Did You Validate Your Ontology? OOPS!

María Poveda-Villalón(✉), Mari Carmen Suárez-Figueroa, and Asunción Gómez-Pérez

Departamento de Inteligencia Artificial, Ontology Engineering Group, Facultad de Informática, Universidad Politécnica de Madrid, Madrid, Spain
{mpoveda,mcsuarez,asun}@fi.upm.es

Abstract. The application of methodologies for building ontologies can improve ontology quality. However, such quality is not guaranteed because of the difficulties involved in ontology modelling. These difficulties are related to the inclusion of anomalies or bad practices within the ontology development. Several authors have provided lists of typical anomalies detected in ontologies during the last decade. In this context, our aim in this paper is to describe OOPS! (OntOlogy Pitfall Scanner!), a tool for detecting pitfalls in ontologies.

Keywords: Pitfalls · Bad practices · Ontology evaluation · Ontology engineering

1 Introduction

The growing interest during the last decades of practitioners in ontology development methodologies has facilitated major progress, transforming the art of building ontologies into an engineering activity. The correct application of such methodologies benefits ontology quality. However, such quality is not totally guaranteed because developers must tackle a wide range of difficulties and handicaps when modelling ontologies [1, 2, 5, 8]. These difficulties can imply the appearance of the so-called anomalies or bad practices in ontologies. Therefore, it is important to evaluate the ontologies before using or reusing them in other ontologies or semantic applications.

One of the crucial issues in ontology evaluation is the identification of anomalies in the ontologies. In this regard, it is worth mentioning that Rector et al. [8] describe a set of common errors made by developers during the ontology modelling. Moreover, Gómez-Pérez [4] proposes a classification of errors identified during the evaluation of different features such as consistency, completeness, and conciseness in ontology taxonomies. Finally, Poveda et al. [7] identify an initial catalogue of common pitfalls.

In this context, our goal within this paper is to present an automated tool to help ontology practitioners by detecting common pitfalls during the ontology development. This tool is called OOPS! (OntOlogy Pitfall Scanner!) and represents a new option for ontology developers within ontology evaluation tools as it enlarges the list of errors detected by most recent and available works (e.g. MoKi[1] [6] and XD Analyzer[2]). In addition, OOPS! can be executed independently of the ontology development platform

[1] https://moki.fbk.eu/moki/tryitout/index.php/Main_Page (Last visit on 14-04-2012).
[2] http://neon-toolkit.org/wiki/XDTools (Last visit on: 14-04-2012).

E. Simperl et al. (Eds.): ESWC 2012 Satellite Events, LNCS 7540, pp. 402–407, 2015.
DOI: 10.1007/978-3-662-46641-4_35

without configuration or installation and it also works with main web browsers (Firefox, Chrome, Safari and Internet Explorer[3]).

The remainder of this paper is structured as follows: Sect. 2 presents the main OOPS! features while Sect. 3 describes its architecture. Finally Sect. 4 outlines some conclusions and future steps to improve OOPS!.

2 OOPS! Features

OOPS! scans ontologies looking for potential pitfalls that could lead to modelling errors [7]. OOPS! is intended to be used by ontology developers during the ontology validation activity, particularly during the diagnosis task. Its main functionality is to analyze ontologies[4] (a) via URL in which an ontology is located or (b) via text input containing the RDF code of the ontology. As a result of the analysis, OOPS! informs developers about which elements of the ontology are possibly affected by pitfalls.

Figure 1 shows OOPS! home page[5] where a user can enter an ontology to be analyzed via URL or by pasting RDF code in the box. This page also presents a brief description of OOPS!.

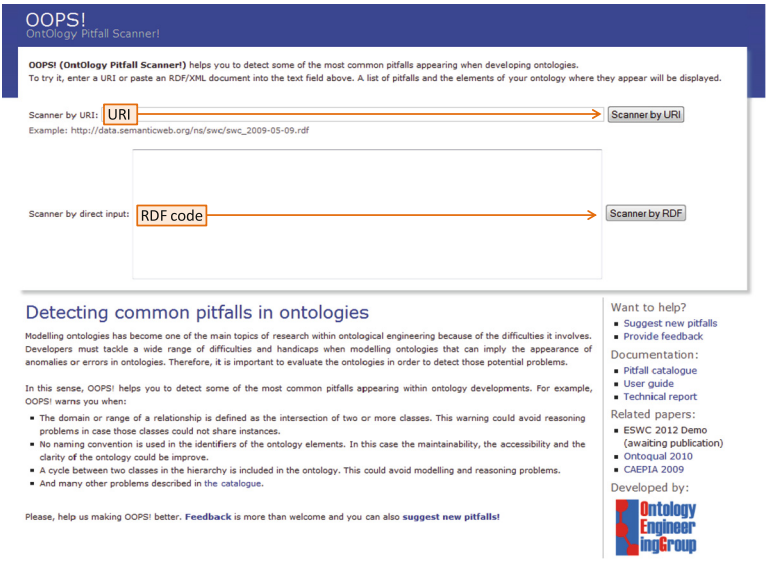

Fig. 1. OOPS! home page

[3] You may experience some layout strange behaviours with Internet Explorer.

[4] The ontology to be analyzed must be implemented in OWL (http://www.w3.org/TR/2009/REC-owl2-primer-20091027/) or RDF (http://www.w3.org/TR/2004/REC-rdf-primer-20040210/).

[5] http://www.oeg-upm.net/oops.

As result of analyzing the ontology provided by the user, OOPS! generates, as it is shown in Fig. 2, a new web page listing the pitfalls appearing in the ontology. This list provides information about (a) how many times a particular pitfall appears, (b) which specific ontology elements are affected by such a pitfall, and (c) a brief description about what the pitfall consist on.

Up to the moment of writing this paper, OOPS! helps to detect a subset of 21 pitfalls of those included in the catalogue.[6] Among others, appearances of pitfalls related to obtaining unexpected inferences (e.g., P6 and P19), to obtaining no inference (e.g., P12 and P13), and to usability issues (e.g., P8 and P11) are considered in OOPS!.

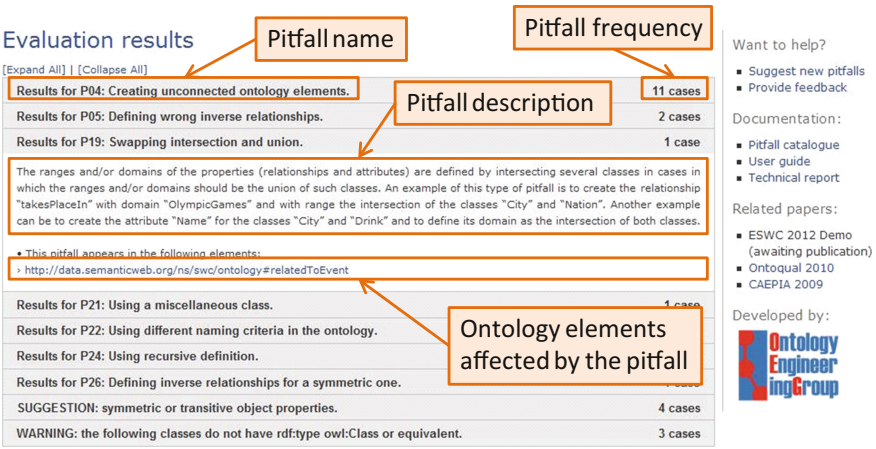

Fig. 2. Example of evaluation results generated by OOPS!

The current pitfall catalogue is included in the OOPS! web site. It is worth mentioning that the catalogue is continuously revised, since new kinds of modelling mistakes could appear as new ontologies are developed and evaluated. For example, pitfalls from P25 to P29 have been implemented in OOPS! extending the previous catalogue published in [6]. In addition, a form to suggest new pitfalls[7] is provided so that users can contribute enlarging the pitfall catalogue.

It is worth mentioning that OOPS! output points to ontology elements identified as potential errors but not necessarily factual errors and it depends on the type of pitfall detected. There are pitfalls that OOPS! detects in an automated way (e.g., P8 and P28) which means that they should be repaired; while others are detected in a semi-automated way (e.g., P13 and P24), which means that they must be manually checked in order to discern whether the elements identified actually contain errors.

[6] http://www.oeg-upm.net/oops/catalogue.jsp.

[7] http://www.oeg-upm.net/oops/submissions.jsp.

3 OOPS! Architecture

In this section OOPS! underlying architecture is presented (see Fig. 3) as well as some technical details. Basically, OOPS! is a web application based on Java EE,[8] HTML,[9] jQuery,[10] JSP[11] and CSS[12] technologies. The web user interface consists on a simple view where the user enters the URL pointing to or the RDF document describing the ontology to be analyzed. Once the ontology is parsed using the Jena API[13] the model is scanned looking for pitfalls, from those available in the pitfall catalogue. During this phase, the ontology elements involved in potential errors are detected as well as warnings regarding RDF syntax and some modelling suggestions are generated. Finally, the evaluation results are displayed by means of the web user interface showing the list of pitfalls appearing, if any, and the ontology elements affected as well as explanations describing the pitfalls.

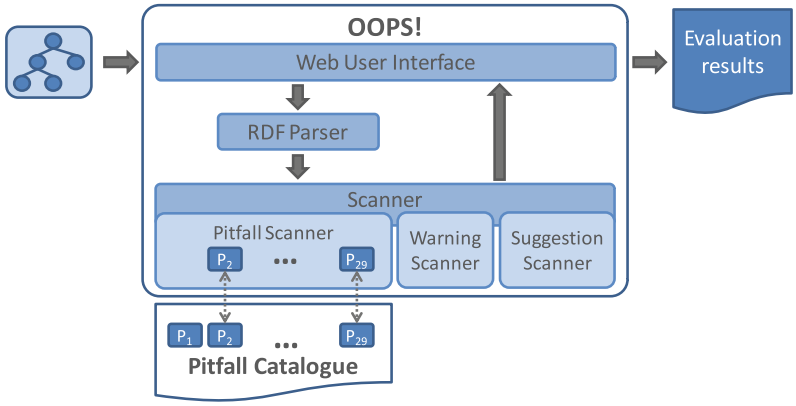

Fig. 3. OOPS! architecture

4 Conclusions and Future Work

In this paper we have presented OOPS! main features and architecture and how this tool represents a step forward within ontology evaluation tools as (a) it enlarges the list of errors detected by most recent and available works (e.g. MoKi [6] and XD Analyzer), (b) it is fully independent of any ontology development environment and (c) it works with main web browsers (Firefox, Chrome, Safari and Internet Explorer).

[8] http://www.oracle.com/technetwork/java/javaee/overview/index.html.
[9] http://www.w3.org/html/wg/.
[10] http://jquery.com/.
[11] http://www.oracle.com/technetwork/java/javaee/jsp/index.html.
[12] http://www.w3.org/Style/CSS/.
[13] http://jena.sourceforge.net/.

OOPS! is currently being tested by Ontology Engineering Group[14] members in order to debug it and extend its functionality. However, OOPS! has been already used by other ontology developers who belong to different organizations (such as AtoS, Tecnalia, *Departament Arquitectura, La Salle* at *Universitat Ramon Llull* and Human Mobility and Technology Laboratory at CICtourGUNE). In fact, OOPS! is freely available to users on the Web. It includes a link to a feedback form[15] so that everyone can test it and provide feedback and suggestions to be included in the tool.

As long as we discover new pitfalls during our research, they will be included in the current pitfall catalogue and implemented in OOPS!. In addition, we plan to improve and extend OOPS! features in the following lines:

- To group and classify pitfalls by categories according to previous ontology quality criteria identified in [3] and [4]. This feature will provide more flexibility to the ontology evaluation, since it will allow users to diagnose their ontologies just with respect to the dimensions they are interested in.
- To increase OOPS! features with guidelines about how to solve each pitfall. This information will ease the task of repairing the ontology after the diagnosis phase.
- To associate priority levels to each pitfall according to their different types of consequences they can convey when appearing in an ontology. This feature will be useful to prioritize actions to be taken during the repairing task.
- To make REST services available in order to allow other developments to use and integrate the pitfall scanner functionalities within their applications.
- To allow users to define their own pitfalls, according with their particular quality criteria, in order to use OOPS! in a customized fashion.

Acknowledgments. This work has been partially supported by the Spanish projects *BabelData* (TIN2010-17550) and *BuscaMedia* (CENIT 2009-1026). We would also like to thank Miguel Ángel García for his technical support.

References

1. Aguado de Cea, G., Gómez-Pérez, A., Montiel-Ponsoda, E., Suárez-Figueroa, M.C.: Natural language-based approach for helping in the reuse of ontology design patterns. In: Gangemi, A., Euzenat, J. (eds.) EKAW 2008. LNCS (LNAI), vol. 5268, pp. 32–47. Springer, Heidelberg (2008)
2. Blomqvist, E., Gangemi, A., Presutti, V.: Experiments on pattern-based ontology design. In: Proceedings of K-CAP 2009, pp. 41–48 (2009)
3. Gangemi, A., Catenacci, C., Ciaramita, M., Lehmann, J.: Modelling ontology evaluation and validation. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 140–154. Springer, Heidelberg (2006)
4. Gómez-Pérez, A.: Ontology evaluation. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies. International Handbooks on Information Systems, pp. 251–274. Springer, Heidelberg (2004)

[14] http://www.oeg-upm.net/.

[15] http://www.oeg-upm.net/oops/form.jsp.

5. Noy, N.F., McGuinness, D.L.: Ontology development 101: A guide to creating your first ontology. Technical report SMI-2001-0880, Standford Medical Informatics (2001)
6. Pammer, V.: PhD Thesis: Automatic Support for Ontology Evaluation Review of Entailed Statements and Assertional Effects for OWL Ontologies. Engineering Sciences. Graz University of Technology. http://know-center.tugraz.at/wp-content/uploads/2010/12/Dissertation_Viktoria_Pammer.pdf
7. Poveda, M., Suárez-Figueroa, M.C., Gómez-Pérez, A.: A double classification of common pitfalls in ontologies. In: OntoQual 2010 - Workshop on Ontology Quality at the 17th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2010). Proceedings of the Workshop on Ontology Quality - OntoQual 2010. CEUR Workshop Proceedings, Lisbon, Portugal, pp. 1–12, 15 October 2010. ISBN: ISSN 1613-0073
8. Rector, A., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., Wang, H., Wroe, C.: Owl pizzas: Practical experience of teaching owl-dl: Common errors and common patterns. In: Motta, E., Shadbolt, N.R., Stutt, A., Gibbins, N. (eds.) EKAW 2004. LNCS (LNAI), vol. 3257, pp. 63–81. Springer, Heidelberg (2004)