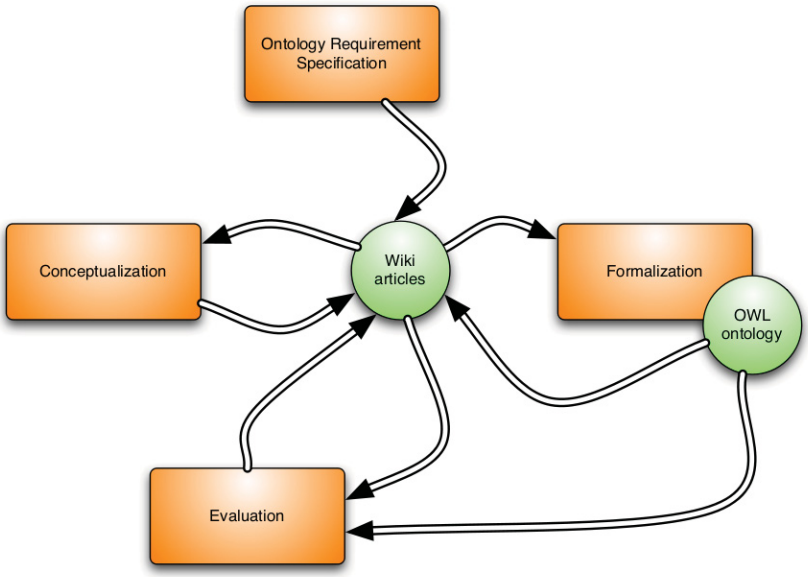

Figure 4.1: Ontology engineering process supported by the ONTOVERSE wiki. The schema modifies Figure 2.4. All protoontological documents (e. g. ORSD, concept lists, etc.) are now created, versioned and collected in the wiki.

articles. This supersedes the use of revision control systems like Subversion[2] (SVN) or the Concurrent Versions System[3] (CVS).

In the ONTOVERSE system each ontology project has its own wiki with a main page. Only project members are able to see articles in the project wiki. The ONTOVERSE wiki exploits the unrestrictive navigation possibility, so the main page serves as starting point of article linking. New wiki articles are created by inserting internal wiki links to a new article.

With a project start the wiki main page (see Fig. 4.2) is automatically created and filled with a link to the project's ORSD, its conceptualization and articles that list actual ontology classes and individuals (see Section 4.3). In this process the ORSD article is automatically prefilled with the subdivision that was developed in Mainz (2006a) for the ONTOVERSE project. Moreover, it contains descriptions for each section that explain what should be worked in. The "Conceptualization" link refers to an empty wiki article as hint for ontology engineers to add articles e. g. collections of relevant terms and concept graphs. Users are not limited, in which articles they want to integrate here, because each project and team has its own necessities. In the BIO2ME wiki, articles for unstructured lists of concepts and properties were created and concepts graphs were added to facilitate the modeling in OWL.

In the backend only one wiki exists, which simulates all project wikis. Fig. 4.3 shows the database schema of the wiki. Every wiki page is saved in table `articles` including title, body text, version number, creator, creation and update time. The assignment to a project is realized for each article by defining the foreign key `project_id`. In the opposite direction each project can possess any number of articles. The other database tables in this figure are explicated in fol-

[2] `http://subversion.tigris.org/`

[3] `http://www.nongnu.org/cvs/`

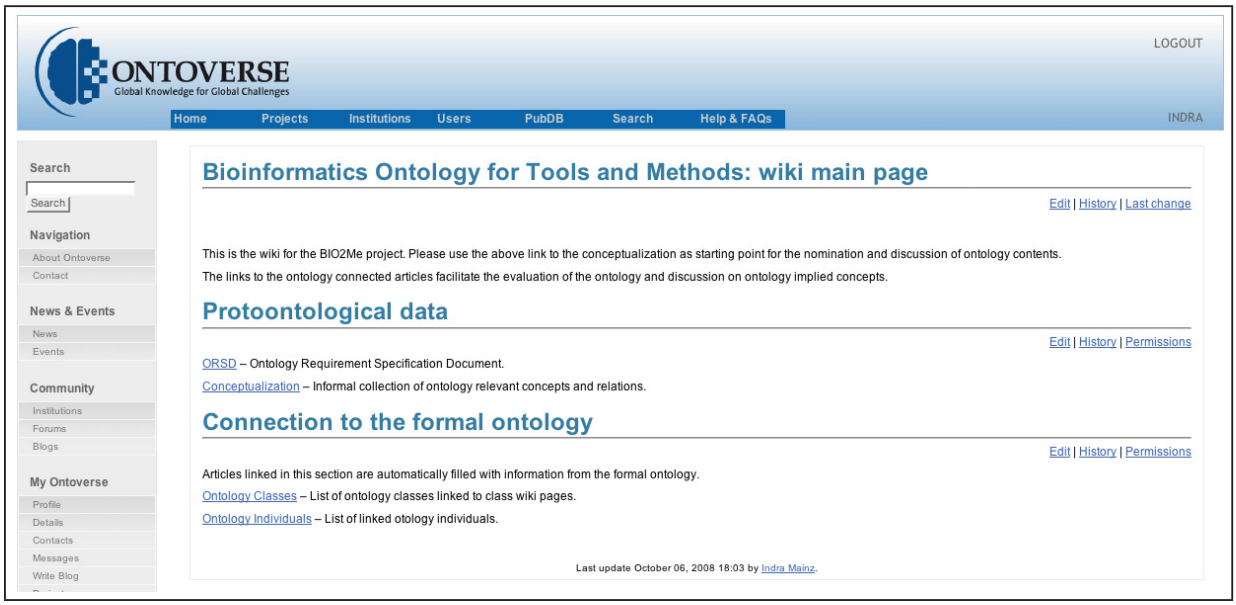

Figure 4.2: Wiki main page. Screenshot of the wiki main page of the BIO2ME project.

lowing sections (4.1.2, 4.1.3 and 4.2). The ONTOVERSE system utilizes the AUTHORIZATION[4] plugin, which facilitates the implementation of access restrictions in RUBY ON RAILS applications. See Paulsen (2007; pages 79ff) for the description of the role-based access model. This implementation also controls the project-specific access to the project affiliated wiki. Internal links between wiki pages are also only allowed between articles of the same project as other articles are not visible in the current project.

4.1 Basic Wiki Features

Section 2.8 describes common wiki features. These features proved themselves as very useful in the design phase of BIO2ME, so they were integrated into the ONTOVERSE wiki. The following sections briefly describe these functions and their implementation with the aid of RUBY ON RAILS plugins.

4.1.1 Text Formatting

Users are able to insert simple wiki-specific text formattings into articles. The utilized markup language is TEXTILE[5,6], which enables easy text formatting in hypertexts for users who are unfamiliar with html. In contrast to html this markup language was developed to be readable for humans and easy to learn. Table 4.1 lists some textile syntax. More information is given on the ONTOVERSE wiki help page[7]. TEXTILE also allows the input of html syntax.

[4] `http://github.com/DocSavage/rails-authorization-plugin/tree/master/`

[5] `http://textism.com/tools/textile/`

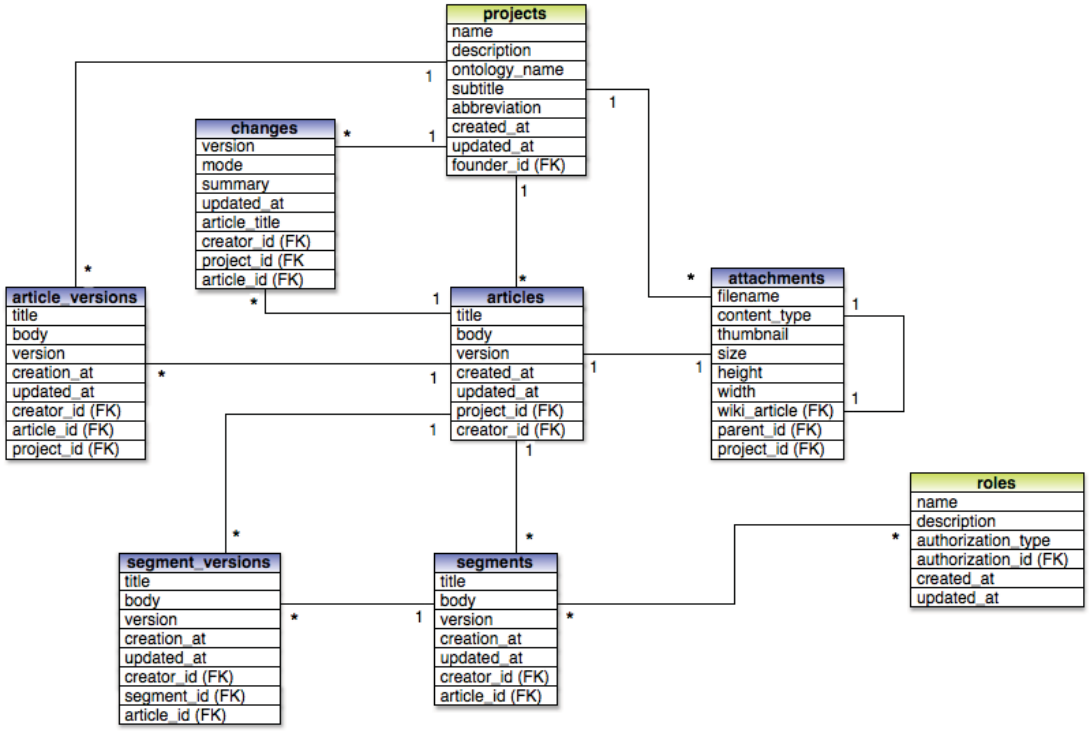

Figure 4.3: Entity Relationship Model of the Wiki. The schema shows the database tables storing the data of the wiki. The green color of the `projects` table indicates that this table is also linked to other tables for it is involved in other platform modules. Foreign keys are marked with FK. (*Source:* Extended after Paulsen, 2007).

The implementation of the ONTOVERSE wiki utilizes the RAILS plugin ACTS_AS_TEXTILED[8]. This facilitates a simple integration of REDCLOTH[9], a RUBY realization of TEXTILE. In the wiki the article's body attribute is set to be textiled. The plugin stores textile in the database and translates it into html on default, because html is displayed more often than textiled text in web pages. Given an article, the simple output of the body attribute delivers html text:

```
>> article.body
=> "<h2>Mind Map</h2> <img src=\"Program_MindMap.png\" alt=\"\" />"
```

If textile formatting is to be rendered, for example on the wiki edit page, it has to be declared like this:

```
>> article.body_source
=> "h2. Mind Map !Program.png!"
```

TEXTILE was customized to meet the ONTOVERSE requirements. For this purpose the markup was extended by some functions for the integration of internal links, attachments and segments.

[6] http://hobix.com/textile/

[7] http://ontoverse.cs.uni-duesseldorf.de/wiki/help

[8] http://errtheblog.com/posts/12-actsastextiled

[9] http://whytheluckystiff.net/ruby/redcloth/

Table 4.1: Wiki Formatting. The table shows some common textile formatting rules for the ONTOVERSE wiki.

Captions	

`h1. Caption 1`
`h2. Caption 2`
`h3. Caption 3`
	Caption 1
	Caption 2
	Caption 3

Font Styles and Phrase Modifiers	
`**bold**`	**bold**
`_italics_`	*italics*
`+underlined+`	underlined
`-deleted-`	~~deleted~~
`%{color:red}red colored text%`	red colored text
Bulleted Lists	

`* First.`
`* Second.`
`* Third.`
	• First.
	• Second.
	• Third.

Enumerations	

`# First.`
`# Second.`
`# Third.`
	1. First.
	2. Second.
	3. Third.

External and internal links	
`"Ontoverse":http://www.ontoverse.org` `"Article":intern`	Ontoverse Article
Footnotes	
`This text has a footnote[1].` `fn1. This is footnote number 1.`	This text has a footnote[1] [1]This is footnote number 1.

These extensions are implemented in a wiki helper method in RAILS (see Section 2.5.1) called `to_html`. The markup of external links, which direct to the provided internet address, is handled by TEXTILE. The translating of links that cause the loading of other project-assigned wiki articles is facilitated for users. The declaration of the article name suffices as article titles are unique for one project. The `articles` table is then searched through with the aid of the article title, to which the link directs, and the project id, which is associated with the current article. If an article satisfies this information, the html image tag is composed and inserted into the current article's body. In this process URL mapping of RAILS is exploited (see Section 2.5.1). More information about customizations for the integration of attachments and segments are given in sections 4.1.2 and 4.2.

The wiki edit page (see Fig. 4.4) provides a summary text field, in which modification comments can be pasted (see Version Management 4.1.3), and a help site where the formatting syntax is

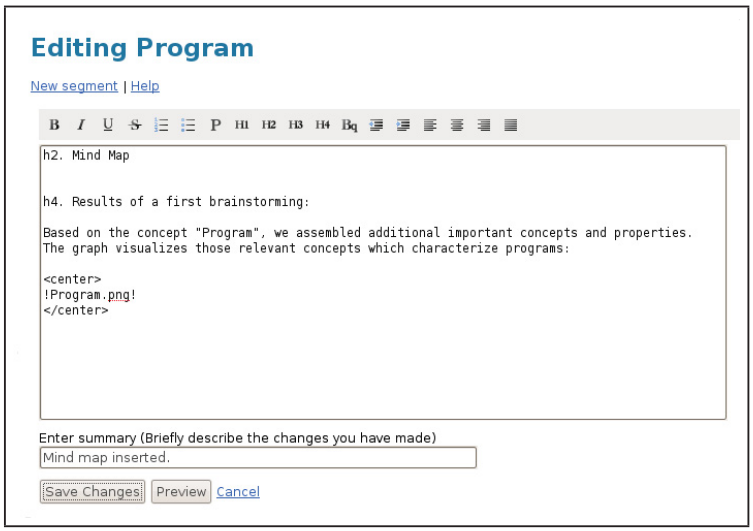

Figure 4.4: Wiki Edit Page. The screenshot shows the rich text editor of the ONTOVERSE wiki.

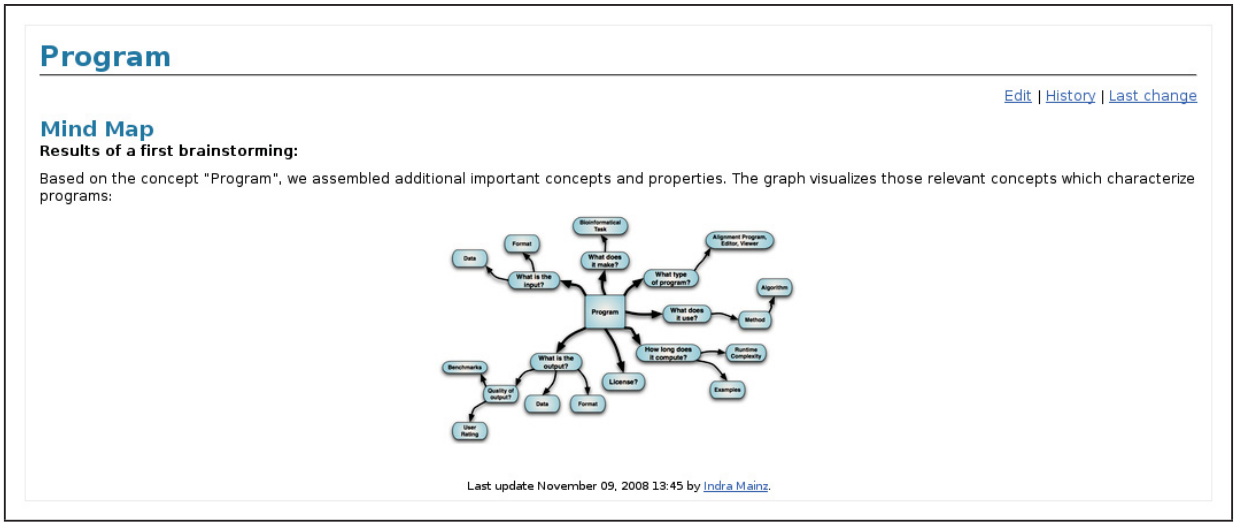

Figure 4.5: Wiki Attachment. Screenshot of the conceptualization of the concept “program”. A mind map is inserted into the article.

---

presented. Moreover, Dr. Ingo Paulsen added javascript functions that facilitate the correct text formatting by providing some buttons for inserting headings, bullets etc.

4.1.2 Attachments

Additionally to wiki articles, the system offers the possibility to attach project-specific files like images or pdf documents. By uploading a file with clicking the “Upload Attachment” button in the project-specific navigation sidebar, the attachment is tied to the project and a special wiki page for the attachment is created. Attachment files can then easily integrate into a project’s article by inserting:

```
!file_name!
```

Images are displayed and serve as link to the image's wiki article (see Fig. 4.5). Other file types are textually linked to their wiki pages. On these attachment wiki pages users can view and download the files. Links to pdf documents are additionally marked by a symbol.

Attachments are particularly helpful in articles dealing with the conceptualization. For example, mind maps can be uploaded to visualize an overview of the domain and concept graphs can point out the connections between domain relevant terms. A listing of all attachments of a project is provided which is helpful to get an overview of available files. The list directly links to the according attachment article in the wiki.

In the backend, the RUBY ON RAILS plugin ATTACHMENT_FU[10] manages the upload and storage of files. By uploading an image file, a thumbnail is created automatically. The plugin provides three storage options, whereof ONTOVERSE uses the file system storage. Metadata of each file is saved in the database, too. Figure 4.3 visualizes the corresponding database table `attachments`. The file name, the content type (e. g. `image/png` or `application/pdf`), the metrics, the name of the thumbnail file, if the attachment is an image, and the id of the parent file of the thumbnail are stored in this table. Each attachment has assigned exactly one attachment wiki article, which is automatically created at upload time. The opposite direction models the relation of the special attachment wiki articles to attachments. The article affiliation of attachments which are inserted into wiki articles are not managed in the database, but by the text markup. TEXTILE substitutes the above wiki syntax into html for images. By regular expressions in a RAILS wiki helper method, these `<img>` tags are extended by hyperlinks to the attachment wiki article or substituted by textual hyperlinks in case of non-image file types, respectively. Each attachment belongs exactly to one project, however a project may possess any number of attachments (see Fig. 4.3).

4.1.3 Version Management

Each article saving triggers a versioning process of the article. This change log enables certain common wiki functions. The "Recent Changes" web page lists all articles sorted in descending order by update time. In the ONTOVERSE system this list is of course restricted to project affiliations. Beside the editor's user name it holds information about the modification type denoted by a single character coding (N - New, M - Modified, R - Reverted, D - Deleted) and an additional modification comment. This comment is either automatically filled in for system generated articles or added by the editor. Therefore the wiki edit page offers a textfield to enter a comment on the modification (see Fig. 4.4). More than the "Recent Changes" section, the wiki provides a history for each article, in which the user is able to revert to previous article versions and a differences function which highlights the changes between two revisions. This helps to retrace modifications.

In the course of versioning, the RUBY ON RAILS plugin ACTS_AS_VERSIONED[11] is utilized. This library adds simple versioning and revision functions for RUBY ON RAILS models. In this case, a copy of an article entry in the database is automatically saved in a versioned table by

[10] `http://svn.techno-weenie.net/projects/plugins/attachment_fu/`

[11] `http://ar-versioned.rubyforge.org/`

saving a wiki article. This requires the versioned table, `article_versions` (see Fig. 4.3), and the "version" attribute in the "article" RUBY model. The versioned table needs to contain a field for those article columns that should be versioned, the article's title and body in this case, a "version" field and the article id as foreign key to the `articles` table. Each entry belongs to exactly one article and one project.

For the purpose of meaningful recent changes listings, the `changes` table was created in the database (see Fig. 4.3). The relevant fields are "mode" and "summary", which store modification type and a comment as described above.

4.1.4 Search

The wiki offers its own search functionality. Therefore a wiki search field is visible in the project-specific sidebar if the user is on a wiki page. After entering a search pattern, all project related article titles and bodies are searched through. The search is realized as phonetic search. This means that search patterns are broken down into its phonemes to build the phonetic representation of the pattern. In this process similar phonemes are united. After that, this phonetic representation is compared to the phonetic presentation of the wiki texts. In this way misspellings are ignored.

This phonetic full text search is implemented with the RUBY ON RAILS plugin ACTS_AS_FERRET[12]. This extension builds on FERRET[13], which is a RUBY version of APACHE LUCENE[14]. ACTS_AS_FERRET supports high speed full text search in defined RUBY model attributes. It automatically and imperceptibly manages the update of the search index in the operating wiki.

4.2 Segments

In comparison to regular wikis, one special feature in the ONTOVERSE wiki is the introduction of so called *segments*. They are sections in an article with their own history and version management. The characteristic of these sections is the ability to be separately locked. This locking is feasible for each combination of user roles (project admin, domain expert and ontology designer; see Section 2.2). If a segment is locked, users holding a disabled role(s) are no longer able to edit or revert this segment (see Fig. 4.7). Project admins nevertheless are able to edit and revert segments as they can still change the permissions on the segment and unlock project admins themselves. Locking of project admins first seems to be preposterous or contradictory but it is implemented because of two reasons. First, special wiki articles contain information extracted from the formal ontology. This data is enclosed in segments because no user should modify them and changes will be overwritten by another article load anyhow (see Section 4.3). The second reason is to prevent segment modifications even for project admins, if a segment

[12] `http://projects.jkraemer.net/acts_as_ferret/`

[13] `http://ferret.davebalmain.com/trac/wiki`

[14] `http://lucene.apache.org/java/docs/`

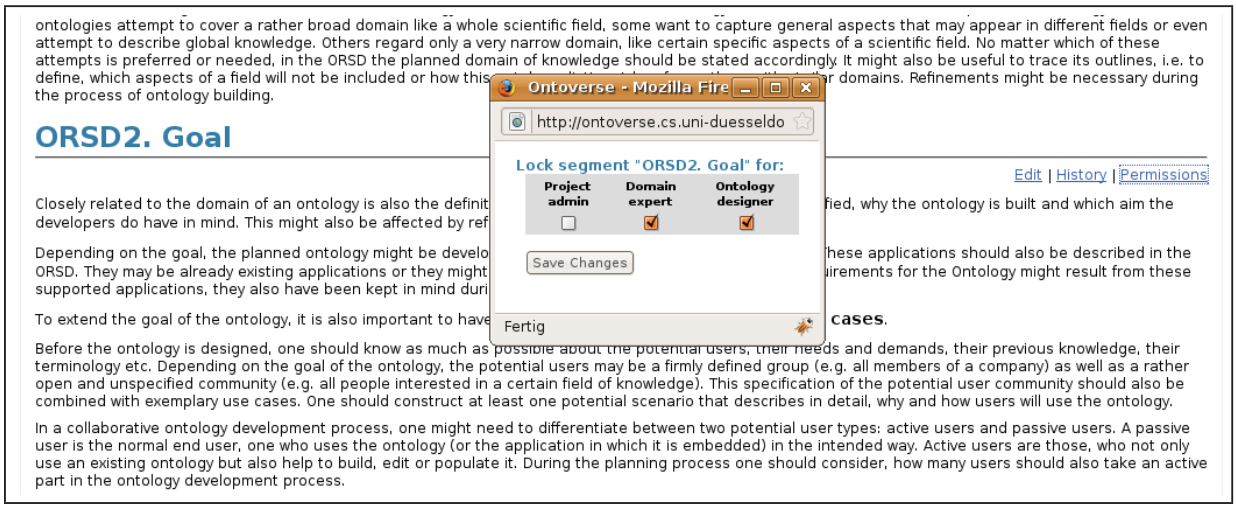

Figure 4.6: Lock dialogue of an ORSD segment. After clicking on “Permissions” a popup window provides the opportunity to choose the roles, for which this segment should be locked. On the right side of the image one can still see the segment menu: “Edit”, “History” and “Permissions”.

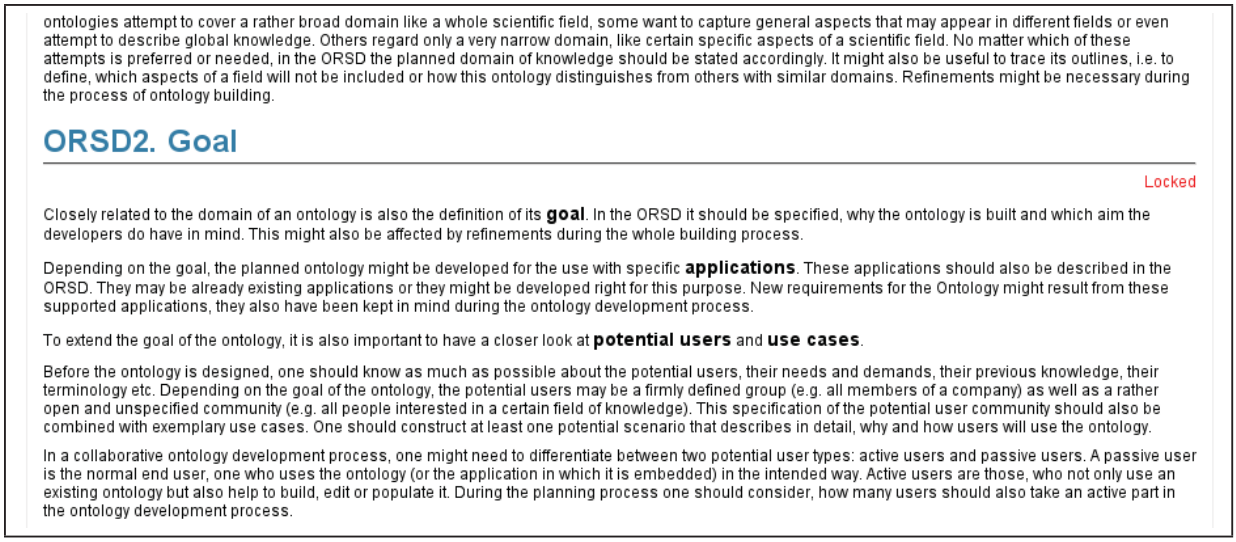

Figure 4.7: Locked segment. The screenshot shows the locked wiki segment after locking illustrated in Figure 4.6. The segment menu is now replaced by a red “Locked” label.

should not be changed anymore. If a project admin wants to modify a locked segment anyway, this should first be discussed among the project members.

The initial idea for segments origined in the realization strategy for the collaborative creation of the ORSD. ORSD sections like motivation, domain and goal should be defined before other ontology engineering processes are started. These sections should not be modified afterwards because the structure of the ontology and the modeling process highly depend on this view on the ontology. Changing these basic properties of an ontology might effect the remodelling of the whole ontology. Single ORSD sections are prefilled as segments in the ORSD article, so

a project admin can decide when these sections are finished and lock them to start the next engineering phases.

A new segment can easily be inserted by clicking on the "New Segment" link of the editor (see Fig. 4.4) or by typing:

```
\segment(Segment title)
   Here is the text of the segment.
\end
```

4.2.1 Implementation

In the backend, a segment is realized in the same way as a wiki article using the ACTS_AS_VERSIONED plugin. Therefore the database comprises a `segments` and a `segment_versions` table. Though, segments are not assigned to projects but to articles, into which they are integrated (see Fig. 4.3).

The segment code block extends TEXTILE. The method in the RUBY article model parses these blocks beginning with `\segment` and ending with `\end`. The segment title is between the parentheses (see example) and it is made sure that each segment has a unique title within one article. The segment is then stored in the `segments` table by inserting the extracted title, the body from within the segment tags and the ID of the current article. The version plugin automatically keeps track of this modification in the `segment_versions` table. Afterwards, the code block is substituted by a segment placeholder which contains the segment id and its current version. The version information facilitates the revision of single segments in an article. As soon as the entire article with all its segments is successfully stored, the `changes` table is filled with the modification and optionally with the user comment. These placeholders are markuped by the RAILS helper method `to_html`. On the basis of the segment id and the version, the accurate segment title and body are fetched from the `segments` table and formatted in html as shown in Figures 4.6 and 4.7.

The locking mechanism utilizes the ONTOVERSE system's authorization solution which is realized by the AUTHORIZATION plugin. This provides the `permit` method for checking authorization in a simple way. The plugin uses the `roles` database table (see Fig. 4.3), which stores the role name, a description, the authorizable type and the authorizable id. The authorizable type defines the RUBY class (model) to which the role is applied. The authorizable id sets specific objects of the authorized model. In the ONTOVERSE wiki the project model and the project id configure a user role. The `roles_users` table stores the mapping of roles to users. Those project-related user roles that have granted access to manipulate a segment are saved in the `roles_segments` table.

4.3 Connection to the Formal Ontology

Ontology engineering is not practicable without evaluating the formal ontology and thereupon refining and extending it. The realization of user roles in the ONTOVERSE system (see Sec-

tion 2.2) allows domain experts not having to be familiar with formal ontology features. To assure the verification by the domain experts anyhow, I implemented a connection between wiki and formal ontology that displays the ontology structure in a more convenient and familiar way.

4.3.1 Frontend

The connection to the formal ontology is completely integrated into the wiki structure. That means the information of the ontology is provided in wiki syntax within special wiki pages. By project creation on the ONTOVERSE platform two wiki articles, a list of ontology classes and of ontology individuals are automatically linked on the main page of this new project's wiki (see Fig. 4.2). By following the link to the special wiki page "Ontology Classes", a connection to the tuple space server is established by the SWATClient (see Section 2.7) and the wiki page is filled with a locked segment listing links to special wiki pages for all ontology classes (see Fig. 4.8). Each class has its own wiki page. Figure 4.9 exemplifies such an article. Titles of those articles are prefixed with "Class:" followed by the local name of the ontology class. The article body is filled with the information from the ontology each time the wiki page is loaded. A segment is created containing links to wiki articles of all direct subclasses of the current class. Similarly there exist segments listing links to direct superclasses and instances. These segments are locked automatically by the system for each project member and are overwritten each time the article is reloaded. The implementation of this reloading was indispensable because there was no possibility to get information about modification times from the tuple space server. That is why no time specific queries on the data were possible and no update of the already fetched information was realized. More than these automatically filled segments, the user is able to add content to the article. Here new super-, subclasses and instances can be proposed or new descriptions added. The modeling of the class might be discussed and ontology designers are able to ask questions about the class. That means an evaluation and refinement is facilitated on these articles. Additionally, these ontology-connected wiki articles provide a direct link to the formal editor. By clicking on this "Show in editor" link, either a new browser window appears loading the editor with the just viewed class in the focus, or the already opened editor changes its focus to this class. The formal editor also has links to the wiki and *vice versa*.

Similar wiki pages are created for each individual in the ontology. Links to these articles are all embraced in the special wiki page "Ontology Individuals" and are prefixed by "Individual:" (see Fig. 4.10). The segments of these articles differ slightly from those contained in the class' pages. They comprise information about the type of the individual and list relations to other individuals.

4.3.2 Implementation

In the backend of this connection to the formal ontology are SQLSpaces (see Section 2.7), which are provided by the group of Prof. Hoppe of the University of Duisburg-Essen. Ontologies are composed of triples (subject, predicate, object), which in ONTOVERSE are stored as tuples in a

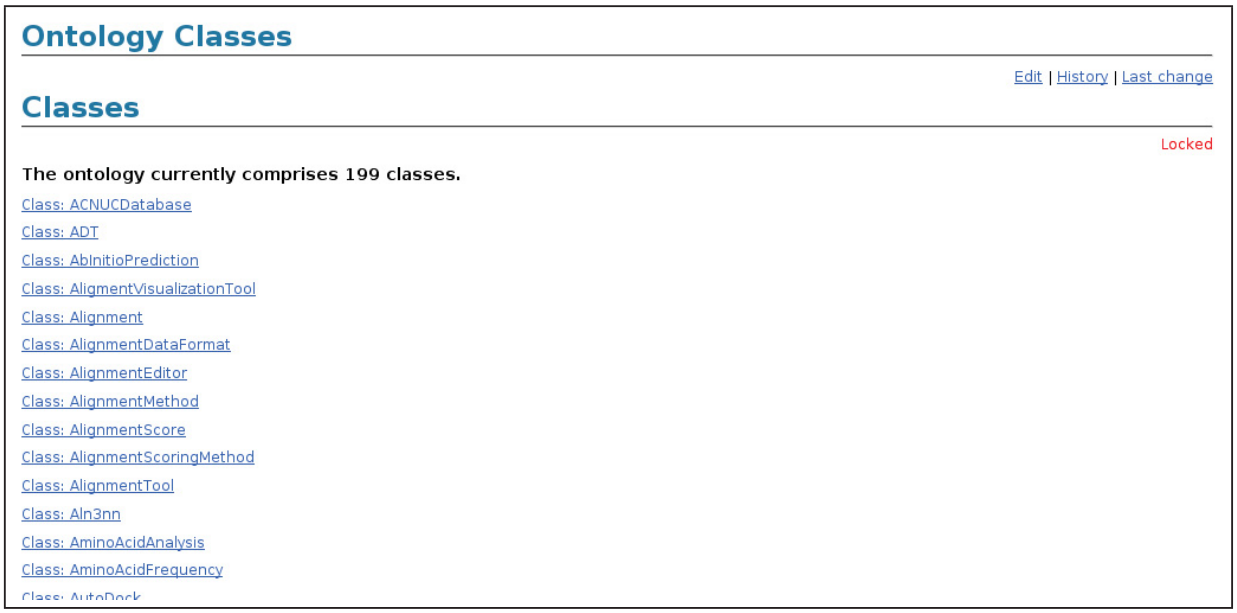

Figure 4.8: Wiki article "Ontology Classes". Screenshot of the special wiki article "Ontology Classes". It consists of a link list to all classes the ontology currently comprehends.

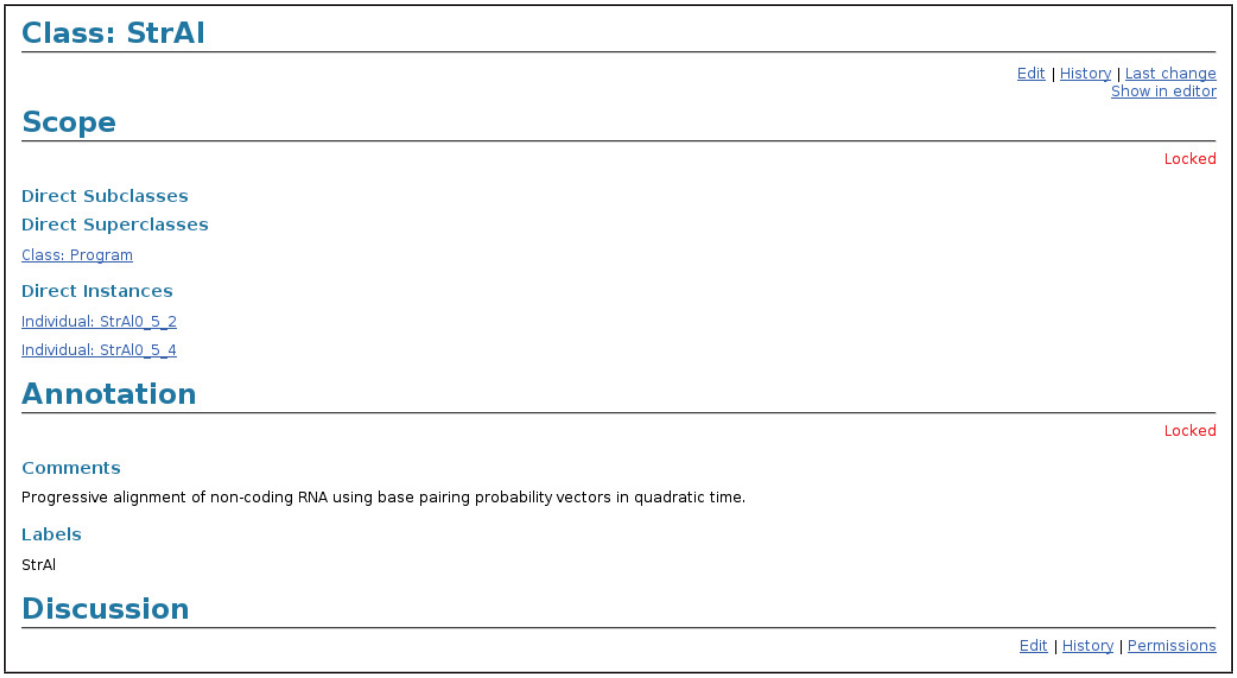

Figure 4.9: Wiki article of class `StrAl`. The screenshot shows the automatically filled "Class: StrAl" wiki page.

tuple space. In addition to subject, predicate and object, the creation and modification time is attached to each tuple. Each ontology project has its own tuple space. Stefan Weinbrenner from the University of Duisburg-Essen provided a RUBY-based SWATClient for the communication

Figure 4.10: Wiki article of instance `StrAl0.5.4`. The screenshot exemplifies an automatically filled instance article in the ONTOVERSE wiki.

between the JAVA-based SQLSpaces and the RAILS platform. This web service was utilized for the implementation of the connection between wiki and ontology.

To unambiguously identify the tuple space belonging to a certain ontology project in the ONTOVERSE system, the namespace of the ontology is used as identifier of the ontology space. So by loading an ontology-linked wiki article, the namespace of the known, article-associated project is loaded from the database and the connection to the identically labeled tuple space is established. Queries can then associatively be formulated in templates. All tuples in the ontology spaces have eight fields. The first four, ID, creation time, modification time and expiration, are not searchable and contain information about the tuple. The others are of type `SQLType::String` and specify the type of the tuple, which in ontology spaces always is "fact" because each tuple represents a fact of the ontology. The last three fields specify the URI of the RDF triple's subject, predicate and object of that fact.

Figure 4.11 shows a code snippet that illustrates the query of ontology data associated with the class `StrAl`. Line one creates a new instance of the RUBY client. In the next line the tuple space is set, from which the information is required, by providing the namespace of the according ontology. The example uses the namespace of BIO2ME, `http://www.ontoverse.org/BIO2Me.owl#`. In lines three to seven the template is constructed by creating a new instance of the RUBY class `Tuple`. As described above, the tuple has to be composed of four fields. These fields are all purported to be of type `SQLType::String`. All tuples in the specified space comprise "fact" as value of the first field, so no further specification is made. The example shows the information search for the

```
1 $client = OntoverseClient.new()
2 $space = $client.set_space("http://www.ontoverse.org/BIO2Me.owl#")
3 template = Tuple.new([Field.new(SQLType::String),
4                       Field.new(SQLType::String,
5                                 "http://www.ontoverse.org/BIO2Me.owl#StrAl"),
6                       Field.new(SQLType::String),
7                       Field.new(SQLType::String)])
8 @tuples = $space.readAll(template)
9 $client.close
```

Figure 4.11: Source code example. The source code exemplifies the usage of the SWATClient web service. One can see the gaining of information about the class `StrAl` of the BIO2ME project for rendering the article "Class: StrAl" (see Fig. 4.9).

---

wiki article "Class: StrAl" (see Fig. 4.9) that is why the second field is forced to be of value "http://www.ontoverse.org/BIO2Me.owl#StrAl". The predicate and object fields are variable and thus not further specified. Line eight shows the search command `readAll`, which passes all tuples that satisfy the template to the array `@tuples`. After getting the resulting tuples, the connection to the client has to be closed (line nine).

The resulting array `@tuples` comprises arrays of tuples with exactly four indices each. This can then be processed. The tuple:

```
("fact", "http://www.ontoverse.org/BIO2Me.owl#StrAl",
 "http://www.w3.org/2000/01/rdf-schema#subClassOf",
 "http://www.ontoverse.org/BIO2Me.owl#Program")
```

for example defines the subclass relationship between class `StrAl` and its superclass `Program`. In this manner all relevant data for the wiki article are extracted.

The coordination between the JAVA-based editor and the RUBY-based wiki is managed with the aid of the session space. For this purpose the wiki and the editor get a callback to a special tuple that informs about the ontology element that is to be focused in the editor or shown in the wiki, respectively.

4.4 Discussion & Conclusions

Current ontology editors barely support the acquisition and informal structuring of ontology relevant knowledge. Moreover, depending on the domain, motivation and application, the ontology has to be built collaboratively. Only this will guarantee a common view and shared definitions of terms and their interconnections. A domain like the BIO2ME domain for example needs a group of engineers to cover the necessary amount of knowledge in bioinformatics and biology. This makes it very difficult and unpractical to create documents like the ORSD and conceptualization lists in a simple document and sharing them via file transfer. The utilization of a wiki is

obvious because it is already established, not only in the internet, to serve exactly this purpose: the support of a collaborative work on documents to share information. That means users are familiar with the concept of wikis, consequently the inhibition level is reduced and the training period is shortened. Changes are directly saved in the current document version and have not to be committed into a repository, so other project members directly can see them. Additionally, it is more useful to work on the same document and to save all documents altogether, accessible for all project members. Modifications can easily be retraced and reverted within the wiki. Most of these functions are offered by revision control systems either, but they are by far less intuitive to handle.

MEDIAWIKI was used for the BIO2ME design phase and proved itself as very helpful. As a consequence, a self-written wiki was integrated into the architecture of the ONTOVERSE system. A group of students tested the common wiki functions of the ONTOVERSE wiki for one semester during a course. They evaluated it to be as good as the MEDIAWIKI from this perspective.

ONTOEDIT (Sure *et al.*, 2002) is a collaborative ontology engineering environment. Its ONTOKICK plugin undertakes the support of users in the creation of requirement specification documents and the extraction of relevant structures for a semi-formal ontology. The plugin also promises the management of competency questions. Another ONTOEDIT plugin, MIND2ONTO, is supposed to integrate brainstorming processes about the semi-formal description of terms and their interconnections. The editor could not be tested, because it was object of a spin-off from the university of Karlsruhe and is now marketed by the ONTOPRISE GMBH[15] as ONTOSTUDIO, which is not freely available. However, the idea of ONTOKICK and the described plugins sounds plausible and its principle is similar to the ONTOVERSE wiki. Though, the popularity and acceptance of wikis provide shorter training periods.

4.4.1 Special ONTOVERSE Wiki Features

To guide new ontology engineers, the ONTOVERSE wiki gives starting points for ontology building. Users are adviced to start an ontology project by the creation of a requirement specification (ORSD). They can find an ORSD template in the project wiki linked in the project's wiki main page. This template is automatically prefilled with an advisable segmentation and descriptions to each section explaining its possible content and aim. This feature of the ONTOVERSE wiki already effected positive feedback from users. These users also wished for more support in the conceptualization phase of ontology engineering. However, in discussion with a colleague from the information sciences, we decided not to insert any guidelines, because this step strongly depends on the domain of the planned ontology and the engineers' preferences. During the conceptualization of BIO2ME I also tried all kinds of possibilities like lists, tables and illustrations. Those utilities were selected that were most helpful for a special task and that were appreciated by the other domain experts, too. The ONTOVERSE FAQs nevertheless include some suggestions, which can be tryed out by the users.

[15] `http://www.ontoprise.de/`

Segments

Segments provide a locking mechanism of text sections in a wiki page, which is an additional advantage in the ONTOVERSE wiki. This is useful to save own knowledge statements in a discussion and to mark a section as accepted by the majority. Also already modeled knowledge can be locked to advise users of that this should not be modified except one wants to remodel it. In the formal ontology-connected wiki articles, locking protects the user to edit segments that are filled by the system and in which modifications would get lost. Additionally, some ORSD sections should not be modified inconsiderately, because this might change the domain and the modeling of the ontology. The segment locking mechanism effects reconsiderations. Locking is in fact contrary to the common wiki idea that is most important for an open community, the free access to edit texts. In the ONTOVERSE wiki open community advantages are not demanded. Only selected experts in a knowledge domain should have access. Furthermore, project administrators are able to modify access rights anytime. The segment locking mechanism is optional, which has not to be used in a project.

Connection to the Formal Ontology

Another main feature of the ONTOVERSE wiki is the connection to the formal ontology. Special wiki articles are created that contain ontological data in a compact way, facilitating verification of the modeling for domain experts. These ontology-connected wiki articles include information about superclasses, subclasses of ontology classes, "type of" relations, object property related individuals and datatype property related values for ontology individuals. These articles provide a good overview and enable discussions about the modeling on the very spot. Direct linking to ontology concepts in the formal editor and *vice versa* in particular facilitates the collaboration of domain experts and ontology designers.

An additional section in these special ontology-connected articles is still under development. This integrates information about the concept's history. It will contain modifications of the class or individual, respectively, such as renaming and deleting operations. This will enable retracing of ontology changes. For this implementation the session space of the tuple spaces is read out to gain access to all operations on the ontology data. By now it is possible to fetch all modifications of a resource but the RUBY interface to the tuple space server so far does not support the query after creation times of the tuple in the session space nor the simple return of this time. The tuple space already includes the information to create this history section and to update those wiki articles, so that they have not to be rebuilt each time the page is loaded, but the RUBY SWATClient has to be extended by the group of the University of Duisburg-Essen.

These special wiki pages of classes and individuals are interlinked. Nonetheless, the ONTOVERSE wiki is not a semantic wiki like the ONTOWIKI[16]. Semantic wikis include different kinds of internal links, which add meaning to relations between two wiki articles. In ONTOVERSE only the two traditional linkings exist: internal and external links. Though, in the

[16] `http://ontowiki.net/Projects/OntoWiki`

special ontology-connected articles, users are able to distinguish relations to other ontology-connected articles, because the linking lists to other class or individual articles are titled with respective relationship. This relation type is automatically extracted from the formal ontology. Thus, users are able to differentiate between wiki linkings in ONTOVERSE, but the computer is not. In context of the intention of the wiki, the support of users in the protoontological phase, the interpretability for the computer is not required. Semantic wikis enable the automatical extraction of ontologies out of the wiki, but in my opinion in the current research status, the complexity and hence the expressiveness of ontologies might suffer from this. So this approach is only feasible for some domains.

Ontology evaluation by domain experts further can be facilitated by providing simple graphs created out of the ontology. ONTOVERSE comprises some ontology visualizations in the formal editor implemented by the group of Prof. Ziegler from the University of Duisburg-Essen. But these are not suitable for this purpose as they are too complex for domain experts and can hardly be incorporated in the wiki structure. One solution would be the generation of a simple concept graph stored in an image file, which then could easily be integrated via the wiki syntax. The image serves as illustration of the current ontology state and should not be modified directly. This extension is yet not being tackled in ONTOVERSE.

4.4.2 Utilized Plugins

The utilization of RUBY ON RAILS plugins notably simplified the implementation of the wiki. These plugins fully exploit features of RUBY ON RAILS and thus are especially effective. Without their employment the implementation of an own wiki would have been too time consuming. Though with the aid of these helpers it was possible to build a wiki especially adjusted to perfectly fitting the ONTOVERSE platform. It would have been more complicated by the use of available wiki frameworks.

The table `changes`, which is relevant for the “Recent Changes” listing of wiki, to some extent contains information that is redundant to the `article_versions` table. The table `article_versions` is needed by the plugin `acts_as_versioned`, so additional version information like modification mode and comment is included into a new table. This table meets the requirements for an unobstructed implementation of the modifications listing.

For the BIO2ME application implementation the framework DEEP SEMANTICS was chosen instead of the tuple space solution. Section 6.5 discusses the differences between their usages.

5

Machine-supported Ontology Extension by Tagging

Ontology extension is a crucial point in ontology engineering (see Section 2.1.4). In the case of BIO2ME, a wide knowledge field is covered that has to be considered and kept up-to-date. Furthermore, it turned out to be difficult to recruit domain experts for ontology extension, since the ontology has not been published yet and its benefit was rather unknown for external experts (see Section 3.2.1). As one consequence, I implemented an automatic tagging mechanism for publications into the ONTOVERSE system, which uses the classes', individuals' and properties' labels of the ontology.

5.1 Idea

The underlying idea of ontology extension by publication tagging is that articles that include ontology relevant notions, the very ontology labels, also might include information about associated concepts, which is not modeled in the ontology yet. The ontology labels form the keywords, because the local names of ontology classes, individuals and properties are subject to restrictions (e. g. no whitespace characters are allowed) and with the aid of labels any number of synonyms and different spellings can be added. That is one reason why BIO2ME was extended by at least one label for each class, individual and property.

Fig. 5.1 schematizes the procedure of tagging-based ontology extension. At first, a collection of relevant publications is selected from a scientific publication database. Additionally, ONTOVERSE provides the possibility of uploading own texts into its publication database PUBDB (see Section 2.2). These scientific articles can be automatically scanned for ontology relevant terms. On the basis of these tagged papers, domain experts can have a close look to those publications, extract new information and insert them manually into the ontology.

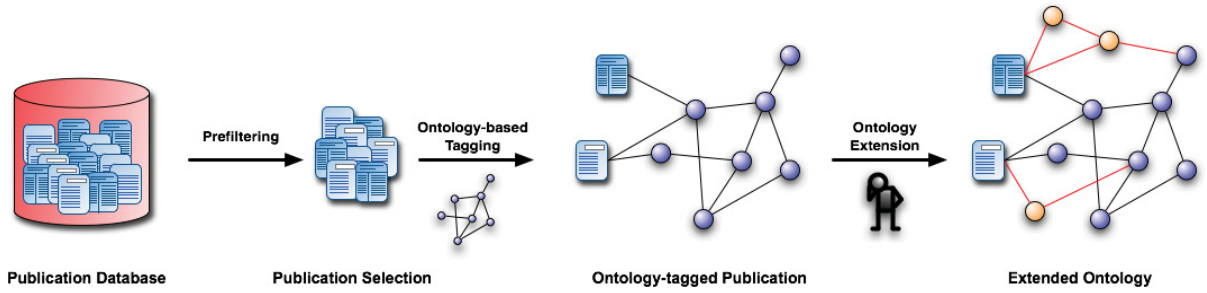

Figure 5.1: Machine-supported ontology extension. The scheme illustrates the ontology extension on the basis of ontology-based tagging.

Table 5.1: Database Table: `keywords`. Listed are the database columns with a short description.

Field	Description
`ID`	primary key
`NAME`	keyword name
`DESCRIPTION`	additional information on the keyword
`SOURCE`	One of:
	`PubMed`, if the keyword is extracted from PUBMED
	`free`, if the keyword is freely assigned by a user
	`ontology: <ontology_name>`, if the keyword is fetched from an ontology.
`CONCEPT`	URI of the concept, if the keyword was extracted from an ontology.

5.2 Process of Tagging

Before the actual tagging process takes place the labels of ontology classes, individuals and properties have to be extracted from the formal ontology, which is persistent in the BIO2ME project space of the SQLSpaces (see Section 2.7) on the ONTOVERSE platform. Therefore the RUBY SWATClient methods to list all ontology concepts, individuals and properties as well as methods to get their labels were utilized. These accessed labels are saved in the database `keywords` table declared with BIO2ME as their source and their actual concept in the concept field. See Table 5.1 for the table description.

On the basis of these keywords the actual tagging process can be started either for one publication or for a selection of them selected in the project's publication collection in ONTOVERSE. The tagging mechanism utilizes regular expressions and is processed as follows (see Fig. 5.2):

In each publication (loop: lines 1 to 15) each ontology label (loop: lines 2 to 14) is searched. The keywords are selected from the database table `keywords` by searching after the "source" field. In the process the keywords are distinguished by whether they contain whitespace characters (`\s`) or not (if-clause lines 3 to 13). If a keyword includes at least one whitespace character (lines 4 to 7) an exact matching of the keyword in the title or abstract is required (lines 4 and 5). The slashes restrict the pattern which is searched for. The `i` behind these slashes causes a case insensitive search. `#{variable}` is special RUBY syntax and means that the content of *`variable`* is included into the string or in this case into the regular expression. Ontology labels

```
 1 papers.each do |paper|
 2   keywords.each do |keyword|
 3     if keyword =~ /\s/
 4       if paper[title] =~ /#{Regexp.escape(keyword)}/i
 5       || paper[abstract] =~ /#{Regexp.escape(keyword)}/i
 6         hits[paper[id]] << keyword
 7       end
 8     else
 9       if (paper[title] =~ /\b#{Regexp.escape(keyword)}[s\b\s]/i)
10       || paper[abstract] =~ /\b#{Regexp.escape(keyword)}[s\b\s]/i)
11         hits[paper[id]] << keyword
12       end
13     end
14   end
15 end
```

Figure 5.2: Exact Matching. The source code is an extract from the RUBY code implemented in ONTOVERSE.

---

are in natural language, so the method `escape()` for the RUBY class of regular expressions, `Regexp`, is utilized, which escapes characters with special meanings in regular expressions. The match can emerge in any textual environment. Matching keywords are saved according to the publication in which they were found (line 6).

If there is no whitespace character in the keyword string (lines 9 to 12), a more restrict regular expression is used (lines 9 and 10). This enables the search of whole words only. The expression begins with metacharacter `\b` matching word boundaries. Word boundaries appear:

1. before the first and the last character in a string, if the first and the last character are word characters,
2. between a word character and a non-word character following right after the word character and
3. between a non-word character and a word character following right after the non-word character.

A to Z, a to z, 0 to 9 and _ are word characters. After the keyword the regular expression allows the single character `s`, a word boundary `\b` or a whitespace character `\s` to match singular and plural occurrences.

5.3 Ontology Extension – A Case Study

To analyze the usability of this tagging mechanism with regard to ontology extension, a collection of relevant publications in bioinformatics was tagged with BIO2ME labels. First, an initial

ontology was used for tagging, then the most tagged publications were selected, analyzed and the extracted information was incorporated into BIO2ME. To demonstrate ontology extension by tagging in more detail the most tagged publication was examined separately.

5.3.1 Publication Selection

The publication tagging regarding ontology extension was tested with the aid of a collection of 211 publications. These papers introduce tools the ontology already contained. Moreover, additional papers were extracted from the biomedical publication database PUBMED by keyword search, e. g. for "alignment", "bioinformatics" and "modeling". All of these publications deal with bioinformatics tools, methods and studies. After inserting them into the BIO2ME's publication collection on the ONTOVERSE system, they were tagged with BIO2ME labels, extracted from the formal ontology before.

5.3.2 Initial Tagging

In a first tagging study the ontology offered 523 unique labels. Labels might appear repeatedly in the ontology, but duplicates are not considered in this analysis. 109 (approximately 21 %) of these keywords were found in 201 publications. This means ten paper were not tagged at all.

Fig. 5.3 shows the frequency distribution of publications that are tagged with a certain number of keywords. Ten publications were not tagged at all and the maximum number of keywords which are found in publications is 17. The numbers of papers that include two to six keywords are the largest. From twelve keywords per publication on, the number of papers decrease and is under the arithmetic mean of 5.70 (including not tagged papers). Publications with more than ten keywords are listed in Table 5.2.

Fig. 5.4 plots the numbers of keywords that occur in a certain number of papers. 414 keywords are not found in any publication. This information was omitted in the histogram to shorten the y-axis. 33 % of the keywords that are found in a publication at all, appear in only one paper. Keywords that appear in more that 14 publications occur infrequently. Table 5.3 lists those ontology labels that are found in at least ten publications. The keyword "sequence" is by far the most occurring as it appears in approximately 64 % of the tagged publications. The other labels listed mainly are terms that deal with alignment programs and RNA structure prediction. The tag "Feature" is artificially found in 36 publications, as it actually labels a chemical file format. There is no correct hit of this keyword in the papers. This label was renamed to "Feature format" afterwards.

The first column of Table 5.4 itemizes the initial number of ontology labels, classes, properties and individuals. Additionally, it provides information about the crosslinking between classes and instances of the ontology, respectively, by listing the numbers of different types of relationships (Instance of relations, Object Property assignments and Datatype Property assignments).

The quality of the revealed tags can be shown by comparison of the assigned BIO2ME labels with PUBMED keywords. PUBMED tags publications with the MESH thesaurus (see Section 2.9). These keywords yet are mostly too general for finding BIO2ME relevant papers. The

Table 5.2: Most tagged Publications of the Initial Tagging Process. Listed are all ontology tagged publications in which more than ten ontology labels were found.

Publication	#Keywords
SARSA: a web tool for structural alignment of RNA using a structural alphabet. (Chang *et al.*, 2008b)	17
R-Coffee: a web server for accurately aligning noncoding RNA sequences. (Moretti *et al.*, 2008)	16
ProfDistS: (Profile-) Distance based phylogeny on sequence - structure alignments. (Wolf *et al.*, 2008)	15
STRAL: progressive alignment of non-coding RNA using base pairing probability vectors in quadratic time. (Dalli *et al.*, 2006)	15
MARNA: multiple alignment and consensus structure prediction of RNAs based on sequence structure comparisons. (Siebert & Backofen, 2005)	14
Multiple sequence alignments of partially coding nucleic acid sequences. (Stocsits *et al.*, 2005)	14
TOPS++FATCAT: fast flexible structural alignment using constraints derived from TOPS+ Strings Model. (Veeramalai *et al.*, 2008)	14
Accelerated probabilistic inference of RNA structure evolution. (Holmes, 2005)	13
Alignment of RNA base pairing probability matrices. (Hofacker *et al.*, 2004)	13
An enhanced RNA alignment benchmark for sequence alignment programs. (Wilm *et al.*, 2006)	13
Progressive multiple sequence alignments from triplets. (Kruspe & Stadler, 2007)	13
R-Coffee: a method for multiple alignment of non-coding RNA. (Wilm *et al.*, 2008)	13
BAliBASE 3.0: latest developments of the multiple sequence alignment benchmark. (Thompson *et al.*, 2005)	12
DNA reference alignment benchmarks based on tertiary structure of encoded proteins. (Carroll *et al.*, 2007)	12
PREDICT-2ND: a tool for generalized protein local structure prediction. (Katzman *et al.*, 2008)	12
RADAR: a web server for RNA data analysis and research. (Khaladkar *et al.*, 2007)	12
Sigma: multiple alignment of weakly-conserved non-coding DNA sequence. (Siddharthan, 2006)	12
Accurate multiple sequence-structure alignment of RNA sequences using combinatorial optimization. (Bauer *et al.*, 2007)	11
Colorstock, SScolor, Ratón: RNA alignment visualization tools. (Bendaña & Holmes, 2008)	11
INFO-RNA–a fast approach to inverse RNA folding. (Busch & Backofen, 2006)	11
MACSIMS: multiple alignment of complete sequences information management system. (Thompson *et al.*, 2006)	11
MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. (Katoh *et al.*, 2002)	11
MASTR: multiple alignment and structure prediction of non-coding RNAs using simulated annealing. (Lindgreen *et al.*, 2007)	11
PRALINE: a multiple sequence alignment toolbox that integrates homology-extended and secondary structure information. (Simossis & Heringa, 2005)	11
PROMALS: towards accurate multiple sequence alignments of distantly related proteins. (Pei & Grishin, 2007)	11
RNALogo: a new approach to display structural RNA alignment. (Chang *et al.*, 2008a)	11
Semiautomated improvement of RNA alignments. (Andersen *et al.*, 2007)	11
SimulFold: simultaneously inferring RNA structures including pseudoknots, alignments, and trees using a Bayesian MCMC framework. (Meyer & Miklós, 2007)	11

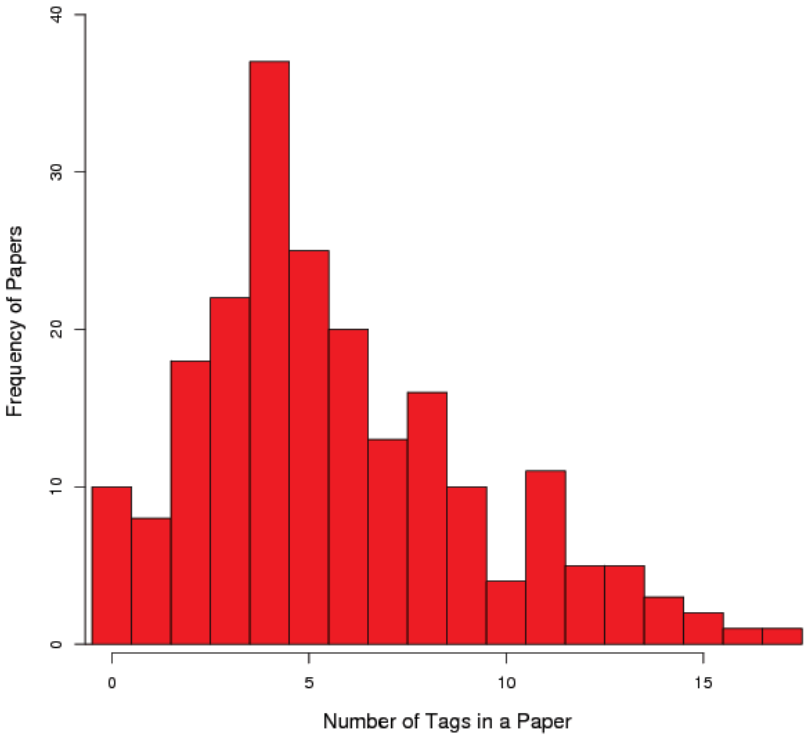

Figure 5.3: Publications Histogram. This diagram displays the frequency of publications, in which a certain number of keywords appear.

Table 5.3: Most occurring Keywords. Ontology keywords which are found in at least ten publications.

Keyword	# Papers	Keyword	# Papers
sequence	129	molecule	25
data	86	structure prediction	19
tool	81	DNA	17
alignment	75	alignment method	14
protein	73	benchmark	14
RNA	58	computational method	14
program	51	package	14
database	50	tree	13
sequence alignment	43	genomics	12
secondary structure	40	RNA molecule	12
Feature	36	progressive	11
RNA sequence	27	DNA sequence	10
multiple alignment	25	structure alignment	10

comparison is exemplified for the most tagged publication for that PUBMED keywords were available, “STRAL: progressive alignment of non-coding RNA using base pairing probability vectors in quadratic time.” (Dalli *et al.*, 2006), came with the following PUBMED tags:

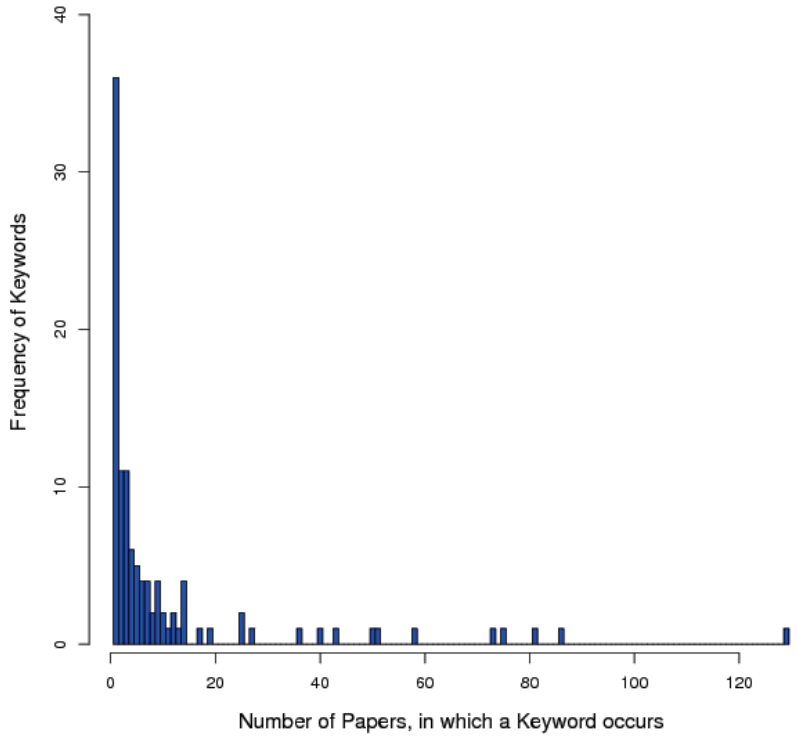

Figure 5.4: Keywords Histogram. The figure shows the frequency of keywords that are found in a certain number of publications.

Table 5.4: Ontology Data during the Extension Process. The table shows the numbers of ontology elements in five different ontology versions described in the text. All versions from left to right are build in succession. Note, that Extension 1 (**Ext. 1**) is based on 27 publications and **Ext. 2** to **Ext. 4** include diverse knowledge sources of only one paper.

	Initial	**Ext. 1**	**Ext. 2**	**Ext. 3**	**Ext. 4**
Labels	523	682	698	703	703
Classes	158	206	209	209	209
Object Properties	19	38	42	42	42
Datatype Properties	23	29	29	29	29
Individuals	254	329	334	339	339
Instance of relations	326	445	454	458	458
Object Property assignments	823	1056	1081	1093	1095
Datatype Property assignments	298	374	380	383	390

"Algorithms", "Base Pairing", "Base Sequence", "Computational Biology", "Models, Statistical", "Molecular Sequence Data", "Nucleic Acid Conformation", "Phylogeny", "Probability", "RNA", "RNA, Untranslated", "Sequence Alignment", "Software", "Time".

Tagging with BIO2ME labels in contrast delivered the following keywords before ontology extension:

> "alignment", "base pairing probability vector", "pairwise alignment", "program", "progressive", "RNA", "RNA sequence", "RNA structure prediction", "Sankoff algorithm", "sequence", "sequence alignment", "sequence-structure alignment", "StrAl", "structure alignment", "structure prediction".

The PUBMED keywords "Algorithms", "Computational Biology", "Software" and "Time" are too general for BIO2ME, since the whole ontology deals with computational biology and algorithms. These keywords only yield a rough classification, which can be used to fetch papers from PUBMED. It is rather a prefilter for ontology-based tagging. The keywords "RNA" and "sequence alignment" are located in both tag collections. However, the PUBMED tags do not provide the information that STRAL is a program, which takes structure information into account for the computation of the sequence alignment. Moreover, the BIO2ME tags join the two PUBMED keywords "Base Pairing" and "Probability" into the more precise "base pairing probability vector", because the published algorithm uses a certain kind of probability. Furthermore, the ontology-based tagging offers the name of the tool and the utilized computational method "sankoff algorithm" and the information that the program can be used for structure prediction.

The publication collection also subsumed papers about bioinformatics tools that were already modeled in the ontology. As expected all tools were found by name. Additionally found keywords in these titles and abstracts were indeed reasonable, too. Those publications revealed new relations, labels and instances as well. For example Holmes (2005) provides the local alignment method "Waterman-Eggert algorithm", which was not modeled in the ontology before. Thus `WatermanEggert` was inserted as new instance of `AlignmentMethod` and related to the described program STEMLOC through the `hasComputationalMethod` property. Further ontology extensions based on publication tagging are described in the next sections.

5.3.3 Extension 1: Analysis of the Most Tagged Abstracts

At first, 27 titles and abstracts from the 28 publications listed in Table 5.2 were analyzed. With the aid of the most tagged paper, Chang *et al.* (Chang *et al.*, 2008b) the analysis process is demonstrated in more detail in Section 5.3.4, that is why it is omitted as a start.

The information provided in the abstract, amongst others, depends on the journal, which published it, as they use different abstract templates. Generally, a publication's title and abstract comprise different information for the ontology. On the one hand the ontology labels can be analyzed and extented by synonyms. On the other hand new ontology concepts and properties can be extracted. The use of BIO2ME as knowledge base in a web application in combination with the artificial syntax of concept's local names stipulate that each ontology element at least has to possess one label. That is the reason why the extension with concepts also yields an addition of ontology labels. The last interesting information retrieved from a paper's title and abstract is the knowledge, in this study in particular the knowledge about bioinformatics tools. Extracting this knowledge results in inter- and intrarelations between ontology classes, individuals and literals. These relationships are formalized by using hierarchical and self-defined ontology properties. The following sections describe these divers extensions.

Synonyms

In going through the titles and abstracts a lot of notions were detected that were already modeled in the ontology, but has not been labeled by these phrases. Due to the freedom of the natural language, it is important to add synonyms of the concept as `rdfs:label` to enable improved tagging and search over the ontology. Such a term found in the titles and abstracts is for example "protein sequence", which is synonymous with the already modeled "amino acid sequence".

Ontology Elements

The analysis of the most tagged abstracts delivered 25 new tools for the ontology. Three of them were databases, which are specialized for benchmarks of alignment tools. The remaining 22 tools were programs and web tools that use new computational methods.

The analyzed publications in general provided no version information of the introduced tools. But the structure of BIO2ME requires the declaration of a program version as individual to enable the automatic comparison of different program versions. If no program version was available, an integer was appended to the programs' name to build the local name of the individual in the ontology. For example R-COFFEE has no published version number, so the instance of the class `RCoffee` is `RCoffee1`. If a new version gets published, the ontology can easily be extended.

Additionally to the detection of new tools, new data types, computational methods and other ontology relevant concepts can be found in the titles and abstracts. The analyses even inspired to remodel some parts of the ontology. An example is shown in Figure 5.5. The notions "nucleic acid sequence" and "NA sequence", respectively, adverted to a missing concept, the initial ontology comprehended the class `Sequence` with its instances `RNASequence` and `DNASequence`. Now it is manually extended by the new subclass `NASequence` with labels "NA sequence" and "nucleic acid sequence", modeled inbetween `Sequence` and the instances.

Information

As already mentioned, the analyses of the most tagged publications yielded several new bioinformatics tools. Ontology relevant information, which can be retrieved out of the titles and abstracts, is the name of the tool, the bioinformatics application and sometimes the biological relevance. Often there are also statements about input and output data. Tool publishing papers naturally include short descriptions of the tool, which can be used to add a characterization via a `rdfs:comment` attribute into the ontology. This information often can be retrieved even from the publication's title. Another interesting information provided by the most abstracts and even titles is the implemented computational method. This can be for example a mathematical model or a newly developed algorithm.

A really useful information is the declaration of the availabilty of the tool. Homepages of tools are often denoted, which provide the download URL (datatype property: `hasDownloadLocation`) or an available web frontend.

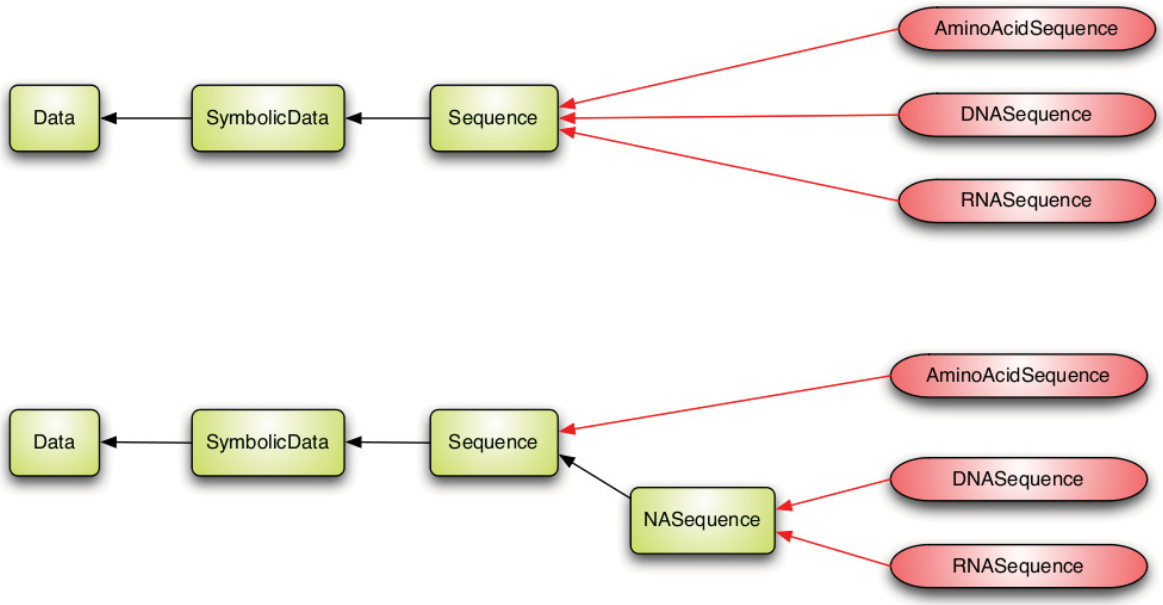

Figure 5.5: Insertion of `NASequence`. This illustration exemplifies the extension of the ontology supported by tagging. The upper schema displays the relevant ontology elements of the initial ontology. The bottom graph shows the current modeling of the ontology.

Sometimes more than one tool can be found in an abstract. For example Wilm *et al.* (2008) and Moretti *et al.* (2008), which publish the tool R-COFFEE, contain the notion "R-Coffee" and additionally the already in the ontology modeled alignment program T-COFFEE. In this case R-COFFEE is an extension of T-COFFEE. Moreover, the program MAFFT was found in there, which was compared with the new tool in the included benchmark. Thus R-COFFEE is modeled in the ontology with the `usesProgram` relation to T-COFFEE and with a `hasBenchmark` reference to the paper. However, most benchmarks are not considered in the abstract so this information cannot always be extracted simply by tagging.

Results

The extension after analyzing the titles and abstracts of 27 most tagged publications (not considering the most-tagged paper, Chang *et al.*, 2008b) results in an ontology with 682 labels (see Table 5.4, 2nd column). Also listed is the count of ontology labels, classes, properties, individuals and interrelations. The most numbers increased by a factor around 1.3; the number of object properties even doubles.

Section 5.3.4 provides an exemplified detailed description of what can be retrieved from a publication.

5.3.4 Analysis of Chang *et al.*, 2008b

To exemplify the process of the tagging-based ontology extension, the publication "SARSA: a web tool for structural alignment of RNA using a structural alphabet." (Chang *et al.*, 2008b)

SARSA: a web tool for structural alignment of RNA using a structural alphabet

SARSA is a web tool that can be used to align two or more RNA tertiary structures. The basic idea behind SARSA is that we use the vector quantization approach to derive a structural alphabet (SA) of 23 nucleotide conformations, via which we transform RNA 3D structures into 1D sequences of SA letters and then utilize classical sequence alignment methods to compare these 1D SA-encoded sequences and determine their structural similarities. In SARSA, we provide two RNA structural alignment tools, PARTS for pairwise alignment of RNA tertiary structures and MARTS for multiple alignment of RNA tertiary structures. Particularly in PARTS, we have implemented four kinds of pairwise alignments for a variety of practical applications: (i) global alignment for comparing whole structural similarity, (ii) semiglobal alignment for detecting structural motifs, (iii) local alignment for finding locally similar substructures and (iv) normalized local alignment for eliminating the mosaic effect of local alignment. Both tools in SARSA take as input RNA 3D structures in the PDB format and in their outputs provide graphical display that allows the user to visually view, rotate and enlarge the superposition of aligned RNA molecules. SARSA is available online at http://bioalgorithm.life.nctu.edu.tw/SARSA/.

Figure 5.6: Tagged Title and Abstract of Chang *et al.*, 2008b. Green highlighted text indicates matched BIO2ME keywords. Yellow text background denotes information that is relevant for the ontology.

with the highest number of keywords found (17 hits) was selected. It is also well suited for the detailed analysis, as it introduces a structure alignment tool and is therefore in the domain of bioinformatics tools I investigated in my Diplom thesis (Mainz, 2006b). Note that the quality and quantity of the extraction of information is highly depending on the knowledge of the researcher who investigates the publication.

The 17 keywords found in the paper's title and abstract are the following (in alphabetical order):

> alignment, alignment method, global alignment, local alignment, molecule, multiple alignment, normalized local alignment, pairwise alignment, PDB format, RNA, RNA molecule, semiglobal alignment, sequence, sequence alignment, structural alignment, tertiary structure, tool

Extension 2: Analysis of title and abstract

Figure 5.6 shows the title and abstract of Chang *et al.* (2008b), in which ontology labels (found tags) are highlighted (green). Information that was manually detected by brainpower and should be modeled in the ontology, is emphasized in yellow. As described in Section 5.3.3, title and abstract of a publication comprise different information that is relevant for the ontology. In the presented abstract few additional synonyms were found. "3D structure" was added for example as label of `TertiaryStructure`, which till then was labeled with "tertiary structure", and

"structural alignment tool" as synonym of "structure alignment tool". Title and abstract of this publication introduce three new bioinformatics tools, the web tool SARSA is already presented in the title, and two structural alignment tools, PARTS and MARTS, provided via SARSA. Classes for these tools were created in BIO2ME along with their instances that model the version of a tool. Since in the publication no version number is available, local names of these instances are composed of the tool's name suffixed by "1" for the first modeled version of the tool. The published computational method of these tools was not modeled in the ontology, thus it was inserted as new instance `VectorQuantificationApproach` with label "vector quantification approach".

In the title and abstract of Chang *et al.* (2008b) a lot of information about these three tools can be extracted. To model the relationship between SARSA and PARTS as well as between SARSA and MARTS, respectively, new object properties had to be inserted into BIO2ME. The object property `gainsAccessTo` with domain `WebServer` and range `Tool` was incorporated into BIO2ME, together with its subproperties `gainsAccessToProgram` (range: `Program`) and `gainsAccessToDatabase` (range: `Database`) and its inverse property `canBeAccessedVia`. Actually, the range of `gainsAccessTo` should be the union of `Database` and `Program`, but with regard to the BIO2ME application and the use of DEEP SEMANTICS, which currently does not handle unions, it was modeled in the way described. Besides the relation between the tools, many attributes of the tools are inclosed in the abstract. A description can be extracted, which is inserted in the ontology using `rdfs:comment`, and the URL of SARSA, which is modeled with the datatype property `isAvailableAt`. Additionally, PARTS and MARTS take RNA tertiary structures as input in the PDB format and put out structural RNA alignments. Differences between PARTS and MARTS are also provided in the abstract. Moreover, the bioinformatics application of the tools is clearly expressed. The newly inserted computational method could also be related to the tools with the self-defined property `usesComputationalMethod`.

After inserting the information of the publication's title and abstract, 15 new ontology labels were created in the ontology. Eight (bold) of them were found by tagging in the paper's title or abstract again:

> 2D structure, **3D structure**, can be accessed via, gains access to, gains access to program, gains access to database, **MARTS**, **PARTS**, protein 2D structure, protein 3D structure, **RNA 3D structure**, **RNA tertiary structure**, **SARSA**, **structural alignment tool**, **vector quantization approach**.

The seven additional labels, which were replenished in the ontology, derived either from similar keywords or newly inserted properties. The latter also get labels, however property labels always are hard to find in a paper. For example "3D structure" was found in the abstract and inserted into the ontology. Based on his knowledge about the domain, a knowledge engineer can then assign the keyword "2D structure" to the according ontology class `SecondaryStructure` for the sake of completeness. Furthermore, protein structures were labeled in the course of labeling RNA structures.

Extension 3: Full-text Analysis

After analyzing the paper's title and abstract, the whole paper was searched through. Out of the full text of this paper some ontology relevant information could be extracted. A lot of information was already provided in the abstract. New attributes for the tools SARSA, PARTS and MARTS are in particular scoring schemes that are utilized and the information that the programs were tested on the operating system Linux. The publication also contains benchmarks of the programs, being modeled by the `hasBenchmark` property. The computational method is described in more detail, which however cannot be fully mapped into the ontology in its current state. The full-text paper also provides detailed information about the input and output of the web tool. The publication for example includes the information that the tool JMOL in version 11.4 is utilized to present the resulting 3D structures.

Extension 4: Analysis of Web Sources

The publication contained the URL of the web tool SARSA, so information from these web sources was integrated in BIO2ME in a last extension study. Submit forms for the use of PARTS and MARTS as well as a help page are linked on the given web page. A more appropriate description for PARTS could be found there. The help page mostly conforms to the publication and an additional usage manual. Table 5.4 shows that only some new property assertions could be gained from these resources. Information like contact data and URLs of web interfaces for the programs could be integrated in the ontology.

5.3.5 Overall results

Table 5.4 lists the ontology changes for all described extension steps. The extensive increase in **Extension 1** becomes apparent. The analysis of title and abstract of the most tagged publication (**Extension 2**) also resulted in an addition of several ontology elements. These two text modules already hold so much ontology relevant information, that the full-text paper did not reveal a lot of new knowledge (**Extension 3**). The detailed description of the computational method for example cannot be completely modeled in the current state of the ontology. Even data format specifications were provided in the abstract, that is one reason why the information extraction of the web resources did not deliver many new ontology data (**Extension 4**).

Table 5.5 shows the most occurring keywords that were integrated in BIO2ME on the basis of all highly tagged publications in Table 5.2. The number of labels increased from 523 to 703 (see Table 5.4). Now 210 of them, instead of 109 in the initial tagging process, are found in 204 publications. Thus not every new ontology label was found in any paper. That means after the ontology extension, three publications that included no ontology label before, are tagged by the new labels. These three papers are Gevorgyan *et al.* (2008) and Valdivia-Granda (2008), which both include the new keyword "algorithm", and Alterovitz *et al.* (2008) which comprises the ontology labels "algorithm", "Java" and "open source".

Table 5.5: Occurrence of Ontology Labels after Ontology Extension by all most tagged titles and abstracts. The table lists new added ontology labels that at least occur in ten publications. Keyword "Feature" was renamed to "feature format" and delivers no artificial hits anymore.

Keyword	# Papers
algorithm	58
NA sequence	38
~~Feature~~	~~36~~
C	22
protein sequence	22
web server	20
free	19
is available at	19
sequence data	13
3D structure	13

Table 5.6 lists the most tagged publications containing more than 15 keywords. Now 49 papers include more than eleven labels. The list does not show any newly most tagged publications in the top 20, but the order in regard of the number of contained keywords has changed.

5.4 Discussion

In this chapter the newly implemented tagging mechanism was described and tested in regard to its usability for this purpose. Based on these studies, the idea turned out to be reasonable for ontology extension. Figure 5.6 illustrates the matching ontology labels (highlighted green) in the initially most tagged title and abstract. Ontology relevant information was detected by my brainpower and highlighted yellow in the text. The location of these colored areas shows that the environment of matching ontology labels indeed contains new, ontology relevant information. In the following sections the tagging approach and the results are dicussed in more detail.

5.4.1 Implementation of tagging

The implementation of tagging as regular expression approach could be proven as effectual in the depicted case study. The described tagging process is a case-insensitive, but otherwise exact matching of ontology labels in title and abstract of a publication. The implementation of a phonetic search is not reasonable, because the ontology labels should be found exactly and no typing errors have to be compensated as in a manual search. A scientific domain additionally does not include a lot of terms that have got many different spellings like surnames, which could rather be found by a phonetic search. As the ONTOVERSE platform includes a phonetic search function, I also tested the tagging based on a phonetic search. The results (not shown) were inferior to the performances of the integrated pattern matching approach.

The distinction of keywords with and without whitespace characters revealed much more specific and reasonable matches in publications than without any differentiation (data not shown).

Table 5.6: Most tagged Publications after tagging based ontology extension. Listed are all ontology tagged publications that include more than 15 ontology labels.

Publication	**#Keywords**
SARSA: a web tool for structural alignment of RNA using a structural alphabet. (Chang *et al.*, 2008b)	25
R-Coffee: a web server for accurately aligning noncoding RNA sequences. (Moretti *et al.*, 2008)	23
ProfDistS: (Profile-) Distance based phylogeny on sequence - structure alignments. (Wolf *et al.*, 2008)	22
Accelerated probabilistic inference of RNA structure evolution. (Holmes, 2005)	21
Alignment of RNA base pairing probability matrices. (Hofacker *et al.*, 2004)	20
An enhanced RNA alignment benchmark for sequence alignment programs. (Wilm *et al.*, 2006)	20
TOPS++FATCAT: fast flexible structural alignment using constraints derived from TOPS+ Strings Model. (Veeramalai *et al.*, 2008)	20
PREDICT-2ND: a tool for generalized protein local structure prediction. (Katzman *et al.*, 2008)	20
Multiple sequence alignments of partially coding nucleic acid sequences. (Stocsits *et al.*, 2005)	19
PROMALS: towards accurate multiple sequence alignments of distantly related proteins. (Pei & Grishin, 2007)	18
STRAL: progressive alignment of non-coding RNA using base pairing probability vectors in quadratic time. (Dalli *et al.*, 2006)	18
Colorstock, SScolor, Ratón: RNA alignment visualization tools. (Bendaña & Holmes, 2008)	17
DNA reference alignment benchmarks based on tertiary structure of encoded proteins. (Carroll *et al.*, 2007)	17
R-Coffee: a method for multiple alignment of non-coding RNA. (Wilm *et al.*, 2008)	17
BAliBASE 3.0: latest developments of the multiple sequence alignment benchmark. (Thompson *et al.*, 2005)	16
MARNA: multiple alignment and consensus structure prediction of RNAs based on sequence structure comparisons. (Siebert & Backofen, 2005)	16
MASTR: multiple alignment and structure prediction of non-coding RNAs using simulated annealing. (Lindgreen *et al.*, 2007)	16
Progressive multiple sequence alignments from triplets. (Kruspe & Stadler, 2007)	16
Sigma: multiple alignment of weakly-conserved non-coding DNA sequence. (Siddharthan, 2006)	16
SimulFold: simultaneously inferring RNA structures including pseudoknots, alignments, and trees using a Bayesian MCMC framework. (Meyer & Miklós, 2007)	16

In other approaches, a lot of irrelevant hits could be detected like matches within words by not using word boundaries. The label for the C programming language for example was found in each publication, because each occurring of the alphabetic character “c” was found. The regular expression for the pattern matching of composed keywords (those containing whitespaces) is less restrictive, because these combinations are not assumed to be found artificially. However, with increasing length of one word keywords, the probability of false positive hits, apart from ambiguities, decreases. The implementation of a tagging approach containing further dis-

tinctions of non-whitespace keywords is a benefit-cost estimation, which means the effort of extending regular expressions might be more time consuming than the results legitimate.

Thus the design of regular expressions is a balancing of restrictiveness, which inhibits false positive matches, against the freedom to find new relevant labels that embrace the searched keyword. The introduced tagging process already reveals satisfying results.

5.4.2 Ontology Labels

Section 5.3 shows the results of a tagging study for a selection of 211 publications, all of them dealing with bioinformatics. First, these papers were tagged by the labels of an initial version of BIO2ME. The most found labels, listed in Table 5.3, are not surprising, because all publications were selected fitting the domain of the ontology. Furthermore, BIO2ME in its former state was in particular modeled for alignment tools. Therefore most frequently found labels like "sequence", "RNA", "protein", "alignment", etc. appear in the ontology. Keywords like "data", "tool", "program", "database" and "computational method" can be traced back to the core structure of BIO2ME, which models basic concepts of bioinformatics tools and methods. The occurrence of "Feature" under the most found keywords shows that false positive matches are not only a matter of syntax, but also of semantics. To avoid those hits, all labels of data formats were suffixed with "format".

Artificial matches could mostly be prevented. However ontology labels are selected meeting two demands: 1. labels are assigned in regard to the BIO2ME information system, which needs a preferably complete acquisition of synonyms to cover possible search patterns of the users. This entails the labeling with abbreviations that might consist of single or few letters like the abbreviation of a structural alignment score "SCI". 2. the labels are used for tagging publications. However, assignments of few letter words are not meaningful in the majority of cases.

After the initial tagging ten publications were not tagged at all. For six of these papers no abstract is available in PUBMED and the titles do not include any keywords. Other three publications without abstracts were tagged by one keyword each. Further three not initially tagged papers deal with bioinformatics research fields that are not modeled in BIO2ME now. These publications were tagged after the extensions. One last not tagged paper, which came with an abstract, could not be tagged even after the extensions. This abstract deals with a "combined transcriptomics/bioinformatics protocol that identifies cell surface glycoproteins" (Aplin & Singh, 2008). "Protocol" is not a bioinformatics tool described in the ontology. Futhermore, this example shows that the keyword "protein" in "glycoprotein" cannot be matched. This is due to the restrictness of the word boundaries, which inhibits hundreds of false positive hits for abbreviations and one letter labels.

All described tagging processes have shown that only a relative small number of ontology labels were found in the publications. This holds for initial as well as for newly inserted labels. It can be explained by the different grade of specificity of the ontology keywords. Some labels in the ontology are very general and some are extremely specific. In publications both of these labels can be found, however only a small number of specific terms are used in general. The classes of biological tasks and data formats for example already consider domains that are not

covered by those tools that are introduced in the publications of the collection. Furthermore, labels of object and datatype properties are extremely difficult to find exactly in text. First and foremost they serve the readability in the BIO2ME application. The label "is available at" for datatype property `isAvailableAt` to specify the homepage of the modeled tool is hardly to find in abstracts. Texts comprise phrases like "is online available at" or "are available at" that cannot be found by the introduced tagging mechanism. Some journals insist on providing "AVAILABILITY:" in the abstract, that is why "availability" was added as label of the property. The problem of finding different times, number and conjugation of verbs cannot be solved with the given approach, but has to be addressed to computer linguists and information extraction.

As expected, the comparison of ontology tags with PUBMED keywords showed that most PUBMED tags are far too coarse for the BIO2ME domain. This is why they are used as prefilter to select publications from PUBMED that deal with ontology relevant topics. Additionally, these tags have the advantage that they are manually linked to the article so no artificial tag is assigned.

5.4.3 Ontology Extension

To show the amount of information that can be extracted due to publication tagging based on ontology labels, an extension predicated on the 28 most tagged publications (see Table 5.2) was performed. Subsequently the most tagged paper (Chang *et al.*, 2008b) was analyzed in more detail to exemplify the information found in a paper's title and abstract. This offered that tagging analysis of title and abstract provides ontology relevant information. Missing concepts, individuals, properties and synonyms of ontology labels could be found. The overvalue of the extension of missing synonyms became apparent for example in the new label "protein sequence", described in Section 5.3, which is a synonym of "amino acid sequence". The new label tagged 22 publications of the collection whereas "amino acid sequence" only was found in seven papers. This aspect of ontology extension supports the completion of some ontology subdomains. Furthermore, new tools and knowledge about these can be found. This is a crucial point in information retrieving. This information is very hard to find automatically and is even depending on the expertise and experience of the knowledge researcher. For example, keywords such as "tool", "program", "computational method" and "algorithm" indicate that the analyzed publication might publish a new program or method.

Knowledge contained in title and abstract varies in a wide range. In Chang *et al.* (2008b) a lot of the available knowledge was covered in title and abstract. Though, other analyzed publications showed only fundamental information in this parts of the paper. Full-text articles mostly offer information about the computational methods and in publications introducing a tool, benchmarks can often be found in the results section. In the majority of cases this information is too detailed as to be mapped in the ontology. However, if the abstract includes no statement on this information, this means a loss of important knowledge about a tool. In those cases the knowledge researcher should analyze the full text, too. Though tagging of whole articles nevertheless is not profitable as the information that a new tool is described in the article, can be found in the abstract. The scientist can decide if the full text has to be scanned either. For a lot of tools the

online availability is offered in the abstract. In these resources the knowledge researcher often finds information like the input and output type of data and their formats.

The composition of the most tagged publications (see Table 5.2) and the most tagged paper itself was not surprising, because the ontology was particularly developed in the field of alignment programs. The most tagged publications mainly deal with alignment programs in a broader sense.

5.4.4 Conclusions

The case study has shown that the implemented tagging method is a sound and helpful filter, which adverts to publications that deal with ontology relevant topics. The approach of pre-filtering publications by keyword search in PUBMED and the subsequent sorting out, or even weighting of ontology significance based on ontology-based tagging, delivers reasonable publication selections for a following analysis. As preliminary filter also the use of publications of a specific journal are thinkable.

The knowledge extraction is still an intellectual effort of knowledge engineers and domain experts. Meaningful hits have to be detected and located in the ontology and modeled accurately afterwards. The quantity and quality of knowledge extracted from publications is indisputable depending on the extractors. Their background knowledge and experiences help to retrieve implicit information, which is hardly to detect automatically even for intelligent information retrieving methods, but also for humans unfamiliar in the field of knowledge. A supposable improvement is the implementation of information extraction methods that promise to find keywords in the text after the root of words, so-called word stemming.

The ONTOVERSE platform still misses a supporting interface for the presentation of tagging results and the facilitation of subsequent analyses. A particularly helpful feature would be the exploitation of ontological information in this presentation, for the tags are not just keywords but are semantically linked via the ontology. By the inclusion of these semantics, the analysis becomes more facilitated, because keywords get a meaning and become interpretable.

In summary, the here introduced semi-automatic approach of ontology extension on the basis of ontology-based tagging is rather simple, but was already shown as being a successful support for knowledge engineers. Nevertheless, the knwoledge engineer has to decide which and how the information is to be (manually) modeled in the ontology.

6

BIO2ME Information System

The previous sections introduced BIO2ME as well as the support of collaborative and machine-supported ontology creation and extension. In this thesis these processes served the building of a knowledge representation of bioinformatics tools and methods. To profit from this knowledge base, the BIO2ME information system (BIS) was developed, which in first instance should make BIO2ME searchable for everyone who is interested. Therefore it was implemented in a RUBY ON RAILS web application (see Fig. 6.1), which enables a facilitated utilization of BIS and offers a potentially wide distribution. This implies a user interface that is easy to handle and does not assume any user knowledge of the data basis in the background. Additionally, the interface should be made as flexible as possible, so that an extended knowledge base can be imported directly without any modification of the web application. The search is described in Section 6.2.

After implementing the search interface, an ontology browser was integrated that facilitates the manual browsing of BIO2ME. Interested users are able to understand the crosslinking in the ontology. Furthermore, users are able to contribute their domain knowledge, insert new tools or additional information about them (see Section 6.3).

Ontologies are liable to the open world assumption. This is an important aspect in the utilization of BIS. If a piece of information is not provided by the information system, this does not mean that it does not exist. On the other side this means that a user, who is aware of such a lack of knowledge in BIO2ME, is invited to insert his knowledge.

6.1 Knowledge Base

The BIO2ME ontology serves as knowledge base for the information system. To satisfy this application, in the first instance it had to be refined. Several rules were established that are founded in the utilization of the Semantic Web framework DEEP SEMANTICS and in the user friendliness of the Web interface.

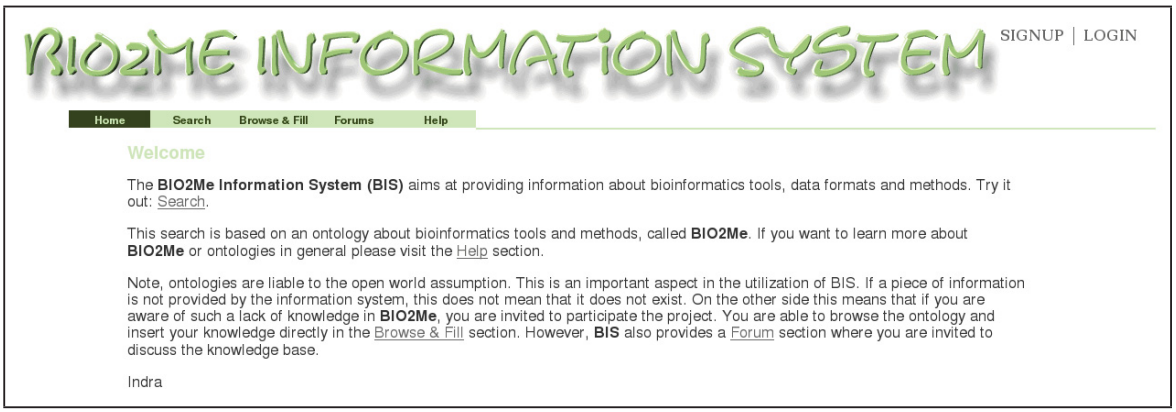

Figure 6.1: Startpage of BIS. The navigation bar at the top enables a fast access to the diverse sections of the information system: Search, Browse & Fill, Forums and Help.

Furthermore, the ontology has to be checked for consistency and all implicit knowledge is infered to become searchable (see Section 6.1.2). Finally, BIO2ME has to be serialized in N-Triple format (see Section 2.1.3) to be parsable by DEEP SEMANTICS.

6.1.1 Requirements

There are some guidelines defined to enable the use of BIO2ME within the Web application. There are two categories of requirements: First, some requirements are based on the deep integration approach of DEEP SEMANTICS (see Setion 2.6), which transfers ontology classes into RUBY classes. Thus, local names of ontology classes are subject of the RUBY syntax. Additionally, the ontology model has to be OWL Lite conform, because DEEP SEMANTICS is predently restricted to OWL Lite constructs and some additional OWL DL features. Second, some requirements are based on usability aspects of the Web application. Taking these requirements into account, the following rules were established:

Deep integration:

- Ontology class names begin with an uppercase character. This is due to the fact that in RUBY class names are constants. The RUBY interpreter distinguishes these from variables by the starting uppercase character. In fact in RUBY each class has a corresponding global constant with the same name as the class. This means that classes are treated like any other RUBY object; they can be copied, passed to methods and used in expressions.
- Ontology class names may consist of any combination of letters, numbers and underscores, just as constants in RUBY.
- The ontology includes no unions. This means amongst others that properties are not allowed to possess more than one domain and one range class.
- No enumerated data ranges: OWL DL allows the definition of a set of concrete data values for a datatype property, these can then be related to individuals through the property.

Usability:

- Insertion of at least one natural language label for each class, individual and property.
- Each property has a short description that can be displayed in the search form to help the user understanding the search field.
- Classes and individuals should be commented.
- All `rdfs:comments` are defined HTML conform to enable a correctly formatted display in the Website.

The initial BIO2ME ontology was adapted according to these rules. Hyphens were removed from local class names. All multiple domains and ranges were remodeled to discard unions, see Section 3.2. Enumerated data ranges were eliminated. For example the datatype property `hasLicenseType` had a selection of several license types modeled as strings in an enumerated datatype range. Now the property is remodeled as object property and the license types are remodeled as instances of the new class `LicenseTypes`, which is the range class of the property. As described in Section 3.2 all classes, instances and properties of BIO2ME were labeled to be displayable in a user friendly way in natural language. Furthermore, each comment was reformatted to HTML.

6.1.2 Ontology Preparation

Fig. 6.2 schematizes the different steps in ontology preparation before it can be used within BIS. BIO2ME was built and is modified with the formal ontology editor PROTÉGÉ (see Section 2.3). The consistency check and inference processes were accomplished by the OWL reasoner PELLET (see Section 2.4) and the serialization with the aid of JENA 2 (see Section 2.4). After that the ontology is deep-integrated (see Section 2.6) into RUBY and is then accessible for the information system. The depicted preparation steps are described in more detail in the next sections.

Consistency Check

The BIO2ME ontology has to be checked for logical correctness each time it has been modified before it can be applied in BIS. This assures that no inconsistent information is modeled in the ontology, because a search over an inconsistent ontology might reveal wrong results. For example, an internal conflict would be the declaration of individual `FASTA_Pearson_format` as instance of class `AlignmentDataFormat` and of class `SequenceDataFormat`, if these two classes are modeled to be disjoint. The conflict follows directly from the definition of disjoint classes to be exactly the disjunction of their instance sets. Though at first glance, alignment data and sequence data are disjoint concepts, but in this case the formats are modeled and these may overlap. A knowledge base cannot comprise a class distinction and a common instance of these classes, because both statements cannot be true at the same time. The consistency

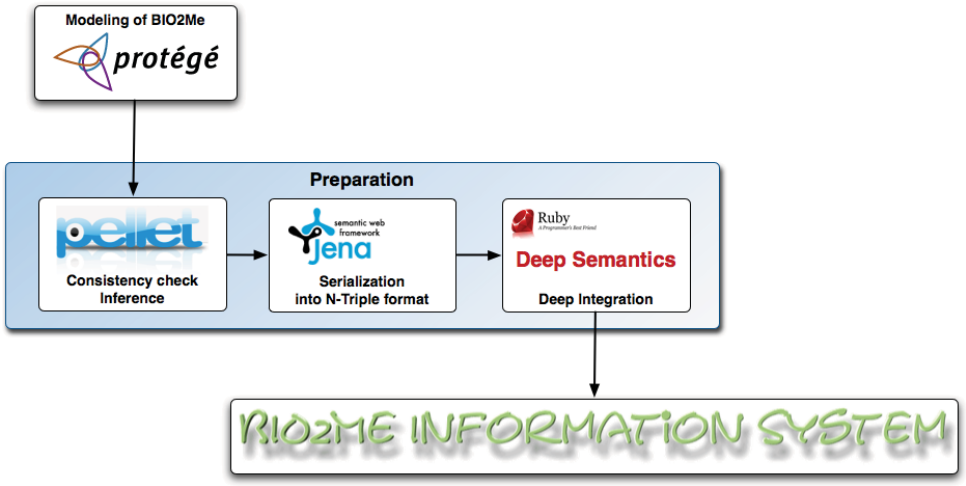

Figure 6.2: BIO2ME Preparation. The scheme shows the processing of BIO2ME from its creation and modification, respectively, over the preparation to the point of its application in the information system. This procedure has to be reiterated each time the OWL ontology is modified.

---

check helps to discover such wrong modelings and the knowledge engineer than has to decide how to erase it. In the described example, the declaration of `AlignmentDataFormat` and `SequenceDataFormat` to be disjoint is false and has to be removed from the ontology model.

The consistency check is carried out outside of the application by a selfmade command line program using the OWL reasoner PELLET. Fig. 6.3 shows a code snippet of the consistency check. PELLET libraries can be used via JENA 2, a JAVA framework for handling ontologies. Lines one to four of Fig. 6.3 read in the ontology from a file into an ontology model (PELLET class). This `OntModel` class provides divers methods to operate on ontologies. In line five a validity report is created. If the ontology is valid, meaning logically consistent (line six), "True" is printed to standard out (line seven). If there exist warnings or errors, these are provided in the console (line ten). The error message of the above described example would look like this:

```
False

Validation Results
==================
Error (KB is inconsistent!):
Individual http://www.ontoverse.org/BIO2Me.owl#FASTA_Pearson_format
is forced to belong to class
http://www.ontoverse.org/BIO2Me.owl#AlignmentDataFormat and its
complement
```

Inference

Ontologies possibly include information that is not explicitly formalized, but is implicitly included. For example the distinction of direct and indirect instances results from this feature. The

```
 1  String ontology_file = "file:BIO2Me.owl";
 2  OntModel model =
 3      ModelFactory.createOntologyModel(PelletReasonerFactory.THE_SPEC);
 4  model.read(ontology_file);
 5  ValidityReport report = model.validate();
 6  if(report.isClean()) {
 7      System.out.println("True");
 8  } else {
 9      System.out.println("False");
10      printIterator(report.getReports(), "Validation Results");
11  }
12  File outfile = new File("Bio2Me_infered.nt");
13  FileOutputStream fos = new FileOutputStream(outfile);
14  model.writeAll(fos, "N-TRIPLE", null);
```

Figure 6.3: Reasoning with PELLET.

individual `StrAl0.5.4` is the direct instance of the ontology class `StrAl`. Because `StrAl` is subclass of `Program`, the information that `StrAl0.5.4` also is an instance of `Program` is contained implicitly due to inheritance. To make all information searchable, the ontology has to be infered before loading into the application. The ontology is automatically infered by writing out OntModel of PELLET.

If the ontology is not valid, the inference aborts with the following error message:

```
ERROR [main] (RDFDefaultErrorHandler.java:40) -
Cannot do reasoning with inconsistent ontologies!
```

Serialization

To process the ontology in the Semantic Web framework DEEP SEMANTICS, BIO2ME has to be serialized in N-Triple format. In Fig. 6.3 line 14 comprises the command to store the ontology model into a N-Triple formatted file.

Deep Integration

By starting the web server, the N-Triple file is read and parsed by DEEP SEMANTICS, which is loaded as plugin into the application. In Section 2.6 the deep integration process is described in general. The OWL ontology is converted into a functional RUBY model, with which the ontology can be accessed like any RUBY model. The advantage of this deep integration is the natural (in a programming sense) handling of the ontology data in the scope of the RUBY application. The ontology model is assigned to the global variable `$bio2me`, which is accessible in the whole application. Hence, the ontology persists in the main space as long as the web server is running. This enables a quick access to the data. The memory complexity of DEEP SEMANTICS

currently does not pose a problem. The web application loaded with the present knowledge base, which comprises 6502 triples, uses only approximately 7 MB cache.

DEEP SEMANTICS provides a divers set of methods to operate on the ontology model. There are methods to list all classes, instances and properties in general and under specific conditions like listing porperties according to their range classes. Furthermore, the deep-integrated ontology model offers dynamically created methods for classes and instances. Thus, for a given OWL class (mapped on a RUBY class, e. g. `StrAl`) all instances and all instances of a certain label (e. g. "StrAl 0.5.4") are retrievable by:

```
$bio2me::StrAl.instances
$bio2me::StrAl.find_instances_by_label("StrAl 0.5.4")
```

Similarly, other attributes like comments, subclasses and superclasses can be retrieved. For a given individual (e. g. `StrAl0.5.4`)

```
$bio2me::StrAl0.5.4.writesFormat
```

delivers an array of all instances that are related to `StrAl0.5.4` with the object property `writesFormat`. This is possible due to the fact that DEEP SEMANTICS declares instance methods for each object and datatype property, which is related to every deep-integrated ontology individual.

6.2 Search

After the creation of the ontology and its preparation the ontology is deep-integrated into a RUBY model in the RAILS application. The search controller is then able to access the ontology data using the methods described above. The search procedure can be divided into several steps that are schematized in Fig. 6.4:

1. Each search process starts with the specification of a concept, for which information is desired. The user has the choice to search programs, databases, computational methods, types of data or data formats, which all are represented by top level concepts in the ontology. For this purpose the interface provides five radio buttons. The initial distinction between these concepts maps ontology classes and facilitates the building of class specific search forms of just these concepts, the ontology provides information about.

2. After the selection of the concept the user wants to get information about, a concept specific search form is created dynamically based on the ontology model. Therefore all object and datatype properties are fetched that possess the selected class as domain. DEEP SEMANTICS provides methods of the ontology model `$bio2me` to directly get properties by their domain, in the following exemplified by the class `Program`:

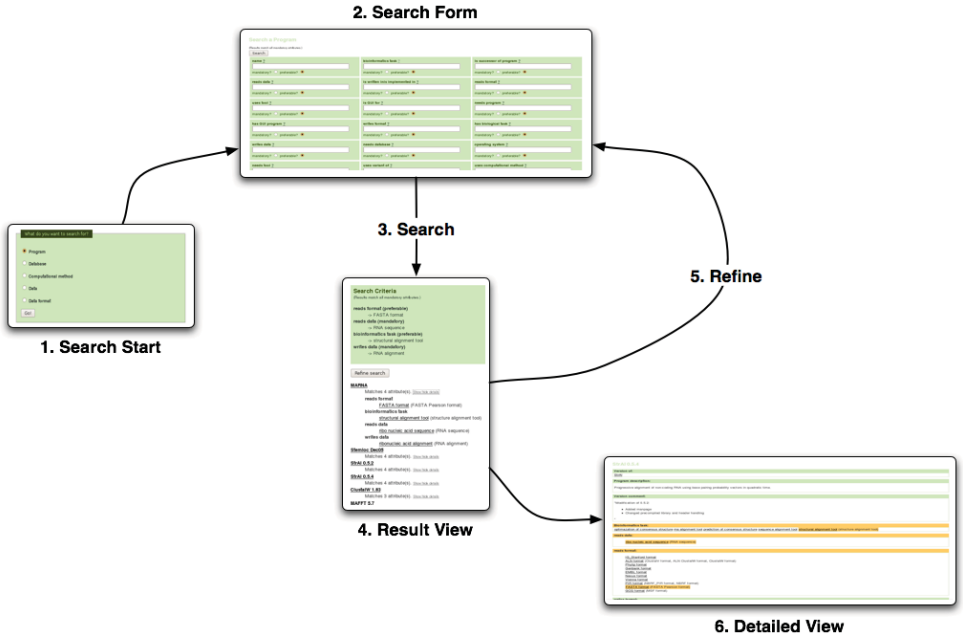

Figure 6.4: Search procedure. The search procedure is splitted into six steps in the user interface of BIS. Detailed descriptions of these processes are given in the text.

```
$bio2me.listObjectPropertiesByDomain($bio2me::Program)
$bio2me.listDatatypePropertiesByDomain($bio2me::Program)
```

For each property a text field is provided in the search form (see Fig. 6.5). Search fields of datatype properties with range `xsd:boolean` are implemented as select boxes, in which the user is able to select nothing, “true” or “false”. Additionally, a name field is offered that allows the direct search for a program, database, method or data of known name. In case a user looks for a program, a bioinformatics task field appears. This information is not represented with the aid of properties in BIO2ME, but about `instance of` relationships, that is why these two fields have to be handled separately.

Furthermore, each search field provides two radio buttons (see Fig. 6.5), which declare whether the according property is a criterion for exclusion and has to be possessed by every result or if it is just preferable to be included. This means, each resulting ontology instance meets all mandatory demands and some may possess also preferable attributes. Additionally, each search field offers a question mark linking to a popup. This displays a help message describing which information is provided by the according property. This text is dynamically generated out of the property’s `rdfs:comment`(s) in the ontology. For this reason each property has to feature a comment formatted in a web browser interpretable form.

Moreover, each text field is provided with auto completion. Thus, by typing letters in a search field an auto completion is triggered on the basis of the labels of all ontology instances that are available for the defined range of the according property. RAILS provides a set of helper methods for creating JavaScript macros, which amongst others facilitate

the implementation of text fields with auto completion. For example the RUBY HTML (RHTML) syntax of the search field for the object property `writesFormat` looks like:

```
text_field_with_auto_complete 'writesFormat', nil, {:size => 40},
  {:url =>{:action => 'auto_complete_for_object_property',
   :property => 'writesFormat', :class => @search,
   :results =>@results}, :method => :get, :skip_style => true}
```

and is translated to the following HTML code:

```
<input id="writesFormat" name="writesFormat[]" size="40"
  type="text"/>
<div class="auto_complete" id="writesFormat_auto_complete">
</div>
<script type="text/javascript">
  //<![CDATA[
    var writesFormat_auto_completer =
      new Ajax.Autocompleter('writesFormat',
        'writesFormat_auto_complete',
        '/search/auto_complete_for_object_property?class=
          Program&amp;property=writesFormat', {method:'get'})
  //]]>
</script>
```

By typing characters, the method `auto_complete_for_object_property` in the search controller is called and searches all labels of all subclasses and instances of the range class, which begins with this character(s). If the array `@results` given by parameter `result` is not empty, the user performs a refinement search (see 4.). In this case not all subclasses and instances of the range class are scanned, but only those labels of instances and their superclasses that are related to the result instances over the specified object property. For datatype properties, names and bioinformatics tasks the auto completion is performed appropriately.

3. After the search form was filled and committed, the search controller performs the search depicted in Fig. 6.6 for each attribute that belongs to a filled and/or mandatory field. In the following the procedure is described on the basis of a search for programs:

   At first it is determined whether the current attribute is mandatory or preferable. If it is mandatory, but the according field was not filled, each instance of program that possesses this attribute is taken into the temporary result set. In case there already exists a temporary result set, each result instance not overlapping is deleted from the set. When the result set subsequently is empty, the search loop is aborted and the result page is rendered without any result. If the temporary result set is not empty the search loop starts again with the next attribute. By the time all desired attributes are processed, the temporary result set is passed to the result view.

   If a field is filled and declared as mandatory, one of the following two alternative actions is executed:

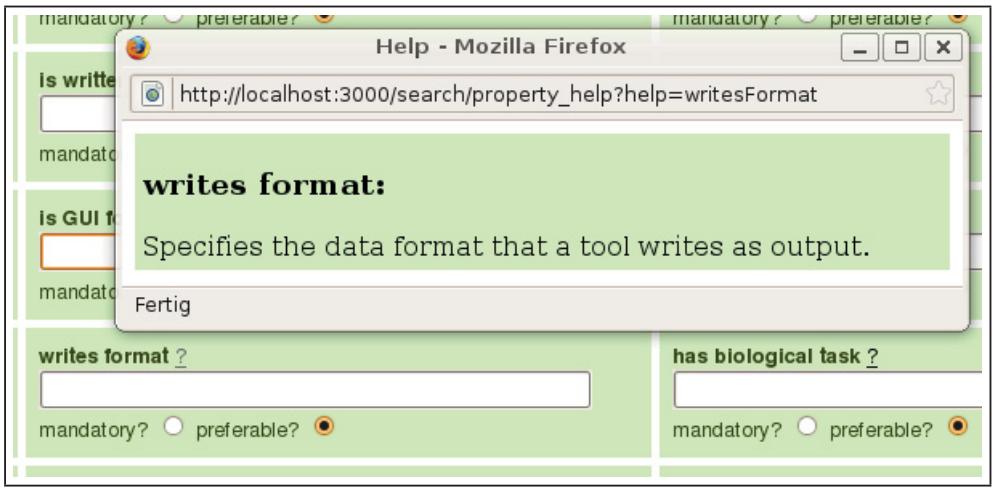

Figure 6.5: Search field with help. Screenshot of a search field and the help window for the according property.

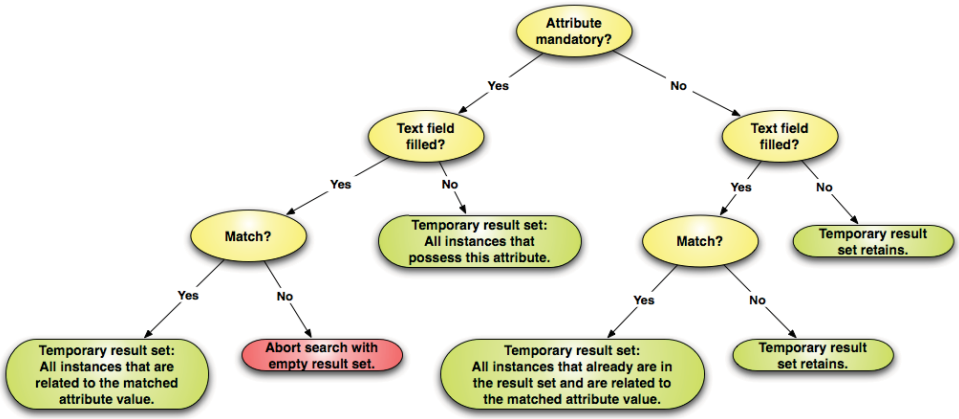

Figure 6.6: Attribute search. The scheme depicts the search for one attribute. Several if-else clauses decide the further progress of the search. At the end of the search either a result set exists and a next attribute is searched (green boxes) or the search is aborted (red box).

(a) If no temporary result set is available, all program versions have to be checked. For the name and bioinformatics task search fields this means, that the provided pattern is searched in labels of all subclasses and instances of the classes `Program` and `BioinformaticsTool`, respectively. All found instances and the instances (including indirect instances) of all found subclasses are added to the preliminary result set. This approach exploits inheritance in the ontology. For example, if a user looks for an alignment program, the bioinformatics tool `AlignmentTool` (subclass of `BioinformaticsTool`) presents a match (see Fig. 6.7). Due to the inheritance, all instances of (e. g.) `SequenceAlignmentTool` are also instances of `AlignmentTool`. Thus, all instances of `SequenceAlignmentTool` and of all other subclasses of `AlignmentTool` are alignment tools and consequently included into the temporary result set.

For object properties (e. g. `writesFormat`) the range class has to be determined first:

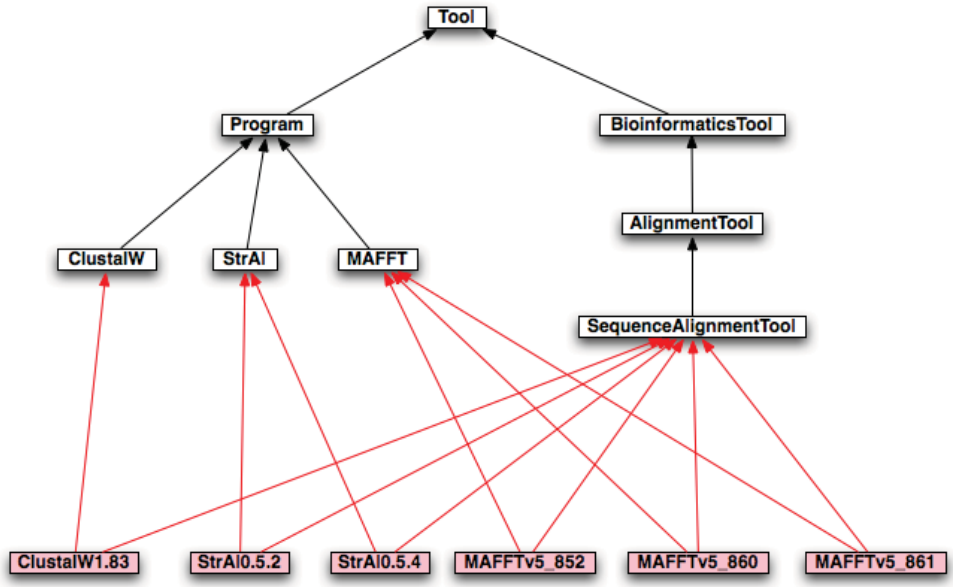

Figure 6.7: Ontology subgraph for `Program` and `BioinformaticsTool`. The graph depicts the modeling of programs. Program versions are instances of class `Program` as well as of `BioinformaticsTool`. The latter represents bioinformatics tasks of the tool. Instances are represented by pink boxes. Black and red arrows depict "is a" and "instance of" relations, respectively.

```
range = $bio2me::writesFormat.range
```

Then the labels of all subclasses and instances of the range class are searched through for the pattern.

```
range.find_subclasses_by_label(pattern)
range.find_instances_by_label(pattern)
```

If a label of a class matches, due to inheritance all instances of this class and all matching instances are added to a value list. Each program version that is related via the object property to at least one possible value instance is then inserted into the preliminary result set (see Fig. 6.8).

In case of a datatype property each value of this property for every program version has to be considered and compared to the search pattern. The temporary result set is then extended by those program versions that include matching datatype values.

(b) If a temporary result set is available, it implies that the search for a previous mandatory attribute was successful. In this case only the program versions of the preliminary results have to be checked. For the name attribute this means that either their labels have to match the search pattern, or the labels of those classes, to which the instances of the temporary result set are tied, and which additionally are subclasses of `Program`. For the bioinformatics task attribute also those classes that possess the program versions as instances and that additionally are subclasses of `BioinformaticsTool` are taken into account. The restriction to both specific superclasses, `Program` and `BioinformaticsTools`, limits the set of classes the individual is instance of to the subset that is relevant for the attribute. In case of

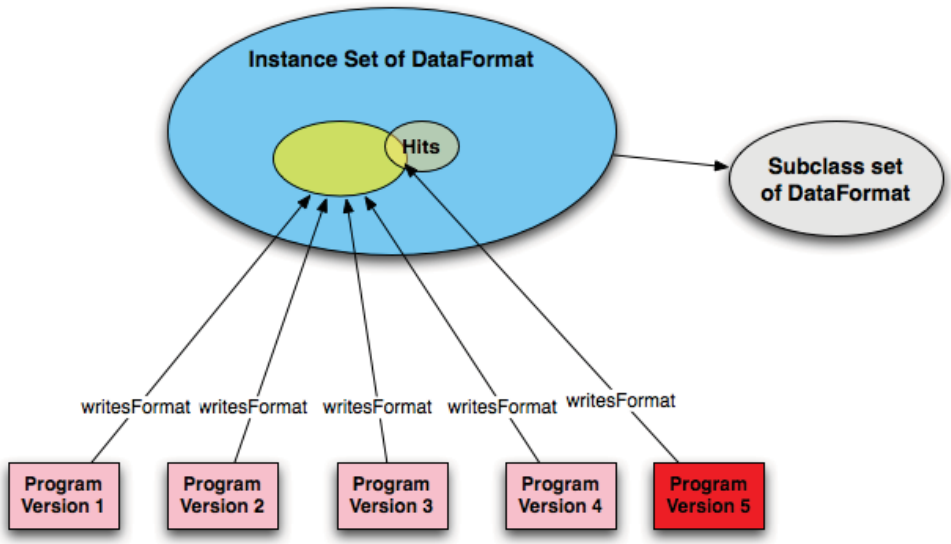

Figure 6.8: Search of Object Properties. The figure schematizes the determination of a program version (red box) that writes a format, which matches the search pattern ("hits"). The green oval depicts the subset of data format instances, which are related to program versions. The program version, which is related to a data format of the intersection of both subsets, is the result of the search.

---

the name attribute the left ontology branch of Fig. 6.7 is significant. Bioinformatics task relies on the right side of the graph.

In case of an object and datatype properties the search is performed equally to the above described approach except that the set of program versions is restricted to the preliminary result set.

For both cases is essential, that if the temporary result set is empty after the described procedure, because either no result for the current mandatory property is found or the current result set does not overlap with the temporary result set, the search procedure is aborted and the search returns no results.

If the search field is filled and declared preferable (see Fig. 6.6) the above described search steps for mandatory properties are examined. Though, if a temporary result set is available, none of these resulting instances are dismissed, because the current preferable attribute is just an additional search criterium. In case no temporary result set is available, the results are stored in a temporary preferable result set, which is adjusted with the preliminary (mandatory) result set, if a mandatory property follows.

The final results are saved in datastructures that allow the comfortable access to preferable properties, which are matched for the respective result.

4. On the basis of this datastructure, the result view comprises detailed information about the search and its results. Fig. 6.9 provides a result list. On top of the window the user can see the precise search criteria he defined. The results are listed underneath sorted in a first instance according to the number of matched properties and in a subsequent level lexicographical by the name. For each result, the number of matches are shown as well as the possibility to display which properties matched. The according range instances that

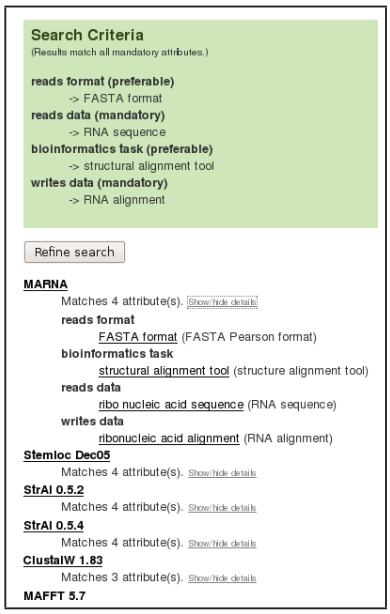

Figure 6.9: Result View. Screenshot of the result list.

caused the match are given by label. If an instance has assigned more than one label, all further labels are attached in parenthesis. This helps the user to recognize those instances. Each result and range instance is linked to a detailed view of it (see 5.) and the result set can be refined by an additional search over this instance set (see 6.).

5. In the detailed view (see Fig. 6.10) the matched attributes are colored orange as well as the actually matching instances and values, respectively. Additionally, the view shows all further information about the selected class or individual that is modeled in BIO2ME. A semantical linking is integrated, which means that related instances (and program subclasses) are linked, so that the user can browse the information without having to know the underlying knowledge base. Furthermore, if the ontology comprises a `rdfs:comment` for the selected object, it is displayed as a short description. Program instances additionally have a version description, for the program classes' comment serves as program description and the comment on the instance provides the version information. The detailed view of program classes allows a listing of all its program versions. All ontology classes and instances are represented by their natural language labels.

6. The result page also comprehends a "Refine search" button (see Fig. 6.9). This enables the refinement of a search. By clicking the button the user is redirected to the search form. The form again is dynamically created based on the ontology information, but this time it is composed of attribute search fields that are actually interrelated to the instances of the current result set. The auto completion function also takes only these results into account to display possible search patterns. The search then proceeds equally to the first search process, with the exception that the previous result set is utilized as preliminary (mandatory) result set.

The search procedure will be demonstrated by an example in the following section.

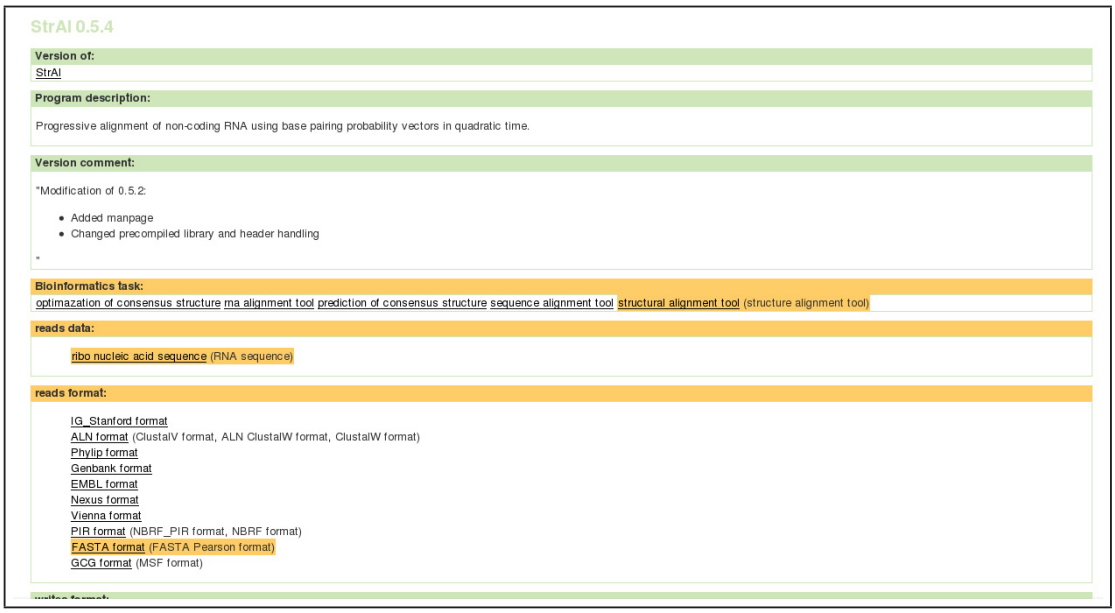

Figure 6.10: Detailed View. Screenshot of a part of the detailed view of instance `StrAl0.5.4`. Highlighted in orange are those properties and instances that matched search criteria.

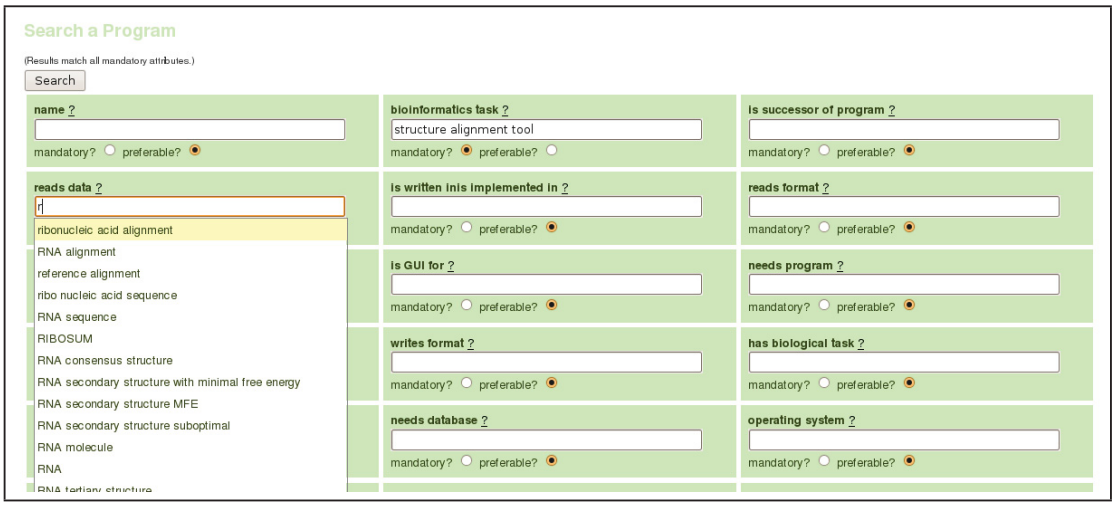

Figure 6.11: Example Search. The search form is depicted showing the auto completion function.

6.2.1 Example

The user searches a program, so he selects “Program” at the beginning of the search and the program specific form is loaded. Fig. 6.11 depicts the search form filled by the user; he is looking for a structure alignment program that reads RNA sequences at the best. Thus, in the bioinformatics task field “structure alignment tool” is inputted and declared as mandatory. Furthermore, “RNA sequence” is given in field “reads data” and marked as preferable. Then he starts the search.

The search returns 15 alignment programs, whereof nine tools fit both requirements (see Fig. 6.12). To narrow down the number of results, the user refines his search by making input data “RNA sequence” mandatory. Additionally, he prefers the input of FASTA formatted

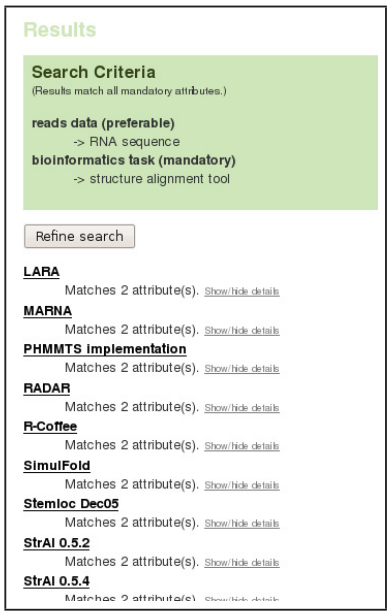

Figure 6.12: Example Result. Cutout of the result set of the example search.

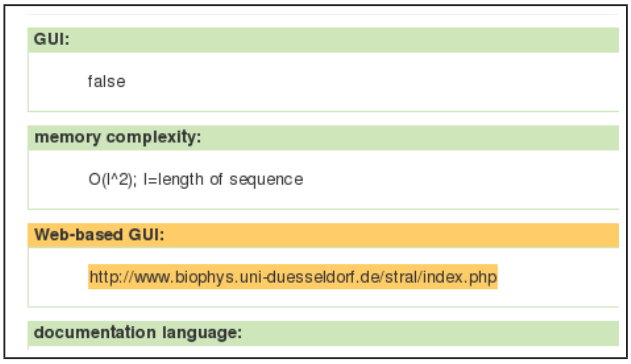

Figure 6.13: Example detailed view. The datatype property is highlighted that provides the URL of the Web-based interface of the program.

---

sequences into the program, so that he has not to convert his data set. Furthermore, the user demands a web-based graphical user interface, because he does not want to be bothered by a local installation. Thus, he clicks onto the "Refine search" button, fills out the dynamically created form with his additional criteria. This reduces the resulting list to programs MARNA and STRAL 0.5.4.

To decide which of these programs he wants to use, he enters the detailed views of both. A benchmark is referenced, which compares these programs. STRAL is tested to perform more accurately in regard to his sequence set, so he decides to use this alignment program. The user has now the possibility to read the referenced publication or visit the program's homepage. If the user wants to test the program, he is allowed to click to the provided URL, which is given in the detailed view under the subtitle "Web-based GUI" (see Fig. 6.13) and runs the program with his sequences.

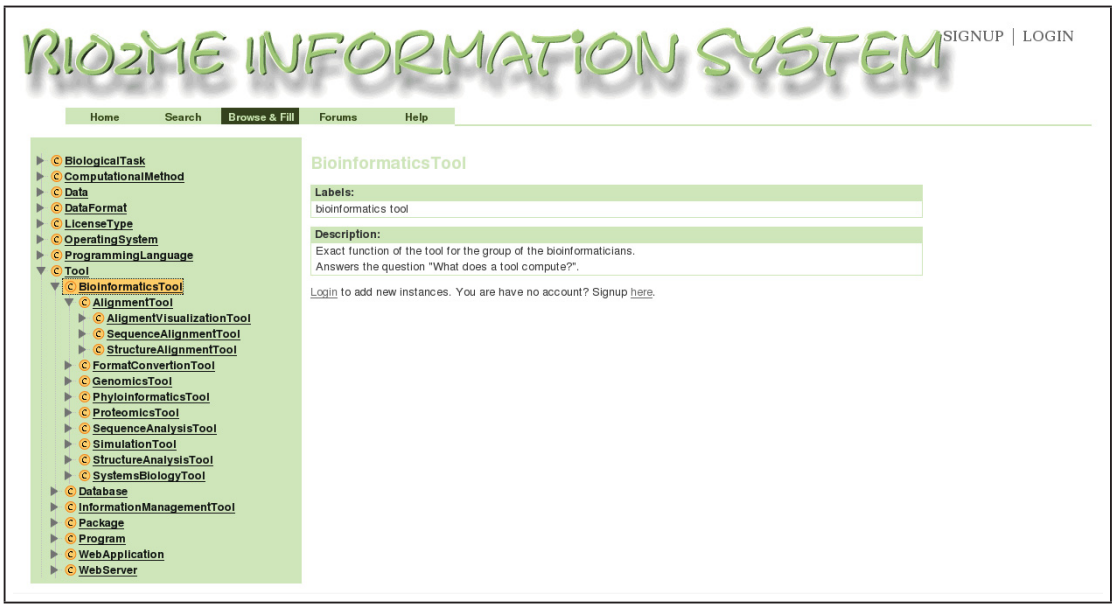

Figure 6.14: Ontology browser. In the ontology tree (left) class `BioinformaticsTool` is selected. The details to this class are shown on the right side.

6.3 Browse & Fill

The "Browse & Fill" section of BIS is intended to deliver interested users insight into the knowledge base BIO2ME. This section should help to get familiar with the ontology and to facilitate the support of the BIO2ME extension.

Fig. 6.14 shows the browser view. On the left hand side the ontology hierarchy tree is displayed consisting of ontology classes and instances. By clicking on a tree element the corresponding information of this concept and instance, respectively, is dynamically loaded from the RUBY ontology model. For classes the assigned `rdfs:labels` and `rdfs:comments` are provided, in case they are modeled in the ontology. The detailed view of instances additionally lists direct "type of" classes, object properties and datatype properties with the instances and values that are related (see Fig. 6.15).

The ontology tree is dynamically built based on the ontology model. The expand and collapse functionalities are implemented using RJS (see Section 2.5.1).

The more ontology-like representation of the information distinguishes the detailed view of the browser from the detailed view in the search section. Classes, instances and properties are constituted by their local names instead of labels and properties are directly identifiable regarding their type (object or datatype property). This presentation of ontology data facilitates the sufficient familiarization with the knowledge base to enable the contribution of missing information without the importunity of formal details.

Figure 6.15: Ontology browser focused on `StrAl0.5.4`. The detailed view of instance `StrAl0.5.4` provides information about its labels, description, parent classes and related instances. Related data values are cutted.

6.3.1 Extension of BIO2ME

By browsing the ontology a user might be interested to contribute his knowledge. In contrast to simple questionnaires (see Section 3.2.1) BIS facilitates the insertion of knowledge directly into the ontology. Users have to signup and login first before they are allowed to extend BIO2ME (see bottom of Fig. 6.14). This barrier helps to assure the quality of BIO2ME.

Class views offer text fields to insert new instances (see Fig. 6.16) by providing a name and label. A new instance with local name “StrAl0.6” and label “StrAl 0.6” can be added to class `StrAl` by

```
new_instance = $bio2me::StrAl.new("StrAl0.6")
new_instance.rdfs_label << "StrAl 0.6"
```

After inserting the instance the user is able to add information about it. Due to that, the individual view offers select boxes to relate the selected instance via object properties to other instances and via datatype properties to datatype values. These select boxes are dynamically filled with properties, whose domains are “type of” classes of the selected instance. After a property was selected, a list of possible range instances appears or a text field for entering the datatype value, respectively (see Fig. 6.17).

Additionally, the extension feature is adjusted to the concept “program”. Programs are specifically modeled in the ontology, as versions of programs are instances of the program class. The extension feature is in particular offered to facilitate the contribution of new tools by external domain experts. That is why the detailed view of class `Program` additionally provides text fields to insert new subclasses. The DEEP SEMANTICS framework does not aim at manipulating

Figure 6.16: Extending of Instances and Program subclasses. All ontology classes can be extended by instances. The `Program` class additionally can be expanded by subclasses modeling new programs.

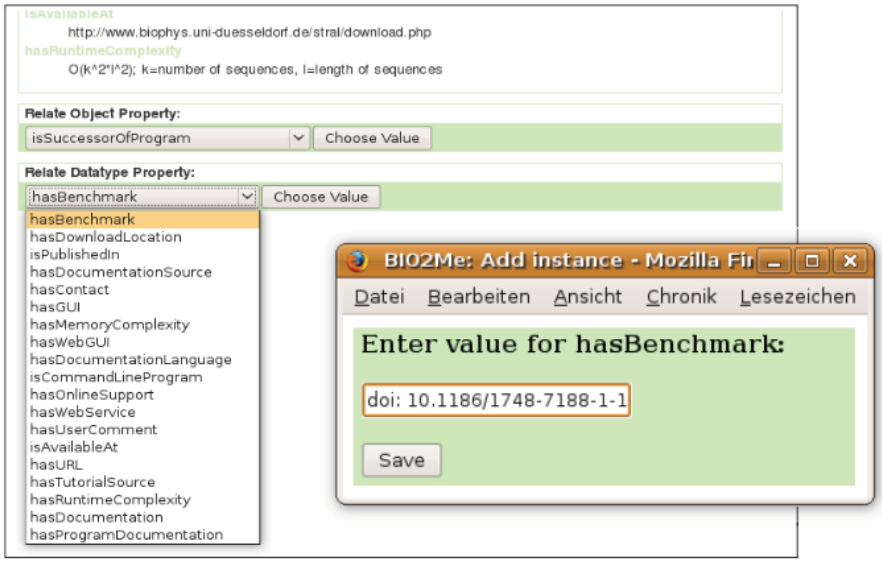

Figure 6.17: Extending of relationships. The selection of a datatype properties is shown, whose value can be inserted in a popup window.

---

the ontology scaffold consisting of class and property definitions. Thus, it had to be extended by methods to add program classes into BIO2ME.

The extension functionality of the BIS ontology browser is not intended to supersede a formal editor, but it supports the enlargement of BIO2ME with programs, instances and new information about instances by domain experts. The refinement and enhancement of the ontology scaffold consisting of classes and property declarations require much more familiarization with ontologies in general and the structure of BIO2ME in detail.

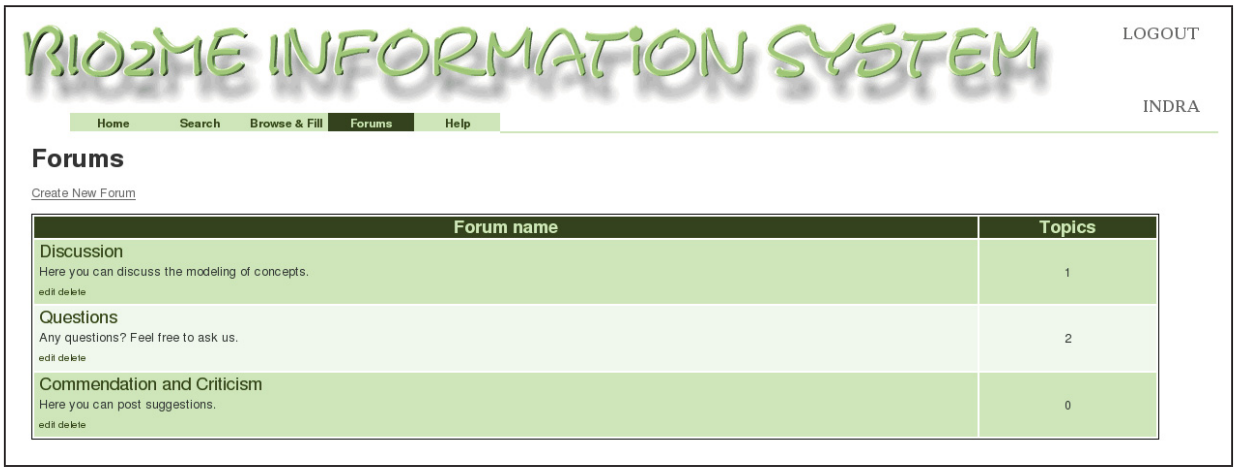

Figure 6.18: Forums. The forums enable discussion about BIS and BIO2ME.

6.4 Forums

BIS also provides forums, which are intended for users who want to comment on the application and the BIO2ME. The use of the ontology browser still requires a certain amount of familiarization depending on the intention of a user to view or extend the ontology. Furthermore, the extension functionality in the application is limited to the integration of new information represented by instances, programs and relating instances. Modifications on the BIO2ME structure can be discussed in the forum and incorporated into the ontology by ontology designers with the aid of a formal editor. Additionally, the forum helps to figure out deficiencies in the ontology and can also be used as discussion place for the ontology extension. Fig. 6.18 shows the start page of the “Forums” section within BIS, from which the user able to enter divers forums.

6.5 Discussion

The BIO2ME information system could also have been developed with an underlying database instead of an ontology, but the modeling of information in form of ontologies is much more comfortable than that of database schemata. The structure and handling of ontology data in form of graphs is less artificial as it meets thought patterns of the human mind. Furthermore, ontologies can be remodeled in a simpler way than database schemata of an already running system. Additionally, the dynamical architecture of BIS restricts the design of the ontology only to a minor degree.

The preparation of BIO2ME before it is integrated into the BIO2ME information system, is less extensive, because it only requires one program fetch and the (re-)start of the web server. Nevertheless, due to the ontology extension functionality of BIS the consistency check and inference should be integrated into the web application.

In the ontology community, there are often attempts to provide Semantic Web applications that are able to operate on any ontology. Due to the experiences gained during the development of

the BIO2ME information system, in my opinion this is quite infeasible. The structure of the ontology and the implementation of the application have to be coordinated to build a comfortable interface and provide as well as exploit all information modeled in the ontology adequately.

6.5.1 Search

The search functionality is mature and yet achieves well-founded results for the information existent in the ontology. Semantics included in BIO2ME is exploited in terms of considering inheritance by searching all individuals that are (direct or indirect) instance of a class, which label matches the search term. Furthermore, the meaning of relations between instances as well as between instances and datatype values are involved in the search. Additionally to the established features, the search and result representation could be refined. For example, the result view could be extended by a clustering of results e. g. based on bioinformatics tasks or other user defined aspects, which are involved in the semantics of BIO2ME.

Furthermore, the current implementation of BIS yet does not use phonetic search. In a first instance I concentrated on the development of a well elaborated search procedure, which exploits advantages of the semantic aspects of ontologies. Moreover, the utilization of auto completion text fields in combination with exact pattern matching is quite sufficient by now as it allows a simple matching approach by simultaneous providing a user friendly search.

The search form will be extended by select boxes for enumerated datatypes as soon as the Semantic Web framework DEEP SEMANTICS will provide the support of them. Then select boxes can dynamically be created based on the ontology data. This would result in a more comfortable user interface. At present the same result could be achieved by simple HTML coding. Datatype properties with values limited to a well-defined set could be handled separately. Therewith the lack of information could be compensated by the web application. Thus, this approach would mean a loss of dynamics in BIS, for the information would then not only base on the ontology, but additional knowledge would be statically included in the application.

On the basis of BIO2ME an automatic version comparison is realizable. For this purpose program versions were modeled by instances of a program class. According to that, the comparison of versions is simply the mapping of related attributes.

6.5.2 Browse & Fill

The BIS ontology browser enables the familiarization with BIO2ME for interested users. It provides a tree view of the ontology as well as detailed information about classes and individuals. Nevertheless, the presented information does not include the ontology scaffold in a whole. Definitions of properties and their domain and ranges are omitted, because the browser addresses interested domain experts and aims at attracting them to contribute their knowledge mainly on instance level. The only exception are programs, which require the inserting of classes. This is in contrast to the ONTOVERSE system, which addresses domain experts with increased interest in ontologies. The extension functionality of BIS does not intend and cannot displace a

real formal ontology editor. It is rather considered to ease taking part in the ontology extension process for domain experts who are not familiar with formal ontologies and are not eager for becoming acquainted with it. The simple filling out of questionnaires like provided so far for requesting domain knowledge (see Section 3.2.1) does not seem to be accepted in the community, that is why the ontology browser aims at forming a combination of offering information about BIO2ME, which makes domain experts feel like they are involved in knowledge engineering tasks, and nevertheless providing an interface that does not bother users with complicated ontological background.

By now the BIO2ME application does not include a persistency layer to store ontology modifications. Users are able to directly operate on the ontology model in RUBY, but this changes exist in the main space and yet expire by shutting down the web server. Mainz (2008) introduces an approach in serializing ontology data based on DEEP SEMANTICS. By integrating persistency, a quality check of new ontology information should be realized. Although a login is required for extending the ontology, the system is free for every interested user.

6.5.3 DEEP SEMANTICS vs. SQLSpaces

ONTOVERSE and BIS utilize different approaches for the integration of ontological data. Both frameworks are in development, which provide disadvantages in so far as full OWL DL support (DEEP SEMANTICS) and insufficient data handling (SQLSpaces) are concerned. Actually the comparison of both is illegal as SQLSpaces are considered for manipulating ontological data and DEEP SEMANTICS in first instance aims at providing a fast access to the information in ontologies. Therefore SQLSpaces offer persistency and (JAVA) methods to add facts, but DEEP SEMANTICS provides a intuitive and natural way of retrieving and handling ontological data. When the ontolgy is deep-integrated, information can be extracted at once. Additionally, DEEP SEMANTICS enabled me a lot of flexibility, for the source code was open to me and a close collaboration with its author facilitated the prompt extension of new methods like `find_instances_by_label()`. The use of established ontology quering languages like SPARQL[1] was not discussed in this thesis; in some tests these could be proved being less flexible and comfortable in contrast to DEEP SEMANTICS.

6.5.4 Conclusions

In consideration of the mass of existing bioinformatics tools, the need for an detailed search is obvious. There exist several web sites that list bioinformatics tools comprehensively. For example, the homepage of Dr. Joe Felsenstein[2] lists tools for phylogenetical analyses and the Online Bioinformatics Resources Collection[3] (OBRC) as well as BioWareDB[4] extend the simple listing

[1] `http://www.w3.org/TR/rdf-sparql-query/`

[2] `http://evolution.genetics.washington.edu/phylip/software.html`

[3] `http://www.hsls.pitt.edu/guides/genetics/obrc`

[4] `http://biowaredb.org/site/`

of bioinformatics tools by a search function. All of these collections have essential disadvantages: for example, they are often badly structured and only provide poor search functionality.

In contrast the BIO2ME information system provides an "intelligent" search for bioinformatics tools, methods and data, which is based on the underlying knowledge base BIO2ME. The tools are captured in regard to their applications, methods, input and output formats as well as several additional attributes (see Chapter 3). Thus, bioinformatics tools can be searched by specifying any combination of these attributes.

The BIO2ME information system is still work in progress and as BIS is not yet published, it is not evaluated by a reasonable number of users. However, the search procedure already is "intelligent" by means of exploiting the full ontological information. The extension facility of BIO2ME provides an auspicious attempt.

7

Summary

The amount of information increases exponentially these days. The life sciences in particular produce masses of data by utilizing modern methods like high-throughput screenings. This vast amount of data has to be analyzed and made accessible to enable the gain of new information and the reuse of data. For example, the size of sequence databases soars as high-throughput sequencing techniques facilitate the fast determination of biomolecular sequences. To access these data in an intelligent way ontologies were established in the life sciences. Ontologies—as considered in this thesis—represent knowledge of a domain in a structured and machine-interpretable way.

In this thesis an ontology for bioinformatics tools and methods (BIO2ME) was extended, which models bioinformatics programs, databases and methods according to (e. g.) their application and data which can be processed. BIO2ME aims at providing a knowledge base for an information system that enables an intelligent search across the plethora of bioinformatics tools, which are developed to face the visualization, processing, archiving and search for the above described amount of biomedical data. Moreover, there exist additional tools that support experimental methods or simulate biological processes. Even for bioinformaticians it is hard to review methods and tools serving a specific purpose. In this thesis the BIO2ME information system was established, which enables the intelligent search based on the semantics of the BIO2ME ontology and is unique in its field of application.

Domain experts are essential for the building of ontologies as they provide and cross-link the knowledge that is modeled in an ontology. Even though ontologies form a well founded field of research, during the development of BIO2ME a lack of the support of domain experts was noticed. Due to this discovery, in this thesis the wiki idea is adapted to the requirements of the informal ontology engineering phases, which comprehend the knowledge acquisition and the interrelation of pieces of information. Wikis are well suited for this task, because they are well established in the collaborative development of documents. Special characteristics of the here implemented Wiki are: the locking function of single sections in a Wiki document, which protects ontology definitions against vandalism, and a connection to a formal ontology model, which in particlular facilitates content-specific evaluation of the ontology.

Furthermore, the enrichment of the ontology with information is a considerable task, and the attracting of domain experts, who wants to contribute their domain knowledge, can be challenging. Based on that, a machine-supported approach for ontology extension was implemented in this thesis, which is based on a special tagging mechanism for documents with keywords from the ontology. This thesis comprises an extensive study, which involves a remarkable enlargement of BIO2ME.

In this thesis the benefit of ontologies could be shown on the basis of an ontology for bioinformatics tools and its application. New techniques were established, which facilitate the collaborative ontology design and the machine-supported extension of ontologies.

8

Zusammenfassung

Das Wissen wächst heutzutage auf vielen Gebieten exponentiell. Vor allem im Bereich der Lebenswissenschaften spricht man von einer Datenexplosion, die mit modernen Methoden wie High-Troughput-Screenings zu erklären ist. Eine Unmenge an Daten kann mit diesen Methoden gewonnen werden, die in geeigneter Weise analysiert und zugänglich gemacht werden müssen. Die Anzahl von Einträgen in Sequenzdatenbanken steigt zum Beispiel rapide, da High-Throughput-Sequenzierungen sogar die schnelle Sequenzbestimmung ganzer Genome ermöglichen. Um diese Daten intelligent zugänglich zu machen, werden immer häufiger Ontologien eingesetzt. Ontologien, wie sie in dieser Arbeit verstanden werden, sind spezielle Datenstrukturen, in denen Wissen eines abgesteckten Fachgebietes in strukturierter Form gespeichert wird, sodass es von Computern interpretiert werden kann.

Eine Ontologie namens BIO2ME (*BioInformatics Ontology for Tools and Methods*) wurde aufgebaut und in dieser Arbeit erweitert, die bioinformatische Werkzeuge und Methoden strukturiert abbildet. Das Ziel ihrer Erstellung ist ihr Einsatz als Wissensbasis in einem Informationssystem, das eine intelligente Suche über die Vielzahl bioinformatischer Werkzeuge ermöglicht. Diese Werkzeuge werden zum einen für die Anzeige, Verarbeitung, Archivierung und Suche der beschriebenen Menge biomedizinischer Daten entwickelt, aber auch für die Unterstützung experimenteller Methoden und Simulation biologischer Vorgänge. Mittlerweile ist es daher eine zunehmende Herausforderung, ein geeignetes Werkzeug für eine bestimmte Aufgabe zu finden. Selbst Bioinformatikern fällt es schwer, den Überblick über Methoden und Werkzeuge ihres Fachgebietes zu behalten. In dieser Arbeit konnte ein solches Informationssystem erfolgreich implementiert werden, das die Semantik der Ontologie für die Suche ausnutzt und somit einzigartig für seine Anwendung ist.

Fachexperten spielen eine entscheidende Rolle vor allem in der informellen Phase des Ontologieaufbaus, in welcher das zu erfassende Wissen gesammelt und zueinander in Beziehung gesetzt wird. Obwohl die Forschung an Ontologien etabliert ist, konnte in dieser Arbeit festgestellt werden, dass für diese kritische Phase nur wenig Software-Unterstützung existiert. Um diesen Mangel zu kompensieren, wurde in dieser Arbeit der Wiki-Ansatz angepasst, der sich

bereits für die kooperative Erstellung von Dokumenten etabliert hat und sich dadurch im Besonderen für die gemeinschaftliche Arbeit der ersten Phasen der Ontologieerstellung qualifiziert. Das Wiki, welches im Rahmen dieser Arbeit in die Ontologie-Plattform ONTOVERSE integriert wurde, umfasst alle hilfreichen Wiki-Funktionen wie Versionierung und Änderungsaufzeichnung. Die Implementierung zeichnet sich jedoch besonders durch zwei zusätzliche Funktionalitäten aus: Zum einen können einzelne Abschnitte eines Wiki-Artikels gesperrt werden. Das ist zum Beispiel hilfreich, um die nachträgliche Änderung an festgelegten Ontologie-Kriterien nur für bestimmte Nutzergruppen zu erlauben. Zum anderen verfügt das Wiki über eine Verbindung zu der formalen Ontologie, welche die inhaltliche Evaluierung der Ontologie enorm erleichtert.

Weiterhin ist die Auffüllung von Ontologien mit Information eine umfassende Aufgabe und die Gewinnung von Fachexperten, die diese unterstützen, eine Herausforderung. Aus diesem Grund wurde in dieser Arbeit zusätzlich eine Methode zur semi-automatischen Unterstützung der Ontologieerweiterung mit Hilfe von Tagging implementiert. Die vorliegende Arbeit beinhaltet zudem eine umfassende Studie dieser Vorgehensweise, aufgrund derer die BIO2ME-Ontologie erheblich erweitert werden konnte.

Anhand einer Ontologie über bioinformatische Werkzeuge und ihrer Anwendung zeigt diese Arbeit den Nutzen des Einsatzes von Ontologien in der Wissenschaft auf und entwickelt Methoden, die dem gemeinschaftlichen Aufbau einer solchen Wissensbasis dienen und deren Erweiterung Computer-gestützt erleichern.

Bibliography

Alterovitz, G., Jiwaji, A. & Ramoni, M. F. (2008). Automated programming for bioinformatics algorithm deployment. *Bioinformatics*, 24(3), 450–451.

Andersen, E. S., Lind-Thomsen, A., Knudsen, B., Kristensen, S. E., Havgaard, J. H., Torarinsson, E., Larsen, N., Zwieb, C., Sestoft, P., Kjems, J. & Gorodkin, J. (2007). Semiautomated improvement of rna alignments. *RNA*, 13(11), 1850–1859.

Apke, S. & Dittmann, L. (2004). Generisches Vorgehensmodell KOWIEN Version 2.0. In *KOWIEN-Projektbericht 7/2004*. Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen),.

Aplin, J. D. & Singh, H. (2008). Bioinformatics and transcriptomics studies of early implantation. *Ann N Y Acad Sci*, 1127, 116–120.

Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Davis, A. P., Dolinski, K., Dwight, S. S., Eppig, J. T., Harris, M. A., Hil, D. P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J. C., Richardson, J. E., Ringwald, M., Rubin, G. M. & Sherlock, G. (2000). Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nat Genet*, 25(1), 25–29.

Baclawski, K. & Niu, T. (2006). *Ontologies for Bioinformatics*. The MIT Press.

Bai, F. & El Jerroudi, Z. (2008). Interactive and collaborative ontology developing. In *Workshop Proceedings der Tagungen Mensch & Computer 2008, DeLFI 2008 und Cognitive Design 2008*. Lucke, U. and Kindsmüller, C. and Fischer, S. and Herczeg, M. and Seehusen, S., pp. 174–179.

Baker, P. G., Brass, A., Bechhofer, S., Goble, C., Paton, N. & Stevens, R. (1998). Tambis–transparent access to multiple bioinformatics information sources. *Proc Int Conf Intell Syst Mol Biol*, 6, 25–34.

Bauer, M., Klau, G. W. & Reinert, K. (2007). Accurate multiple sequence-structure alignment of rna sequences using combinatorial optimization. *BMC Bioinformatics*, 8, 271.

Bendaña, Y. R. & Holmes, I. H. (2008). Colorstock, sscolor, ratón: Rna alignment visualization tools. *Bioinformatics*, 24(4), 579–580.

Benson, D. A., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J. & Wheeler, D. L. (2008). Genbank. *Nucleic Acids Res*, 36(Database issue), D25–D30.

Berners-Lee, T., Hendler, J. & Lassila, O. (2001). The semantic web: Scientific american. *Scientific American*, pp. 28–37.

Bodenreider, O. & Stevens, R. (2006). Bio-ontologies: current trends and future directions. *Brief Bioinform*, 7(3), 256–274.

Busch, A. & Backofen, R. (2006). Info-rna–a fast approach to inverse rna folding. *Bioinformatics*, 22(15), 1823–1831.

Carroll, H., Beckstead, W., O'Connor, T., Ebbert, M., Clement, M., Snell, Q & McClellan, D. (2007). Dna reference alignment benchmarks based on tertiary structure of encoded proteins. *Bioinformatics*, 23(19), 2648–2649.

Chang, T.-H., Horng, J.-T. & Huang, H.-D. (2008). RNALogo: a new approach to display structural RNA alignment. *Nucleic Acids Res*, 36(Web Server issue), W91–W96.

Chang, Y.-F., Huang, Y.-L. & Lu, C. L. (2008). SARSA: a web tool for structural alignment of RNA using a structural alphabet. *Nucleic Acids Res*, 36(Web Server issue), W19–W24.

Consortium, Gene Ontology (2001). Creating the gene ontology resource: design and implementation. *Genome Res*, 11(8), 1425–1433.

Consortium, Gene Ontology (2006). The gene ontology (go) project in 2006. *Nucleic Acids Res*, 34(Database issue), D322–D326.

Consortium, Gene Ontology (2008). The gene ontology project in 2008. *Nucleic Acids Res*, 36(Database issue), D440–D444.

Dalli, D., Wilm, A., Mainz, I. & Steger, G. (2006). Stral: progressive alignment of non-coding rna using base pairing probability vectors in quadratic time. *Bioinformatics*, 22(13), 1593–1599.

Davidson, S. B., Overton, C. & Buneman, P. (1995). Challenges in integrating biological data sources. *J Comput Biol*, 2(4), 557–572.

De Roure, D., Jennings, N.R. & Shadbolt, N.R. (2003). *Grid Computing: Making the Global Infrastructure Reality.*, chapter The Semantic Grid: A future e-Science infrastructure.

Ebersbach, A., Glaser, M. & Heigl, R. (2008). *Social Web*. Uni-Taschenbücher.

Fernandez, O. (2005). Deep integration of ruby with semantic web ontologies.

Fernández López, M. (2001). Overview of methodologies for building ontologies. In *Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods (KRR5)*. Stockholm, Sweden.

Freeman, E., Hupfer, S. & Arnold, K. (1999). *Javaspaces Principles, Patterns, and Practice: Principles, Patterns and Practices*. Addison-Wesley Longman, Amsterdam.

Garrett, J. J. (2005). Ajax: A new approach to web applications. http://www.adaptivepath.com/ideas/essays/archives/000385.php.

Gelernter, D. (1985). Generative communication in linda. *ACM Trans. Program. Lang. Syst.*, 7(1), 80–112.

Gevorgyan, A., Poolman, M. G. & Fell, D. A. (2008). Detection of stoichiometric inconsistencies in biomolecular models. *Bioinformatics*, 24(19), 2245–2251.

Goble, C., Corcho, O., Alper, P. & De Roure, D. (2006). *Discovery Science*, chapter e-Science and the Semantic Web: A Symbiotic Relationship, pp. 1–12. Heidelberg: Springer.

Gómez-Pérez, A., Fernández, M. & de Vicente, A. J. (2004). Towards a method to conceptualize domain ontologies.

Gruber, T. (2005). Ontology of folksonomy: A mash-up of apples and oranges.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2), 199–220.

Grüninger, M. & Fox, M. S. (1995). Methodology for the design and evaluation of ontologies. In *IJCAI'95, Workshop on Basic Ontological Issues in Knowledge Sharing, April 13, 1995*.

Hey, T. & Trefethen, A. E. (2005). Cyberinfrastructure for e-science. *Science*, 308(5723), 817–821.

Hofacker, I. L., Bernhart, S. H. F. & Stadler, P. F. (2004). Alignment of rna base pairing probability matrices. *Bioinformatics*, 20(14), 2222–2227.

Holmes, I. (2005). Accelerated probabilistic inference of rna structure evolution. *BMC Bioinformatics*, 6, 73.

Jurafsky, D. & Martin, J. H. (2008). *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics. 2nd edition.* Prentice-Hall, 2 edition.

Karp, P. D. (1995). A strategy for database interoperation. *J Comput Biol*, 2(4), 573–586.

Katoh, K., Misawa, K., Kuma, K. & Miyata, T. (2002). Mafft: a novel method for rapid multiple sequence alignment based on fast fourier transform. *Nucleic Acids Res*, 30(14), 3059–3066.

Katzman, S., Barrett, C., Thiltgen, G., Karchin, R. & Karplus, K. (2008). Predict-2nd: a tool for generalized protein local structure prediction. *Bioinformatics*.

Keseler, I. M., Collado-Vides, J., Gama-Castro, S., Ingraham, J., Paley, S., Paulsen, I. T., Peralta-Gil, M. & Karp, P. D. (2005). Ecocyc: a comprehensive database resource for escherichia coli. *Nucleic Acids Res*, 33(Database issue), D334–D337.

Khaladkar, M., Bellofatto, V., Wang, J. T. L., Tian, B. & Shapiro, B. A. (2007). Radar: a web server for rna data analysis and research. *Nucleic Acids Res*, 35(Web Server issue), W300–W304.

Kruspe, M. & Stadler, P. F. (2007). Progressive multiple sequence alignments from triplets. *BMC Bioinformatics*, 8, 254.

Lassila, O. & McGuinness, D. L. (2001). The role of frame-based representation on the semantic web. Technical Report KSL-01-02, Knowledge Systems Laboratory, Stanford University. Stanford, California.

Leuf, B. & Cunningham, W. (2001). *The Wiki way: quick collaboration on the Web*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Lindgreen, S., Gardner, P. P. & Krogh, A. (2007). Mastr: multiple alignment and structure prediction of non-coding rnas using simulated annealing. *Bioinformatics*, 23(24), 3304–3311.

Mainz, D. (2008). *Deep integration of the OWL ontology language into Ruby using metaprogramming*. Dissertation, Heinrich-Heine-Universität Düsseldorf.

Mainz, I. (2006). Development of a prototype ontology for bioinformatics tools. (in german). Bachelor thesis, Heinrich-Heine-Universität Düsseldorf.

Mainz, I. (2006). Statistik von RNA-Struktur-Alignments. Diplom thesis, Heinrich-Heine-Universität Düsseldorf.

Mainz, I., Weller, K., Paulsen, I., Mainz, D., J. & von Haeseler, A. (2008). Ontoverse: Collaborative ontology engineering for the life sciences. *Information Wissenschaft & Praxis*, 2, 91–99.

Malzahn, N., Weinbrenner, S., Hüsken, P., Ziegler, J. & Hoppe, H. U. (2007). Collaborative ontology development - distributed architecture and visualization. In *Proceedings of the German E-Science Conference*. Max Planck Digital Library. Open-Archive-Publikation.

Mathes, A. (2004). Folksonomies: Cooperative Classification and Communication Through Shared Metadata. http://adammathes.com/academic/computer-mediated-communication/folksonomies.html.

McCray, A. T. (2006). Conceptualizing the world: lessons from history. *J Biomed Inform*, 39(3), 267–273.

Meyer, I. M. & Miklós, I. (2007). Simulfold: simultaneously inferring rna structures including pseudoknots, alignments, and trees using a bayesian mcmc framework. *PLoS Comput Biol*, 3(8), e149.

Moretti, S., A., Wilm, G., Higgins D., I., Xenarios & Notredame, C. (2008). R-Coffee: a web server for accurately aligning noncoding RNA sequences. *Nucl. Acids Res.*, **36** Web Server Issue, W10–3.

Paulsen, I. (2007). *Collaborative Knowledge Management in the Life Sciences Network*. PhD thesis, Heinrich-Heine-Universität Düsseldorf.

Paulsen, I., Mainz, D., Weller, K., Mainz, I., Kohl, J. & von Haeseler, A. (2007). Ontoverse: Collaborative knowledge management in the life sciences network. In *Germany eScience Conference 2007, Max Planck Digital Library, ID 316588.0.*

Pei, J. & Grishin, N. V. (2007). Promals: towards accurate multiple sequence alignments of distantly related proteins. *Bioinformatics*, 23(7), 802–808.

Peters, I. & Stock, W. G. (2007). Folksonomy and information retrieval. In *In Proceedings of the 70th Annual Meeting of the American Society for Information Science and Technology.*, volume 45. pp. 1510–1542. CD-ROM.

Siddharthan, R. (2006). Sigma: multiple alignment of weakly-conserved non-coding dna sequence. *BMC Bioinformatics*, 7, 143.

Siebert, S. & Backofen, R. (2005). MARNA: multiple alignment and consensus structure prediction of RNAs based on sequence structure comparisons. *Bioinformatics*, **21**(16), 3352–3359.

Simossis, V. A. & Heringa, J. (2005). Praline: a multiple sequence alignment toolbox that integrates homology-extended and secondary structure information. *Nucleic Acids Res*, 33(Web Server issue), W289–W294.

Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L. J., Eilbeck, K., Ireland, A., Mungall, C. J., Consortium, O. B. I., Leontis, N., Rocca-Serra, P., Ruttenberg, A., Sansone, S.-A., Scheuermann, R. H., Shah, N., Whetzel, P. L. & Lewis, S. (2007). The obo foundry: coordinated evolution of ontologies to support biomedical data integration. *Nat Biotechnol*, 25(11), 1251–1255.

Stevens, R., Goble, C. A. & Bechhofer, S. (2000). Ontology-based knowledge representation for bioinformatics. *Brief Bioinform*, 1(4), 398–414.

Stocsits, R. R., Hofacker, I. L., Fried C. & Stadler, P. F. (2005). Multiple sequence alignments of partially coding nucleic acid sequences. *BMC Bioinformatics*, (6), 160.

Sure, Y. (2002). A tool-supported methodology for ontology-based knowledge management.

Sure, Y., Erdmann, M., Angele, J., Staab, S., Studer, R. & Wenke, D. (2002). Ontoedit: Collaborative ontology engineering for the semantic web. In *In Proceedings of the first International Semantic Web Conference 2002 (ISWC 2002)*. Sardinia, Italy.

Thompson, J. D., Higgins, D. G. & Gibson, T. J. (1994). CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucl. Acids Res.*, **22**, 4673–4680.

Thompson, J. D., Koehl, P., Ripp, R. & Poch, O. (2005). Balibase 3.0: latest developments of the multiple sequence alignment benchmark. *Proteins*, 61(1), 127–136.

Thompson, J. D., Muller, A., Waterhouse, A., Procter, J., Barton, G. J., Plewniak, F. & Poch, O. (2006). MACSIMS: multiple alignment of complete sequences information management system. *BMC Bioinformatics*, 7, 318.

Valdivia-Granda, W. (2008). The next meta-challenge for Bioinformatics. *Bioinformation*, 2(8), 358–362.

Veeramalai, M., Ye, Y. & Godzik, A. (2008). Tops++fatcat: fast flexible structural alignment using constraints derived from tops+ strings model. *BMC Bioinformatics*, 9(1), 358.

Wilm, A., Higgins, D. G. & Notredame, C. (2008). R-Coffee: a method for multiple alignment of non-coding RNA. *Nucl. Acids Res.*

Wilm, A., Mainz, I. & Steger, G. (2006). An enhanced rna alignment benchmark for sequence alignment programs. *Algorithms Mol Biol*, 1, 19.

Wolf, M., Ruderisch, B., Dandekar, T., Schultz, J. & Müller, T. (2008). Profdists: (profile-) distance based phylogeny on sequence - structure alignments. *Bioinformatics*.

Wyckoff, P., McLaughry, S. W., Lehman, T. J. & Ford, D. A. (1998). T spaces. *IBM Systems Journal*, 37(3).