OWL2Graffoo: Import Ontology in diagrams.net with Graffoo

Jakub Duchateau[1 [0009−0009−5090−8192]] and Christophe Debruyne[1 [0000−0003−4734−3847]]

Montefiore Institute, University of Liège, Belgium
{Jakub.Duchateau,C.Debruyne}@uliege.be

Abstract. We present a plugin for the popular diagramming tool diagrams.net (formerly draw.io) that imports OWL ontologies using the Graffoo visual syntax. Using a generic diagramming tool allows further manual modification and layout customisation of the ontology visualisation. The layout data can be stored with the ontology, allowing a workflow of recreating the diagram, while keeping the layout. We also introduce "Graffoo Compact", a more compact variant of Graffoo aimed at reducing the space required when visualising complex ontologies.

Keywords: Ontology · Diagrams · Visualization · Graffoo.

1 Introduction

Creating clear and effective visualizations of ontologies is essential for documenting the ontology. While various tools exist for ontology engineering, visualisation, or diagramming [3], we lacked a single solution combining three capabilities: converting an OWL ontology into a diagram using an established notation, permitting layout customisation, and preserving this layout when updating the ontology. The necessity for this combined functionality was directly motivated by our work on the RML Execution Report (RER) vocabulary[1], where existing tools did not support our iterative workflow.

One visualisation of RDF and in particular of OWL ontologies is Graffoo [1]. It was proposed with DiTTO to convert diagrams into OWL, but the reverse was missing. We therefore implemented an open-source import plugin for diagrams.net (formerly draw.io), utilising Luigi Asprino's shape library[2].

We contribute the plugin source code[3] under an Apache 2.0 license, and *Graffoo Compact*, a compact notation and layouting guidelines for Graffoo.

2 Workflow

Existing ontology visualisation systems often couple the modelling and visualisation steps. In contrast, our tool allows users to develop their ontology in

[1] RER namespace is http://w3id.org/dre/rer.
[2] https://github.com/luigi-asprino/Graffoo4DrawIO
[3] **https://github.com/jduchateau/OWL2Graffoo**

their preferred editor (e.g., Protégé[4]) and then import it into diagrams.net for visualisation and manual refinement of the diagram.

During import, the plugin generates classes (standard Graffoo or a UML variant, see Section 4), natively supports data, object, and annotation properties, and creates a visual prefix table. For the initial placement of the elements, the plugin applies one of diagrams.net's built-in layout algorithms, a fast organic layout, to provide a basic starting point. The diagram author is then expected to manually refine the diagram's layout to their specific needs. Note that Graffoo's "facility" properties, which use dotted lines to illustrate informal or relaxed property usage for human comprehension, are not generated automatically, as they represent tacit knowledge rather than strict formal OWL-axioms, and must be added manually. Additionally, each generated class and property includes a hyperlink to its URI, remaining clickable in exported SVG and PDF visualisations to let users explore the ontology. This transforms a static diagram into a hyperlinked object, allowing one to easily explore the contents of an ontology by following the IRIs.

To support iterative ontology development, the plugin allows exporting a manually adjusted layout as RDF. Users can subsequently append these layout triples directly to their primary ontology file. This ensures that when the combined ontology is re-imported, the manually adjusted layout is gracefully preserved, facilitating an iterative documentation workflow.

We provide a dedicated deployment hosted on GitLab Pages.[5] This deployment is pre-configured with the Graffoo shape library and our plugin; a link can be found in the source code[3].

3 Application: RER Vocabulary

The development of the RML Execution Report (RER) vocabulary served as the primary motivation for this work. To prepare the ontology for public release, we needed a stable, static diagram for the report and documentation, as well as a WebVOWL [4] diagram for an interactive, dynamic view. We adopted the Graffoo notation for its adoption within the Semantic Web community and its clarity. However, we quickly ran into the problem of large diagrams due to their notation. This subsequently inspired the development of *Graffoo Compact*, in which we devised more compact representations of certain ontological aspects. The advantage of our plugin is that its layout export feature allowed us to refine the RER ontology while keeping the documentation diagram in sync. In other words, we gained time by limiting the generation of diagrams for documentation as recreating the diagram after updates required minimal manual intervention.[6]

[4] https://protege.stanford.edu

[5] Because custom plugins cannot be loaded directly into the official Web application of diagrams.net

[6] Requiring manual interventions was mostly limited to edge-to-edge connections (such as for representing sub-properties), which are not natively supported by diagrams.net, and thus have to be done manually with absolute positions.

4 Graffoo Compact

As diagrams in Graffoo can take quite a lot of space when ontologies are larger, we introduce *Graffoo Compact* to more efficiently use the space on a canvas. The opportunity for a compact notation was already identified as future work in the initial Graffoo article [1], but, to the best of our knowledge, no such compact version was yet published. Graffoo Compact (Fig. 1) revises the initial specification by addressing cases where classes have numerous data properties or subclasses.

First, the UML variant allows classes with multiple data properties to list them directly within the class box, similar to UML attributes. Second, drawing inspiration from WebVOWL's compact notation [4], the subclass and property variants replace the generic labelled directed edges for predicates with specific arrow styles, reducing the number of labels on the diagram.

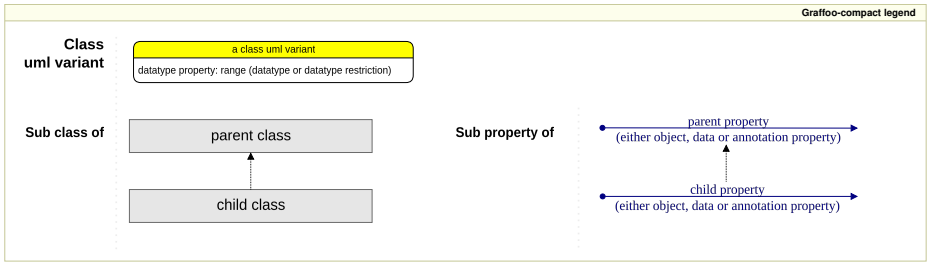

Fig. 1. We introduce UML-like variants for classes and simplified subclass/subproperty arrows, for the original Graffoo legend, we refer to [1].

5 Related Work

Multiple tools have been proposed in ontology engineering. [3] has done a survey of tool usage in 2024. More specifically, such as OntoGraf (a bundled plugin of Protégé), Graffoo [1] which is the basis of this work, and WebVOWL [4]. While WebVOWL operates primarily as an interactive ontology viewer, our solution uses diagrams.net to make diagram customizable. Contrary to DiTTO [2], which generates OWL from diagrams, this plugin imports an OWL ontology and generates the visualisation in diagrams.net. Related work focuses on generating OWL from diagrams, whereas we focus on generating diagrams from RDF and on keeping track of layout when diagrams need to be recreated as ontologies evolve over time.

6 Conclusions

This plugin enables the creation of ontology visualisation diagrams and updating them as the ontology evolves. By enabling the import of OWL ontologies into diagrams.net, developers can produce customizable visualisation artifacts. When the ontology is updated, the diagram can be recreated with the layout preserved from previous iterations. We used this tool to generate and maintain RER's graphical representation of the ontology. Whilst the plugin answered our needs for quickly generating visual representations of a vocabulary, we acknowledge that the plugin and the *Graffoo Compact* notation have not yet undergone a user evaluation with a broader user base, and we recognize the potential of additional features, such as background colors based on prefixes and finding a workaround for the placement of arrows between arrows or storing multiple positions for targets of data and object properties.

Acknowledgments. This work was supported by the Fonds de la Recherche Scientifique – FNRS under Grant n° MIS F.4016.24.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Falco, R., Gangemi, A., Peroni, S., Shotton, D., Vitali, F.: Modelling OWL Ontologies with Graffoo. In: Presutti, V., Blomqvist, E., Troncy, R., Sack, H., Papadakis, I., Tordai, A. (eds.) The Semantic Web: ESWC 2014 Satellite Events. pp. 320–325. Springer International Publishing, Cham (2014). https://doi.org/10.1007/978-3-319-11955-7_42
2. Gangemi, A., Peroni, S.: DiTTO: Diagrams Transformation inTo OWL. In: Blomqvist, E., Groza, T. (eds.) Proceedings of the ISWC 2013 Posters & Demonstrations Track, Sydney, Australia, October 23, 2013. CEUR Workshop Proceedings, vol. 1035, pp. 5–8. CEUR-WS.org (2013)
3. Gomes, C.A.D., Monfardini, G.K.d.S.Q., Salamon, J.S., Sangali, R.d.S., Timoteo, I.S.R., Barcellos, M., Souza, V.E.S.: Investigating tool usage in ontology engineering: A survey. In: Fonseca, C.M., Emygdio, J.L. (eds.) Proceedings of the 17th Seminar on Ontology Research in Brazil (ONTOBRAS 2024) and 8th Doctoral and Masters Consortium on Ontologies (WTDO 2024), Vitória, Brazil, October 07-10, 2024. CEUR Workshop Proceedings, vol. 3905. CEUR-WS.org (2024), https://ceur-ws.org/Vol-3905/paper6.pdf
4. Lohmann, S., Negru, S., Haag, F., Ertl, T.: Visualizing ontologies with VOWL. Semantic Web **7**(4), 399–419 (2016). https://doi.org/10.3233/SW-150200