Visualizing Ontologies with VOWL

Editor(s): Krzysztof Janowicz, University of California, Santa Barbara, US; Stefan Schlobach, Vrije Universiteit Amsterdam, The Netherlands
Solicited review(s): Luca Gilardoni, Quinary; 2 anonymous reviewers

Steffen Lohmann [a,*], Stefan Negru [b,**], Florian Haag [a], Thomas Ertl [a]
[a] *Institute for Visualization and Interactive Systems (VIS), University of Stuttgart, Universitätsstraße 38, 70569 Stuttgart, Germany*
E-Mail: {steffen.lohmann, florian.haag, thomas.ertl}@vis.uni-stuttgart.de
[b] *Faculty of Computer Science, Alexandru Ioan Cuza University, Strada General Henri Mathias Berthelot 16, 700483 Iasi, Romania*
E-Mail: stefan.negru@info.uaic.ro

Abstract. The Visual Notation for OWL Ontologies (VOWL) is a well-specified visual language for the user-oriented representation of ontologies. It defines graphical depictions for most elements of the Web Ontology Language (OWL) that are combined to a force-directed graph layout visualizing the ontology. In contrast to related work, VOWL aims for an intuitive and comprehensive representation that is also understandable to users less familiar with ontologies. This article presents VOWL in detail and describes its implementation in two different tools: ProtégéVOWL and WebVOWL. The first is a plugin for the ontology editor Protégé, the second a standalone web application. Both tools demonstrate the applicability of VOWL by means of various ontologies. In addition, the results of three user studies that evaluate the comprehensibility and usability of VOWL are summarized. They are complemented by findings from an interview with experienced ontology users and from testing the visual scope and completeness of VOWL with a benchmark ontology. The evaluations helped to improve VOWL and confirm that it produces comparatively intuitive and comprehensible ontology visualizations.

Keywords: OWL, VOWL, ontology, visualization, notation

1. Introduction

Ontologies have received a lot of attention with the rise of the Semantic Web as a way to give information well-defined meaning [6]. Nowadays, they are used in many different contexts to structure and organize information [91]. As a consequence, an increasing number of people in modern knowledge societies come into contact with ontologies. They are no longer exclusively used by ontology experts but also by other user groups, ranging from domain experts to non-expert users. However, especially these casual ontology users often have difficulties to understand ontologies.

Visualizations can help in this regard by assisting in the development, exploration, verification, and sense-making of ontologies [33,53,60]. They are particularly useful to casual users, but can also provide a new perspective for ontology experts.

Although several visualizations for ontologies have been developed in the last couple of years, they often focus on certain aspects of ontologies and are hard to read for casual users. Furthermore, many visualizations are tailored to specific tasks or use special types of diagrams that must first be learned to understand the ontology representation. A review of existing ontology visualizations is given in Section 2 of this article.

The Visual Notation for OWL Ontologies (VOWL) aims to fill this gap by defining a comprehensive visual language for the representation of ontologies that can also be understood by casual ontology users with only little training. It is designed for the Web Ontology Language (OWL) [100], which has become the de facto standard to define ontologies. VOWL specifies graph-

[*]Corresponding author.
[**]Stefan Negru is now with MSD IT Global Innovation Center.

ical depictions for most language constructs of OWL that are combined to a graph visualization representing the ontology. It can be used to generate static ontology visualizations, but also enables the interactive exploration of ontologies and the customization of the visual layout.

We compared an early version of VOWL [76] to the UML-based visualization of ontologies [75]. Based on insights from that comparison and other feedback, we reworked the notation and developed VOWL 2, with significant improvements and more precise mappings to OWL [68]. VOWL 2 will be presented in Section 3, followed by the description of two VOWL implementations in Section 4: ProtégéVOWL, a plugin for the ontology editor Protégé, and WebVOWL, a responsive web application based on open standards.

VOWL and its implementations have been evaluated in several user studies that are summarized in Section 5. In that section, we also report on an interview with experienced ontology users providing additional insights with regard to VOWL. Finally, an evaluation of the visual scope and completeness of VOWL based on a benchmark ontology is provided in Section 5, before we conclude this article in Section 6.

2. Related Work

A number of visualizations for ontologies have been presented in the last couple of years. Surveys can be found in [22,53,60], whereas comparative evaluations of selected visualizations are given in [31,36, 54], among others. Several of the visualizations have been implemented as standalone applications, but most are realized as plugins for ontology editors like Protégé [93].

2.1. Graph Visualizations of Ontologies

Many approaches visualize ontologies as graphs, which reflects the way concepts and relationships are organized in OWL ontologies. The graphs are typically rendered in force-directed, radial, or hierarchical layouts, often resulting in appealing visualizations. However, only few visualizations show complete ontology information—i.e., all classes and properties along with their attributes, such as *functional* or *transitive* for properties—, but most approaches focus on certain aspects.

For instance, OWLViz [49], OntoTrack [61], and KC-Viz [72] visualize merely the class hierarchy of ontologies. GLOW [48] also focuses on the class hierarchy, but provides different layouts in multiple views and is capable to visualize additional property relations using hierarchical edge bundling [47]. A related multi-view approach has been presented in [86], consisting of a radial layout, an indented tree, and a graph visualization. It also depicts some relations besides the inheritance structure of the class hierarchy, but does not provide sufficient detail for a qualitative interpretation of these relations. The same holds for BioMixer [30] that offers different views and graph layouts as well. Likewise, OntoGraf [25], FlexViz [26], OLSVis [98], and OWLPropViz [102] represent various types of property relations, but do not show datatype properties and property characteristics required to fully understand the information modeled in ontologies.

A smaller number of works provide more comprehensive graph visualizations that represent all key elements of ontologies. Unfortunately, the different ontology elements are often hard to distinguish in these visualizations. TGViz [1] and NavigOWL [52], for example, use plain node-link diagrams where all nodes and links look the same except for their color. This is different in SOVA [7] and GrOWL [59], which define more elaborate notations using different symbols, colors, and node shapes. However, as the notations of both SOVA and GrOWL rely on symbols from Description Logic [2] and contain many abbreviations, they are not that suitable for casual users. Furthermore, visualizations created with SOVA and GrOWL feature a large number of crossing edges, which has a negative impact on their readability [68].

Finally, there are 3D graph visualizations for ontologies, such as OntoSphere [9], Onto3DViz [35], and OntoSELF [89], as well as tools that use hyperbolic trees to visualize ontologies, such as OntoRama [23] and Ontobroker [27]. However, these works again focus only on specific aspects of ontologies, such as the class hierarchy or relationships between certain OWL elements.

2.2. Specific Diagram Types

While the reported graph visualizations use basic node-link diagrams to represent ontologies, there are also a number of works that apply other diagram types. For instance, OWL-VisMod [32] and Jambalaya [94] use treemaps, among others, to depict the class hierarchy of ontologies. Jambalaya additionally provides a nested graph visualization that allows to split up the class hierarchy into different views. A similar visualization is used in Knoocks [58], which is moreover complemented by a view that shows individuals and their relations. Also related is the visualization approach of CropCircles [103], which represents the class hierarchy as nested circles similar to treemaps. However, all these works address once again mainly the visualization of inheritance relations, without considering all the other property relations that are usually contained in ontologies.

The Cluster Maps approach also provides a visualization based on nested circles that has been applied to ontologies [28]. Instead of showing the class hierarchy, like in CropCircles, Cluster Maps visualize individuals grouped by the classes they are instances of. Each individual is depicted as a small sphere that is placed inside a larger circle representing the class it belongs to. Cluster Maps have, for instance, been integrated into the DOPE browser [95]. A similar visualization is used in the VIScover system [62,78], which provides interactive filtering and basic editing options. While being appealing visualizations, Cluster Maps and VIScover show only a selection of classes along with the individuals that are members of these classes, but not complete ontology information.

OntoTrix [3] represents ontologies by means of the NodeTrix [46] visualization technique, which is a combination of node-link diagrams and adjacency matrices. Like Cluster Maps, OntoTrix focuses on the visualization of individuals and their connections within the ontology.

In other words, most of the latter approaches address the visualization of what is known as the Assertional Box (ABox) in Description Logic [2], whereas VOWL deals primarily with the visualization of the Terminological Box (TBox), like most other ontology visualizations.

2.3. UML-based Ontology Visualizations

A powerful type of diagram related to OWL and often reused to visualize ontologies is the class diagram of the Unified Modeling Language (UML) [80]. Precise mappings between OWL and UML class diagrams are specified in the Ontology Definition Metamodel (ODM) [81]. Related approaches have, for instance, been presented in [12,16,21,55,82].

There are numerous editors for UML diagrams, but as conventional UML editors cannot read and visualize OWL files, special ontology editors or plugins for UML editors have been developed. Examples include OWLGrEd [4], Visual Ontology Modeler (VOM) [57], and TopBraid Composer [96].

A major drawback of these attempts is that they require knowledge about UML. Although many users with an IT background are familiar with UML diagrams, people from other domains have difficulties interpreting them correctly, as we found in a user study [75]. In addition, there are also conceptual limitations when using UML class diagrams for the visualization of ontologies, since UML has been designed for the representation of software rather than knowledge [41,56].

2.4. Other Diagrammatic Ontology Visualizations

A UML-related type of diagram for ontology visualization is used in OntoViz [88]. It groups classes and datatypes in boxes that are linked by properties. VisioOWL [29] is another UML-related attempt that has been implemented as a template for the diagram editor Microsoft Visio. However, both types of diagrams share the limitation of UML class diagrams that the resulting visualizations are rather difficult to read for lay users.

This is different in Graffoo [24], which aims at an easy-to-understand notation for OWL, similar to VOWL. It comes with a comprehensive specification [83] and has been implemented as a GraphML extension for the diagram editor yEd. In contrast to VOWL, Graffoo is primarily a modeling language. It is therefore rather related to the idea of UML-based modeling than to the visualization approach that is followed by VOWL. This is also the case for OWLeasyViz [15], which proposes a lightweight ontology editor with a simple user interface including a nested ontology visualization.

Diagrammatic approaches of ontology visualization have also be been developed in other contexts. For instance, COE [42] adopts the popular idea of Concept Maps [79] and applies it to the visualization of OWL ontologies. A similar attempt has been made with Concept Diagrams [51] that consider the logic of OWL in particular. Both approaches aim for a formal representation of ontologies that precisely expresses the OWL semantics. However, they do not propose intuitive ontology visualizations that are immediately understandable to casual users.

2.5. RDF Visualizations of Ontologies

Since any OWL ontology can be represented as an RDF graph, it may also be visualized using the common RDF graph notation, consisting of plain nodes and links that form a graph [70]. Such RDF visualizations of OWL ontologies can, for instance, be generated with the RDF validation service of the W3C [99], which outputs static images. Interactive visualizations of RDF graphs are created by tools like RDF Gravity [34], IsaViz [85], or Welkin [71].

However, RDF visualizations of ontologies quickly become large in size, with plenty of nodes and edges, as many OWL constructs are represented by multiple triples in RDF. The visualizations are often hard to read due to their size and for the fact that they provide a one-to-one representation of the RDF triples and not a visualization of the semantics of the OWL elements.

Finally, there are Linked Data visualizations that are also RDF-focused and usually do not consider and visualize complete ontologies [17]. One such tool is RelFinder [44] that visualizes relationships between individuals described by ontologies. Another example is gFacet [45] where individuals are grouped by the classes they are members of and filtered by selecting linked individuals or data values. A related attempt has been implemented in OOBIAN Insight [69], which also groups individuals by their classes and visualizes links between them.

All these tools provide some insight into the relationships of a limited set of classes and/or individuals, but they do not visualize complete ontology information. Furthermore, they are once again rather focused on the ABox instead of the TBox of ontologies which is addressed in VOWL. This is also the case for the tool LodLive [14], which enables the visual exploration of Linked Data using a graph visualization as well.

2.6. *Discussion of Related Work*

Looking at the related work, some common characteristics stand out: Most ontology visualizations utilize a well-known type of diagram (node-link diagram, treemap, UML class diagram, etc.), are two-dimensional, and focus on specific ontology aspects. Only few approaches aim at a comprehensive ontology visualization. Even less provide an explicit description of the visual notation, i.e., a specification that precisely defines the semantics and mappings of the graphical elements. Often, there is no clear visual distinction between different property types or even between classes, properties, and individuals.

Furthermore, several of the works implement a stepwise approach of ontology exploration, where only a root class is shown at the beginning and the user has to navigate through the visualization (e.g., by expanding and collapsing visual elements). VOWL rather pursues an approach that provides users with a complete overview of the ontology and let them subsequently explore parts of it in depth, following the popular Visual Information Seeking Mantra of "overview first, zoom and filter, then details-on-demand" [87]. We decided on this approach, as we consider it important to give users a visual impression of the size and topology of the ontology before they start to explore it any further. However, it has its limitations and challenges, especially when it comes to the visualization of very large ontologies, as we will discuss in Section 6.

Most importantly, VOWL aims for an intuitive visualization that is also understandable to users less familiar with ontologies, while most of the related work has rather been designed for ontology experts.

3. Visual Notation for OWL Ontologies (VOWL)

So far, there are two versions of VOWL that we have specified at `http://purl.org/vowl/spec/`. While the first version emerged from the idea of providing an integrated representation of classes and individuals [76], VOWL 2 focuses on the visual representation of classes, properties, and datatypes.[1] As mentioned before, this information is known as the Terminological Box (TBox) in Description Logic and distinguished from the individuals and data values that make up the Assertional Box (ABox). The TBox is usually most important to understand the conceptualization described in an ontology and therefore considered 'first-class citizen' in most ontology visualizations. This is in contrast to Linked Data visualizations which typically have a stronger focus on the ABox, as we already discussed in Section 2. A good example in this regard is the visual query language QueryVOWL [39] that is based on VOWL but addresses the querying of Linked Data, and therefore focuses on the ABox.

In contrast, VOWL primarily represents the TBox and only optionally integrates ABox information in the visualization. We recommend to display detailed information about individuals in another part of the user interface (e.g., a sidebar) that is linked with the visualization, as in the tool WebVOWL (cf. Section 4). This design decision is supported by comments from a user study on an earlier version of VOWL [75], expressing

[1]This article refers to version 2 of VOWL if not otherwise stated.

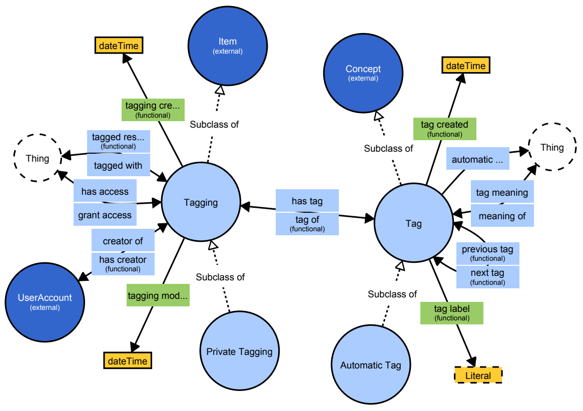


Fig. 1. A small ontology (MUTO [63]) visualized with VOWL.

concerns about the scalability of an integrated TBox and ABox visualization. Even with few individuals per class, additional information, such as property values of individuals, would be difficult to include in the visualization without creating lots of clutter.

VOWL uses a graph visualization, as this is considered an intuitive way to represent the structure of ontologies, which has been confirmed in a comparative evaluation [31]. Figure 1 shows the VOWL visualization of a small ontology. It represents version 1.0 of the Modular Unified Tagging Ontology (MUTO) [63,64], and has been created with the tool WebVOWL that is described in Section 4.

3.1. Basic Building Blocks of VOWL

The basic building blocks of VOWL are a clearly defined set of graphical primitives and a color scheme. Both express specific attributes of the OWL elements (e.g., datatype or object property, different class and property characteristics, etc.), while considering possible combinations thereof. In addition, VOWL defines explicit splitting rules that specify which elements are multiplied in the graph visualization and in which way.

We chose this systematic and modular approach in order to improve the semantics of the visual language and to facilitate its implementation. For instance, the small set of graphical primitives that are systematically varied and combined match well with object-oriented programming. The same holds for the style information that is specified in a modular way in the CSS stylesheet complementing the VOWL specification.

Furthermore, the graph structure was defined in a way that it can be easily visualized (e.g., by avoiding edges between property labels), and developers were given some freedom in the parameterization of VOWL

Table 1
Graphical primitives used in VOWL

Primitive	Application	Primitive	Application
○	classes	▭	datatypes, property labels
⌒	properties	┅	special classes/properties
▷ ▶	property directions	text number symbol	labels, cardinalities

(e.g., by using an abstract color scheme that can be customized as required).

3.1.1. Graphical Primitives

Table 1 lists the small set of graphical primitives that VOWL is based on, along with the ontology elements they are applied to. Classes are depicted as circles that are connected by lines representing the properties with their domain and range axioms. Property labels and datatypes are displayed in rectangles, and text is used for labels and cardinality constraints.

Where available and desired, the number of individuals that are members of a class can be visually indicated by the circle size. VOWL does not specify a particular scaling method for the circle radius, but developers are free to choose a scaling that meets their preferences. A logarithmic or square-root scaling is, however, recommended, as this allows to recognize differences between classes with comparably few individuals, while at the same time limiting the maximum size of the class nodes.

If information about individuals is not available or considered in the visualization, the circles of all common classes have the same predefined size. An exception are the circles of classes that represent `owl:Thing`: They are shown in a smaller size, as they usually do not carry relevant domain information (cf. Figure 1). Even though all individuals in an ontology are instances of `owl:Thing` according to the OWL specification [20], this is irrelevant for the visualization so that `owl:Thing` has always a fixed size.

Most connecting lines in VOWL have an arrowhead that points to the class or datatype that is defined as the range of the property represented by the line. If no domain and/or range axiom is defined for a property, `owl:Thing` is used as domain and/or range. An exception are datatype properties without a defined range, where `rdfs:Literal` is used as range instead.

Lines of inverse properties have arrowheads at both ends, and the direction of the respective property is highlighted when hovering over the label (or when selecting it in touch contexts). Subproperty axioms are

Table 2
Excerpt of the VOWL color scheme

Name	Color	Application
General	🟦	classes, object properties, disjointness
External	🟦	external classes and properties
Deprecated	⬜	deprecated classes and properties
Datatype	🟨	datatypes, literals
Datatype property	🟩	datatype properties
Highlighting	🟥	circles, rectangles, lines, borders, arrows

also indicated by interactive highlighting instead of explicit connections between the property labels (as in an earlier version of VOWL) to reduce the number of edge crossings and to facilitate the implementation of VOWL.

Rectangles representing property labels have no border to distinguish them from those representing datatypes. The labels depict the text for the element given with `rdfs:label` in the language selected by the user. If no label is set for an element, the last part of the IRI is taken, i.e., the string that follows the last slash (/) or hash (#) character. Long labels are abbreviated, but the full text is always available on demand (e.g., displayed in a tooltip and/or a sidebar).

3.1.2. *Color Scheme*

VOWL defines a color scheme for a better distinction of the different elements (cf. excerpt in Table 2). The colors are specified by their function in an abstract way and relative to the canvas color to leave room for customization. Although concrete color codes are recommended, developers may choose different colors as long as they comply with the abstract descriptions provided in the VOWL specification.

The color scheme also specifies how the colors should relate to each other in order to encode the VOWL semantics. For example, the *external* color is defined to be a "dark version of the *general* color", and the *datatype* color is described as a "light color that is clearly different from the other colors". Where several of the color mappings may apply, priority rules are specified. One example for such a rule is that the deprecated color has priority over the external color, as the deprecation information is considered to be more important in most cases.

The recommended color scheme has been designed in accordance with the descriptions: For instance, and in line with the above example, the recommended external color (dark blue) is similar to the general color (light blue), whereas the color for datatype properties (light green) is clearly different. In other cases, established associations have been reused, such as the color gray indicating deactivation (and deprecation) or red as a very visible signal color for the highlighting.

However, colors are not required in order to use VOWL visualizations—they are also comprehensible when printed in black and white or viewed by colorblind people. Details that rely on color, such as the subtle distinction of `owl:Class` and `rdfs:Class`, can either be neglected or may alternatively be added as text information in these cases. While the notations of object and datatype properties also differ only in color, they can still be distinguished without colors, as object properties usually point to classes and datatype properties to datatypes. Other information is provided as text, so that it remains available in the absence of colors.

The text labels also help to make the visualization more self-explanatory, as we found in a user study [68]. Accordingly, apart from the priority rules for colors, the VOWL specification includes rules on how to combine multiple text labels. In most cases, they are simply displayed as a comma-separated list (e.g., "deprecated, external").

3.2. *Visual Elements*

VOWL defines visual elements for most language constructs of OWL, including those of RDF and RDFS reused in OWL. The representations are based on the graphical primitives and color scheme introduced above; a selection is shown in Figure 2.

As already mentioned, redundancy is deliberately introduced in some cases. One such example is the visual element for *external classes*, which we call classes whose base IRI differs from that of the visualized ontology. It has both the external color, as defined by the color scheme, and the hint "external" beneath its label. Other examples are the visual elements for class disjointness and logical disjunction that complement the mathematical symbol or text label with illustrations reminiscent of Venn diagrams. These illustrations help to communicate the underlying set operations to users who are not familiar with the mathematical symbols or names of the concepts.

Some of the visual elements were inspired by UML class diagrams, such as the notation for cardinality constraints or for subclass relations, with the latter being similar to the generalization and realization notations of UML. The representations of these elements were considered intuitive by most participants of a user study, in which we compared an earlier version

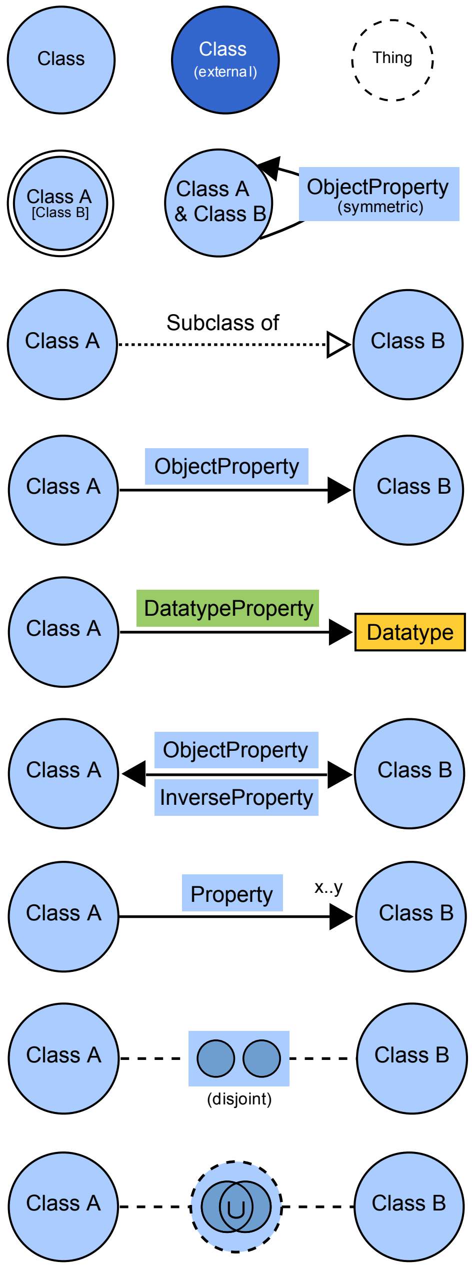


Fig. 2. Selection of visual elements from the VOWL specification.

of VOWL to the UML-based visualization of ontologies [75].

It is important to note that some OWL elements are merged in the visualization. One example are equivalent classes, which are visually indicated by one circle with a double circle border in the graphical representation. The idea behind this merging is that equivalent classes share the same properties, among others, and can therefore be represented by only a single element in the visualization. The labels of merged elements are shown in brackets so that they are still available to the viewer.

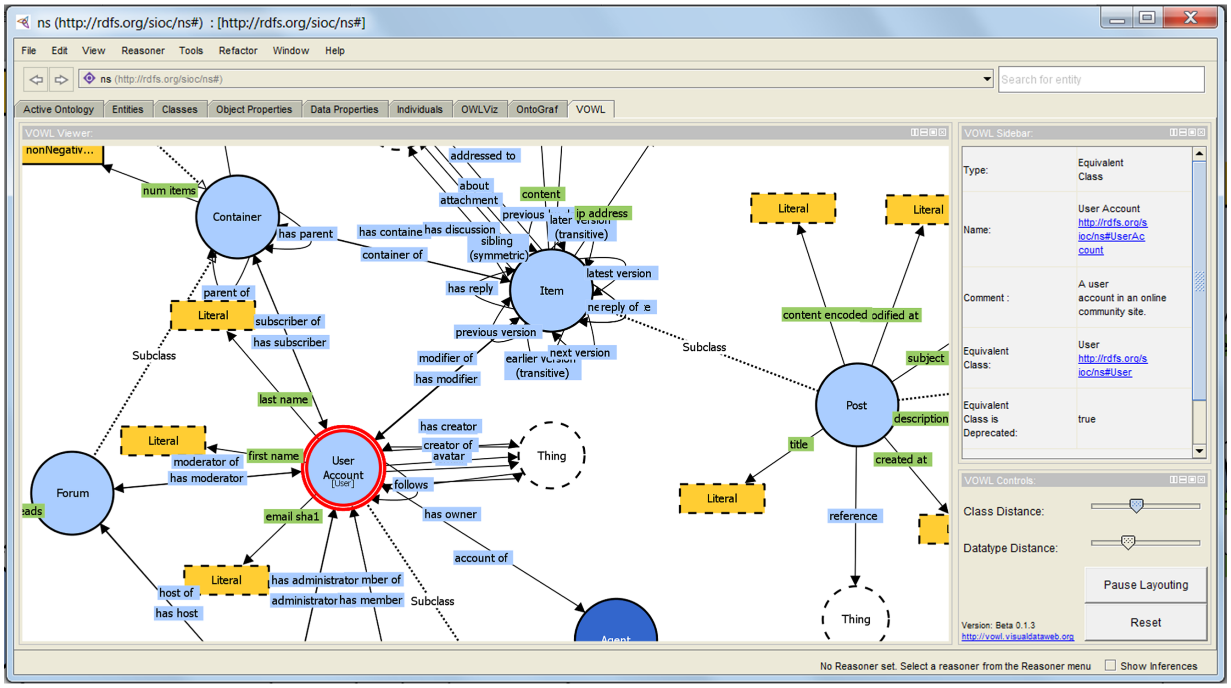
The graphical representations of further OWL elements are defined in the VOWL specification [77]. Most of the remaining representations are variations of the ones presented in Figure 2 and visualize specific property characteristics (e.g., *functional*, *transitive*) or other set operators (e.g., *intersection*, *complement*).

3.3. Graph Visualization

The visual elements are combined to a graph that represents the entire ontology (or a particular part of interest). By default, VOWL graphs are visualized in a force-directed layout. Such a layout tends to arrange the nodes in a way that highly connected classes are placed more to the center of the visualization, whereas less connected ones are placed rather in the periphery. Thus, the force-directed layout helps to reflect the relative importance of the classes in the resulting graph visualization, as the number of connections of a class is often an indication of its importance in the ontology [84]. Moreover, graph layouts created with force-directed algorithms are perceived to be aesthetically pleasant, since all edges have roughly the same length and since they tend to avoid edge crossings, which increases the readability of the visualization [5].

In the same spirit as some elements are merged in VOWL, others are multiplied so that they may appear more than once in the graph. This helps to relax the energy of the force-directed layout and reduces the visual prominence of generic ontology concepts. Multiplication is based on the aforementioned splitting rules that determine whether there is no multiplication for elements, multiplication for each connected class, or multiplication across the entire graph. For instance, the generic class `owl:Thing` is multiplied in a way that it is added once for every class it is connected to, whereas datatype nodes appear once for every datatype property (cf. Figure 1).

While the multiplication increases the number of nodes, it does not affect the number of edges which remains constant. However, as the force-directed layout tends to place the multiplied nodes in the neighborhood of the nodes they are connected to, the multiplication helps to avoid overly long edges and edge crossings that would have a negative impact on the readability of the visualization.



Type:	Equivalent Class
Name:	User Account http://rdfs.org/sioc/ns#UserAccount
Comment :	A user account in an online community site.
Equivalent Class:	User http://rdfs.org/sioc/ns#User
Equivalent Class is Deprecated:	true

Class Distance:
Datatype Distance:
Pause Layouting
Reset
Version: Beta 0.1.3
http://vowl.visualdataweb.org
No Reasoner set. Select a reasoner from the Reasoner menu
Show Inferences

Fig. 3. The user interface of ProtégéVOWL consists of the ontology visualization, a sidebar, and a control panel.

4. Implementations of VOWL

VOWL has been implemented in two different tools so far that demonstrate its applicability: ProtégéVOWL has been realized as a Java-based plugin for the Desktop version of the ontology editor Protégé [92], whereas WebVOWL is a standalone application based on open web technologies. Both tools are released under the MIT license and are publicly available at `http://vowl.visualdataweb.org`. A demonstration of ProtégéVOWL was first shown at ESWC 2014 [67], while WebVOWL was first demonstrated at EKAW 2014 [66].

4.1. ProtégéVOWL: VOWL Plugin for Protégé

In accordance with VOWL, ProtégéVOWL focuses on the visualization of the ontology schema, i.e., the classes, properties, and datatypes (TBox), while it does not consider individuals and data values (ABox information) for the time being. It is deployed as a JAR file that needs to be copied to the *plugins* folder of the Protégé installation, and is compatible with version 4 or higher of Protégé.

Figure 3 shows a screenshot of ProtégéVOWL depicting version 1.35 of the SIOC Core Ontology [8]. The user interface consists of three parts: The *VOWL Viewer* containing the ontology visualization, the *VOWL Sidebar* listing details about the selected element, and the *VOWL Controls* allowing to adapt and pause the force-directed graph layout.

4.1.1. VOWL Viewer

ProtégéVOWL makes use of the visualization toolkit Prefuse [43] to render the visual elements and to arrange them in a force-directed graph layout. It accesses the ontology representation provided by the OWL API of Protégé and transforms it into the data model required by Prefuse. The OWL elements are mapped to the graphical representations as specified by VOWL and combined to a graph.

Prefuse uses a physics simulation to generate the force-directed graph layout consisting of three different forces: Edges act as springs, while nodes repel each other, and drag forces ensure that nodes settle. The forces are iteratively applied, resulting in an animation that dynamically positions the nodes [43].

Users can zoom into the visualization to explore certain ontology parts in detail or zoom out to analyze the global structure of the ontology. They can pan the background and move elements around via drag and drop to optimize the graph visualization and adapt it to their needs. Whenever a node is dragged, the rest of the nodes are repositioned with animated transitions by the force-directed algorithm (if not paused).

4.1.2. VOWL Sidebar

If an element is selected in the visualization, details about it are shown in the sidebar, such as for the selected class "User Account" in Figure 3. These details are provided by the OWL API and include the type and IRI of the element as well as ontology comments, among others. IRIs are displayed as hyperlinks that can be opened with a web browser or other tools for further exploration.

4.1.3. VOWL Controls

The spring forces are separately adaptable for class and datatype nodes with the two sliders in the control panel. This allows to fine-tune the force-directed layout in accordance with the size and structure of the visualized ontology. For instance, datatypes can be positioned in close proximity to the classes they are connected with in order to visually group these elements.

Finally, the force-directed algorithm can be paused via the control panel. This does not only reduce the load of the processor but also allows to rearrange elements without an immediate update of the force-directed layout.

However, the current version of ProtégéVOWL does not implement all visual elements defined in the VOWL specification. A more complete implementation of VOWL is provided by WebVOWL that is described in the following.

4.2. WebVOWL: Web Implementation of VOWL

Implementing visualizations as plugins for ontology editors like Protégé has the advantage that one does not have to deal with ontology processing. The parsing of the ontology files and their transformation into an efficient data structure is done by the ontology editor, or the library that is used for this purpose, such as the OWL API [50] in case of Protégé 4 and 5.

This is different in WebVOWL, which is not a plugin like ProtégéVOWL but a standalone application. Instead of being tied to a particular OWL parser, WebVOWL defines a JSON schema that ontologies need to be converted into. This makes WebVOWL independent from a particular OWL parser and, in principle, broadly applicable. However, since the OWL API is currently the reference implementation for creating, manipulating, and serializing OWL, we also use it for ontology processing in relation with WebVOWL at the moment.

4.2.1. Ontology Processing

The JSON schema has been designed with regard to VOWL, i.e., its structure differs from common OWL serializations in order to enable an efficient generation of the graph visualization. Due to this fact, it is also different from other JSON schemas that emerged in the context of the Semantic Web, such as RDF/JSON [19] or JSON-LD [90].

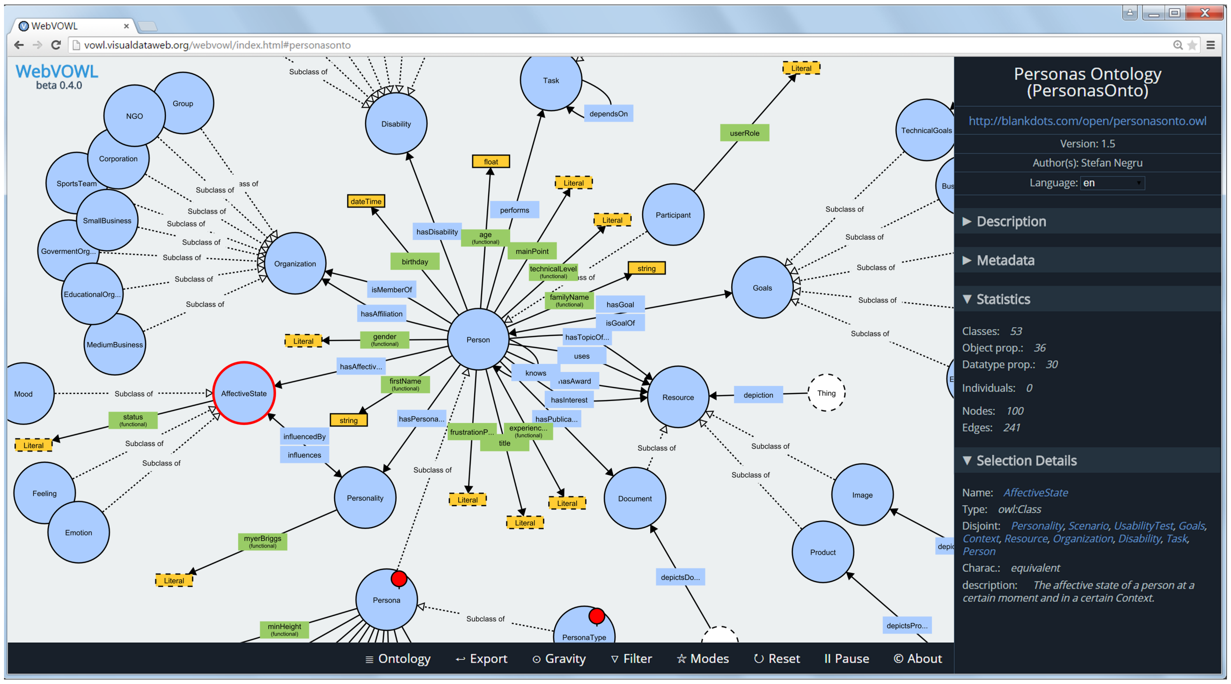
The VOWL-JSON file contains the classes, properties, and datatypes of the ontology along with corresponding type information (`owl:Class`, `owl:ObjectProperty`, `xsd:dateTime`, etc.). Additional characteristics (inverse, functional, deprecated, etc.) as well as annotations (ontology title, version, etc.) and optional ontology metrics (number of classes, properties, etc.) are separately listed. If a VOWL-JSON file does not contain any ontology metrics, they are computed by WebVOWL at runtime. In case an ontology defines individuals, these are listed inside the classes they are members of in the VOWL-JSON file.

Even though WebVOWL is implemented in JavaScript, the transformation of the OWL ontology into the required JSON structure can also be realized with other programming languages. By default, WebVOWL is deployed with a Java-based OWL2VOWL converter that utilizes the aforementioned OWL API [50]. As mentioned before, the converter accesses the ontology representation provided by the OWL API and transforms it into the JSON format required by WebVOWL.

4.2.2. User Interface and Visualization

Figure 4 shows a screenshot of WebVOWL (version 0.4.0) visualizing the Personas Ontology (version 1.5) [73,74]. The general user interface of WebVOWL is similar to that of ProtégéVOWL, consisting of the ontology visualization in the main view, a sidebar listing details, and a menu containing controls, filters, and modes.

The SVG visualization is generated from the JSON file at runtime. WebVOWL renders the graphical elements according to the VOWL specification, i.e., it uses the SVG code and CSS styling information provided by the specification. The force-directed graph layout is realized with the JavaScript library D3 [10]. It is based on a physics simulation similar to the one of Prefuse used in ProtégéVOWL (cf. Section 4.1.1). The forces cool down in each iteration and the force-directed algorithm stops automatically after some time to remove load from the processor and provide a stable visualization. The algorithm is triggered again each



Personas Ontology (PersonasOnto)

http://blankdots.com/open/personasonto.owl

Version: 1.5

Author(s): Stefan Negru

Language: en

▶ Description

▶ Metadata

▼ Statistics

Classes: *53*
Object prop.: *36*
Datatype prop.: *30*

Individuals: *0*

Nodes: *100*
Edges: *241*

▼ Selection Details

Name: *AffectiveState*
Type: *owl:Class*
Disjoint: *Personality, Scenario, UsabilityTest, Goals, Context, Resource, Organization, Disability, Task, Person*
Charac.: *equivalent*
description: *The affective state of a person at a certain moment and in a certain Context.*

Fig. 4. WebVOWL has a similar user interface as ProtégéVOWL, consisting of the ontology visualization, a sidebar, and a menu with controls.

time the user changes the graph layout, as long as the algorithm has not been paused via the menu.

Users can interact with the visualization in a similar fashion as in ProtégéVOWL. They can zoom in and out, pan the background, and move elements around to adapt the force-directed layout. Certain elements (e.g., subproperties and inverse properties) are interactively highlighted according to the VOWL specification. Details about selected elements are once again listed in the sidebar (along with hyperlinks, whenever IRIs are available), such as for the selected class "AffectiveState" in Figure 4.

Furthermore, the sidebar displays metadata about the visualized ontology, such as its title, namespace, author(s), and version, as well as a description text (if provided) and the aforementioned metrics contained in the JSON file or computed at runtime. It also lists annotations and custom properties that are not shown in the visualization. All this information is grouped and displayed in an accordion widget to save screen space.

The language of the text labels in the visualization and sidebar can be changed in case of multilingual ontologies. All available languages are provided in a dropdown list from which the user can choose one.

Moreover, and in contrast to ProtégéVOWL, WebVOWL considers individuals if contained in the visualized ontology. In accordance with the VOWL specification, the size of the circles indicates the number of individuals that are instances of the respective class. When a class is selected by the user, the corresponding individuals are listed in the sidebar, with their IRIs as hyperlinks.

4.2.3. *Controls and Modes*

Like ProtégéVOWL, WebVOWL comes with two *gravity* sliders that allow to adapt the forces of class and datatype nodes, respectively. It also provides a *pause* button to suspend the automatic layout in favor of a manual positioning of the nodes.

In addition, WebVOWL implements some features that extend the ones provided by ProtégéVOWL. One such feature is a special *pick-and-pin mode* inspired by the RelFinder user interface [65]: It allows to decouple selected nodes from the force-directed layout and fix them at freely chosen positions on the canvas. Pinned nodes are indicated by a needle symbol (cf. classes "Persona" and "PersonaType" in Figure 4) that can be removed to release the nodes and reintegrate them into the force-directed layout.

Another mode allows to deactivate the *node scaling*, so that the number of individuals per class is no longer visually indicated by the size of the circles. A third mode switches to a more *compact notation*, where some of the redundant information coding is removed from the visualization, such as the text labels that indicate subclass relations or external and deprecated classes. While these text labels are useful for people who are exposed to VOWL for the first time, they are no longer required by users familiar with VOWL, as the information can also be derived from the line style or color of the elements.

4.2.4. Filters

WebVOWL provides a number of *filters* that help to reduce the size of the VOWL graph in order to focus on certain aspects and to increase its visual scalability. By default, disjointness information is not shown in the visualization but only listed in the sidebar (cf. Figure 4). This is due to the fact that disjoint relations tend to clutter the graph, without being of large interest to the user. However, this filter can be deactivated at any time if disjointness information should be included in the VOWL graph.

Other filters remove *datatype properties* (along with the datatypes they are pointing to) or *set operators* (i.e., classes representing union, intersection, or complement operations). The latter filter has, for instance, been activated in the visualization of Figure 4 so that set operators are not shown in the VOWL graph. There is also a filter for *solitary subclasses*, a term we introduced to refer to subclasses that are only connected to their superclass but no other classes. We identified all these elements as candidates that can be removed from the VOWL visualization without destroying the overall image of the ontology.

We consider it important to systematically remove complete sets of elements from the VOWL graph instead of using filters that remove only single elements and are less predictable. An exception is the slider that controls the *degree of collapsing* of the VOWL graph, and that is provided as another filter in WebVOWL. It removes classes from the graph based on their node degree (not considering datatype properties), starting with the classes having the lowest degree. This filtering is based on the observation that classes with many object properties are often central hubs within ontologies, while classes with only few object properties are typically less important. It can be used to reduce the size of VOWL visualizations of large ontologies by only showing a selection of highly connected classes along with their properties and hiding the rest.

Finally, WebVOWL allows to export the complete or filtered VOWL visualization as SVG image that can be opened in other programs, scaled without loss of quality, edited, shared, and printed. Apart from that, the native functionality of the web browser can be used, for instance, to print the VOWL visualization or to save it as a PDF file.

4.2.5. Discussion of WebVOWL

WebVOWL is one of the first tools for ontology visualization that is completely implemented with open web technologies on the client side. It works in all major browsers, except for the current version of Internet Explorer (version 11), which lacks support for some SVG features. However, cross-browser compatibility is an issue of any application based on open web standards and a general drawback compared to Java-based tools like ProtégéVOWL.

Most of the ontology visualizations reviewed in Section 2 are implemented in Java. Exceptions are the tools BioMixer [30] and OLSVis [98], which also use open web technologies. However, they are less generic than WebVOWL but oriented towards the biomedical domain and integrated in a corresponding toolsets. Their visualizations are also less detailed and show only parts of ontologies or mappings across several ontologies rather than complete ontology overviews.

Other visualization tools that run in the web browser, such as FlexViz [26] or OOBIAN Insight [69], use technologies like Adobe Flash or Microsoft Silverlight that require proprietary browser plugins. While the tool LodLive [14] is also based on open web technologies, it focuses on the visual exploration of Linked Data and not on the visualization of ontologies, as we discussed in Section 2.

Moreover, WebVOWL has been designed with different interaction contexts in mind, including settings using touch interfaces. For instance, zooming can either be performed with the mouse wheel, a double click/tap, or two fingers zooming gestures. As some features may not be available in all interaction contexts (e.g., mouseover effects), we took care that they are not crucial for the interaction and understanding or are differently realized in those contexts.

An instance of WebVOWL is publicly available at `http://vowl.visualdataweb.org`. It allows users to load custom ontologies and to interactively explore the resulting VOWL visualizations. It demonstrates the usefulness of WebVOWL especially for casual ontology users who do not want to install an ontology editor or related tool on their machine in order to visualize some ontology of interest.

5. Evaluations of VOWL

We conducted several user studies to evaluate the comprehensibility and usability of VOWL, and tested its visual scope and completeness with a benchmark ontology. This section reports on the different evaluations.

5.1. Comparative Evaluations with Lay Users

We have compared VOWL to other ontology visualizations in three qualitative user studies [67,68,75]. None of the participants in these studies had extensive prior knowledge about ontologies and the Semantic Web, so they could be regarded as either lay users or casual ontology users.

All studies started with a brief explanation of the examined notations and underlying ontology concepts. We then presented various ontologies displayed with VOWL and other ontology visualizations that VOWL was compared to. All studies had a counterbalanced within-subject design so that the participants could directly compare the different visualization approaches.

We asked questions about the ontologies, which had to be solved by looking at the visualizations. Among those questions, some referred to the general structure of the visualized ontologies (e.g., "What is the approximate number of visible classes?"), while others focused on particular ontology elements (e.g., "Which property is the inverse of property *X*?"). Moreover, we asked the participants for comments on the visualizations, considering aspects such as general overview, choice of shapes and colors, the level of comprehensibility or confusion when looking at the visualization, perceived completeness of the displayed information, and suggestions for improvement.

Ontology visualizations that aim to provide an overview of the ontology structure and approximate OWL feature completeness typically follow a node-link diagram approach (cf. Section 2). Hence, the OWL visualizations we compared VOWL to were based on some kind of node-link diagram: In the first user study [75], we compared an early version of VOWL to the visualization of OWL using UML class diagrams. The second study [67] focused on ProtégéVOWL and compared it to SOVA [7], another visualization plugin for Protégé. The SOVA plugin also served as a reference in the third user study, next to the GrOWL visualization [59], but this time we used the WebVOWL implementation of VOWL for the comparative evaluation [68].

Overall, participants considered all presented visualizations comprehensible. The inherent possibility to visually distinguish basic elements, such as properties and classes, was stated to be important, and a small number of crossing edges was confirmed to support the overview.

Short descriptive labels on elements, as they were included in VOWL, were generally seen as helpful. In the case of special property characteristics and modifiers, such as *functional* or *transitive*, the unabbreviated textual description was considered indispensable to easily interpret the visualizations.

The aggregation and multiplication of nodes, as featured in VOWL, was met with mixed reactions. Remembering the explanations on the visual notation, some of the participants quickly understood that generic elements, such as `owl:Thing`, would appear several times, and that groups of equivalent classes were merged into a single node. Others had expected a different behavior—only one representation of the unique class `owl:Thing` and one separate node per equivalent class—and thus required a moment to adjust to the visualization.

The study participants welcomed the continuous zoom, but asked for additional highlighting features (e.g., an indication of all nodes that are directly linked to the selected one) as well as a feature to search and highlight specific elements. Even though the search functionality of the web browser can be used for the latter purpose in case of WebVOWL, participants regarded this as too restrictive.

In summary, the lay users in the three comparative studies could solve most tasks well with VOWL. When a preference for one of the visualizations had to be stated, it leaned toward VOWL for the majority of participants, based on aspects such as clarity and distinguishability of elements, ease of use with interactive layouting and highlighting features, as well as aesthetics. The interested reader can find more details on the reported user studies in [67,68,75].

5.2. Interviews with Experienced Ontology Users

To gain additional insight into how users perceive VOWL, and to become more aware of the differences between lay users and experienced ontology users, we have also evaluated VOWL with users who had already worked with ontology editors and formal OWL syntax.

5.2.1. Participants

We recruited five users experienced with ontologies for this evaluation. Their expertise varied between knowing the major OWL features from working with existing ontologies and being very familiar with the specifics of OWL and able to define custom ontology elements. Four of the participants had some prior experience with editing ontologies in Protégé.

We chose this type of participants to contrast their statements with the findings from the aforementioned user studies (cf. Section 5.1). Moreover, we wanted to identify requirements that would not become apparent to users who have never worked (or are just starting to work) with ontologies.

5.2.2. Materials and Equipment

WebVOWL (version 0.2.11) was executed in a Mozilla Firefox browser (version 32) for the interview. It was initially showing the FOAF vocabulary [11], while three other ontology visualizations were available from the menu (SIOC [8], MUTO [63], and PersonasOnto [73]). In addition, the VOWL specification [77] was opened in another tab of the web browser, and ProtégéVOWL (cf. Section 4.1) was running in Protégé 4.3.0 on the same computer, initially showing one of the four aforementioned ontologies.

A number of questions were prepared to provide some rough guidance through the features of VOWL and to ensure touching upon a wide range of visualization aspects. Some of the questions explicitly referred to the visualization (e.g., “Are any displayed visual elements inherently unclear? Is any further information needed to understand them?”), while others dealt with interactive features (e.g., “Are links between elements expressed sufficiently clearly by the interactive highlighting?”). Furthermore, some final questions were related to the degree of information obtainable with VOWL (e.g., “What information is missing?”). These questions were printed on paper and remained with the conductor of the interview.

5.2.3. Procedure

All interviews were conducted by the same person in a quiet room. One of the five participants attended alone, while the other four took part in groups of two. This helped to stimulate discussions between the participants while they were using the visualization. Participants were asked to “think aloud” and to describe both what they were seeing and what they would like to do or find when exploring the visualization.

To ensure that all participants started out the same, they were provided with a short introduction explaining that both WebVOWL and ProtégéVOWL display one ontology at a time, and that all elements are defined in a specification document. However, in contrast to the previous user studies, no systematic introduction to the VOWL notation was given this time. We wanted the participants to explore and familiarize with VOWL on their own, without any prior knowledge of the meaning of shapes, colors, and symbols.

At the same time, we wanted to make sure that the participants would come across all key elements and features of VOWL. Therefore, we basically allowed free exploration, but occasionally resorted to ask some of the prepared questions when we found the participants did not have an idea for a particular action on their own. For example, when the participants had not noticed the multiplication of datatypes by themselves after a while, the interviewer would read out the text “Can you find datatype *X*? How often does it appear in the visualization? Do you consider this appropriate?” to give participants an incentive to think about the respective feature. Explanations for elements and features were provided upon request, unless participants could figure out the meaning themselves.

5.2.4. Results

Overall, the results showed that the participants had no problems in understanding the general VOWL visualizations and distinguishing classes, properties, and datatypes. Three of the participants instantly identified these ontology elements, while the other two could quickly associate the concepts with their visual representations, as well. Detailed results are presented in the following.

Layout and Navigation All study participants spoke favorably about the force-directed layout, as it helped them to easily recognize clusters and inheritance structures given in the ontologies. They stated that highly connected elements could easily be recognized due to the layout, and that the gravity options, which control how strongly nodes are attracted to each other, were beneficial for improving the readability of the visualizations.

The pick-and-pin feature was thought of as useful. However, several participants could imagine further automatic layouts besides the force-directed one (e.g., hierarchical layouts) and an ability to focus single elements that the remaining elements should align around (i.e., some kind of pivot navigation). One participant suggested the addition of a minimap to ease the navigation in large ontologies.

Node Multiplication The multiplication of nodes was considered beneficial for the simplicity of the graph structure and the layout of the visualization. Only one participant did not see the multiplication of datatypes as desirable, though he was not confused by it either.

One type of questions that could not immediately be answered due to the multiplication of the datatypes was "What datatype properties with range *X* exist in the ontology?" (because a datatype *X* may be represented by several nodes, connections to which can thus not be counted at a single glance). Most participants agreed that they would normally not want to extract such information anyway.

Equivalent Classes The meaning of double-ringed classes as groups of merged equivalent classes was not immediately clear to any of the participants. Once the meaning had been explained, most participants considered the visual representation appropriate. However, one participant wondered why one of the equivalent class names is presented in a larger font and therefore more prominent than the others.

Contrary to the studies with lay users (Section 5.1), all participants were content with the merging of equivalent classes into a single visual element. Yet, one participant remarked that for editing and determining the provenance of property definitions, the equivalent class nodes might optionally have to be splittable.

Properties In general, object and datatype properties were instantly found and recognized for what they were. Three participants asked for options to hide either property labels (thus only showing unlabeled property edges) or datatype nodes (thus showing labeled datatype properties without range information), as they deemed that information to be relevant only in certain situations. This reiterates the wish expressed by some of the lay users from the previous studies to optionally hide some textual information.

Inverse properties were correctly identified by most participants. One of them explicitly praised the way how pairs of inverse properties were displayed as bidirectional links with arrows at both sides. However, the participants generally had to be pointed to the interactive highlighting feature that indicates which property is associated with which direction. Likewise, the highlighting of subproperties was not noticed unless explicitly pointed out. With that explanation, however, the visual representation was comprehensible to participants who had experience with the concept of subproperties.

Set Operators The participants were not sure whether they could recognize the visual notation for classes based upon set operations (union, intersection, and complement) on their own. Three of them quickly figured out the meaning because of the mathematical symbols ($\cup$, $\cap$, $\neg$), but did not notice the Venn diagrams at first. Two participants thought the combination of the dashed node border, the Venn diagrams, and the logical symbols is a visual overload that should be tackled. Another one remarked that the notation would be more consistent with the rest of VOWL if the symbols were replaced by text labels.

Filtering Several participants asked for additional ways of filtering elements in the graph. For instance, one participant could imagine advanced filter criteria, such as ranges of properties to hide the respective property edges, while two others thought an option for selectively collapsing and expanding subtrees or subgraphs was missing. The possibilities to hide datatype properties and weakly connected subclasses included in WebVOWL version 0.2.11 were considered promising steps in this direction.

Colors Most colors were not commented on a lot. Two participants stated the color coding was not clear to them, especially with respect to external classes. One of them pointed out that the dark color of external elements visually emphasizes these elements, while he did not see a particular reason why external classes should call for attention. Two participants would have wished for colors that express the class hierarchy in some way, such as the specialization level in the inheritance tree, though they admitted that such a color coding would probably only be conceivable for strictly hierarchical inheritance structures rather than the inheritance structures possible with OWL.

Completeness When asked whether VOWL lacks any information, three participants answered that the amount of information currently displayed is almost too much, for which some overlapping labels were given as proof. Participants agreed that *disjoint* relationships between classes should not be displayed by default but only in cases where disjointness is of particular interest, as it is recommended and implemented.

One participant would have preferred some more explicit information in the sidebar of WebVOWL, such as an indication of namespaces along with the labels. The same participant also asked for a way to display other class restrictions that are currently not supported by VOWL, such as restricting the instances of a class to a specific set of individuals. Beside these omissions, the visualization was stated to be "quite complete" in terms of OWL features considered relevant by the participants.

Application Areas Finally, the interview participants were asked how they could imagine to use VOWL. All participants agreed with the basic idea of finding out about the structure and content of an ontology, and some mentioned techniques such as navigating through an ontology by starting with some core elements. One participant could imagine using VOWL to align several ontologies.

Usage for ontology engineering and editing tasks, such as adding, modifying, or removing ontology elements, was suggested by all participants, which also included using VOWL for debugging purposes. One participant mentioned that VOWL could be helpful in teaching contexts, such as lectures on the Semantic Web, in order to explain OWL concepts to learners.

5.3. Benchmark Test of the VOWL Visualization

VOWL has additionally been tested with OntoViBe, the Ontology Visualization Benchmark [37], available at `http://ontovibe.visualdataweb.org`. OntoViBe provides an ontology comprising of a comprehensive set of OWL 2 [101] language constructs and systematic combinations thereof. It has been designed to support the testing of feature completeness of ontology visualizations. The elements in OntoViBe are named in a self-descriptive manner to allow for an easier interpretation and analysis.

Figure 5 shows the VOWL visualization of version 2.1 of OntoViBe [38]. It has been created with WebVOWL (version 0.4.0), which provides a complete implementation of VOWL 2 in contrast to ProtégéVOWL. We have annotated the visualization to point to some aspects discussed in the following.

Looking at the VOWL visualization of OntoViBe, some effects of the force-directed layout are noticeable right away. A strongly connected class ("PropertyOwner") appears to be very central to the ontology, due to the large number of nodes it is connected to (annotation ① in Figure 5). The convenient radial placement of outbound and inbound edges, which mostly avoids overlapping arrowheads, is immediately visible. Other graphical features that can directly be seen are the merging of equivalent classes into single nodes (also visible in annotation ①), and the use of human-readable labels, where available.

OntoVibe includes a small class hierarchy that is clearly arranged in the VOWL visualization (annotation ②). One of the classes in the hierarchy ("MultiSubclass") has a larger size than the others, as it contains individuals. The exact number of individuals that are members of this class is displayed beneath the class name (in this case "4"). Moreover, the class hierarchy (in annotation ②) contains further equivalent classes merged into a single node ("Subclass") and a deprecated class that is imported from another ontology ("DeprecatedImportedClass").

The latter class is an example of how possible conflicts with regard to the visualization are resolved by VOWL. Usually, external elements are visualized with a dark blue background, such as the imported class in Figure 5 (cf. annotation ③). However, as the deprecated color overrides the external color according to the priority rules of VOWL, the "DeprecatedImportedClass" (in annotation ②) has a gray background instead of a blue one. The deprecated color also overrides the standard colors of object and datatype properties (see annotation ④ for an example).

Another kind of avoided conflict concerns the geometry, as multiple properties between the same pair of classes do not visually obstruct one another but are rendered as curved multi-edges (also visible in annotation ④). The same applies to sets of cyclic properties whose domain and range axioms point to the same class (annotation ⑤).

Cardinality constraints are correctly combined and depicted in the VOWL visualization: Annotation ⑥ shows an exact cardinality, and annotation ④ a minimum cardinality; in annotation ①, a range of cardinalities is shown that has been composed from the minimum and maximum cardinality expressions defined for that class and property. Also, global cardinality constraints as well as logical property characteristics are depicted (cf. indications "functional", "inverse functional", and "symmetric" in annotation ④). Equivalent properties are correctly merged in the VOWL visualization too (annotations ④ and ⑦).

Moreover, the visualization reveals that some parts of OntoViBe are only little connected. There are several classes that are not linked to any other class, and two separate subgraph structures that are easy to spot (annotations ⑦ and ⑧). While all OWL classes are subclasses of `owl:Thing` according to the OWL specification [20], and therefore implicitly connected, these implicit connections are not shown in VOWL, as they would clutter the visualization without any benefit for the interpretation of the ontology.

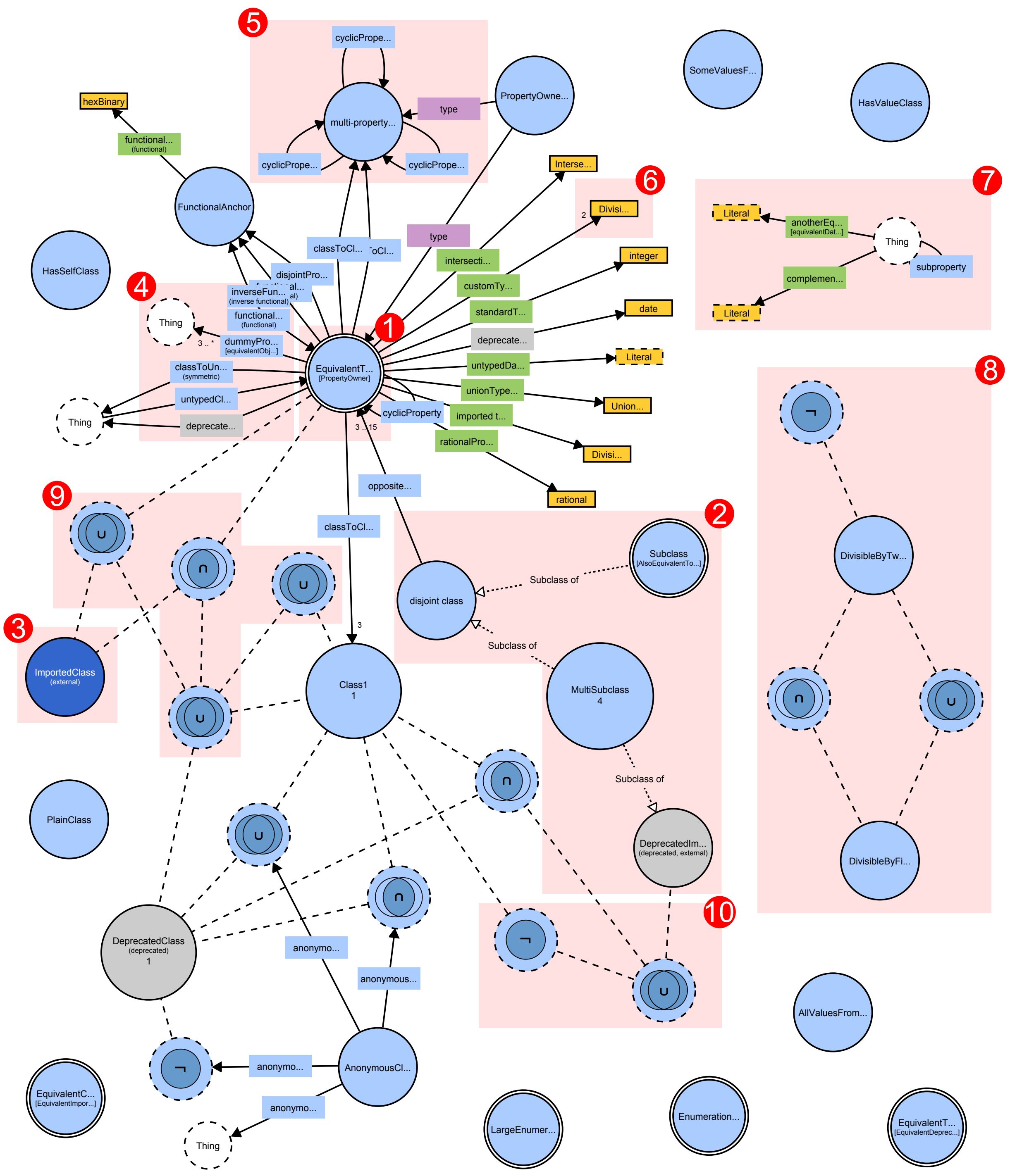


Fig. 5. Benchmark ontology (OntoViBe 2 [38]) visualized with VOWL. The annotations have been added to indicate aspects of interest that are referred to in the paper. They are *not* part of the actual VOWL visualization.

Note that the layout of the VOWL graph has been manually adapted for Figure 5 in order to fit well into the available space. In particular, the distances of unconnected nodes have been reduced to get a more compact visualization. Normally, the disconnected ontology parts would tend to drift away from the central ontology part due to the force-directed layout, which makes them even easier to spot.

The visualization of custom data ranges is not properly supported for the time being, as VOWL mainly focuses on classes and property relations. This is seen by nodes that display only the names of the data ranges but no more information, such as the node with the label "Divisi..." (for "DivisibleByFiveEnumeration", annotation ⑥). This data range is additionally represented by a class that is linked with set operators representing other datatypes of OntoViBe, as can be seen in annotation ⑧ which contains a corresponding class with the label "DivisibleByFi..." (for "DivisibleByFiveEnumeration").

Besides the lack of support for custom data ranges, only two omissions in the current VOWL syntax become apparent that might be crucial for the interpretation of some ontologies: On the one hand, unions, intersections, and complements of classes have only been specified for anonymous classes in VOWL. So far, no notation for named classes that are defined by these set operations exists.

On the other hand, in the case of set operation classes that refer to other set operation classes, the nesting direction is not yet displayed in VOWL: A union including an intersection and an intersection including a union look the same and can hence not be immediately distinguished in the visualization (cf. annotation ⑨). This can be even more irritating if a complement class is involved, such as in annotation ⑩.

Apart from these minor issues that can be addressed in a future iteration of the visual notation, VOWL provides a complete and correct representation of the OntoViBe ontology. Therefore, the notation can be expected to consistently visualize also a wide variety of other ontologies.

6. Conclusion and Future Work

The main goal of VOWL is to provide a comprehensive yet comprehensible visual language for the representation of OWL ontologies. This article described the key considerations, features, and capabilities of VOWL, as well as two implementations: a plugin for the widely used ontology editor Protégé and a responsive web application. Both implementations demonstrate the applicability and usability of VOWL by means of several ontologies. The comprehensibility of VOWL has been confirmed by a number of user studies conducted with different user groups, while its visual expressiveness and completeness have been tested with a benchmark ontology.

The ontologies used in the evaluations of VOWL were of relatively moderate size. However, there is no upper limit for the size of ontologies. On the other hand, graph visualizations are only viable up to a certain size, at which the overview is lost and the graph is no longer easily usable. The visualization of large-scale ontologies with VOWL is an issue that has not yet been sufficiently addressed.

We have done first steps by designing interactive features to filter elements and reduce the overall graph size. Future solutions might consider both automatic and manual methods to detect ontology components that carry context-specific importance, so that parts of the visualization can be hidden or bundled where appropriate. In addition, clever strategies from the field of graph visualization may be incorporated into VOWL to improve the visualization of large ontologies. However, this is a general issue that does not only affect VOWL but most graph visualizations of ontologies.

A related issue is the fact that ontology elements have no inherent spatial information. Therefore, all elements are initially placed in a random manner in the force-directed layout. While this does not influence a single session of work, it prevents users from creating a "mental map" of the visualization that remains constant for several sessions, since the elements are placed at different locations every time the VOWL graph is rendered. Future work will have to develop reasonable guidelines on how to best place ontology elements so their positioning follows a reproducible pattern. In addition, implementations of VOWL may provide options to save customized VOWL visualizations and allow to reopen them with exactly the same layout and visual settings at a later point in time.

Future work will also be concerned with incorporating the remaining OWL language constructs into VOWL. Although VOWL already considers a large portion of the language constructs, it is not yet complete, in particular with regard to OWL 2. Therefore, we plan to further extend VOWL to turn it into a visual language that can represent OWL ontologies as completely as possible.

Even though VOWL has been designed for OWL, it might also be used to represent related languages and models, as long as they can be converted into a VOWL-compatible format (e.g., the VOWL-JSON format used in WebVOWL). In particular, the WebVOWL implementation can be integrated well with other tools due to its modular architecture and the separation of the visualization from the rest of the user interface. For instance, WebVOWL has already been integrated into the tool PURO Modeler[2] and the Linked Open Vocabularies (LOV) service[3]. There are also ongoing efforts to integrate WebVOWL into WebProtégé[4] [97].

In related endeavours, we have looked into how VOWL could be reused to support the visual querying of Linked Data [39,40], to visualize text [18], or the evolution of ontologies [13]. Apart from that, we hope that VOWL and its implementations will be useful to other researchers and developers and in related application areas, such as ontology editing or alignment, as well as in teaching and training contexts.

Acknowledgements

The authors would like to thank David Bold, Vincent Link, and Eduard Marbach for their excellent contributions to the implementation of VOWL and their valuable feedback on the notation. The authors would also like to thank all participants of the user studies, and all ontology users who took part in the interviews.

References

[1] H. Alani. TGVizTab: An ontology visualisation extension for Protégé. In *Proceedings of the 2nd Workshop on Visualizing Information in Knowledge Engineering (VIKE '03)*, 2003.

[2] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[3] B. Bach, E. Pietriga, I. Liccardi, and G. Legostaev. OntoTrix: A hybrid visualization for populated ontologies. In *Proceedings of the 20th International Conference on World Wide Web (WWW '11), Companion Volume*, pages 177–180. ACM, 2011.

[4] J. Bārzdiņš, G. Bārzdiņš, K. Čerāns, R. Liepiņš, and A. Sproģis. OWLGrEd: A UML style graphical notation and editor for OWL 2. In *Proceedings of the 7th International Workshop on OWL: Experiences and Directions (OWLED '10)*, volume 614 of *CEUR-WS*, 2010.

[2] `http://lod2-dev.vse.cz/puromodeler`

[3] `http://lov.okfn.org/dataset/lov`

[4] `http://github.com/VisualDataWeb/webprotege`

[5] C. Bennett, J. Ryall, L. Spalteholz, and A. Gooch. The aesthetics of graph visualization. In *Proceedings of the International Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging (CAe '07)*, pages 57–64. Eurographics Association, 2007.

[6] T. Berners-Lee, J. A. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, 2001.

[7] T. Boinski, A. Jaworska, R. Kleczkowski, and P. Kunowski. Ontology visualization. In *Proceedings of the 2nd International Conference on Information Technology (ICIT '10)*, pages 17–20. IEEE, 2010.

[8] U. Bojārs and J. Breslin. SIOC core ontology specification. `http://rdfs.org/sioc/spec/`, 2010.

[9] A. Bosca, D. Bonino, and P. Pellegrino. OntoSphere: More than a 3D ontology visualization tool. In *Proceedings of the 2nd Italian Semantic Web Workshop (SWAP '05)*, volume 166 of *CEUR-WS*, 2005.

[10] M. Bostock, V. Ogievetsky, and J. Heer. D3 – data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011.

[11] D. Brickley and L. Miller. FOAF Vocabulary Specification. `http://xmlns.com/foaf/spec/`, 2014.

[12] S. Brockmans, R. Volz, A. Eberhart, and P. Löffler. Visual modeling of OWL DL ontologies using UML. In S. A. McIlraith, D. Plexousakis, and F. van Harmelen, editors, *The Semantic Web – ISWC 2004*, volume 3298 of *LNCS*, pages 198–213. Springer, 2004.

[13] M. Burch and S. Lohmann. Visualizing the evolution of ontologies: A dynamic graph perspective. In *Proceedings of the International Workshop on Visualizations and User Interfaces for Ontologies and Linked Data (VOILA '15)*, volume 1456 of *CEUR-WS*, pages 69–76, 2015.

[14] D. V. Camarda, S. Mazzini, and A. Antonuccio. LodLive, exploring the web of data. In *Proceedings of the 8th International Conference on Semantic Systems (I-SEMANTICS '12)*, pages 197–200. ACM, 2012.

[15] N. Catenazzi, L. Sommaruga, and R. Mazza. User-friendly ontology editing and visualization tools: The OWLeasyViz approach. In *Proceedings of the 13th International Conference Information Visualisation (IV '09)*, pages 283–288. IEEE, 2009.

[16] S. Cranefield. UML and the semantic web. In *Proceedings of the 1st Semantic Web Working Symposium (SWWS '01)*, pages 113–130. IOS press, 2001.

[17] A.-S. Dadzie and M. Rowe. Approaches to visualizing linked data: A survey. *Semantic Web*, 2(2):89–124, 2011.

[18] S. Dasiopoulou, S. Lohmann, J. Codina, and L. Wanner. Representing and visualizing text as ontologies: A case from the patent domain. In *Proceedings of the International Workshop on Visualizations and User Interfaces for Ontologies and Linked Data (VOILA '15)*, volume 1456 of *CEUR-WS*, pages 83–90, 2015.

[19] I. Davis, T. Steiner, and A. L. Hors. RDF 1.1 JSON Alternate Serialization (RDF/JSON). `http://www.w3.org/TR/rdf-json/`, 2013.

[20] M. Dean, G. Schreiber, S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. OWL web ontology language reference. `http://www.w3.org/TR/owl-ref/`, 2004.

[21] D. Djurić, D. Gašević, V. Devedžić, and V. Damjanović. A UML profile for OWL ontologies. In U. Aßmann, M. Aksit,

and A. Rensink, editors, *Model Driven Architecture*, volume 3599 of *LNCS*, pages 204–219. Springer, 2005.

[22] M. Dudáš, O. Zamazal, and V. Svátek. Roadmapping and navigating in the ontology visualization landscape. In K. Janowicz, S. Schlobach, P. Lambrix, and E. Hyvönen, editors, *Knowledge Engineering and Knowledge Management*, volume 8876 of *LNAI*, pages 137–152. Springer, 2014.

[23] P. Eklund, N. Roberts, and S. Green. OntoRama: Browsing RDF ontologies using a hyperbolic-style browser. In *Proceedings of the 1st International Symposium on Cyber Worlds (CW '02)*, pages 405–411. IEEE, 2002.

[24] R. Falco, A. Gangemi, S. Peroni, D. Shotton, and F. Vitali. Modelling OWL ontologies with graffoo. In V. Presutti, E. Blomqvist, R. Troncy, H. Sack, I. Papadakis, and A. Tordai, editors, *The Semantic Web: ESWC 2014 Satellite Events*, volume 8798 of *LNCS*, pages 320–325. Springer, 2014.

[25] S. Falconer. OntoGraf. `http://protegewiki.stanford.edu/wiki/OntoGraf`, 2010.

[26] S. M. Falconer, C. Callendar, and M.-A. Storey. A visualization service for the semantic web. In P. Cimiano and H. S. Pinto, editors, *Knowledge Engineering and Management by the Masses*, volume 6317 of *LNAI*, pages 554–564. Springer, 2010.

[27] D. Fensel, S. Decker, M. Erdmann, and R. Studer. Ontobroker: The very high idea. In *Proceedings of the 11th International Florida Artificial Intelligence Research Society Conference (FLAIRS '98)*, pages 131–135. AAAI Press, 1998.

[28] C. Fluit, M. Sabou, and F. van Harmelen. Ontology-based information visualization: Toward semantic web applications. In *Visualizing the Semantic Web*, pages 36–48. Springer, 2002.

[29] J. Flynn. VisioOWL. `http://www.semwebcentral.org/projects/visioowl/`, 2012.

[30] B. Fu, L. Grammel, and M.-A. Storey. BioMixer: A web-based collaborative ontology visualization tool. In *Proceedings of the 3rd International Conference on Biomedical Ontology (ICBO '12)*, volume 897 of *CEUR-WS*, 2012.

[31] B. Fu, N. F. Noy, and M.-A. Storey. Indented tree or graph? a usability study of ontology visualization techniques in the context of class mapping evaluation. In H. Alani, L. Kagal, A. Fokoue, P. Groth, C. Biemann, J. Parreira, L. Aroyo, N. Noy, C. Welty, and K. Janowicz, editors, *The Semantic Web – ISWC 2013*, volume 8218 of *LNCS*, pages 117–134. Springer, 2013.

[32] F. J. García-Peñalvo, R. Colomo-Palacios, J. García, and R. Therón. Towards an ontology modeling tool. A validation in software engineering scenarios. *Expert Systems with Applications*, 39(13):11468–11478, 2012.

[33] V. Geroimenko and C. Chen. *Visualizing the Semantic Web: XML-Based Internet and Information Visualization*. Springer, 2nd edition, 2006.

[34] S. Goyal and R. Westenthaler. RDF Gravity. `http://semweb.salzburgresearch.at/apps/rdf-gravity/`.

[35] S. S. Guo and C. W. Chan. A tool for ontology visualizaiton in 3D graphics: Onto3DViz. In *Proceedings of the 23rd Canadian Conference on Electrical and Computer Engineering (CCECE '10)*, pages 1–4. IEEE, 2010.

[36] S. S. Guo and C. W. Chan. A comparison and analysis of some ontology visualization tools. In *Proceedings of the 23rd International Conference on Software Engineering & Knowledge Engineering (SEKE '11)*, pages 357–362. Knowledge Systems Institute Graduate School, 2011.

[37] F. Haag, S. Lohmann, S. Negru, and T. Ertl. OntoViBe: An ontology visualization benchmark. In *Proceedings of the International Workshop on Visualizations and User Interfaces for Knowledge Engineering and Linked Data Analytics (VISUAL '14)*, volume 1299 of *CEUR-WS*, pages 14–27, 2014.

[38] F. Haag, S. Lohmann, S. Negru, and T. Ertl. OntoViBe 2: Advancing the ontology visualization benchmark. In P. Lambrix, E. Hyvönen, E. Blomqvist, V. Presutti, G. Qi, U. Sattler, Y. Ding, and C. Ghidini, editors, *Knowledge Engineering and Knowledge Management*, volume 8982 of *LNAI*, pages 83–98. Springer, 2015.

[39] F. Haag, S. Lohmann, S. Siek, and T. Ertl. QueryVOWL: A visual query notation for linked data. In F. Gandon, C. Guéret, S. Villata, J. Breslin, C. Faron-Zucker, and A. Zimmermann, editors, *The Semantic Web: ESWC 2015 Satellite Events*, volume 9341 of *LNCS*, pages 387–402. Springer, 2015.

[40] F. Haag, S. Lohmann, S. Siek, and T. Ertl. QueryVOWL: Visual composition of SPARQL queries. In F. Gandon, C. Guéret, S. Villata, J. Breslin, C. Faron-Zucker, and A. Zimmermann, editors, *The Semantic Web: ESWC 2015 Satellite Events*, volume 9341 of *LNCS*, pages 62–66. Springer, 2015.

[41] L. Hart, P. Emery, B. Colomb, K. Raymond, S. Taraporewalla, D. Chang, Y. Ye, E. Kendall, and M. Dutra. OWL full and UML 2.0 compared. Technical report, OMG, 2004.

[42] P. Hayes, T. C. Eskridge, R. Saavedra, T. Reichherzer, M. Mehrotra, and D. Bobrovnikoff. Collaborative knowledge capture in ontologies. In *Proceedings of the 3rd International Conference on Knowledge Capture (K-CAP '05)*, pages 99–106. ACM, 2005.

[43] J. Heer, S. K. Card, and J. A. Landay. Prefuse: A toolkit for interactive information visualization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '05)*, pages 421–430. ACM, 2005.

[44] P. Heim, S. Hellmann, J. Lehmann, S. Lohmann, and T. Stegemann. RelFinder: Revealing relationships in RDF knowledge bases. In T.-S. Chua, Y. Kompatsiaris, B. Mérialdo, W. Haas, G. Thallinger, and W. Bailer, editors, *Semantic Multimedia*, volume 5887 of *LNCS*, pages 182–187. Springer, 2009.

[45] P. Heim, J. Ziegler, and S. Lohmann. gFacet: A browser for the web of data. In *Proceedings of the International Workshop on Interacting with Multimedia Content in the Social Semantic Web (IMC-SSW '08)*, volume 417 of *CEUR-WS*, pages 49–58, 2008.

[46] N. Henry, J.-D. Fekete, and M. J. McGuffin. NodeTrix: A hybrid visualization of social networks. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1302–1309, 2007.

[47] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748, 2006.

[48] W. Hop, S. de Ridder, F. Frasincar, and F. Hogenboom. Using hierarchical edge bundles to visualize complex ontologies in GLOW. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing (SAC '12)*, pages 304–311. ACM, 2012.

[49] M. Horridge. OWLViz. `http://protegewiki.stanford.edu/wiki/OWLViz`, 2010.

[50] M. Horridge and S. Bechhofer. The OWL API: A java API for OWL ontologies. *Semantic Web*, 2(1):11–21, 2011.

[51] J. Howse, G. Stapleton, K. Taylor, and P. Chapman. Visualizing ontologies: A case study. In L. Aroyo, C. Welty, H. Alani, J. Taylor, A. Bernstein, L. Kagal, N. Noy, and E. Blomqvist, editors, *The Semantic Web – ISWC 2011*, volume 7031 of *LNCS*, pages 257–272. Springer, 2011.

[52] A. Hussain, K. Latif, A. Rextin, A. Hayat, and M. Alam. Scalable visualization of semantic nets using power-law graphs. *Applied Mathematics & Information Sciences*, 8(1):355–367, 2014.

[53] A. Katifori, C. Halatsis, G. Lepouras, C. Vassilakis, and E. Giannopoulou. Ontology visualization methods – a survey. *ACM Computer Surveys*, 39(4), 2007.

[54] A. Katifori, E. Torou, C. Halatsis, G. Lepouras, and C. Vassilakis. A comparative study of four ontology visualization techniques in Protégé: Experiment setup and preliminary results. In *Proceedings of the International Conference on Information Visualisation (IV '06)*, pages 417–423. IEEE, 2006.

[55] E. F. Kendall, R. Bell, R. Burkhart, M. Dutra, and E. K. Wallace. Towards a graphical notation for OWL 2. In *Proceedings of the 6th International Workshop on OWL: Experiences and Directions (OWLED '09)*, volume 529 of *CEUR-WS*, 2009.

[56] K. Kiko and C. Atkinson. A detailed comparison of UML and OWL. Technical Report TR-2008-004, University of Mannheim, 2005.

[57] R. Kost. VOM – Visual Ontology Modeler. `http://thematix.com/tools/vom/`, 2013.

[58] S. Kriglstein. OWL ontology visualization: Graphical representations of properties on the instance level. In *Proceedings of the 14th International Conference on Information Visualisation (IV '10)*, pages 92–97. IEEE, 2010.

[59] S. Krivov, R. Williams, and F. Villa. GrOWL: A tool for visualization and editing of OWL ontologies. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):54–57, 2007.

[60] M. Lanzenberger, J. Sampson, and M. Rester. Visualization in ontology tools. In *Proceedings of the International Conference on Complex, Intelligent and Software Intensive Systems (CISIS '09)*, pages 705–711. IEEE, 2009.

[61] T. Liebig and O. Noppens. OntoTrack: A semantic approach for ontology authoring. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(2-3):116–131, 2005.

[62] T. Liebig, O. Noppens, and F. W. von Henke. VIScover: Visualizing, exploring, and analysing structured data. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST '09)*, pages 259–260. IEEE, 2009.

[63] S. Lohmann. Modular Unified Tagging Ontology (MUTO). `http://purl.org/muto/core`, 2011.

[64] S. Lohmann, P. Díaz, and I. Aedo. MUTO: The modular unified tagging ontology. In *Proceedings of the 7th International Conference on Semantic Systems (I-SEMANTICS '11)*, pages 95–104. ACM, 2011.

[65] S. Lohmann, P. Heim, T. Stegemann, and J. Ziegler. The RelFinder user interface: Interactive exploration of relationships between objects of interest. In *Proceedings of the 15th International Conference on Intelligent User Interfaces (IUI '10)*, pages 421–422. ACM, 2010.

[66] S. Lohmann, V. Link, E. Marbach, and S. Negru. WebVOWL: Web-based visualization of ontologies. In P. Lambrix, E. Hyvönen, E. Blomqvist, V. Presutti, G. Qi, U. Sattler, Y. Ding, and C. Ghidini, editors, *Knowledge Engineering and Knowledge Management*, volume 8982 of *LNAI*, pages 154–158. Springer, 2015.

[67] S. Lohmann, S. Negru, and D. Bold. The ProtégéVOWL plugin: Ontology visualization for everyone. In V. Presutti, E. Blomqvist, R. Troncy, H. Sack, I. Papadakis, and A. Tordai, editors, *The Semantic Web: ESWC 2014 Satellite Events*, volume 8798 of *LNCS*, pages 395–400. Springer, 2014.

[68] S. Lohmann, S. Negru, F. Haag, and T. Ertl. VOWL 2: User-oriented visualization of ontologies. In K. Janowicz, S. Schlobach, P. Lambrix, and E. Hyvönen, editors, *Knowledge Engineering and Knowledge Management*, volume 8876 of *LNAI*, pages 266–281. Springer, 2014.

[69] Maisis. OOBIAN Insight. `http://dbpedia.oobian.com`.

[70] F. Manola and E. Miller. RDF primer. `http://www.w3.org/TR/2004/REC-rdf-primer-20040210/`, 2004.

[71] S. Mazzocchi and P. Ciccarese. Welkin. `http://simile.mit.edu/welkin/`.

[72] E. Motta, P. Mulholland, S. Peroni, M. d'Aquin, J. M. Gomez-Perez, V. Mendez, and F. Zablith. A novel approach to visualizing and navigating ontologies. In L. Aroyo, C. Welty, H. Alani, J. Taylor, A. Bernstein, L. Kagal, N. Noy, and E. Blomqvist, editors, *The Semantic Web – ISWC 2011*, volume 7031 of *LNCS*, pages 470–486. Springer, 2011.

[73] S. Negru. PersonasOnto. `http://blankdots.com/open/personasonto.html`, 2014.

[74] S. Negru and S. Buraga. Persona modeling process: From microdata-based templates to specific web ontologies. In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development (KEOD '12)*, pages 34–42. SciTePress, 2012.

[75] S. Negru, F. Haag, and S. Lohmann. Towards a unified visual notation for OWL ontologies: Insights from a comparative user study. In *Proceedings of the 9th International Conference on Semantic Systems (I-SEMANTICS '13)*, pages 73–80. ACM, 2013.

[76] S. Negru and S. Lohmann. A visual notation for the integrated representation of OWL ontologies. In *Proceedings of the 9th International Conference on Web Information Systems and Technologies (WEBIST '13)*, pages 308–315. SciTePress, 2013.

[77] S. Negru, S. Lohmann, and F. Haag. VOWL: Visual notation for OWL ontologies. `http://purl.org/vowl/`, 2014.

[78] O. Noppens and T. Liebig. Understanding large volumes of interconnected individuals by visual exploration. In E. Franconi, M. Kifer, and W. May, editors, *The Semantic Web: Research and Applications*, volume 4519 of *LNCS*, pages 799–808. Springer, 2007.

[79] J. D. Novak and A. J. Cañas. The theory underlying concept maps and how to construct them. Technical Report IHMC CmapTools 2006-01, Florida Institute for Human and Machine Cognition, 2006.

[80] OMG. Unified Modeling Language. `http://www.uml.org`.

[81] OMG. Ontology Definition Metamodel, Version 1.1. `http://www.omg.org/spec/ODM/1.1/`, 2014.

[82] F. S. Parreiras, T. Walter, and G. Gröner. Visualizing ontologies with UML-like notation. In *Proceedings of the 2nd International Workshop on Ontology-Driven Software Engineering (ODiSE '10)*, pages 4:1–4:6. ACM, 2010.

[83] S. Peroni. Graffoo specification. `http://www.essepuntato.it/graffoo/specification/current.html`, 2013.

[84] S. Peroni, E. Motta, and M. D'Aquin. Identifying key concepts in an ontology, through the integration of cognitive principles with statistical and topological measures. In J. Domingue and C. Anutariya, editors, *The Semantic Web*, volume 5367 of *LNCS*, pages 242–256. Springer, 2008.

[85] E. Pietriga. IsaViz: A visual authoring tool for rdf. `http://www.w3.org/2001/11/IsaViz/`.

[86] V. P. Sasa Kuhar. Ontology visualization for domain experts: A new solution. In *Proceedings of the 16th International Conference on Information Visualisation (IV '12)*, pages 363–369. IEEE, 2012.

[87] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages (VL '96)*, pages 336–343. IEEE, 1996.

[88] M. Sintek. OntoViz. `http://protegewiki.stanford.edu/wiki/OntoViz`, 2007.

[89] R. Somasundaram. OntoSELF: A 3D ontology visualization tool. Master thesis, Miami University, 2007.

[90] M. Sporny, G. Kellogg, and M. Lanthaler. JSON-LD 1.0 – A JSON-based Serialization for Linked Data. `http://www.w3.org/TR/json-ld/`, 2014.

[91] S. Staab and R. Studer. *Handbook on Ontologies*. Springer, 2nd edition, 2009.

[92] Stanford. Protégé. `http://protege.stanford.edu`.

[93] Stanford. Protégé Visualizations. `http://protegewiki.stanford.edu/wiki/Visualization`.

[94] M.-A. Storey, N. F. Noy, M. Musen, C. Best, R. Fergerson, and N. Ernst. Jambalaya: An interactive environment for exploring ontologies. In *Proceedings of the 7th International Conference on Intelligent User Interfaces (IUI '02)*, pages 239–239. ACM, 2002.

[95] H. Stuckenschmidt, F. van Harmelen, A. de Waard, T. Scerri, R. Bhogal, J. van Buel, I. Crowlesmith, C. Fluit, A. Kampman, J. Broekstra, and E. van Mulligen. Exploring large document repositories with RDF technology: The DOPE project. *IEEE Intelligent Systems*, 19(3):34–40, 2004.

[96] TopQuadrant. TopBraid Composer. `http://www.topquadrant.com/topbraid/`.

[97] T. Tudorache, C. Nyulas, N. F. Noy, and M. A. Musen. WebProtégé: A collaborative ontology editor and knowledge acquisition tool for the web. *Semantic Web*, 4(1):89–99, 2013.

[98] S. Vercruysse, A. Venkatesan, and M. Kuiper. OLSVis: An animated, interactive visual browser for bio-ontologies. *BMC Bioinformatics*, 13(116), 2012.

[99] W3C. W3C RDF Validation Service. `http://www.w3.org/RDF/Validator/`.

[100] W3C. OWL – Semantic Web Standards. `http://www.w3.org/2004/OWL/`, 2004.

[101] W3C OWL Working Group. OWL 2 Web Ontology Language Document Overview (Second Edition). `http://www.w3.org/TR/owl2-overview/`, 2012.

[102] L. Wachsmann. OWLPropViz. `http://protegewiki.stanford.edu/wiki/OWLPropViz`, 2008.

[103] T. D. Wang and B. Parsia. CropCircles: Topology sensitive visualization of OWL class hierarchies. In I. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. Aroyo, editors, *The Semantic Web – ISWC 2006*, volume 4273 of *LNCS*, pages 695–708. Springer, 2006.