Undefined 1 (2009) 1–5
IOS Press

OOPS! (OntOlogy Pitfall Scanner!): supporting ontology evaluation on-line

Editor(s): Name Surname, University, Country
Solicited review(s): Name Surname, University, Country
Open review(s): Name Surname, University, Country

María Poveda-Villalón [a*] Mari Carmen Suárez-Figueroa [a] Miguel Ángel García-Delgado [a] and Asunción Gómez-Pérez [a],

[a] *Ontology Engineering Group, Escuela Técnica Superior de Ingenieros Informáticos, Universidad Politécnica de Madrid, Spain*
E-mail: {mpoveda, mcsuarez, magarcia, asun}@fi.upm.es

Abstract. This paper presents a system report about OOPS! (OntOlogy Pitfall Scanner!), a tool for detecting pitfalls in ontologies and targeted at newcomers and domain experts unfamiliar with description logics and ontology implementation languages. OOPS! extends the list of pitfalls detected by most recent and available approaches and allows selecting subset of pitfalls to be analysed according to different evaluation dimensions. In addition, the system also provides an indicator (critical, important, minor) for each pitfall according to their possible negative consequences. The system, which is free and online available, operates independently of any ontology development platform. Two interfaces are provided in order to make use of the system, namely 1) a web user interface supporting human-interaction with the system and 2) a RESTful web service supporting other systems to access OOPS! evaluation results. Currently, OOPS! has been used by developers from more than 50 countries (>2000 times), embedded within 3 third-party software developments and used in several enterprises, supporting both ontology development processes and training activities.

Keywords: ontology, ontology evaluation, knowledge representation, common pitfalls

1. Introduction

The vision of the Semantic Web originally proposed by Berners-Lee et al. [1] has promoted the continued growth of dataset published as well as the publication of ontologies used to model and enrich semantically these data. In this scenario, as important of the quality of the data published should be the quality and validation of the ontologies bringing semantic to such data. These ontologies should be published according to LD principles, but they also have to be evaluated at different stages, both during their development and their publication. Thus, the evaluation of those ontologies have become a significant challenge.

Ontologies developed following a particular methodology (e.g., NeOn Methodology [24], OnToKnowledge [22], DILIGENT [14]) have normally higher quality than those built without using methodological guidelines. However, to apply methodologies for building ontologies do not guarantee to have final products completely free of errors. Some anomalies can appear in developed ontologies since ontologists should handle diverse difficulties and handicaps when modeling ontologies ([6], [2], [19], [12]). Therefore, in any ontology development project it is vital to perform the ontology evaluation activity since this activity checks the technical quality of an ontology against a frame of reference.

In the last decades a huge amount of research on ontology evaluation has been performed. Some of these efforts have defined a generic quality evaluation framework ([11], [10], [23], [27] [8]), other authors have proposed to evaluate ontologies depending on the fi-

nal (re)use of them ([24]), others have formalized quality models based on features, criteria and metrics ([4], [8]), and in recent times methods for pattern-based evaluation have also emerged ([17], [7]). The author in [20] provides a recapitulation of generic guidelines and specific techniques for ontology evaluation.

In addition to frameworks, models, guidelines and approaches, several tools for helping users in the task of evaluating ontology have been proposed. These tools suffer from one or more of the following weak points: (a) there are tools developed as plug-ins for desktop applications (XD-Analyzer and Radon, plug-ins for NeOn Toolkit, and Ontocheck, a plug-in for Protégé), what implies the user should install both the tool and the ontology editor. In this case, it could happen that the tool is outdated, and sometimes incompatible, as long the core desktop applications evolve to new versions; (b) other tools are based on wiki technologies (such as MoKi), what forces an installation process to set up the wiki system; and (c) in most of the analyzed evaluation tools the number of problems detected is comparatively limited .

In this paper, we present OOPS! (OntOlogy Pitfall Scanner!), an on-line service intended to help ontology developers, mainly newcomers, during the ontology validation activity [25] that, in contrast to the previously mentioned tools: (a) is executed independently of any ontology development platform without configuration or installation, (b) works with main web browsers (Firefox, Chrome, Safari and Internet Explorer), (c) enlarges the list of errors detected by most recent and available works (like MoKi and XD Analyzer), and (d) provides both a web-based human interface and a RESTful service to be used by applications. To this end, OOPS! supports the (semi-)automatic diagnosis of OWL ontologies using an on-line and evolving catalogue of pitfalls.

The rest of the paper is organized as follows. Section 2 introduces the approach followed for addressing the ontology evaluation activity including main features and components of the proposed system. Section 3 presents the web user interface provided for human interaction with the system while the web service provided is described in Section 4. Section 5 reviews related works about ontology evaluation tools. Finally, Section 6 presents some concluding remarks and future lines of work.

While our previous papers [15] and [16] have focused on the research and analysis work behind OOPS!, this paper gives an overview of the technological and practical point of view of the system.

2. Automating ontology evaluation

There are a number of approaches for ontology evaluation as for example “golden standard”, “application-based”, “data-driven” and assessment by humans that could be applied to different levels on the ontologies as well summarized and reviewed by Brank and colleagues [3]. One well-known technique for quality analysis in any domain is following a checklist of issues to have into account, both for ensuring that good practices are followed and that bad practices are avoided. Based on this approach we have created a catalogue of bad practices (40 up to the time of writing this document) and automated the detection of as many of them as possible (32 currently).

Along this work we call these bad practices “pitfall” in order to identify characteristics that often represent a problem or that could lead to errors in ontologies; however, this is not always the case. In other words, depending on the ontology at hand, pitfalls can or cannot represent an actual error.

This catalogue of potential errors[1] is both based on existing works and new common errors found after the manual review of more than 30 ontologies [16]. Figure 1 shows the list of pitfall classified by ontology evaluation dimensions, namely structural, functional and usability-profiling, according to the definitions provided in [10]. For each pitfall the following information is provided in the catalogue:

Title: this field provides a brief description of what the pitfall is about.

Description: it provides detailed explanation of what the pitfall consist on and bibliographical references, when possible, including as an example, a situation or modeling in which the pitfall has occurred.

Elements affected: this field could have different type of values, namely a) "ontology" if the pitfall affect the ontology itself instead of particular elements or b) "specific ontology elements" where the elements could be classes, object properties or datatype properties. It is worth noting that if a given pitfall affects the ontology itself this pitfall could appear at most 1 in the ontology, while a pitfall affecting a list of elements could appear more than once, for example, a set of classes, or a list of properties, etc.

[1] The current catalogue of pitfalls is available at http://oops.linkeddata.es/catalogue

Importance level: it is obvious that not all the pitfalls are equally important; their impact in the ontology will depend on multiple factors. For this reason, the pitfall catalogue has been extended with information about how critical the pitfalls are. We have identified three levels:

- **Critical:** It is crucial to correct the pitfall. Otherwise, it could affect the ontology consistency, reasoning and applicability, among others. For example, the consequences of "P19. Swapping intersection and union" could lead to logical inconsistencies in the ontology, which represents a critical error when reasoning over the populated ontology.
- **Important:** Though not critical for ontology function, it is important to correct this type of pitfall. For example, the logical consequences of "P25. Defining a relationship inverse to itself" are the same as if such relationship were defined as symmetric. However, the latter option, that is, using the ontology implementation language constructors for the purpose they were conceived, is a sign of good modeling and understanding of the underlying semantics.
- **Minor:** It does not represent a problem. However, correcting it makes the ontology better organized and user friendly. For example, the pitfall "P22. Using different naming criteria in the ontology" is about the appearance of the ontology and does not compromise the proper ontology functioning.

In addition, Figure 1 indicates which pitfalls are not implemented yet within the system (marked by "*").

2.1. System design

Figure 2 presents the underlying architecture of OOPS!. In order to produce a list of evaluation results, the system takes as input the ontology to be analyzed. The system is accessible to humans by means of a web user interface and to machines throughout a web RESTful service that will be detailed in Section 3 and Section 4 respectively. The input ontology could be entered by its URI or the OWL code [2], which describes the ontology to be analyzed. Once the ontology is parsed using the Jena API[3], the "Pitfall Scanner" module inspects the declared ontology looking for pitfalls among those available in the catalogue. At the moment of writing this system report 32 out of the 40 pitfalls defined in the catalogue are implemented within the system (see Figure 1). During this scanning phase, the ontology elements prone to potential errors are detected, whereas some modeling suggestions are generated by the "Suggestion Scanner" module. Then, the evaluation results are provided including the list of pitfalls detected, if any, and the ontology elements affected, as well as explanations describing the findings. The system allows not only analyzing all the automated pitfalls, but also choosing specific pitfalls or predefined groups according to the pitfall classification presented in this Figure 1.

Finally, a conformance badge is provided to the user together with the HTML code for including it within the ontology documentation website. There are 4 badges indicating whether the ontology is a) free of pitfalls[4]; b) minor pitfalls detected; b) important pitfalls detected and c) critical pitfalls detected. Main goal of this badges is to encourage users to improve their ontologies getting better scored badges.

3. Web user interface

Before realizing the need of providing both human-oriented and machine-readable (see Section 4) access to OOPS!, its main goal was only to provide free and online ontology evaluation features to ontologists by means of a website. This human-based online service is currently available at http://oops.linkeddata.es. The web application provides a user interface based on HTML[5], jQuery[6], JSP[7] and CSS[8] technologies. The user enters in the homepage the ontology URI or pass its OWL code. By pressing the evaluation button the user will get back the evaluation results for all the implemented pitfalls. In case the user would like to analyze only the appearance of a subset of pitfalls, he/she

[2] At the moment of writing this report the only serialization accepted is RDF/XML due to technical problems. We are currently working on it in order to accept other serializations, e.g., turtle.

[3] http://jena.sourceforge.net

[4] It should be noted that not all the pitfalls are automated and that some detected pitfalls might not be considered as such due to modeling decision or specific requirements as mentioned in Section 1. Therefore after the execution the user should review the provided results and the pitfalls that are not automated yet.

[5] http://www.w3.org/html/wg/

[6] http://jquery.com/

[7] http://www.oracle.com/technetwork/java/javaee/jsp/index.html

[8] http://www.w3.org/Style/CSS/

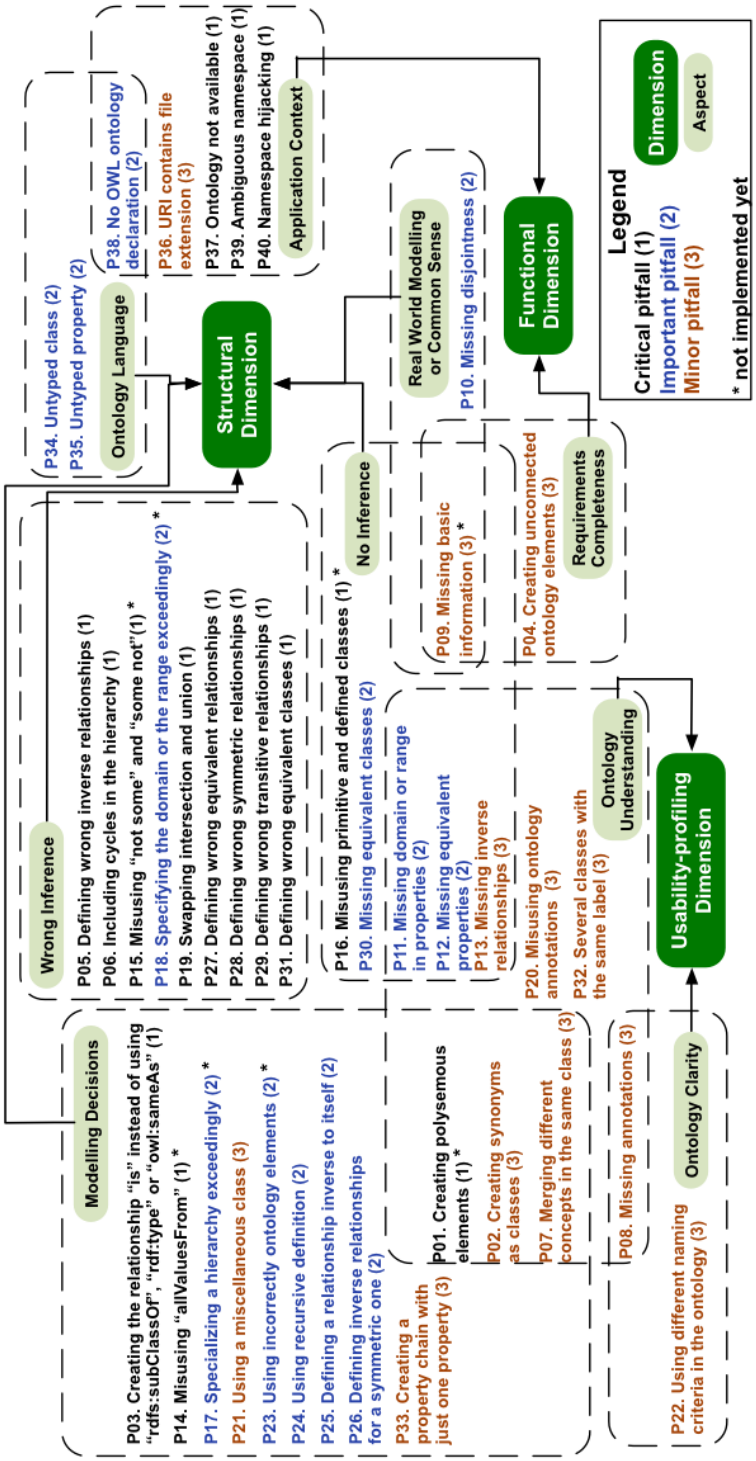

Fig. 1. Pitfall classification by ontology evaluation dimensions [10]

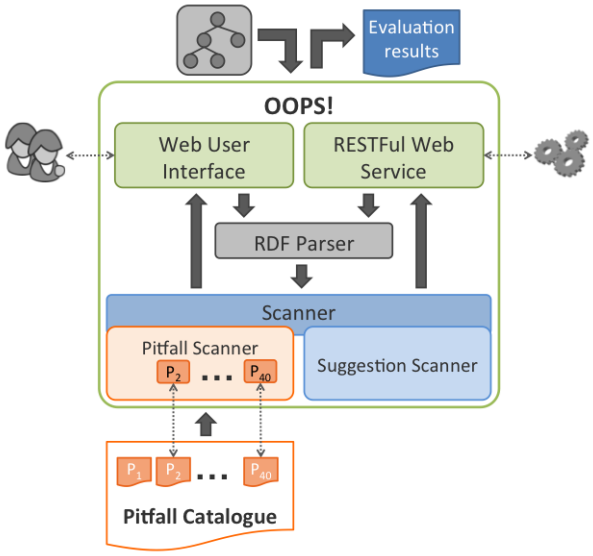

Fig. 2. Design of OOPS! modules

should go to "advanced evaluation"[9]. In this case, the user might choose between a) selecting a number of pitfalls as shown in Figure 3 or b) selecting groups of pitfalls according to the ontology evaluation dimensions and aspects as shown in Figure 4.

Finally, the evaluation results are displayed in the web user interface, which shows the list of pitfalls detected, if any, and the ontology elements affected, the importance level of the detected pitfalls, as well as explanations describing the findings as Figure 5 illustrates.

4. Web service

In order to allow third-party software to integrate OOPS! features a RESTful web service is available at http://oops-ws.oeg-upm.net/rest. The web service uses Jena API as RDF parser like the web user interface (see Section 3) plus Jersey framework [10] to build a system compliance with RESTful architecture. Some examples of systems that are already integrating OOPS! web service will be shown in Subsection 4.1. The request should be invoke by an HTTP POST which body is XML with the following fields:

OntologyUrl: this field contains the ontology URI. it should be noted that this field is mandatory in case the OntologyContent field is empty.

OntologyContent: in this field the ontology RDF source code in entered. It is important to mention that the RDF code must be contained inside `<![CDATA[ RDF code ]]>`. This field is mandatory in case OntologyUrl field is empty.

Pitfalls: this field indicates the list of pitfalls to be scanned. In case the user would like to analyzed only a subset of the implemented pitfalls the number of each selected pitfall with a coma separator if more than one pitfall should be included in this field. For instance: “04,11,21”. This field is optional, if no pitfalls are selected the web service will analyze all the implemented ones.

OutputFormat: this field indicates which response format is demanded. The output formats available at the moment of writing this report are RDF file or XML file. The possible values for this field are “XML” or ńńRDF/XML”. If any other value is entered, the default output will be RDF/XML. This field is optional.

It is worth noting that the user must enter either the OntologyUrl or the OntologyContent value, if none of them is sent, an RDF error message will be received as response. If both of them are entered, the service will use the OntologyContent.

4.1. Examples of systems using OOPS! web service

Some third-party software, mainly ontology registries, already incorporate the ontology evaluation features provided by OOPS!. For example, Figure 6 shows how Ontohub[11], an open ontology repository which supports ontology collection, retrieval, development, mapping, translation, etc. includes ontology evaluation features by means of the web service here presented[12].

In the context of the READY4SmartCities project[13] an ontology catalogue for smart cities and related domains is being developed. As part of its functionality, the catalogue offers evaluation result for each ontology provided by the OOPS! web service as shown in Figure 7.

[9] http://oops.linkeddata.es/advanced
[10] https://jersey.java.net/
[11] https://ontohub.org/
[12] See http://goo.gl/GPe2Or for more details about the integration project.
[13] http://www.ready4smartcities.eu/

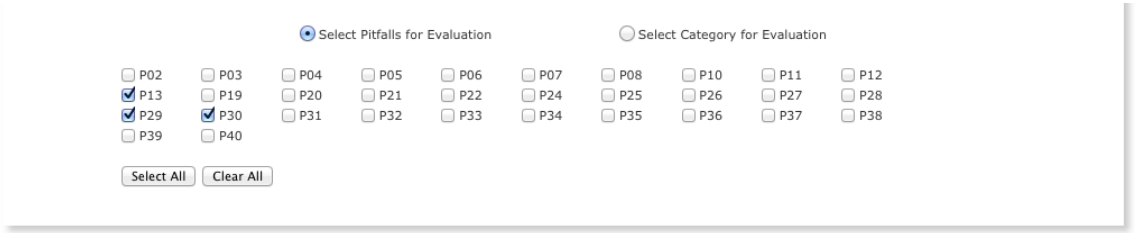

Fig. 3. Selecting particular pitfalls to be analyzed

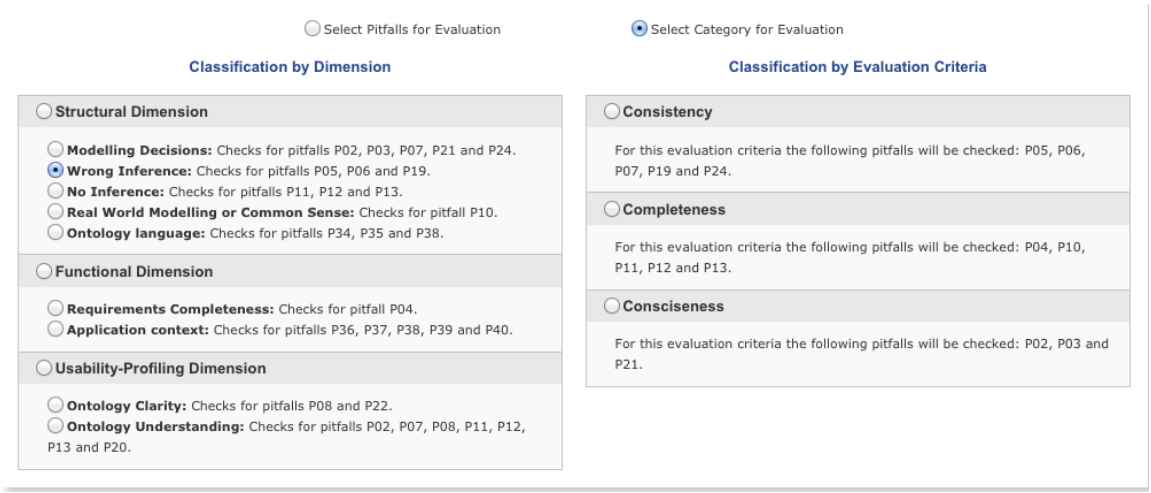

Fig. 4. Selecting pitfalls to be analyzed by ontology evaluation dimensions and aspects

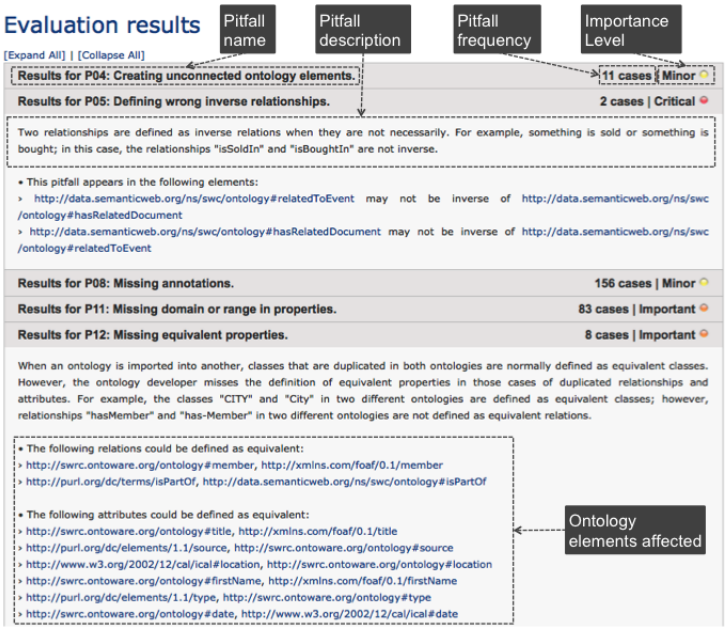

Fig. 5. OOPS! output example through web user interface

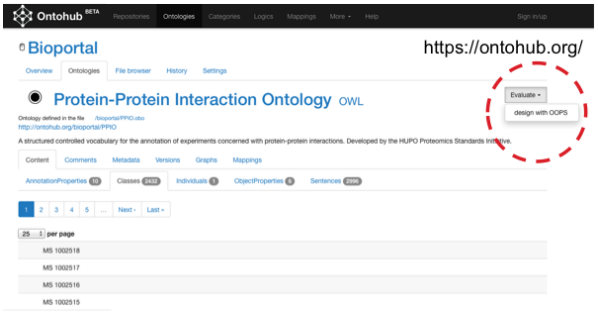

Fig. 6. OOPS! integration within Ontohub

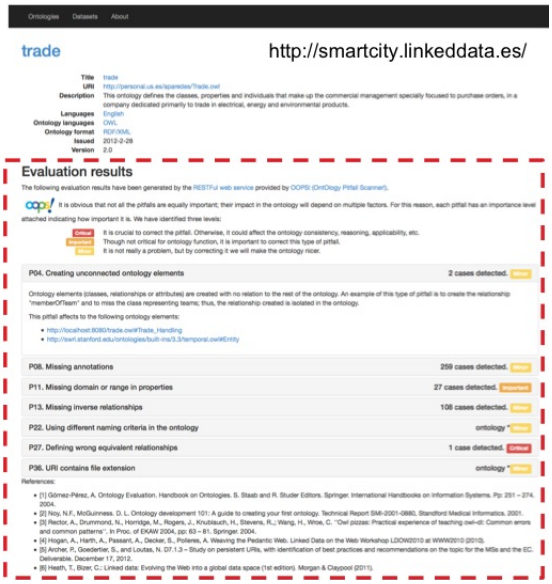

Fig. 7. OOPS! integration within smart cities ontology catalogue

Finally, the DrOntoAPI[14] also provides ontology evaluation functionality based on OOPS! RESTful web service.

5. Existing ontology evaluation systems

First steps towards the automation of ontology evaluation started in the noughties with the development of ODEclean by Fernández-López and Gómez-Pérez in 2002 [9]. ODEClean, a plug-in for WebODE ontology editor, provided technological support for the Ontoclean Method [28]. Later in 2004, ODEval [5] was developed in order to ease the evaluation of RDF(S) and DAML+OIL concept taxonomies. It should be mentioned that none of these systems support ontology evaluation for OWL ontologies.

Regarding tools for ontology evaluation supporting OWL ontologies we should consider plug-ins for desktop applications as XDTools plug-in[15] and OntoCheck[16] [21] that are plug-ins for NeOn Toolkit and Protégé respectively. While the former system checks some structural and architectural ontology features, the latter is focused on metadata aspects. Main drawback of this type of tools is their dependency on an associated ontology editor.

Within the web-based systems we should mention, on the one hand, MoKi[17] [13] that consists on a wiki-based ontology editor that incorporates ontology evaluation functionalities. Even though MoKi is a web-based application it does involve installation process as it is built upon a wiki platform that should be setup. On the other hand, OQuaRE[18] [8] is a web application that extracts quality measurements from the ontology structure and compares these measurements to certain predefined values. While this system provides numerous inspectors for ontology characteristics, it is not clear which parts of the ontology should be modified and in which sense in order to avoid the pointed problems.

Finally we can mention eyeball[19] which is a command-line system and an Java API which provides logical consistency checking and inspectors for common errors related to RDF and OWL models.

To the best of the authors' knowledge there the system described in this report provides there wider range of detections together with a comprehensible and user-friendly interface.

6. Conclusions and future lines

An overview of OOPS! functionalities has been presented along this system report. The importance of a system like the one presented here is the difficulty from ontology developers, mainly newbies and developers with no high background in description logics, to evaluate their ontologies. Main contributions here described are 1) a user-friendly web interface that pro-

[14] http://sourceforge.net/projects/drontoapi/

[15] http://neon-toolkit.org/wiki/XDTools

[16] http://protegewiki.stanford.edu/wiki/OntoCheck

[17] https://moki.fbk.eu/website/index.php

[18] http://miuras.inf.um.es:9080/oqmodelsliteclient/

[19] https://jena.apache.org/documentation/tools/eyeball-getting-started.html

vides useful information about what the pitfalls consist on and which ontology elements are affected and 2) a web service providing ontology evaluation results that can be integrated within third-party software.

It is worth mentioning that the presented system has been widely accepted by the semantic web community and experts in other areas as supported by the following facts:

- OOPS! has been broadly accepted by a high number of users worldwide and has been executed more than 2000 times from 50 different countries.
- It has been continuously used from very different geographical locations.
- It is integrated with third-party software (see Section 4) and locally installed in private enterprises (e.g., Semantic Arts[20] and Raytheon[21]).

Even though there are still several complex issues to address, in the future, we see the following lines of work:

First, we are currently working in the definition of a formal model to describe the ontology evaluation framework. In this sense, we are developing an ontology describing the pitfalls and the evaluation results when executing OOPS!. For doing so, we have taken as reference the "Quality Model Ontology" [18] which describes the elements of a quality model (e.g., quality characteristics, quality measures), and which also provides the means for capturing the values for quality measures obtained in the evaluation of characteristics that are described in a quality model. The resulting ontology will be used and instantiated by the RESTful web service in order to provide the RDF output.

Second, in order to provide a complete service regarding ontology quality, it would be needed to include methodological guidelines addressing also ontology repair, not only the ontology diagnosis phase. This feature will help users to take decisions and actions in order to improve their models after evaluating them.

Third, another important improvement is the automation of the remaining 8 pitfalls and the enhancement of some of the already implemented ones. In order to address this improvements it would be needed to make extensive use of natural language processing techniques as analyzed in [26]. In addition, as is it being done, we plan to keep extending the pitfall catalogue and implementing the detection of as many pitfalls as possible.

Finally, the integration of OOPS! within existing ontology editors, such as WebProtege or the NeOn Toolkit, would be very convenient for the users since they would not need to change platforms to repair their ontologies after the diagnosis phase

Acknowledgments This work has been supported by the 4V project (TIN2013-46238-C4-2-R) funded by the Ministerio de Economía y Competitividad in Spain. We would like to thank Alvaro Pino for his support developing the advanced evaluation features.

[20] http://semanticarts.com/

[21] http://www.raytheon.com/

References

[1] T. Berners-Lee, J. Hendler, O. Lassila, et al. The semantic web. *Scientific american*, 284(5):28–37, 2001.

[2] E. Blomqvist, A. Gangemi, and V. Presutti. Experiments on pattern-based ontology design. In *Proceedings of the fifth international conference on Knowledge capture*, pages 41–48. ACM, 2009.

[3] J. Brank, M. Grobelnik, and D. Mladenić. A survey of ontology evaluation techniques. In *In In Proceedings of the Conference on Data Mining and Data Warehouses (SiKDD 2005*, 2005.

[4] A. Burton-Jones, V. C. Storey, V. Sugumaran, and P. Ahluwalia. A semiotic metrics suite for assessing the quality of ontologies. *Data & Knowledge Engineering*, 55(1):84–102, 2005.

[5] O. Corcho, A. Gómez-Pérez, R. González-Cabero, and M. C. Suárez-Figueroa. Odeval: a tool for evaluating rdf (s), daml+ oil, and owl concept taxonomies. In *Artificial Intelligence Applications and Innovations*, pages 369–382. Springer, 2004.

[6] G. A. De Cea, A. Gómez-Pérez, E. Montiel-Ponsoda, and M. C. Suárez-Figueroa. Natural language-based approach for helping in the reuse of ontology design patterns. In *Knowledge Engineering: Practice and Patterns*, pages 32–47. Springer, 2008.

[7] R. Djedidi and M.-A. Aufaure. Onto-evo a l an ontology evolution approach guided by pattern modeling and quality evaluation. In *Foundations of Information and Knowledge Systems*, pages 286–305. Springer, 2010.

[8] A. Duque-Ramos, J. T. Fernández-Breis, R. Stevens, N. Aussenac-Gilles, et al. Oquare: A square-based approach for evaluating the quality of ontologies. *Journal of Research and Practice in Information Technology*, 43(2):159, 2011.

[9] M. FernĞndez-lŮpez, A. GŮmez-pŐrez, and F. InformĞtica. The integration of ontoclean in webode. In *In Proc. of the EON2002 Workshop at 13th EKAW*, 2002.

[10] A. Gangemi, C. Catenacci, M. Ciaramita, and J. Lehmann. Modelling ontology evaluation and validation. In *Proceedings of the 3rd European Semantic Web Conference (ESWC2006), number 4011 in LNCS, Budva*. Springer, 2006.

[11] A. Gómez-Pérez. Ontology evaluation. In *Handbook on ontologies*, pages 251–273. Springer, 2004.

[12] N. F. Noy, D. L. McGuinness, et al. Ontology development 101: A guide to creating your first ontology, 2001.

[13] V. Pammer. *Automatic support for ontology evaluation: review of entailed statements and assertional effects for OWL ontologies*. na, 2010.

[14] H. S. Pinto, S. Staab, and C. Tempich. Diligent: Towards a fine-grained methodology for distributed, loosely-controlled and evolving. In *Ecai 2004: Proceedings of the 16th European Conference on Artificial Intelligence*, volume 110, page 393. IOS Press, 2004.

[15] M. Poveda-Villalón, A. Gómez-Pérez, and M. C. Suárez-Figueroa. Oops!(ontology pitfall scanner!): An on-line tool for ontology evaluation. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 10(2):7–34, 2014.

[16] M. Poveda-Villalón, M. C. Suárez-Figueroa, and A. Gómez-Pérez. Validating ontologies with OOPS! In *Knowledge Engineering and Knowledge Management*, pages 267–281. Springer, 2012.

[17] V. Presutti, A. Gangemi, S. David, G. Aguado de Cea, M. Suárez-Figueroa, E. Montiel-Ponsoda, and M. Poveda. NeOn Deliverable D2. 5.1. A Library of Ontology Design Patterns: reusable solutions for collaborative design of networked ontologies. *NeOn Project. http://www. neon-project. org*, 2008.

[18] F. Radulovic. A software quality model for the evaluation of semantic technologies. Master's thesis, Facultad de Informática (Universidad Politécnica de Madrid), 2011.

[19] A. Rector, N. Drummond, M. Horridge, J. Rogers, H. Knublauch, R. Stevens, H. Wang, and C. Wroe. Owl pizzas: Practical experience of teaching owl-dl: Common errors & common patterns. In *Engineering Knowledge in the Age of the Semantic Web*, pages 63–81. Springer, 2004.

[20] M. Sabou and M. Fernandez. Ontology (network) evaluation. In *Ontology engineering in a networked world*, pages 193–212. Springer, 2012.

[21] D. Schober, I. Tudose, V. Svatek, and M. Boeker. Ontocheck: verifying ontology naming conventions and metadata completeness in protégé 4. *Journal of biomedical semantics*, 3(2):1–10, 2012.

[22] S. Staab, R. Studer, H.-P. Schnurr, and Y. Sure. Knowledge processes and ontologies. *IEEE Intelligent systems*, 16(1):26–34, 2001.

[23] D. Strasunskas and S. L. Tomassen. The role of ontology in enhancing semantic searches: the evoqs framework and its initial validation. *International journal of Knowledge and Learning*, 4(4):398–414, 2008.

[24] M. C. Suárez-Figueroa. *NeOn Methodology for Building Ontology Networks: Specification, Scheduling and Reuse*. PhD thesis, Universidad Politecnica de Madrid, Spain, June 2010.

[25] M. C. Suárez-Figueroa, G. A. d. Cea, and A. Gómez-Pérez. Lights and shadows in creating a glossary about ontology engineering. *Terminology*, 19(2):202–236, 2013.

[26] M. C. Suárez-Figueroa, M. Kamel, and M. Poveda-Villalón. Benefits of natural language techniques in ontology evaluation: the oops! case. *10th International Conference on Terminology and Artificial Intelligence (TIA 2013)*, page 107, 2013.

[27] D. Vrandečić. *Ontology evaluation*. Springer, 2009.

[28] C. Welty and N. Guarino. Supporting ontological analysis of taxonomic relationships, 2001.