Modelling OWL ontologies with Graffoo

Riccardo Falco[1], Aldo Gangemi[2,3], Silvio Peroni[1,2],
David Shotton[4], and Fabio Vitali[1]

[1] Department of Computer Science and Engineering, University of Bologna (Italy)
riccardo.falco@studio.unibo.it, silvio.peroni@unibo.it, fabio@cs.unibo.it
[2] STLab-ISTC, Consiglio Nazionale delle Ricerche (Italy)
aldo.gangemi@cnr.it
[3] Laboratoire d'Informatique de Paris Nord, Université Paris 13 (France)
[4] Oxford e-Research Centre, University of Oxford (UK)
david.shotton@oerc.ox.ac.uk

Abstract. In this paper we introduce *Graffoo*, i.e., a graphical notation to develop OWL ontologies by means of *yEd*, a free editor for diagrams.

Keywords: DiTTO, Graffoo, OWL, yEd, graphical notation

1 Introduction

In many contexts where the use of formal languages is needed (e.g., software development, GUI implementation, ontology engineering), the adoption of an appropriate graphical notation simplifies the design and property checking of a system since it enables an overview of the system that is difficult to have using textual syntaxes. Within the Semantic Web domain, this seems to be particularly true when developing ontologies: graphical languages, among the others, seem to support ontology modelling and understanding as well as the discussion between all the involved actors (i.e., domain experts, knowledge engineers and final users). Designing a graphical notation specific for OWL requires to consider what are the appropriate requirements that such notation should address according to different kinds of players coming from several academic and industrial contexts. At various SW conferences and workshops, we started brainstorming informally about the ideal features of a graphical notation for OWL ontologies, and we identified the following requirements:

1. *Oriented to OWL.* The notation should address all the capabilities of OWL.
2. *Graphical elements to make modelling and understanding easy.* It should facilitate users in dealing with *modelling* and *understanding* of ontologies.
3. *Colours are a complementary aid, not a fundamental discriminant.* Although the use of different colours helps to reduce the cognitive effort of users [1] in the aforementioned activities, each graphical element of the notation should be clearly recognisable even when it is presented in a grey scale.
4. *Invent the notation, not the editor.* The effort of creators of a notation should concern the development of the notation itself, since diagram applications usually provide mechanisms to extend them with additional notations easily.

Existing tools and notations developed to deal with modelling and understanding activities seem to be hardly appropriate to address all the aforementioned features – either because they do not address all OWL 2 capabilities [5], or because they were developed to address modelling tasks (e.g., ontology editors [3]) or understanding tasks (e.g., documentation generators [6]) but never both.

The aim of this demonstration is to show how to create OWL-aware ontology diagrams by using *Graffoo* (Section 2), a graphical notation for OWL ontologies that tries to address the aforementioned requirements. We accompany the discussion of Graffoo with preliminary outcomes of a comparative user testing session, and we briefly present the extension of DiTTO [2] (i.e., an online service that converts diagrams into OWL ontologies) we developed to convert Graffoo diagrams into proper OWL sources in Manchester Syntax. Finally, in Section 3, we conclude the paper sketching out some future works.

2 A graphical framework for OWL ontologies

The *Graphical Framework For OWL Ontologies*, a.k.a. *Graffoo*[5], is a graphical notation that addresses all the requirements introduced in Section 1. All the graphical elements of Graffoo, summarised in Fig. 1, have been developed using the standard library of *yEd*[6], i.e., a free diagram editor running on Windows, Mac and Linux. The Graffoo graphical elements are available online and can be loaded as a proper section in the yEd palette, as shown in Fig. 2. To add the Graffoo graphical elements to yEd one needs to download the Graffoo *.graphml* file[7], and then to import it as a palette – by selecting that file in the window that appears clicking on "Edit / Manage Palette / Import Section" in the yEd tool bar, and then by adding and including it in the available palettes.

Graphical elements. All the ontological entities (i.e., ontologies, classes, properties, datatypes, and individuals) can be defined either as an *IRI* surrounded by angular brackets (e.g., `<http://xmlns.com/foaf/0.1/Person>`) or as a *CURIE* with a *prefix* (e.g., `foaf:Person`). All the prefixes can be defined within a particular box (entitled "Prefixes") as a list of prefix-IRI pairs (e.g., `foaf: <http://xmlns.com/foaf/0.1/>`). In Graffoo there are two different kinds of graphical elements, i.e., *blocks* (or nodes) and *arcs*. Blocks are used to define classes and class restrictions (yellow rectangles with solid and dotted borders respectively), datatypes and datatype restrictions (green rhomboids with solid and dotted borders respectively), individuals (pink circles with solid black border), ontologies (boxes with light-blue heading and dotted black border), additional axioms in Manchester Syntax for all those constructs that are not directly supported by a particular graphical element (light-blue and folded boxes), and rules (boxes with light-grey heading and black dashed border). Arcs

[5] Available at http://www.essepuntato.it/graffoo.
[6] Available at http://www.yworks.com/en/products_yed_about.html.
[7] Available at http://www.essepuntato.it/graffoo/sources.

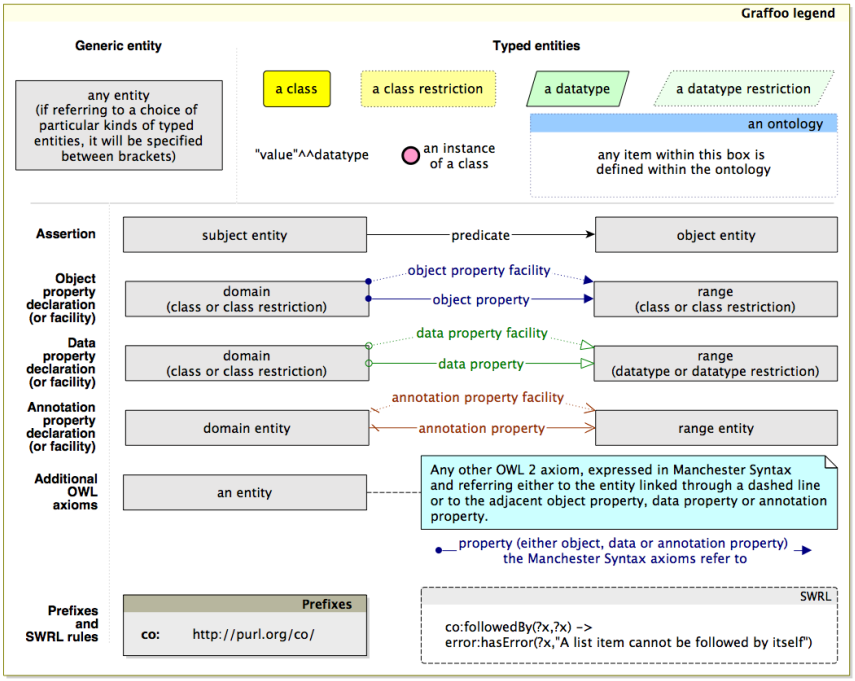

Fig. 1. The full set of graphical elements of Graffoo.

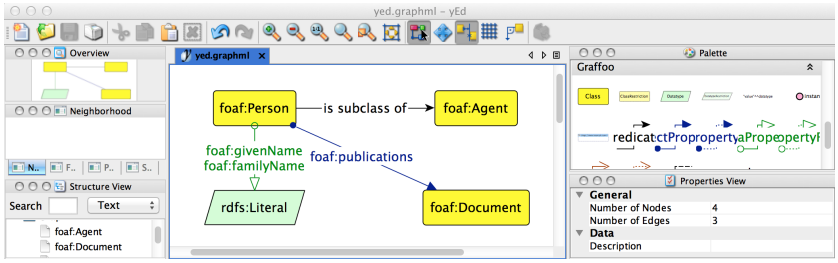

Fig. 2. The Graffoo palette in yEd.

are used to define assertions (black lines ending with a solid arrow[8]), annotation properties (orange lines beginning with backslash and ending with a dashed arrow), data properties (green lines beginning with an empty circle and ending with an empty arrow), and object properties (blue lines beginning with a solid

[8] All the assertions defining typical OWL axioms, such as sub-class axioms, equivalent axioms, etc., can be expressed by natural language names in Graffoo (e.g., by using "is subclass of" instead of "rdfs:subClassOf", "is equivalent to" instead of "owl:equivalentClass" and "owl:equivalentProperty", etc.).

circle and ending with a solid arrow). In addition to these graphical elements, there is a particular kind of graphical element (named *property facilities*, i.e., arcs having dotted border and referring to data, object and annotation properties), that were studied to decrease the cognitive effort of users when understanding an ontology. For instance, they allow one to say explicitly that a certain property can be used in the context of two classes without declaring them as domain and range. The full specification of Graffoo graphical elements is available at `http://www.essepuntato.it/graffoo/specification/current.html`.

Usability. As a preliminary study, we performed a comparative user testing session so as to gather some evidences on the usability of Graffoo when modelling OWL ontologies. We asked eleven PhD students in Computer Science and Law (with no expertise in ontologies and Semantic Web technologies) to use four different tools – i.e., the Manchester Syntax, Protégé, E/R as introduced in [2], and Graffoo – for modelling small OWL ontologies (with 5-15 entities). All the four tools were appropriately introduced to PhD students during six lectures of two hours each and, at the end of the last lecture, we asked them to answer a questionary containing ten likert questions according to the *System Usability Scale* (*SUS*)[9] – sub-scales of pure *Usability* and pure *Learnability* were considered as well [4]. As shown in Table 1, the SUS score for Graffoo was the highest (58.9, in a 0-100 range), meaning that it was perceived more usable than the others – even if none of the differences between SUS scores was statistically significant.

Table 1. SUS (mean) scores of all the notations/tools involved.

Tool	SUS	Learnability	Usability
Manchester Syntax	45.9 (s.d. 15.9)	45.4 (s.d. 21.1)	46.0 (s.d. 16.0)
Protégé	50.9 (s.d. 15.5)	45.4 (s.d. 20.4)	52.3 (s.d. 15.2)
E/R	50.4 (s.d. 17.6)	51.1 (s.d. 22.7)	50.3 (s.d. 17.7)
Graffoo	58.9 (s.d. 16.0)	54.4 (s.d. 23.2)	59.1 (s.d. 15.3)

DiTTO extension. *DiTTO* [2] is a Web service available at `http://www.essepuntato.it/ditto` developed, originally, to transform E/R diagrams into OWL ontologies according to three distinct conversion strategies. The core of DiTTO – i.e., a set XSLT 2.0 documents included in a Java Web Application Archive (i.e., a WAR file) served as a Tomcat application – has been recently extended with additional XSLT documents that apply several rewriting templates to the source file of the Graffoo diagram created through yEd, and return the converted OWL ontology in Manchester Syntax. This extension allows one to use some features available in yEd to simplify the work of ontology engineers. In particular, it is possible to add annotations to ontological entities by using the preference panel (as shown in Fig. 3), avoiding the use of the graphical element for additional axioms to specify such annotations as common axioms.

[9] All data are available at http://www.essepuntato.it/graffoo/preliminary-test.

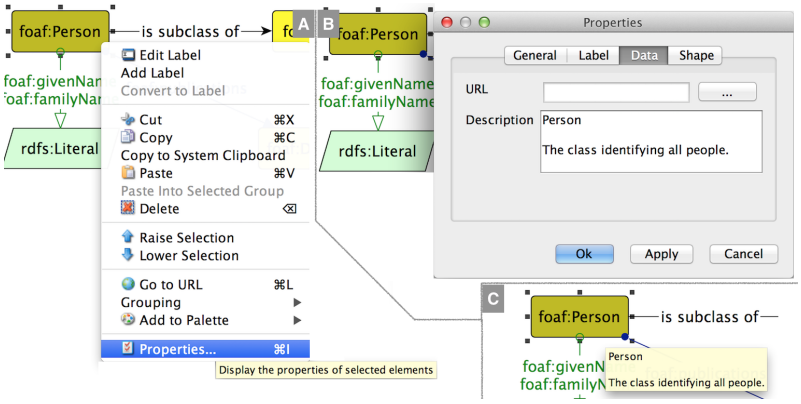

Fig. 3. The steps to add *rdfs:label* and *rdfs:comment* annotations to ontological entities. A: right-click on the entity to access its properties; B: add label (first line) and comment (following lines) as free text in the "Description" field of the "Data" panel. C: hover the pointer on the entity to show the annotations.

3 Conclusions and future works

In this paper we have introduced Graffoo, a graphical notation to model OWL ontologies. We have shown Graffoo graphical elements, we have described preliminary outcomes of a user testing session, and we have presented an extension of DiTTO for transforming Graffoo diagrams into OWL. We plan to extend Graffoo (and the related DiTTO extension) with a new compact syntax (in order to create, for instance, subclass axioms involving class restrictions when declaring properties), to generate Graffoo diagrams from OWL ontologies, and to perform additional usability evaluations to confirm the results sketched out herein.

References

1. Chalmers, P. A. (2003). The role of cognitive theory in human–computer interface. Computers in Human Behavior, 19(5): 593–607. DOI: 10.1016/S0747-5632(02)00086-9
2. Gangemi, A., & Peroni, S. (2013). DiTTO: Diagrams Transformation inTo OWL. In Proceedings of the ISWC 2013 Posters & Demonstrations Track. Retrieved from http://ceur-ws.org/Vol-1035/iswc2013_demo_2.pdf
3. García-Barriocanal, E., Sicilia, M. A., & Sánchez-Alonso, S. (2005). Usability Evaluation of Ontology Editors. Knowledge Organization, 32(1): 1–9.
4. Lewis, J. R., & Sauro, J. (2009). The Factor Structure of the System Usability Scale. In Proceedings of HCSE 2009: 94–103. DOI: 10.1007/978-3-642-02806-9_12
5. Negru, S., Haag, F., & Lohmann, S. (2013). Towards a Unified Visual Notation for OWL Ontologies: Insights from a Comparative User Study. In Proceedings of i-Semantics 2013: 73–80. DOI: 10.1145/2506182.2506192
6. Peroni, S., Shotton, D., & Vitali, F. (2013). Tools for the Automatic Generation of Ontology Documentation: A Task-Based Evaluation. International Journal on Semantic Web and Information Systems, 9(1): 21–44. DOI: 10.4018/jswis.2013010102