Towards an Extensible Web-Based Open-Source Graphical Ontology Editing Framework

Sergejs Kozlovičs*

Institute of Mathematics and Computer Science, University of Latvia (Riga, Latvia)
Raina blvd. 29, LV-1459, Riga, Latvia
sergejs.kozlovics@lumii.lv

Abstract. We describe work in progress towards an 1) extensible, 2) web-based, and 3) open-source ontology editor. The domain logic is the same as in our classical desktop-based graphical ontology editor OWLGrEd. Thus, we are bringing the same set of features to the web environment. Moreover, it becomes possible to implement additional web-based features, such as collaborative editing and integration with third-party applications.

Keywords: OWLGrEd · web · graphical ontology editor · extensible ontology framework

1 Introduction

Visual presentations have a strong psychological advantage over textual languages. Thus, it seems unnatural that there are well-defined standards for storing ontologies textually, but no de facto standard for representing them graphically. Luckily, there are multiple tools trying to fill this gap, including OWLViz[1], TopBraid Composer[2], OntoDia, and VOWL [10,9].

OWLGrEd[3] is a Windows desktop tool, which proved to be a powerful graphical editor for OWL 2.0 ontologies [1,12]. It uses UML-like graph diagrams for main OWL constructs and Manchester syntax for auxiliary constructs [11,14]. OWLGrEd is customizable and extensible. In a recent review, where 33 ontology visualization tools were compared, only free OWLGrEd and commercial TopBraid (worth $3450 per license in 2019) supported all the OWL constructs considered [3]. With this demo, we present work in progress of moving OWLGrEd to the web environment. The web-based OWLGrEd is expected to become a Visio-like "Google Docs" for editing ontologies graphically.

* Copyright © 2019 for this paper by its author. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

[1] https://github.com/protegeproject/owlviz

[2] https://www.topquadrant.com/products/topbraid-composer/

[3] http://owlgred.lumii.lv/

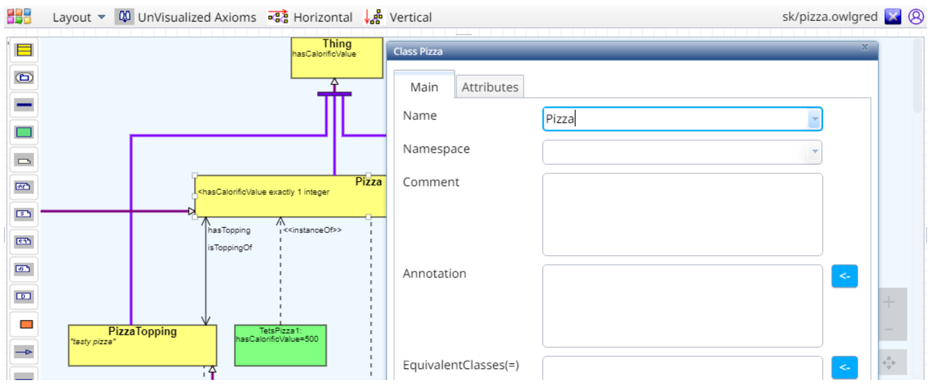

Fig. 1. Client-side user interface in the web-based OWLGrEd (a real screenshot).

2 OWLGrEd Features

In OWLGrEd, most OWL constructs have a one-to-one mapping to UML constructs (e.g., OWL classes map to UML classes, while OWL properties map to UML associations and attributes). OWLGrEd has a compact syntax, where unnecessary anonymous classes are not visualized. The main strength of OWLGrEd is the ability to tune graphical presentations to meet various specific needs. For instance, domain-specific profiles can be used to modify graphical elements depending on their annotations (e.g., to specify a diamond shape at one end of the connector for UML-like compositions or to alter shapes of particular classes) [2]. The visual appearance of elements (such as their shapes and colors as well as the layout and structure) can be changed either manually, or programmatically via predefined or user-defined extensions [4].

Our goal is to retain the same set of features as in the web-based OWLGrEd.

3 Moving OWLGrEd to the Web

To simplify the transition to the web and to minimize the maintenance costs of existing OWLGrEd features, we retained the code implementing the domain logic in the classical desktop OWLGrEd. However, we still had to perform the next two steps.

First, we had to move to the web browser the two main UI components, the graph diagram editor and the dialog engine (see Figure 1).[4] Both components are essential for a typical scenario: the user starts with an empty diagram or imports an existing ontology, then adjusts the diagram incrementally (by adding, moving, and deleting diagram elements and by specifying their properties in dialog windows), and finally exports the ontology in some serialization format.

[4] We use the *ajoo* library for editing graph-like diagrams [13]. We use the DoJo Toolkit for visualizing dialog windows, `https://dojotoolkit.org/`.

The graph diagram editor relies on an advanced layout algorithm to obtain the initial layout (e.g., to arrange the whole diagram hierarchically, symmetrically, or universally) as well as to adjust the existing layout incrementally while the user edits the diagram [5]. The dialog engine, in its turn, uses a variation of the same algorithm to visualize GUI dialogs on-the-fly (some of them can be generated by OWLGrEd extensions at runtime) [7].

Second, we had to deal with network-related issues such as user management, connection management, data synchronization, and client-server code interoperation. For that, we rely on webAppOS[5], our open-source cloud-based infrastructure for developing and running web applications [8]. The infrastructure was specially designed for migrating existing standalone applications by providing a single-PC illusion for them. OWLGrEd is a guinea pig within the webAppOS project. All client-side and server-side code interoperation is factored out by webAppOS. Besides, webAppOS provides a means to upload, download, and open OWLGrEd projects in a way similar to Windows Explorer or macOS Finder. Existing ontologies (.owl files) can be imported into the web-based OWLGrEd and exported back in the desired serialization format by means of the web-based "Browse for file" and "Save as" dialogs, which provide the same user experience as in the classical desktop-based OWLGrEd.

4 Bringing Additional Features

In webAppOS, data can be synchronized not only between the server and the client but also with other authorized clients (e.g., debuggers or third-party applications). Thus, the full OWLGrEd project currently being edited (including all ontology diagrams) can be manipulated from the outside via a web socket using a documented protocol. One of such third-party applications can be a collaborative bot for synchronizing diagrams between multiple editors.

Similar features have already been offered by WebProtégé, which uses a proprietary webhooks-based service Slack[6] for that [6]. Our goal is not to compete, but complement WebProtégé with the ability to edit ontologies *graphically*. We are considering the idea of using WebProtégé as domain storage for ontologies being edited graphically in the web-based OWLGrEd.

5 The Demo

The web-based OWLGrEd is expected to be fully open-source, with the ability to launch it on a private server or as a cross-platform standalone desktop application, with no technical and licensing obstacles of the classical OWLGrEd.

The demo reflecting the current work in progress can be found at `http://webappos.org/owlgred`. We show how to 1) import and adjust an existing ontology; 2) create an ontology from scratch and export it to an .owl file; 3)

[5] `http://webappos.org`
[6] `https://slack.com`

manipulate the diagram programmatically from the outside; 4) upload and open an OWLGrEd project from the webAppOS desktop.

Acknowledgments. The work has been supported by European Regional Development Fund within the project #1.1.1.2/16/I/001, application #1.1.1.2/VIAA/1/16/214 "Model-Based Web Application Infrastructure with Cloud Technology Support".

References

1. Barzdins, J., Barzdins, G., Cerans, K., Liepins, R., Sprogis, A.: OWLGrEd: a UML style graphical notation and editor for OWL 2. In: Proceedings of OWLED 2010 (2010)
2. Cerans, K., Liepins, R., Sprogis, A., Ovcinnikova, J., Barzdins, G.: Domain-specific OWL ontology visualization with OWLGrEd. In: The Semantic Web: ESWC 2012 Satellite Events. pp. 419–424. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
3. Dudáš, M., Lohmann, S., Svátek, V., Pavlov, D.: Ontology visualization methods and tools: a survey of the state of the art. The Knowledge Engineering Review 33, e10 (2018)
4. Čerāns, K., Ovčiņņikova, J., Liepiņš, R., Grasmanis, M.: Extensible visualizations of ontologies in OWLGrEd. ESWC 2019 Satellite Events (2019)
5. Freivalds, K., Ķikusts, P.: Optimum layout adjustment supporting ordering constraints in graph-like diagram drawing. In: Proceedings of the Latvian Academy of Sciences, Section B. vol. 55, pp. 43–51 (2001)
6. Horridge, M., Gonçalves, R.S., Nyulas, C.I., Tudorache, T., Musen, M.A.: WebProtégé: A cloud-based ontology editor. In: Companion Proceedings of The 2019 World Wide Web Conference. pp. 686–689. WWW '19, ACM, New York, NY, USA (2019)
7. Kozlovics, S.: Calculating The Layout For Dialog Windows Specified As Models. In: Scientific Papers, University of Latvia. vol. 787, pp. 106–124 (2012)
8. Kozlovičs, S.: The web computer and its operating system: A new approach for creating web applications. Proceedings of the 15th International Conference on Web Information Systems and Technologies (2019)
9. Lohmann, S., Negru, S., Haag, F., Ertl, T.: Visualizing ontologies with VOWL. Semantic Web 7(4), 399–419 (2016)
10. Mouromtsev, D., Pavlov, D., Emelyanov, Y., Morozov, A., Razdyakonov, D., Galkin, M.: The simple, web-based tool for visualization and sharing of semantic data and ontologies. In: ISWC P&D 2015, CEUR (10 2015)
11. OMG: OMG Unified Modeling Language (OMG UML), Version 2.5. OMG Document Number: formal/15-03-01, `http://www.omg.org/spec/UML`
12. Ovčiņņikova, J., Čerāns, K.: Advanced UML style visualization of OWL ontologies. Proceedings of the Second International Workshop on Visualization and Interaction for Ontologies and Linked Data co-located with the 15th International Semantic Web Conference (ISWC 2016) CEUR 1704, 136–142 (2016)
13. Sprogis, A.: ajoo: Web based framework for domain specific modeling tools. Frontiers in Artificial Intelligence and Applications Volume 291: Databases and Information Systems IX (2016)
14. W3C: OWL 2 web ontology language Manchester syntax (second edition) (11 December 2012), `https://www.w3.org/TR/owl2-manchester-syntax/`