DemoEffTE: A Demonstrator of Dependency-aware Evaluation of Test Cases over Ontology

Lavdim Halilaj halilaj@cs.uni-bonn.de
University of Bonn / Fraunhofer IAIS, Germany

Irlán Grangel-González grangel@cs.uni-bonn.de
University of Bonn / Fraunhofer IAIS, Germany

Maria-Esther Vidal vidal@cs.uni-bonn.de
Fraunhofer IAIS, Germany / Universidad Simon Bolivar, Venezuela

Steffen Lohmann steffen.lohmann@iais.fraunhofer.de
Fraunhofer IAIS, Germany

Sören Auer soeren.auer@tib.eu
Technische Informationsbibliothek, Hannover, Germany

Keywords

Ontology Engineering | Test-Driven Ontology Development | Test Cases | Dependency Graph

Abstract

Traditional approaches, which follow a *test-driven development* technique, allow a set of test cases to be exhaustively evaluated ensuring that each modification of an ontology does not violate predefined requirements. However, the time required for the evaluation of test cases is high and usually represents a bottleneck in an ontology development process. The *EffTE* framework tackles this problem; it relies on a graph-based model of the dependencies between test cases to support users during an ontology development process. Traversing the dependency graph is realized using breadth-first search along with a mechanism that tracks *tabu* test cases, i.e., test cases that will be ignored for further evaluation due to faulty parent test cases. As a result, the number of test cases that are evaluated is minimized, thus reducing the time required for validating an ontology after each modification. We demonstrate the benefits of prioritization and selection of the test cases to be evaluated with *DemoEffTE* . Attendees will observe the behavior of both a naive approach and the *EffTE* framework on different configuration settings such as different: (1) ontology size; (2) topology of the dependency graph of the test cases; and (3) number of test cases. The demo is available at: http://vocol.iais.fraunhofer.de/DemoEffTE .

1. Introduction

The development of domain-specific ontologies requires joint efforts among different groups of stakeholders, such as knowledge engineers and domain experts. Functional requirements can be expressed through Competency Questions, which are questions that the underlying ontology should be able to answer [4] . However, the concurrent definition of ontology concepts often results in a violation of the defined requirements, or creates design issues like duplicate entries or missing documentation. For example, in the large and monolithic DBpedia [1] ontology, version 2016-04 only 556 from a total of 2,849 properties have associated label descriptions or comments [3] . To ensure that any ontology modification has only the expected effects, a set of test cases can be defined based on Competency Questions. This is similar to the test-driven software development principles, where test cases that represent requirements, are defined before the code is actually written [1] .

Commonly, the time demand for evaluating predefined test cases is high and represents a bottleneck in the ontology development process. Therefore, with a naive approach, test cases are exhaustively evaluated to identify any not intended modification or design issue introduced during the concurrent development of the ontology. Since much effort is invested in creating and extending ontologies for various domains, it is crucial to make ontology development and maintenance cost-effectively [2] .

On the other hand, modeling the relationship between test cases using a dependency graph enables prioritization and selection of test cases to be evaluated. As a result, the number of test cases that are evaluated is minimized, thus reducing the time required for ontology validation after each modification. In order to illustrate the benefits of a test cases dependency graph compared to the cases where such a graph is not used, we designed *DemoEffTE,* a naive and dependency-aware test cases evaluation demonstrator. Attendees will be able to choose various options related to the: (1) ontology size; (2) topology of dependency graphs; and (3) number of test cases and observe results like total evaluation time and number of evaluated test cases. The demo includes a set of test cases to be evaluated that provides a comparison of the performance between a naive approach and *EffTE* .

2. The DemoEffTE Architecture

The architecture of *DemoEffTE* is illustrated in Figure 1 and is composed of two main components: (1) the GUI; and (2) *Test Cases Processing (TCP)* . The GUI provides the possibility for the users to specify the approach to be tested; ontology size; number of test cases and the typology of the dependency graph. The *TCP* component receives as input the selections made by users. Next, according to the chosen options, the *TCP* component validates the ontology file against a set of test cases, which are defined by the user and stored in the *Test Cases DB* module. The *Test Case Selection* is responsible for prioritization and selection of the test cases to be evaluated. Each test case is evaluated in the *Test Case Evaluation* . The collected results are returned to the GUI, where can further be explored by attendees.

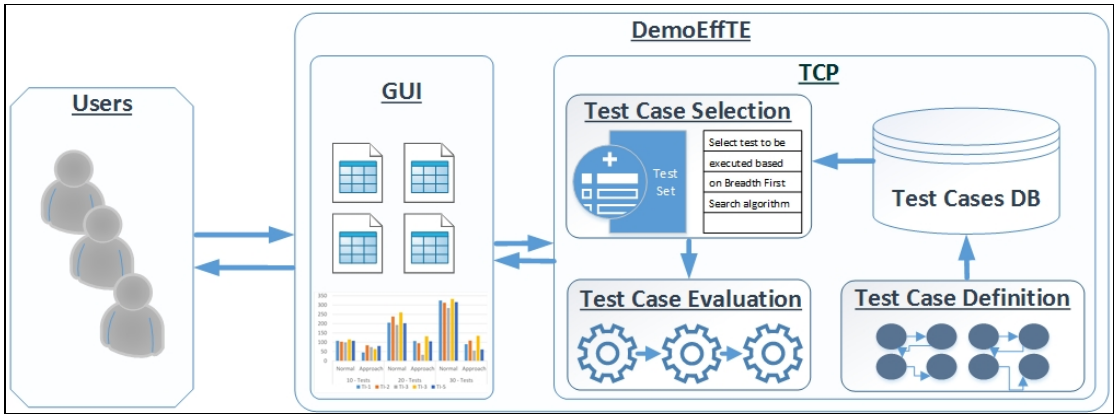

Figure 1. The architecture of the DemoEffTE demonstrator. Users are able to choose different scenarios for evaluation of the EffTE approach compared to a naive approach, as well as a set of test cases and their dependencies; the result of the validation process i s returned as output.

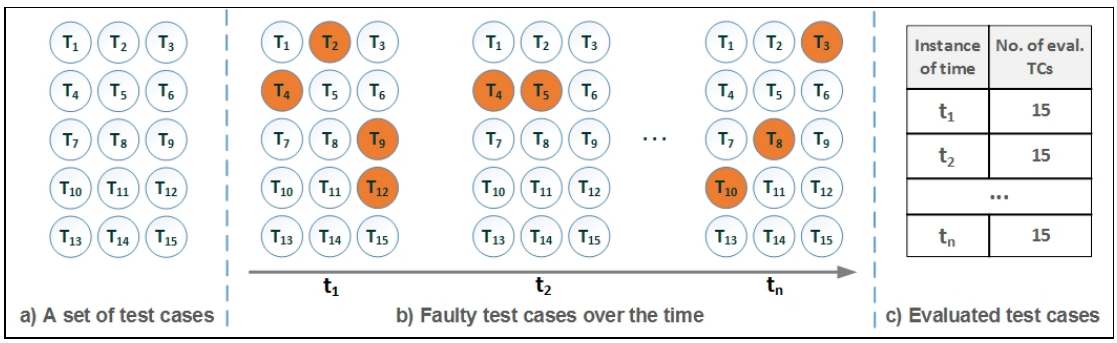

Figure 2. A Naive Approach. (a) Test Cases (TCs) to check if an ontology satisfies the design requirements;

(b) faulty test cases (orange) over time after each ontology modification; and (c) number of evaluated test cases (TCs) per instance of time. Every instan ce of time, the set of test cases is completely evaluated.

DemoEffTE demonstrates the evaluation of a set of 15 test cases using the naive approach and the *EffTE* framework. Figure 2 illustrates that in different instances of time, the number of evaluated test cases after each ontology modification using a naive approach is always the same, i.e., 15 test cases. On the other hand, as shown in Figure 3 , the number of evaluated test cases in different instance of times is reduced. This comes as a result of considering the parent-child relationships between test cases modeled as a dependency graph, where a faulty parent indicates a set a faulty children test cases.

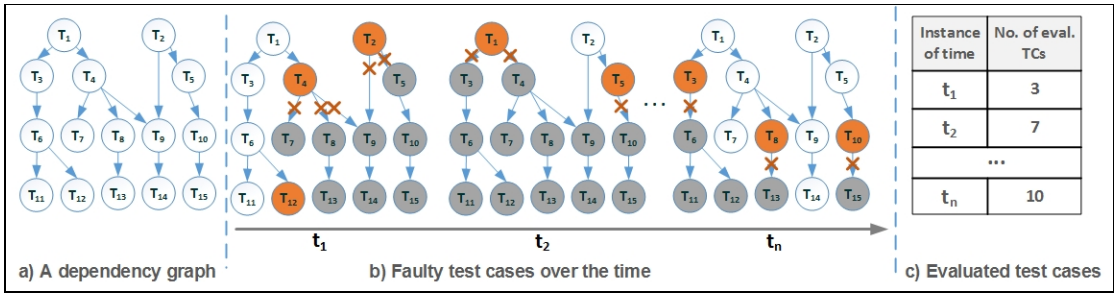

Figure 3. The EffTE Approach. (a) A test case dependency graph; (b) faulty test cases over time (orange nodes are faulty test cases), gray nodes represent test cases ignored for evaluation; and (c) number of evaluated test cases (TCs) per instance of time. Depen dencies enable to ignore faulty test cases.

The results are visualized using D3.js library according to the selected options. Additionally, *DemoEffTE* illustrates traversals of a dependency graph according to a breadth-first search algorithm. This illustration is enabled in an animation view developed on Cytoscape.js [2] library. Bar charts that depict the results are designed using Highcharts.js [3] . The *TCP* component is implemented using Java Apache Jena libraries version 3.0.1. The entire implementation of *DemoEffTE* is deployed on a virtual server with AMD Opteron 2.3 GHz CPU, 4 cores, 8 GB RAM operating in SUSE Linux Enterprise Server 12 SP1.

3. Demonstration of Scenarios

We design several scenarios to illustrate the efficiency of the *EffTE* approach. Thus, attendees will be able to observe how a naive approach and the *EffTE* approach exhibit different behavior on the evaluation of a given set of test cases. *DemoEffTE* demonstrates the execution of both approaches in five instances of time. In the first four time instances, various test cases are randomly chosen to fail. No faulty test cases are included in the instance number five, in order to show the behavior of both approaches on the same conditions. Two evaluation metrics will be shown: (1) evaluation time; and (2) number of evaluated test cases. The following scenarios are demonstrated:

Effect of an Ontology Size . The impact of the ontology size will be observed on a set of test cases. We asses the efficiency of a naive approach and the *EffTE* framework on three different ontologies: (1) *FOAF* : an ontology used to describe persons, activities, and relationships between them and other objects; (2) *Schema.org* ontology describes entities, actions, and their relations to promote the usage of the structured data; and (3) *DBpedia* , a cross-domain ontology for describing the information extracted from Wikipedia infoboxes. Table 1 describes these ontologies in terms number of triples, different subjects, properties, and objects.

Ontology	# triples	# subjects	# properties	# objects
FOAF	631	86	15	192
Schema.org	8,103	1,569	13	3,545
DBpedia	30,793	3,986	23	16,807

Table 1. Description of Ontologies. Different ontology sizes in terms of number of triples, subjects, properties, and objects.

Impact of the Topology of the Dependency Graph . The goal of this scenario is to show the impact of the topology of dependency graph in the evaluation of set of test cases. Figure 4 shows three topologies with 10 test cases each where test case has different dependencies between each other. The topologies are randomly generated using RStudio [4] *randomDag* [5] function. Attendees will be able to visualize the results from evaluation of a given set of test cases produced over a period of five instances of time.

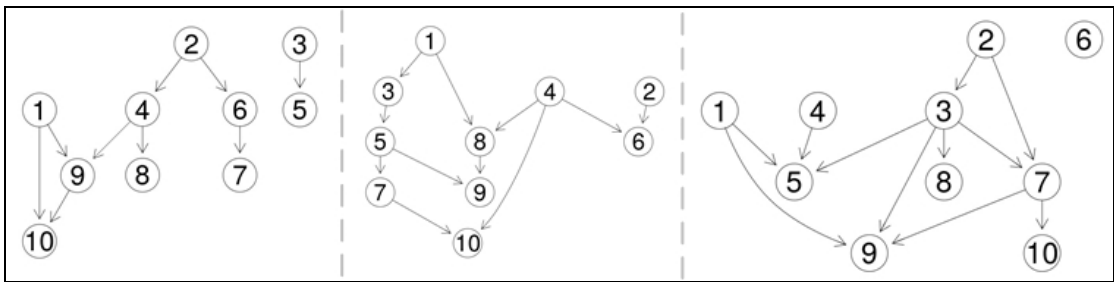

Figure 4. Test Cases. Three different topologies of dependency graphs between test cases.

Impact of the Number of the Test Cases. We demonstrate three different dependency graphs composed of 10, 20, and 30 test cases, respectively. Figure 5 illustrates the dependency graphs randomly generated using RStudio *randomDag* function. Similar to other scenarios, for each set, a random number of test cases are chosen to fail using a *Random Distribution* function in RStudio. Attendees will be able to visualize the results produced over a period of time.

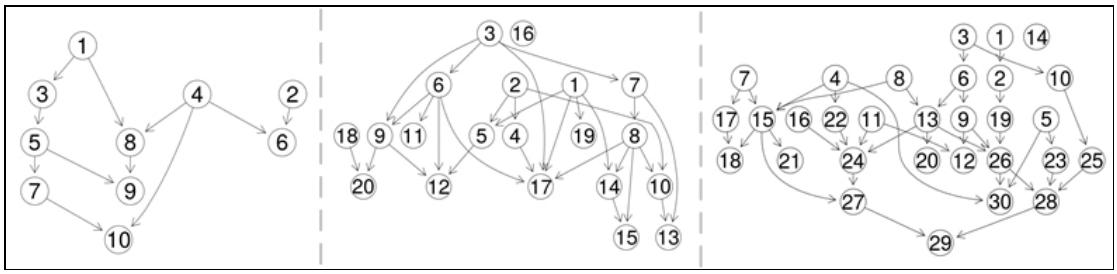

Figure 5. Test Cases. Three different sets composed of 10, 20, and 30 test cases, respectively.

4. Conclusions

DemoEffTE provides a visual comparison how modeling relationships between test cases using dependency graphs affects the performance of the overall evaluation time after each ontology change. Particularly, *DemoEffTE* shows a trade-off between a naive approach that exhaustively evaluates all test cases and the *EffTE* framework in three different scenarios. Attendees will be able to explore all the scenarios, understand why modeling relationship of test cases using dependency graphs play an important role in ontology development process. Furthermore, they will observe how this dependency graph modeling enables minimization of the test cases to be evaluated, thus reducing the time required for validating an ontology after each modification.

Acknowledgements

This work has been supported by the European Union's Horizon 2020 programme for the project BigDataEurope (grant no. 644564), and the German Federal Ministry of Education and Research (BMBF) for the projects Industrial Data Space (grant no. 01IS15054) and SDI-X (grant no. 01IS15035C).

References

1. Fraser, S. Beck, K., Caputo, B., Mackinnon, T., Newkirk, J., & C. Poole. (2003). Test driven development (tdd). In 4th International Conference on Extreme Programming and Agile Processes in Software Engineering, pages 459–462. Springer-Verlag.
2. Mehrotra, M. (2002). Ontology analysis for the semantic web. In Ontologies and the Semantic Web: AAAI Workshop (Technical Report WS-02-11). AAAI Press.
3. Mihindukulasooriya, N., Poveda-Villalón, M., García-Castro, R., & Gómez-Pérez, A. (2016). Collaborative Ontology Evolution and Data Quality - An Empirical Analysis. In 13th OWL: Experiences and Directions Workshop.
4. Uschold, M. & Gruninger, M. (1996). Ontologies: principles, methods and applications. Knowledge Engineering Review, 11(2):93–136.

Footnotes

1 http://wiki.dbpedia.org/services-resources/ontology/ [back]

2 http://js.cytoscape.org/ [back]

3 https://www.highcharts.com/ [back]

4 https://www.rstudio.com/ [back]

5 https://rdrr.io/rforge/pcalg/man/randomDAG.html [back]