Agile Knowledge Graph Testing with TESTaLOD*

Valentina Anita Carriero[1], Fabio Mariani[2], Andrea Giovanni Nuzzolese[1], Valentina Pasqual[2], and Valentina Presutti[1]

[1] STLab, ISTC-CNR, Rome, Italy
[2] FICLIT, University of Bologna, Bologna, Italy

Abstract. XD is an agile methodology for knowledge graph construction focused on the continuous and iterative interaction between the development team and the testing team. This interaction is aimed at formally validating the resulting knowledge graphs with respect to clear ontological commitments identified as *competency questions*. In this paper we present and demonstrate TESTaLOD, which is a tool designed and implemented for supporting the testing team of XD projects with an easy-to-use web-based toolbox.

Keywords: Knowledge graph testing · eXtreme Design · ArCo

1 Introduction

In this paper we demonstrate TESTaLOD, which is a tool designed and implemented for supporting knowledge graph testing. When introducing the eXtreme Design (XD) methodology, [1] remarks that ontology testing is a crucial part of the ontology engineering process. In this context, TESTaLOD is implemented for supporting and automatising the testing activity of XD methodology, aiming at reducing the manual effort of testers. XD's Unit testing approach requires the formulation of unit tests modelled as competency questions. TESTaLOD supports the user in such procedure offering the possibility to perform automatically backward execution of unit tests implemented during the testing activity. The results of this analysis are shown in a simple online graphic interface.

2 Related work

Many languages, frameworks, methodologies and tools have been developed for evaluating and testing ontologies. Among the available tools, it is worth mentioning RDFUnit, built to perform a test-driven evaluation of linked data quality [5]. It can run automatically a set of manually generated test cases. Those test cases are executed as SPARQL queries against an endpoint; thus, an endpoint is always

required. Instead, TESTaLOD provides the possibility to run test cases without a SPARQL endpoint. This functionality addresses the requirement of enabling the testing process without having a SPARQL endpoint available, which can be most likely the case when a knowledge graph is embryonic. SHARK (SHACL Reasoning over Knowledge-graphs), which reuses RDFUnit tool to run the tests and give the results, is a test-driven ontology validation framework for testing an ontology using predefined or custom SHACL tests [7]. Taking SHARK as reference framework due to its easy-to-use web interface, our platform has been developed implementing new technical features to support XD methodology, i.e. the possibility to perform automatically backward execution requires several files to be uploaded at the same time, from different sources. In [6] it is also proposed OOPS! (OntOlogy Pitfall Scanner!), which is another Web application we took in consideration. It is a diagnosis tool that, in addition to the detection of potential errors in ontologies – that is based on a catalogue of the most common *pitfalls* when developing an ontology – recommends some tips to repair them. Its detection methods mainly rely on structural pattern matching and linguistic analysis.

3 The TESTaLOD web tool

TESTaLOD is a Web application designed for providing the XD methodology with a testing toolbox for supporting knowledge graph (KG) testing[3]. XD is an iterative and incremental methodology, and involves different actors: (i) a designer team, in charge of modelling a KG; (ii) a testing team, disjoint from the designer team, which takes care of testing the KG; (iii) a customer team, who elicits the requirements that are translated into ontological commitments (i.e. competency questions and other constraints) that guide the KG development. TESTaLOD uses the *TestCase* OWL meta model introduced in [2] as the reference schema for representing unit tests, a means for validating ontology as well as data commitment. In such a schema, a unit test is modelled as a competency question (CQ) expressed in natural language and associated with a corresponding SPARQL query. Additionally, an expected result and a reference data sample can be provided[4]. This allows to validate the CQ by executing the SPARQL query with respect to the data sample, assessing the correspondence to the expected result. TESTaLOD implements a two-step workflow as presented in Figure 1, which is powered by a Web-based user interface that allows a user to select and automatically execute an arbitrary number of defined test cases modelled by using the TestCase OWL meta model.

The first step requires a user to provide one or more test cases as input. Those test cases can be either retrieved from a Github repository (cf. Figure 1

[3]Though there is no consensus on what a KG is [3], in the scope of this paper we informally refer to a KG as linked (open) data including both OWL and RDF entities, and both schema axioms and factual data.

[4]More information on test structure and examples are provided in TESTaLOD documentation's section: http://testalod.herokuapp.com/documentation

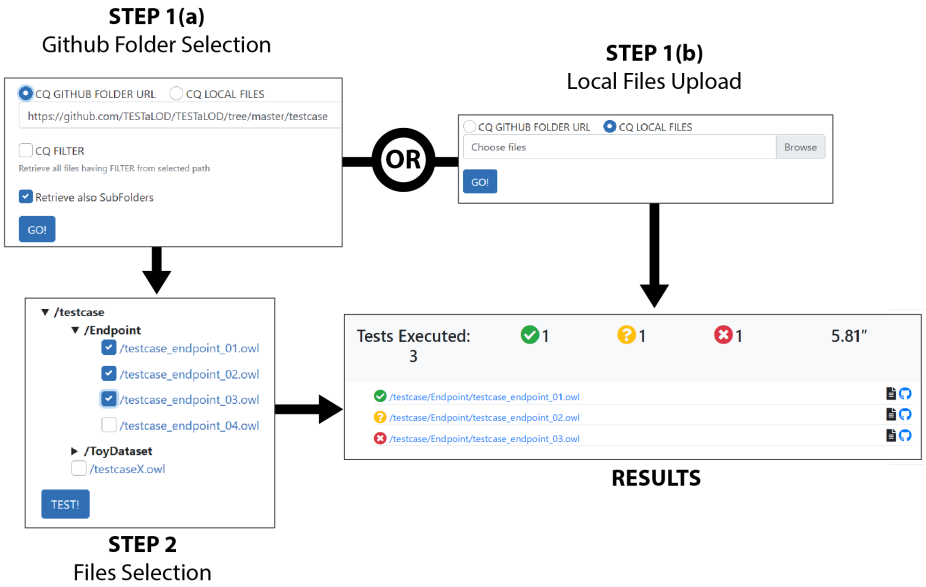

Fig. 1: Workflow implemented by TESTaLOD based on the user interface.

step 1(a)) or directly uploaded from a local file system (cf. Figure 1 step 1(b)). When step 1(a) is selected by the user, TESTaLOD retrieves all the test cases available in the repository by recursively traversing the subfolders reachable from the repository root. After all the test cases are retrieved, the user is presented with an additional view (step 2). This view allows the user to decide whether or not to execute all the test cases found in the repository selectively. On the contrary, if step 1(b) is selected, then all uploaded local files are directly tested. Both alternatives allow the user to visualise a view that reports the output of the automatic execution of the selected test cases. There are 3 possible outcomes resulting from the execution of a test case with TESTaLOD, i.e. (i) the test case is fully successful, thus the corresponding record in the user interface is green coloured; (ii) the test case is partially successful (the test results do not match completely the expected results), thus the corresponding record in the user interface is yellow coloured; (iii) the test case is fully unsuccessful, thus the corresponding record in the user interface is red coloured.

TESTaLOD has been designed and implemented in the context of the ArCo project [4]. ArCo is the Italian Cultural Heritage knowledge graph, which is developed by following eXtreme Design (XD), focused on ontology design patterns (ODPs) reuse. The ArCo KG consists of a network of 7 vocabularies and 169 million triples about 820 thousand cultural entities. The testing activities have been carried out by using TESTaLOD: at each ArCo new realease (or minor change) all ArCo test cases have been executed on TESTaLOD to validate updates of the whole KG. The test cases defined for the ArCo project and the

source code of TESTaLOD are both publicly available on GitHub[5]. A running demo of TESTaLOD is available online[6]. In the context of the demonstration track of the conference we will show how to use TESTaLOD in order to validate the requirements elicited for the ArCo project and represented as test cases formalised by using the TestCase OWL meta model.

4 Conclusions and future work

In this paper we demonstrated TESTaLOD, a tool supporting knowledge graph test focused on reducing the human effort at testing time. An ongoing work is a new tool section that offers a graphical interface for compiling test case OWL files with all the required annotations and for storing OWL files on a selected Github folder, hence, ready to be tested. This update, in addition to facilitating and speeding up the test activity, would be a great support for the creation of TestCase OWL files, helping users in test cases' validation and avoiding TestCase meta model misuse.

References

[1] Eva Blomqvist et al. "Experimenting with eXtreme Design". In: *Proc. of EKAW 2010*. (Lisbon, Portugal). Vol. 6317. Springer, 2010, pp. 120–134.

[2] Eva Blomqvist et al. "Engineering Ontologies with Patterns - The eXtreme Design Methodology." In: *Ontology Engineering with Ontology Design Patterns*. Vol. 25. Studies on the Semantic Web. IOS Press, 2016.

[3] Piero Andrea Bonatti et al. "Knowledge Graphs: New Directions for Knowledge Representation on the Semantic Web (Dagstuhl Seminar 18371)". In: *Dagstuhl Reports* 8.9 (2019), pp. 29–111.

[4] Valentina Anita Carriero et al. "ArCo: the Italian Cultural Heritage Knowledge Graph". In: *Proc. of ISWC 2019 - accepted for publication*. (Auckland, New Zealand). 2019.

[5] Dimitris Kontokostas et al. "Test-driven Evaluation of Linked Data Quality". In: *Proc. of WWW 2014*. Seoul, Korea: ACM, 2014, pp. 747–758.

[6] María Poveda-Villalón et al. "OOPS! (OntOlogy Pitfall Scanner!): An Online Tool for Ontology Evaluation". In: *Int. J. Semant. Web Inf. Syst.* 10.2 (2014), pp. 7–34.

[7] Gustavo Correa Publio. "SHARK: A Test-driven Framework for Design and Evolution of Ontologies". In: *Proc. of ESWC 2018*. 2018.

[5] The test cases are available at https://github.com/ICCD-MiBACT/ArCo/tree/master/ArCo-release/test/CQ, while the source code of the tool is available at https://github.com/TESTaLOD/TESTaLOD.

[6] http://testalod.herokuapp.com/