OntoViBe: An Ontology Visualization Benchmark

Florian Haag[1], Steffen Lohmann[1], Stefan Negru[2], and Thomas Ertl[1]

[1]Institute for Visualization and Interactive Systems, University of Stuttgart,
Universitätsstraße 38, 70569 Stuttgart, Germany
{Florian.Haag,Steffen.Lohmann,Thomas.Ertl}@vis.uni-stuttgart.de
[2]Faculty of Computer Science, Alexandru Ioan Cuza University,
Strada General Henri Mathias Berthelot 16, 700483 Iasi, Romania
stefan.negru@info.uaic.ro

Abstract. A variety of ontology visualizations have been presented in the last couple of years. The features of these visualizations often need to be tested during their development or for evaluation purposes. However, in particular for the testing of special concepts and combinations thereof, it can be difficult to find suitable ontologies. We have developed OntoViBe, an ontology covering a wide variety of OWL 2 language constructs for the purpose of testing ontology visualizations. We describe the design principles underlying OntoViBe and present the supported features in coverage matrices. Finally, we load OntoViBe with ontology visualization tools and point to some noteworthy aspects of the respective visualizations that become apparent and demonstrate how OntoViBe can be used for testing ontology visualizations.

Keywords: Ontology, visualization, benchmark, evaluation, OWL.

1 Introduction

Developing and working with ontologies can be supported by ontology visualizations. Over the past years, a number of visualization approaches geared towards the peculiarities of ontologies have been proposed. Most of the available approaches use node-link diagrams to depict the graph structure of ontologies, while some apply other diagram types like treemaps or nested circles [7,9,11].

During the development of such ontology visualizations, testing with a variety of existing ontologies is required to ensure that the concepts from the underlying ontology language are adequately represented. The same needs to be done to determine the features of an ontology visualization and get an impression of how different ontology language constructs are visually represented. Still, repeatedly loading a set of ontologies that cover a wide variety of language constructs can be a tedious task, even more so as the most common constructs tend to appear over and over in each of the tested ontologies. In order to help that process with respect to ontologies based on the OWL 2 Web Ontology Language, we developed OntoViBe, an Ontology Visualization Benchmark.

Basically, OntoViBe is an ontology that has been designed to incorporate a comprehensive set of OWL 2 language constructs and systematic combinations thereof. While it is oriented towards OWL 2, it also includes the concepts of OWL 1 due to the complete backwards compatibility of the two ontology languages, i.e., all OWL 1 ontologies are valid OWL 2 ontologies [17].

As opposed to most other benchmarks found in the computing world, OntoViBe is not meant for testing the scalability of visualizations with respect to the number of elements contained in ontologies, but rather aims for the scope of visualizations in terms of supported features. Related to this, it focuses on the representation of what is usually called the TBox of ontologies (i.e., the classes, properties, and datatypes), while it does not support the testing of ABox information (i.e., individuals and data values), which is the focus of most related work.

2 Related Work

Several benchmarks for ontology tools have been developed in the past. One well-known benchmark in this area is the Lehigh University Benchmark (LUBM), published by the SWAT research group of Lehigh University [8]. It consist of three components: 1) an ontology of moderate size and complexity describing concepts and relationships from the university domain, 2) a generator for random and repeatable instance data that can be scaled to an arbitrary size, and 3) a set of test queries for the instance data as well as performance metrics.

Since the LUBM benchmark is bound to the university domain, the SWAT research group developed another benchmark that can be tailored to different domains [18]. It uses a probabilistic model to generate an arbitrary number of instances based on representative data from the domain in focus. As an example, the Lehigh BibTeX Benchmark (LBBM) has been created with the probabilistic model and a BibTex ontology. Another extension of LUBM has been proposed with the University Ontology Benchmark (UOBM) [12]. UOBM aims to include the complete set of OWL 1 language constructs and defines two ontologies, one being compliant with OWL Lite and the other with OWL DL. Furthermore, it adds several links to the generated instance data and provides related test cases for reasoners.

All these benchmarks focus primarily on performance, efficiency, and scalability, but do not address the visual representation of ontologies. Furthermore, they are mainly oriented towards instance data (the ABox), while systematic combinations of classes, properties, and datatypes (the TBox) are not further considered. Even though UOBM provides comparatively complete TBox information, it has been designed to test OWL reasoners and not ontology visualizations. This is also the case for JustBench [4], which uses small and clearly defined ontology subsets to evaluate the behavior of OWL reasoners.

There are also some benchmarks addressing specific aspects of ontology engineering. A number of datasets and test cases emerged, for instance, as part of the Ontology Alignment Evaluation Initiative (OAEI) [1]. A related dataset

has been created in the OntoFarm project, which provides a collection of ontologies for the task of of testing and comparing different ontology alignment methods [16]. An extension of the OntoFarm idea is the MultiFarm project, which offers ontologies translated into different languages with corresponding alignments between them [13]. Overall, the test cases are intended to evaluate and compare the quality and performance of matching algorithms, in the latter case with a special focus on multilingualism.

The W3C Web Ontology Working Group has also developed test cases for OWL 1 [6] and OWL 2 [15]. They are meant to provide examples for the normative definition of OWL and can, for instance, be used to perform conformance checks. However, there is not yet any benchmark particularly addressing the visualization of ontologies to the best of our knowledge. To close this gap, we developed OntoViBe, which will be described in the following.

3 Ontology Visualization Benchmark (OntoViBe)

The structure and content of OntoViBe is based on the OWL 2 specifications [17], with the following requirements:

- A wide variety of OWL 2 language constructs must appear. This includes constructs such as class definitions or different kinds of properties, as well as modifiers for these, such as *deprecated* markers.
- Subgraphs that represent compound concepts must appear. This includes small groups of classes that are coupled by a particular property.

Moreover, we tried to keep the overall ontology as small as possible in number of elements. Like this, rather than a mere enumeration of the elements and concepts supported by OWL 2, chances are that the ontology can be completely displayed and grasped "at a single glance" and thereby convey a complete impression of the features supported by the visualization being examined.

OntoViBe was assembled by creating an instance of each of the subgraph structures. Where possible, classes were reused to keep the ontology small. For instance, to include the OWL element *object property*, a subgraph structure consisting of two classes connected by an object property was added. Hence, two classes were inserted into the ontology, and an object property that uses either of the two classes as its domain and range, respectively, was defined. Furthermore, the element *datatype property* needed to appear in the ontology. A compact subgraph structure to express that element consists of a class linked to a datatype property. As the class does not need to have any specific characteristics of its own, one of the two previously inserted classes could be reused.

After that, some of the existing elements were modified to cover all elements and features that we wanted to consider at least once in the ontology. For example, some properties were declared as *functional* or *deprecated*. For any element type that still did not appear in the ontology, a minimal number of extra classes were added (the addition of extra properties and datatypes was not necessary).

Listing 1.1. Concepts based upon set operators are featured in two variants, a small set with two elements, and a larger one with more elements.

```
30 this:UnionClass a owl:Class ;
31   owl:unionOf ( this:Class1 this:DeprecatedClass ).
32
33 this:LargeUnionClass a owl:Class ;
34   owl:unionOf ( this:UnionClass other:ImportedClass this:PropertyOwner ).
```

Listing 1.2. OntoViBe defines custom OWL data ranges.

```
94 this:DivisibleByFiveEnumeration a rdfs:Datatype ;
95   owl:equivalentClass [
96     a rdfs:Datatype ;
97     owl:oneOf ( 5 10 15 20 )
98   ].
99
100 this:UnionDatatype a rdfs:Datatype ;
101   owl:unionOf ( this:DivisibleByTwoEnumeration this:
        DivisibleByFiveEnumeration ).
```

Lastly, all elements in the ontology were named in a self-descriptive manner to allow for an easier interpretation and analysis. For instance, a deprecated class is called `DeprecatedClass`, while the larger of the union classes is called `LargeUnionClass`.

3.1 Exemplary Parts of OntoViBe

Many of the structures could be added in a straightforward way. In some cases, further considerations were required to adequately address the more flexible features of OWL.

Concepts defined based upon set operators (*unionOf*, *intersectionOf*, *complementOf*) come in two variants. One of them uses a set comprising two elements as an example for a small set, while the other features more set elements, usually three (Listing 1.1).

OntoViBe also includes OWL data ranges (Listing 1.2). Visualizations may or may not represent the exact definitions of these data ranges, but even if they do not, support for datatype properties with custom data ranges needs to be tested. Therefore, custom data ranges are used by some datatype properties in OntoViBe, while common datatypes are used for most other properties (Listing 1.3).

In order to check how imported ontology elements are treated, OntoViBe consists of two components. The core ontology[1] contains most of the definitions, but a few classes, properties, and datatypes are defined in an additional module[2], whose content is imported into the core ontology (Listing 1.4).

[1] `http://ontovibe.visualdataweb.org/1.0#`

[2] `http://ontovibe.visualdataweb.org/1.0/imported#`

Listing 1.3. Both custom and common datatypes are used by properties.

```
this:standardTypeDatatypeProperty a owl:DatatypeProperty ;
  rdfs:domain this:PropertyOwner ;
  rdfs:range xsd:integer .

this:customTypeDatatypeProperty a owl:DatatypeProperty ;
  rdfs:domain this:PropertyOwner ;
  rdfs:range this:DivisibleByFiveEnumeration .
```

Listing 1.4. Some of the definitions are imported from a separate ontology module.

```
<http://ontovibe.visualdataweb.org/1.0#> a owl:Ontology ;
  owl:versionIRI <http://ontovibe.visualdataweb.org/1.0#> ;
  owl:imports <http://ontovibe.visualdataweb.org/1.0/imported#> ;
  <http://purl.org/dc/elements/1.1/title> "Ontology Visualization Benchmark
       (OntoViBe)" .
```

Listing 1.5. Sets of properties connected to the same classes allow for testing whether a visualization positions such properties in a non-overlapping way. This example shows two cyclic properties connected to the same class.

```
this:cyclicProperty2 a owl:ReflexiveProperty ;
  rdfs:domain this:MultiPropertyOwner ;
  rdfs:range this:MultiPropertyOwner .

this:cyclicProperty3 a owl:ObjectProperty ;
  rdfs:domain this:MultiPropertyOwner ;
  rdfs:range this:MultiPropertyOwner .
```

In ontology visualizations, sets of properties between the same pair of classes (or the same class and literal) pose a particular challenge, as they may lead to overlapping and thus illegible representations. Several of these cases have been considered in OntoViBe. For instance, Listing 1.5 shows two cyclic properties (i.e., properties whose domain and range are identical) connected to the same class.

Finally, a few of the ontology elements are provided with labels, to check how visualizations cope with multilingual labels that may also contain non-ASCII characters (Listing 1.6). For all non-ASCII characters, the escaped ASCII representation is used in the ontology file, as that maximizes the chances for a good compatibility with the parser reading the file.

3.2 Verification of Coverage and Omissions

To verify that OntoViBe covers most of the features defined by the OWL 2 specifications, we provide two coverage matrices. Table 1 juxtaposes the elements of OntoViBe with systematically listed OWL 2 features as described in the specifications. Table 2 shows which OntoViBe elements use which concrete OWL 2 identifiers, as per the IRIs declared in the OWL 2 Namespace Document [3].

Listing 1.6. Multilingual labels, some of which contain characters from different scripts, exist for a few of the ontology elements.

```
this:importedTypeDatatypeProperty a owl:DatatypeProperty ;
  rdfs:domain this:PropertyOwner ;
  rdfs:range other:DivisibleByThreeEnumeration ;
  rdfs:label "imported type datatype property"@en ;
  rdfs:label "propri\u00E9t\u00E9 d'un type de donn\u00E9es import\u00E9"
      @fr ;
  rdfs:label "\u4E00\u79CD\u5BFC\u5165\u7C7B\u578B\u7684\u6570\u636E\u7C7B\
      u578B\u6027\u8D28"@zh-Hans .
```

The tables also reveal some parts of OWL that are intentionally not included in OntoViBe:

Cardinalities: OntoViBe defines only a few cases of cardinality constraints for properties: Structurally, these can be distinguished as *no cardinality*, *cardinality on one end of a property relation* and *cardinality on both ends of a property relation*. Regarding the concrete cardinality constraints applied, exact cardinality, a minimum cardinality, and a combination of a minimum and a maximum cardinality are included in OntoViBe. Moreover, one of the cardinality constraints is qualified and thus applies only to instances of a specific class. These cases can thus only be used for checking whether cardinalities are displayed at all.
We have opted against integrating all supported cardinalities in OntoViBe, as the number of possible combinations would be considerable—in particular, when considering that "special values" such as zero and one might be displayed in special ways. The total number of properties in OntoViBe would have to be increased while providing only minor additional insight into the tested visualization.

Annotations: Informative metadata has no effect on the conceptual structure of an ontology, which is focused in OntoViBe. For that reason, only the most prevalent metadata attributes, such as labels or the ontology title, have been integrated into OntoViBe.

Equivalent constructs: In cases of conceptually equivalent ways to express statements in OWL, only one way was integrated into OntoViBe. For instance, deprecation of ontology elements can either be expressed by adding the `owl:deprecated` attribute or by declaring the element as belonging to one of the classes `owl:DeprecatedClass` or `owl:DeprecatedProperty`.

Deprecated elements: Deprecated language constructs of OWL itself are not used in OntoViBe. An example is `owl:DataRange` that has been deprecated as of OWL 2 in favor of `rdfs:Datatype` [3].

Moreover, statements referring to particular individuals have not been included, as OntoViBe focuses on visualizations of the TBox of ontologies.

4 Examples of Application

In the following, we demonstrate the usefulness of OntoViBe by applying four ontology visualizations to it: SOVA, VOWL, OWLViz, and OntoGraf. The latter two come with the default installation of the popular ontology editor Protégé [2] (desktop version 5.0), while the first two are comparatively well-specified with regard to the visual elements they are based on. Moreover, we analyze the ontology documentation generated for OntoViBe by the Live OWL Documentation Environment (LODE).

We present screenshots of all these ontology visualizations that give an impression of the supported features. We point out peculiarities of the visualization approaches and their implementations that become apparent based on OntoViBe. By this, we would like to provide some examples of how to use OntoViBe for the qualitative analysis of ontology representations, and to confirm that such an analysis is feasible by using OntoViBe.

It should be noted that a comprehensive analysis of ontology visualizations requires additional methods, such as a checklist comprising further evaluation criteria. These methods are typically not generic but tailored to the type of visualization. For instance, measures for graph visualizations of ontologies could include the total number of edges and edge crossings. However, such additional measures are outside the scope of this work.

4.1 SOVA

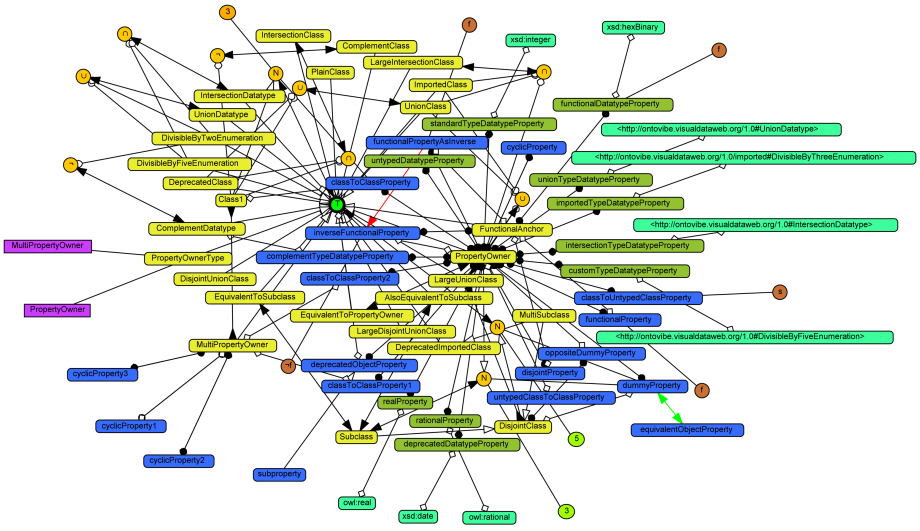

Fig. 1. OntoViBe visualized with SOVA.

SOVA is a plugin for Protégé that provides graph visualizations of ontologies [5]. When displaying OntoViBe in SOVA 0.8.4 (Figure 1), the distinction of classes, object and datatype properties is instantly visible—though relying purely on colors. Based on OntoViBe, support for functional, inverse functional, and symmetric properties can be seen, as they are marked by little brown circles with short abbreviations for the property characteristics.

Furthermore, some of the limitations of the SOVA implementation can be identified. `PropertyOwner` and `MultiPropertyOwner` are classes, but at the same time, they are instances of the class `PropertyOwnerType`. SOVA displays them twice, once as classes and once as individuals, rather than as a single concept. Cardinality constraints are displayed as small colored circles, which are easy to spot—for a combined minimum and maximum cardinality constraint, however, only the lower bound (the number 5 in the green circle) is shown. Overall, the visualization contains many edges and edge crossings, which significantly reduce its readability. A large number of these edges result from the fact that all implicit subclass relations to `owl:Thing` are depicted in the SOVA visualization, and that every piece of information is shown in a separate node.

4.2 VOWL

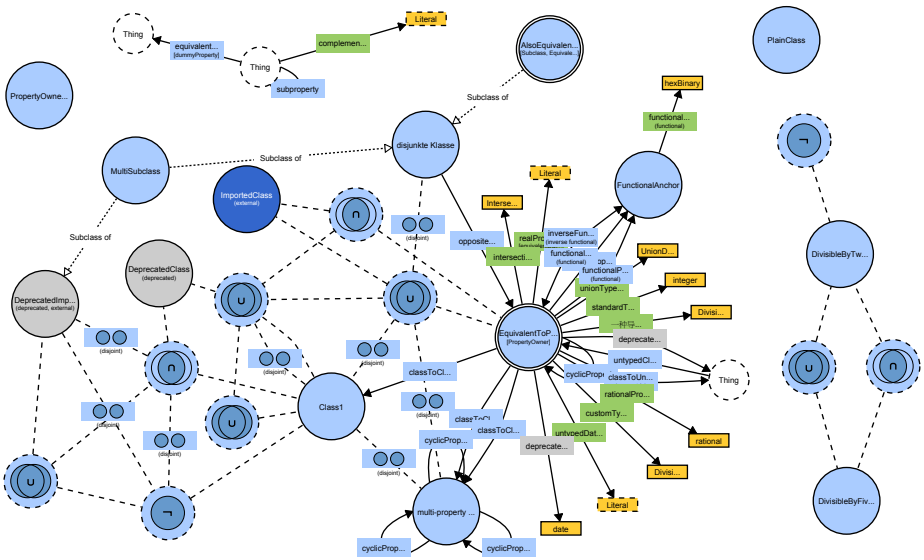

Fig. 2. OntoViBe visualized with VOWL.

VOWL, the Visual Notation for OWL Ontologies, was developed as a means to both obtain a structural overview of OWL ontologies and recognize various attributes of ontology elements at a glance [11]. It has been implemented

in two different tools, a plugin for Protégé and a responsive web application. Figure 2 has been created with version 0.2.15 of the web application (called WebVOWL [10]) that is available at `http://vowl.visualdataweb.org`.

Applying VOWL to OntoViBe shows some typical characteristics of VOWL visualizations, such as equivalent classes being represented as one class with a double border, other special elements being multiplied in the visualization (e.g., `owl:Thing`), or text in brackets below the labels indicating attributes such as *functional* or *symmetric*. Like in SOVA, custom data ranges are not (yet) completely shown, as is seen by the respective nodes that simply display the names of the data ranges (e.g., "Divisib..." for "DivisibleByFiveEnumeration") but no more information on how they are defined. Moreover, it becomes apparent that the WebVOWL implementation tends to route edges between the same pairs of nodes in a way so as to avoid overlapping labels. As the implicit subclass relations to `owl:Thing` are not shown in VOWL, the nodes of the graph visualization are less connected than in SOVA.

4.3 OWLViz

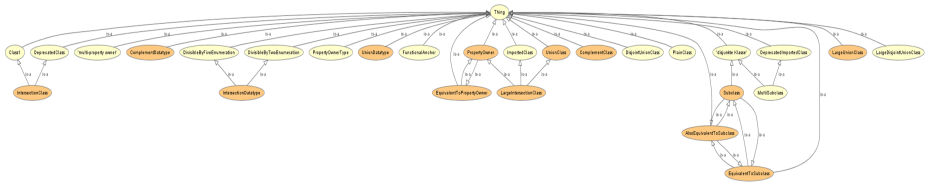

Fig. 3. OntoViBe visualized with OWLViz.

The OWLViz visualization is aimed at visualizing exclusively the hierarchical class structure of ontologies. When used to visualize OntoViBe (Figure 3), the fact that only inheritance ("is-a") relationships are shown by OWLViz 4.1.2, gets apparent. Also, it gets clear that equivalence relationships between classes are expressed as bidirectional inheritance. Other property relations are not visualized in OWLViz, and also further class or property characteristics are not included, making OWLViz an ontology visualization with very limited expressiveness.

4.4 OntoGraf

OntoGraf (Figure 4) depicts property relations between classes with colored lines. Given that OntoViBe includes properties of different types and with different characteristics, it is notable that these are not displayed by OntoGraf 1.0.1 in an inherently distinct way. Again, the graph is highly connected, as the implicit subclass relations to `owl:Thing` are explicitly shown.

Like in OWLViz (Section 4.3), equivalence between classes is displayed by two opposite inheritance arrows. Additionally, classes that are equivalent to others

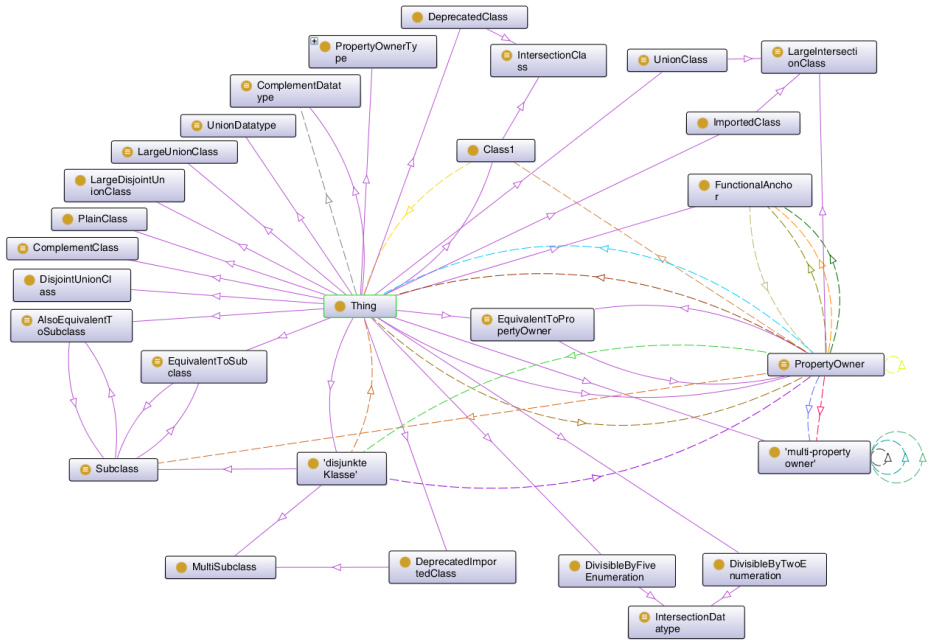

Fig. 4. OntoViBe visualized with OntoGraf.

are highlighted by an equivalence symbol. Other than that, the test shows that OntoGraf copes well with several cyclic properties applied to the same class.

4.5 LODE

LODE is a documentation generator for ontologies [14]. While LODE is not a visualization approach, the output of version 1.2 after processing OntoViBe can be examined in a similar fashion. Features that get apparent in the excerpt shown in Figure 5 include the transformation of the camel-cased element names into separate words (e.g., `DeprecatedClass` becomes *deprecated class*), and the lists of superclasses, subclasses, and connected properties per class.

5 Conclusion and Future Work

Based on the OWL 2 specifications, we have defined OntoViBe, a benchmark ontology for testing ontology visualizations. We did not focus on scalability or other common benchmark goals, such as execution speed, but rather on feature completeness and flexibility in terms of combination of elements with regard to the OWL 2 specifications. Since OWL may further evolve in the future, OntoViBe needs to keep being updated accordingly.

Features not included in OntoViBe may be considered for future adjuncts of the ontology. For instance, these could be separate modules that focus on

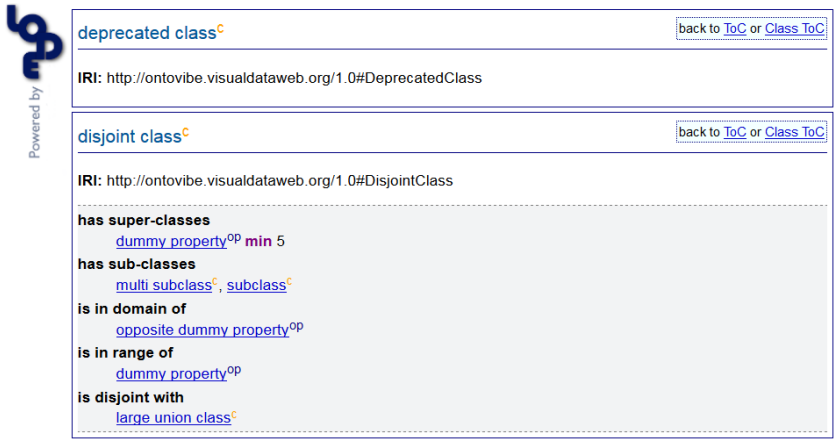

Fig. 5. Excerpt of the HTML documentation for OntoViBe generated by LODE.

testing specific aspects, such as combinations of cardinality constraints or the population of OntoViBe with individuals and other ABox information.

Generally, we hope that our experiences from the development of OntoViBe can benefit other projects, including benchmark data models beyond the task of ontology visualization.

References

1. Ontology alignment evaluation initiative. `http://oaei.ontologymatching.org`
2. Protégé ontology editor. `http://protege.stanford.edu`
3. The OWL 2 schema vocabulary (OWL 2). `http://www.w3.org/2002/07/owl.rdf` (2009)
4. Bail, S., Parsia, B., Sattler, U.: JustBench: A framework for OWL benchmarking. In: Proceedings of the 9th International Semantic Web Conference (ISWC '10), pp. 32–47. Springer (2010)
5. Boinski, T., Jaworska, A., Kleczkowski, R., Kunowski, P.: Ontology visualization. In: Proceedings of the 2nd International Conference on Information Technology (ICIT '10). pp. 17–20. IEEE (2010)
6. Carroll, J.J., Roo, J.D.: OWL web ontology language test cases. `http://www.w3.org/TR/owl-test/` (2004)
7. Dudáš, M., Zamazal, O., Svátek, V.: Roadmapping and navigating in the ontology visualization landscape. In: Proceedings of the 19th International Conference on Knowledge Engineering and Knowledge Management (EKAW '14), pp. 137–152. Springer (2014)
8. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. Web Semantics 3(2–3), 158–182 (2005)
9. Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., Giannopoulou, E.: Ontology visualization methods – a survey. ACM Computing Surveys 39(4), 10:1–10:43 (2007)

10. Lohmann, S., Link, V., Marbach, E., Negru, S.: WebVOWL: Web-based visualization of ontologies. In: Proceedings of EKAW 2014 Satellite Events. Springer (to appear)
11. Lohmann, S., Negru, S., Haag, F., Ertl, T.: VOWL 2: User-oriented visualization of ontologies. In: Proceedings of the 19th International Conference on Knowledge Engineering and Knowledge Management (EKAW '14), pp. 266–281. Springer (2014)
12. Ma, L., Yang, Y., Qiu, Z., Xie, G., Pan, Y., Liu, S.: Towards a complete OWL ontology benchmark. In: Proceedings of the 3rd European Semantic Web Conference (ESWC '06), pp. 125–139. Springer (2006)
13. Meilicke, C., García-Castro, R., Freitas, F., Van Hage, W.R., Montiel-Ponsoda, E., Ribeiro De Azevedo, R., Stuckenschmidt, H., Šváb Zamazal, O., Svátek, V., Tamilin, A., Trojahn, C., Wang, S.: MultiFarm: A benchmark for multilingual ontology matching. Web Semantics 15, 62–68 (2012)
14. Peroni, S., Shotton, D., Vitali, F.: The live OWL documentation environment: A tool for the automatic generation of ontology documentation. In: Proceedings of the 18th International Conference on Knowledge Engineering and Knowledge Management (EKAW '12). pp. 398–412. Springer (2012)
15. Smith, M., Horrocks, I., Krtzsch, M., Glimm, B.: OWL 2 web ontology language conformance (second edition). `http://www.w3.org/TR/owl2-conformance/` (2012)
16. Šváb, O., Svátek, V., Berka, P., Rak, D., Tomášek, P.: OntoFarm: Towards an experimental collection of parallel ontologies. In: Poster Track of ISWC 2005 (2005)
17. W3C OWL Working Group: OWL 2 web ontology language document overview (second edition). `http://www.w3.org/TR/owl2-overview/` (2012)
18. Wang, S.Y., Guo, Y., Qasem, A., Heflin, J.: Rapid benchmarking for semantic web knowledge base systems. In: Proceedings of the 4th International Semantic Web Conference (ISWC '06), pp. 758–772. Springer (2005)

Table 1. Coverage table of OntoViBe elements with respect to OWL 2 features.

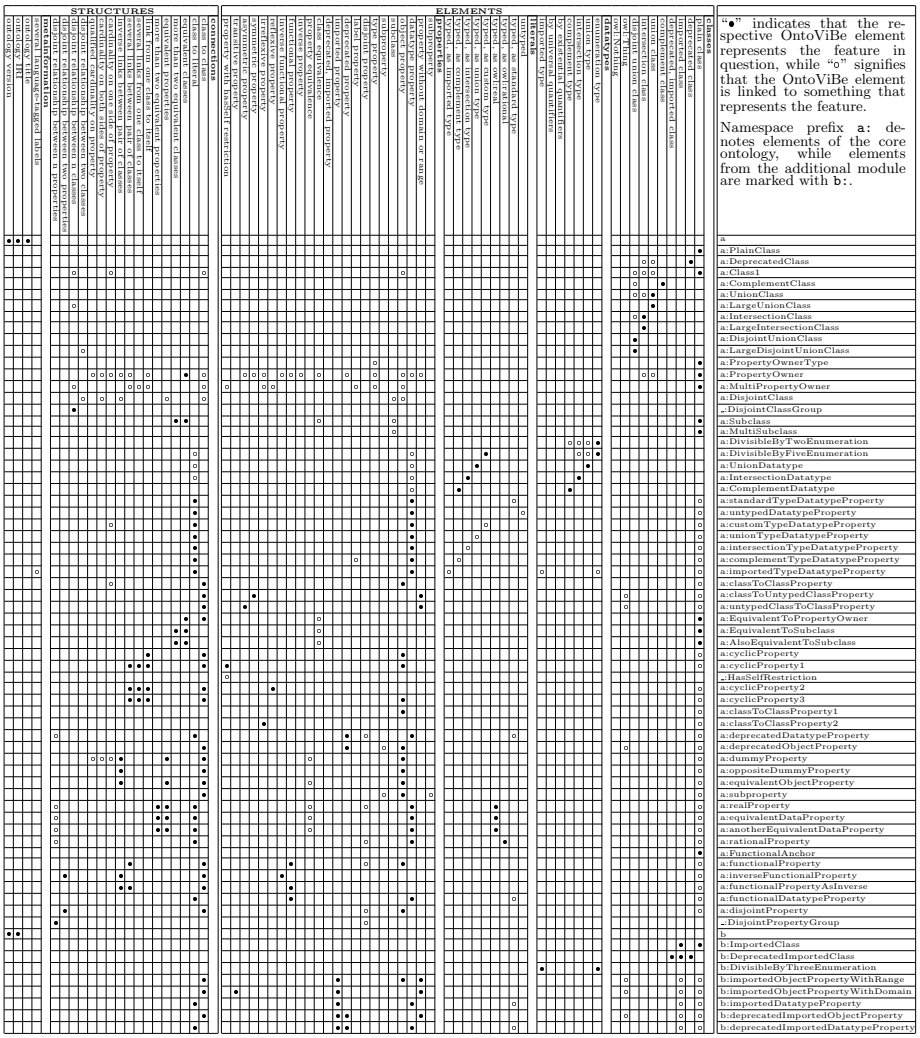

Table 2. Coverage table of OntoViBe elements with respect to OWL 2 identifiers.