WebVOWL: Web-Based Visualization of Ontologies

Steffen Lohmann[1(✉)], Vincent Link[1], Eduard Marbach[1], and Stefan Negru[2]

[1] Institute for Visualization and Interactive Systems (VIS), University of Stuttgart, Universitätsstr. 38, 70569 Stuttgart, Germany
steffen.lohmann@vis.uni-stuttgart.de
[2] Faculty of Computer Science, Alexandru Ioan Cuza University, Strada General Henri Mathias Berthelot 16, 700483 Iasi, Romania
stefan.negru@info.uaic.ro

Abstract. We present WebVOWL, a responsive web application for the visualization of ontologies. It implements the Visual Notation for OWL Ontologies (VOWL) and is entirely based on open web standards. The visualizations are automatically generated from JSON files, into which the ontologies need to be converted. An exemplary OWL2VOWL converter implemented in Java and based on the OWL API is currently used for this purpose. The ontologies are rendered in a force-directed graph layout according to the VOWL specification. Interaction techniques allow to explore the ontologies and customize their visualizations.

1 Introduction

Ontology visualizations are important to knowledge engineering, as they can assist in the exploration, verification, and sensemaking of ontologies [8,11]. They are particularly useful for people less familiar with ontologies, but can also give expert users a new perspective on ontologies. The Visual Notation for OWL Ontologies (VOWL) provides a visual language for the user-oriented representation of ontologies [11]. In contrast to related work, VOWL aims for an intuitive visualization that is also understandable to casual ontology users.

As a first implementation of VOWL, a plugin for the ontology editor Protégé has been developed [10]. In this paper, we present WebVOWL, an alternative and more advanced implementation of VOWL entirely based on open web standards. WebVOWL is easy to use and understand and therefore also appropriate for casual ontology users, as confirmed by an evaluation [11]. It features several interaction techniques that allow to explore the ontology and customize its visualization. It is released under the MIT license and available at http://vowl.visualdataweb.org.

WebVOWL is the first ontology visualization tool of its kind to the best of our knowledge. Like ProtégéVOWL, most other ontology visualizations are implemented in Java and are provided as plugins for ontology editors [5,8,11]. WebVOWL is independent of any ontology editor. It works with all modern web browsers that implement SVG, CSS, and JavaScript to a sufficient degree.

P. Lambrix et al. (Eds.): EKAW 2014 Satellite Events, LNAI 8982, pp. 154–158, 2015.
DOI: 10.1007/978-3-319-17966-7_21

Related tools running in web browsers, such as FlexViz [6], are based on technologies like Adobe Flex that require proprietary browser plugins. LodLive [3] is technically comparable to WebVOWL but focuses on the visual exploration of Linked Data and not on the visualization of ontologies.

In the following, we describe the technical realization and the interactive features of WebVOWL. The interested reader is referred to the related EKAW paper [11] and VOWL specification [12] for more information on VOWL. The EKAW paper also reports on a user study where VOWL and its implementation in WebVOWL have been evaluated. In the demo at EKAW, we will visualize several ontologies with WebVOWL, showcase the interactive features of the tool, and give conference attendees the opportunity to visualize and explore ontologies of interest by their own.

2 Ontology Preprocessing

Instead of being tied to a particular OWL parser, we defined a JSON schema for WebVOWL that ontologies need to be converted into. The JSON schema is optimized with regard to VOWL, i.e., its format differs from typical OWL serializations in order to enable an efficient generation of the interactive graph visualization. For that reason, it is also different from other JSON schemas that emerged in the context of the Semantic Web, such as RDF/JSON [4] and JSON-LD [13]. The VOWL-JSON file contains the TBox of the ontology, i.e., the classes, properties, and datatypes along with type information (*owl:ObjectProperty*, *xsd:dateTime*, etc.). ABox information (individuals and data values) is not considered for the time being but planned to be integrated in the future. Additional characteristics of the elements (inverse, functional, deprecated, etc.) as well as header information (ontology title, version, etc.) and optional ontology statistics (number of classes, properties, etc.) are separately listed. If no ontology statistics are provided in the JSON file, they are computed in WebVOWL at runtime.

Even though WebVOWL is based on JavaScript, the transformation of the OWL ontology into JSON does not need to be performed with JavaScript but can also be done with other programming languages. We implemented a Java-based OWL2VOWL converter, using the well-tested OWL API of the University of Manchester [7]. The converter accesses the ontology representation provided by the OWL API and transforms it into the JSON format required by WebVOWL.

3 VOWL Visualization

The VOWL visualization is generated from the JSON file at runtime. WebVOWL renders the graphical elements according to the VOWL specification, i.e., it takes the SVG code and CSS styling information provided by the specification. The force-directed graph layout is created with the JavaScript library D3 [2]. It uses a physics simulation where the forces are iteratively applied, resulting in an animation that dynamically positions the nodes of the graph. The energy of the

forces cools down in each iteration and the layout animation stops automatically after some time to provide a stable graph visualization and remove load from the processor.

An example of the resulting graph visualization is given in Figure 1. It shows a screenshot of WebVOWL (version 0.2.13) used to visualize revision 1.35 of the SIOC Core Ontology [1]. Metadata about the ontology, such as its title, namespace, author(s), and version, is shown in the sidebar, along with the ontology description and aforementioned statistics. An accordion widget helps to save screen space in the sidebar.

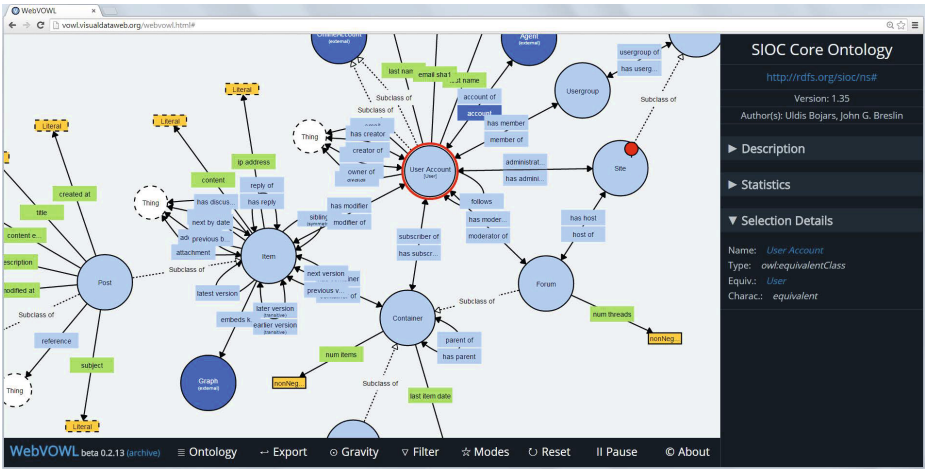

Fig. 1. SIOC Core Ontology visualized with WebVOWL. The class *User Account* has been selected and the class *Site* has been pinned.

4 Interaction and Exploration

Users can optimize the graph visualization and adapt it to their needs by rearranging nodes via drag and drop. The graph layout can also be adjusted by modifying the forces between nodes through the *gravity* settings. Datatypes have a separate force so that they can be placed in close proximity to the classes they are linked with.

Whenever a node is dragged, the force-directed algorithm is triggered and the rest of the nodes are repositioned with animated transitions. To prevent this behavior, users can *pause* the automatic layout in favor of a manual positioning of the nodes. In addition, WebVOWL implements a "pick and pin" *mode* inspired by the RelFinder [9]: It allows to decouple selected nodes from the automatic layout and pin them at custom positions on the canvas. Pinned nodes are indicated by a needle symbol (cf. class *Site* in Figure 1) that can be removed to recouple the nodes with the force-directed layout.

Users can zoom in and out, pan the background, and select elements in the graph visualization. Details about selected elements are displayed in the accordion widget of the sidebar, such as for the selected class *User Account* in Figure 1. IRIs are provided as hyperlinks that can be opened in another tab of the web browser for further exploration. Related elements (e.g., subproperties) are interactively highlighted according to the VOWL specification.

Another group of interactive features are *filters* that help to reduce the size of the VOWL graph and increase its visual scalability. Two filters are currently implemented, one for datatypes and another for "solitary subclasses", which are subclasses only connected to their superclass but not to other classes.

Finally, WebVOWL allows to export the ontology visualization as SVG image that can be opened in other applications, scaled without loss of quality, edited, shared, and printed.

5 Conclusion and Future Work

The current WebVOWL implementation already considers a large portion of the OWL 2 language constructs. We aim to further extend the VOWL specification, OWL2VOWL converter, and WebVOWL to cover as many OWL constructs as possible. Furthermore, we plan to develop features that allow to hide selected subgraphs for a more compact visualization of large ontologies. Future work will also need to address features that help to keep a "mental map", for example, by allowing users to save a VOWL visualization with all its settings as an annotated JSON file. Other open issues are multi-language support, improved search functionality, and the inclusion of ABox data in the JSON schema, converter, and WebVOWL user interface.

WebVOWL can be used in different interaction contexts, including settings with touch interfaces. For instance, zooming can either be performed with the mouse wheel, a double click/tap, or a two fingers zooming gesture (on multi-touch devices). A desirable application of WebVOWL would be a web service where users could upload arbitrary ontologies to get them visualized. Apart from that, we hope that WebVOWL will be useful to others and in related projects, such as ontology editing, ontology alignment, or Linked Data exploration.

References

1. Bojārs, U., Breslin, J.: SIOC Core Ontology Specification. http://rdfs.org/sioc/spec/ (2010)
2. Bostock, M., Ogievetsky, V., Heer, J.: D3: Data-driven documents. IEEE Transactions on Visualization and Computer Graphics **17**(12), 2301–2309 (2011)
3. Camarda, D.V., Mazzini, S., Antonuccio, A.: LodLive, exploring the web of data. In: I-SEMANTICS 2012, pp. 197–200. ACM (2012)
4. Davis, I., Steiner, T., Hors, A.L.: RDF 1.1 JSON Alternate Serialization (RDF/JSON). http://www.w3.org/TR/rdf-json/ (2013)

5. Dudáš, M., Zamazal, O., Svátek, V.: Roadmapping and navigating in the ontology visualization landscape. In: Janowicz, K., Schlobach, S., Lambrix, P., Hyvönen, E. (eds.) EKAW 2014. LNCS, vol. 8876, pp. 137–152. Springer, Heidelberg (2014)
6. Falconer, S.M., Callendar, C., Storey, M.-A.: A visualization service for the semantic web. In: Cimiano, P., Pinto, H.S. (eds.) EKAW 2010. LNCS, vol. 6317, pp. 554–564. Springer, Heidelberg (2010)
7. Horridge, M., Bechhofer, S.: The OWL API: A java API for OWL ontologies. Semantic Web **2**(1), 11–21 (2011)
8. Lanzenberger, M., Sampson, J., Rester, M.: Visualization in ontology tools. In: CISIS 2009, pp. 705–711. IEEE (2009)
9. Lohmann, S., Heim, P., Stegemann, T., Ziegler, J.: The RelFinder user interface: interactive exploration of relationships between objects of interest. In: IUI 2010, pp. 421–422. ACM (2010)
10. Lohmann, S., Negru, S., Bold, D.: The protégéVOWL plugin: ontology visualization for everyone. In: Presutti, V., Blomqvist, E., Troncy, R., Sack, H., Papadakis, I., Tordai, A. (eds.) ESWC 2014 Satellite Events. LNCS, vol. 8798, pp. 394–399. Springer, Heidelberg (2014)
11. Lohmann, S., Negru, S., Haag, F., Ertl, T.: VOWL 2: user-oriented visualization of ontologies. In: Janowicz, K., Schlobach, S., Lambrix, P., Hyvönen, E. (eds.) EKAW 2014. LNCS, vol. 8876, pp. 266–281. Springer, Heidelberg (2014)
12. Negru, S., Lohmann, S., Haag, F.: VOWL: Visual notation for OWL ontologies. http://purl.org/vowl/spec/ (2014)
13. Sporny, M., Kellogg, G., Lanthaler, M.: JSON-LD 1.0 - A JSON-based Serialization for Linked Data. http://www.w3.org/TR/json-ld/ (2014)