Automatic Conformance Test Data Generation Using Existing Ontologies in the Web

Irlán Grangel-González, Raúl García-Castro

Ontology Engineering Group,
Departamento de Inteligencia Artificial
Facultad de Informática, Universidad Politécnica de Madrid, Spain
irlangrangel2003@gmail.com, rgarcia@fi.upm.es

Abstract. Conformance evaluation is key in the semantic technology landscape and semantic technologies should be continuously evaluated in order to ensure their conformance using affordable evaluation methods. This paper presents two methods to automatically increase the expressivity of an ontology, using existing ontologies in the Web in one of them and maximizing OWL 2 expressivity in the other; a process for generating conformance test data that relies in these methods; and three test suites that have been generated using Ontology Design Patterns as input for the generation process.

1 Introduction

Conformance evaluations have the goal of evaluating the conformance of semantic technologies with regards to existing specifications. Since for these technologies the most relevant specifications are those of ontology representation languages (i.e., RDF(S), OWL and OWL 2), conformance evaluations deal with evaluating up to what extent semantic technologies adhere to the specification of ontology representation languages [1].

Conformance evaluation is key in the semantic technology landscape because, on the one hand, existing specifications are not fully implemented by every tool and, on the other hand, specifications evolve over time which leads to the coexistence of different tools and specifications in certain time periods.

For this reason, semantic technologies should be continuously evaluated in order to ensure their conformance; however, this requires having affordable evaluation methods that cover the different aspects of evaluation: from the creation of test data to the analysis of results.

The work presented in this paper aims to facilitate the definition of new conformance test data. The manual definition of conformance test data followed up to now does not scale and, therefore, the coverage of the resulting data is not exhaustive enough.

The main contributions of this paper are: two methods to automatically increase the expressivity of an ontology, using existing ontologies in the Web in one of them and maximizing OWL 2 expressivity in the other; a process for

Proceedings of the Second International Workshop on
Evaluation of Semantic Technologies (IWEST 2012)
May 28th, Heraklion, Greece
CEUR Workshop Proceedings Vol. 843

generating conformance test data that relies in these methods; and three test suites that have been generated using Ontology Design Patterns as input for the generation process.

The rest of the paper is organized as follows: Section 2 defines our scope regarding conformance evaluation and section 3 presents existing conformance test data for evaluations. Then, section 4 presents the two methods for increasing ontology expressivity and section 5 details the process to generate test data using such methods. Section 6 shows how we have generated different test suites following such process and, finally, section 7 draws some conclusions from the work presented in the paper and outlines future work.

2 Evaluating Semantic Technology Conformance

There are several aspects of semantic technology conformance with respect to an ontology language specification, since it can be evaluated with regard to the ontology language model (since different tools have different internal representation models), to the ontology language serialization (both normative and non-normative ones), and to ontology manipulation (at different levels of granularity and complexity).

In this work, we focus in conformance regarding the ontology language model and in the evaluation approach proposed in [2]. In this approach, during the evaluation, a common group of tests is executed in two steps. Starting with a file containing an ontology, the execution consists in importing the file with the ontology into the origin tool and then exporting the ontology to another file.

After a test execution, we have two ontologies, namely, the original ontology and the final ontology exported by the tool. By comparing these ontologies we can know up to what extent the tool conforms to the ontology language.

3 Existing Conformance Test Data

Different works have dealt with the creation of conformance test data for evaluating semantic technologies. In this section we present them and describe the motivation of our work.

The W3C ontology language specifications include test case definitions for RDF(S) [3], OWL [4] and OWL 2 [5], which illustrate the correct usage of the ontology languages and the resolution of issues considered by the Working Groups. These test cases mainly cover conformance with regards to the ontology language semantics but also cover ontology language model and serialization conformance, both with correct and incorrect ontologies.

Conformance test data has also been defined for reasoners implementing the OWL 2 RL/RDF rules language [6], this work complements the W3C OWL 2 test data and a fraction of it is part of the official OWL 2 test suite.

The RDF(S) and OWL Interoperability Benchmarking activities [7, 2] involved the evaluation of the interoperability of semantic technologies using an

interchange language. Two test suites were used in these activities covering the RDF(S) and OWL Lite languages. Besides information about tool interoperability, these activities also provided information about their conformance, mainly in terms of the ontology language model.

In the scope of the SEALS Yardsticks for Ontology Management evaluation campaign [8], the abovementioned test suites were extended to cover OWL DL and OWL Full. To define the OWL DL test suite, a keyword-driven ontology generator[1] was used to facilitate the manual creation of test ontologies; this generator allowed to significantly increase the amount of test ontologies.

Current ontology generators (e.g., the Lehigh University Benchmark [9]) could be used in conformance evaluations; however, they cover a predefined part of the specification regardless of the number or size of ontologies generated and they are not customizable in terms of coverage.

The test ontologies defined in the abovementioned works share some characteristics: they have been defined manually, which is costly and prone to errors; are simple, so they can be used to test in isolation specific characteristics of the language; and rarely include real-world uses in their ontology constructs.

Therefore, the expressivity covered with these ontologies is low and does not allow making an exhaustive evaluation of tool conformance in terms of the ontology language model.

In this work, we aim for the automated generation of conformance test data (i.e., ontologies) that, besides being recognized as a best practice in software testing [10], provides a scalable way of defining conformance test data. This will allow us to obtain an affordable way of increasing the expressivity of ontologies and, consequently, of increasing the exhaustiveness of conformance evaluations. Furthermore, this expressivity increase comes from real-world ontologies so the obtained test data resembles actual ontology modelling patterns.

4 Methods to increase ontology expressivity

This section presents two methods to increase ontology expressivity. First, we define a method that increases ontology expressivity using existing ontologies in the Web. Then, we define a method that maximizes ontology expressivity.

4.1 Method to increase ontology expressivity using existing ontologies

The aim of this method is to increase the expressivity of an ontology taking into account how its components are used in existing ontologies available in the Web. Starting from an ontology (O), for each class (C_i) in the ontology the steps to perform are

[1] `http://knowledgeweb.semanticweb.org/benchmarkinginteroperability/OWLDLGenerator/`

1. To search (discarding duplicates) those ontologies in the Web (O_i) that contain the class identifier (i.e., the IRI fragment identifier of class C_i). In the case of not finding the class identifier in the existing ontologies, the method finishes without adding anything to the ontology.
2. To extract a module (M_i) from each of the ontologies found (O_i), taking as the module signature the class with the same identifier as C_i. In practice, there are cases where the module cannot be extracted, for example, when the ontology is not available, causes parsing problems or is not in the OWL ontology language.
3. To add the extracted modules (M_i) to the initial ontology (O), replacing the namespace of the found ontologies (O_i) by the namespace of the starting one (O).
4. To detect inconsistencies in the updated ontology after all the modules have been included. In the case of finding any inconsistency, the axioms that cause the inconsistency are removed from the ontology.

As an example, figure 1 shows a fragment of the ontology generated from the *Parameter* Ontology Design Pattern ontology[2] (the components of the initial ontology are shown in bold). The resulting ontology contains 177, classes 47 object properties and 5 datatype properties (from the 8 classes, 8 object properties and 1 datatype property in the initial ontology). As can be seen, the initial ontology has been enriched with class hierarchies, disjoint classes, transitive properties, and universal, existential, and cardinality restrictions.

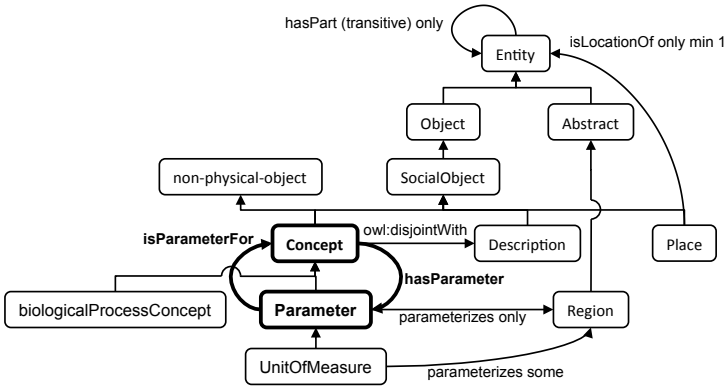

Fig. 1: Fragment of the ontology generated from the *Parameter* ODP.

There are two parameters that affect the results of this method. One is the number of classes (C_i) that are selected from the starting ontology (O); the other is the number of ontologies that are selected from those obtained in the Web

[2] `http://ontologydesignpatterns.org/wiki/Submissions:Parameter`

search in the step 1 above (O_i). In section 6.2 we show how we have identified that the method is more effective (in terms of expressivity increase and efficiency) when it selects all the classes in the starting ontology and takes into account the first 15 ontologies found in the Web.

4.2 Method to maximize ontology expressivity

The goal of this method is to increase the expressivity of an ontology to cover all the existing constructs in the OWL 2 ontology language [11], that is, so the ontology has a full expressivity ($\mathcal{SROIQ}(\mathcal{D})$) and contains every OWL 2 construct not related to such expressivity (e.g., self-restriction, key and property chain). Starting from an ontology (O), the steps to perform are

1. To calculate the expressivity of the ontology (O).
2. To identify the types of axioms needed in the ontology (O) to maximize its expressivity. For example, if the ontology has $\mathcal{SROIQ}$ expressivity, axioms for data values or data types should be added.
3. For each axiom type identified, to add one axiom of such type (A_i) to the ontology (O). In order to build the axiom (A_i), the structure of the ontology (O) is analysed to check whether the axiom can be defined by reusing existing classes and properties in the ontology. Every required class and property that cannot be reused from the ontology is created with a fictitious identifier. For example, adding a reflexive property axiom requires having an object property. To satisfy this requirement, if the ontology has object properties, one of them is randomly chosen; if not, a fictitious object property is created.
4. To detect inconsistencies in the updated ontology after all the axioms have been included. In the case of finding any inconsistency, the axioms that cause the inconsistency are removed from the ontology.

5 Expressive test data generation process

This section presents the process followed to generate expressive test data for conformance evaluations.

Our requirements for test data generation are: a) to use any set of OWL ontologies as input, either OWL version 1 or 2; b) to generate ontologies that reuse parts of real-world ontologies; and c) to generate valid OWL 2 ontologies.

The process, presented in figure 2, starts with a set of ontologies and ends with three different test suites: one that includes the original ontologies (to be used for baseline comparisons), another that includes the original ontologies enriched with axioms found in existing ontologies, and a third one that includes the original ontologies enriched up to a maximum expressivity.

To generate these test suites, we use the methods presented in section 4. To generate the test suite with maximum expressivity, instead of starting from the original set of ontologies, the enriched ontologies are taken as input.

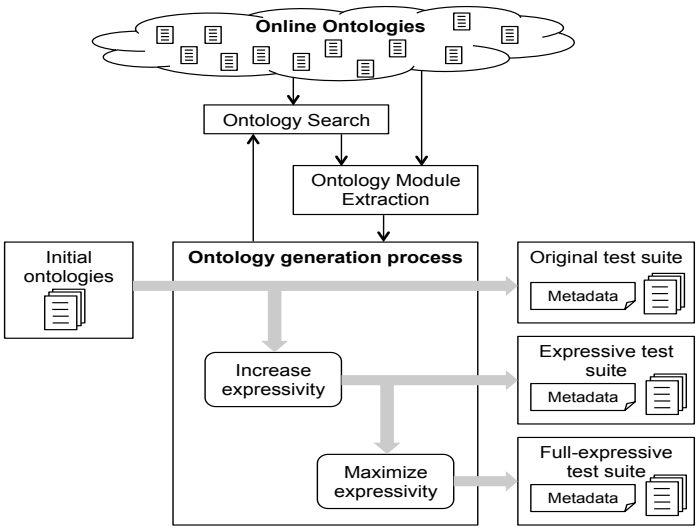

Fig. 2: Conformance Test Data Generation Process

We have implemented the previous process in the OWL2EG (OWL 2 Expressive Generator) Java application[3], which has the following characteristics:

- For searching ontologies in the Web, there are multiple systems available (e.g., Watson[4], Sindice[5], Swoogle[6], or Falcon-S[7]); we chose Watson because it is the only one that provides unlimited access to its functionalities through programming interfaces, while the other systems have use limitations [12].
- For extracting ontology modules, there are multiple approaches available [13–15]; we chose the locality-based module extraction approach proposed by Cuenca-Grau et al. and implemented in the OWL API[8].

  There is one constraint when using the OWL API to extract ontology modules; the implemented algorithm only works with OWL ontologies and not with ontologies in other languages (e.g., RDF-S or DAML). Because of this, we only search for OWL ontologies in Watson, even if we disregard plenty of ontologies in other languages.
- In each test suite, besides generating the ontologies, the tool also generates metadata that allow managing these ontologies in an automated way. These

[3] http://knowledgeweb.semanticweb.org/benchmarking_interoperability/OWL2EG/
[4] http://watson.kmi.open.ac.uk/
[5] http://sindice.com/
[6] http://swoogle.umbc.com/
[7] http://ws.nju.edu.cn/falcons/
[8] http://owlapi.sourceforge.net/

metadata are defined according to the SEALS ontologies[9] and allow the test suites to be processed by the SEALS Platform [16].

6 Generating expressive ontology test data

This section presents how we have generated a set of test suites using the generator presented in section 5, which implements the methods presented in section 4. We describe the ontologies selected as input for the generation process, the tuning of the generator parameters, and the resulting test suites.

6.1 Selecting input ontologies

One of the requirements for the test data generation process and also for the test suites is that test ontologies reuse real-world ontologies. We have tackled this requirement in two fronts: in the generation process by extending ontologies with information from existing ontologies in the Web and in the selection of input ontologies for such process by selecting ontologies created from content Ontology Design Patterns.

Design patterns were first largely used in the Software Engineering field [17] and in the last years their use has spread in the Ontology Engineering field where they are defined as modelling solutions to solve recurrent ontology design problems [18]. There are different types of Ontology Design Patterns (ODPs) grouped into six families: Structural, Correspondence, Content, Reasoning, Presentation, and Lexico-Syntactic. In order to define test data for conformance and interoperability evaluations, we will focus on Content ODPs.

Content ODPs provide solutions to recurrent domain modeling problems [18]. These patterns are relevant to our case because they represent best practices in ontology modelling frequently used by ontology developers and they are defined using simple OWL ontologies.

For defining the test suites, we have used as input for the generation process the 81 Content ODPs available in the OntologyDesignPatterns.org pattern repository[10].

6.2 Tuning the configuration parameters

This section presents the experiment that guided us to choose the optimal configuration parameters for the method to increase ontology expressivity using existing ontologies. As mentioned in section 4, two parameters affect the method results: the number of ontologies selected from those obtained in the Web search and the number of classes selected from the starting ontology.

For this analysis, we have defined an *Ontology expressivity increase* metric that is calculated by counting the different types of expressivity added to an ontology. For example, if qualified cardinality restrictions ($\mathcal{Q}$) and nominal concept

[9] `http://www.seals-project.eu/ontologies/`

[10] `http://ontologydesignpatterns.org/wiki/Submissions:ContentOPs`

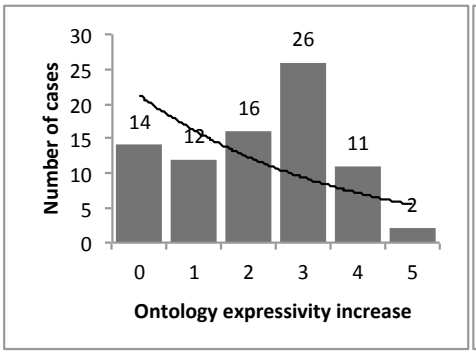

Fig. 3: Frequency of expressivity increase for 10 ontologies and all classes.

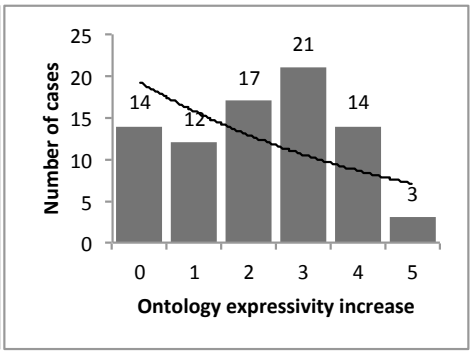

Fig. 4: Frequency of expressivity increase for 15 ontologies and all classes.

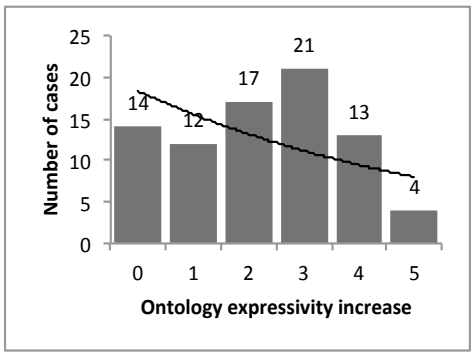

Fig. 5: Frequency of expressivity increase for 20 ontologies and all classes.

inclusion ($\mathcal{O}$) are added to one ontology, we state that the expressivity increase for that ontology is two.

Regarding the first parameter, we have executed the method twenty times varying the number of ontologies selected from those found using every value from 1 to 20; in every case all the classes in the origin ontology have been taken into account. Figures 3, 4, and 5 present the histograms with the frequency of Ontology expressivity increase for the values of 10, 15, and 20, respectively. The X axis represents the Ontology expressivity increase value and the Y axis the number of cases (i.e., generated ontologies) with a certain ontology expressivity increase.

The histograms show, on the one hand, a significant expressivity increase between 10 and 15 and, on the other hand, that between 15 and 20 the increase is similar (this is also confirmed by the asymptotic behavior of the exponential trend line in these two cases). Therefore, bearing in mind the trade-off between

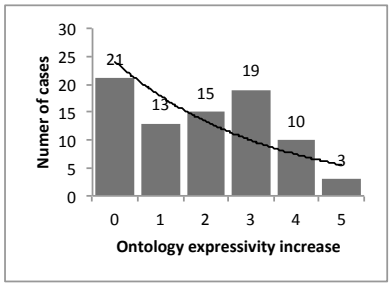

Fig. 6: Frequency of expressivity increase for 15 ontologies and half of the classes.

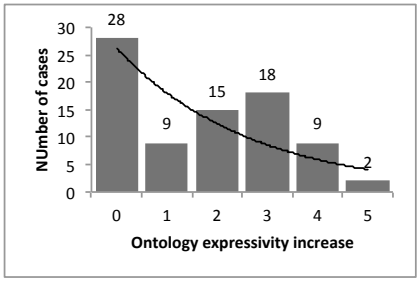

Fig. 7: Frequency of expressivity increase for 15 ontologies and one class.

time efficiency and expressivity increase, we chose 15 as the preferred number of ontologies to be selected from the Web search.

Regarding the second parameter, we have executed the method taking into account one, half and all the classes in the ontology. Figures 4, 6, and 7 present the histograms with the frequency of Ontology expressivity increase for the value of 15 and taking into account all the classes, half the classes, and one class in the ontology, respectively.

For this parameter, when all the classes are taken into account is when more expressive ontologies are obtained. This is logical, since in this case there are more chances of finding classes in the Web and, therefore, of adding more modules to the ontology.

6.3 Resulting test suites

After executing the OWL2EG generator using the ontologies and parameters mentioned in the previous sections, we obtained three different test suites:

- The **OWL Content Pattern Test Suite** (OCPTS), which includes the ontologies extracted from the OntologyDesignPatterns.org pattern repository.
- The **OWL Content Pattern Expressive Test Suite** (OCPETS), which includes the ontologies obtained after applying the method to increase ontology expressivity using existing ontologies to the OCPTS ontologies.
- The **OWL Content Pattern Full Expressive Test Suite** (OCPFETS), which includes the ontologies obtained after applying the method to maximize ontology expressivity to the OCPETS ontologies.

Next, we analyze the expressivity increase obtained in the OCPETS through the application of the method for increasing expressivity using existing ontologies; the histogram with the expressivity increase per ontology can be seen in figure 4. As the figure shows, the method added new expressivity to 83% of the ontologies. For 14 ontologies expressivity was not increased. This is due to:

Table 1: Percentage of expressivity types in the test suites.

	OCPTS	**OCPETS**		**OCPFETS**	
Expressivity type	**%**	**%**	**% inc.**	**%**	**% inc.**
ALC	63.0	88.9	25.9	100	11.1
Transitive roles ($\mathcal{R}_+$)	14.8	70.4	55.6	100	29.6
Role hierarchies ($\mathcal{H}$)	58.0	49.4	-8.6	100	50.6
Limited complex role inclusion ($\mathcal{R}$)	2.5	2.5	0	100	97.5
Nominal concept inclusion ($\mathcal{O}$)	2.5	64.2	61.7	100	35.8
Inverse roles ($\mathcal{I}$)	87.7	95.1	7.4	100	4.9
Cardinality restrictions ($\mathcal{N}$)	49.4	76.5	27.2	100	23.5
Qualified cardinality restrictions ($\mathcal{Q}$)	3.7	2.5	-1.2	100	97.5
Datatype properties, data values or data types ($\mathcal{D}$)	40.7	88.9	48.1	100	11.1

- In 9 cases, the search for the class identifier in existing ontologies did not return any ontology. This was because the Content ODP only contained highly-specific domain classes that do not appear in other ontologies; for example, the class *RTMS_Code* that belongs to the *RtmsMapping* pattern and the class *AquaticSpecies* that belongs to the *SpeciesNames* pattern.
- In 5 cases, when adding modules to the ontology no expressivity was added. However, we still included the modules in the ontologies so the new classes and properties could later support reaching full expressivity while using real-world entities as much as possible.

Table 1 shows the percentage of ontologies in the three test suites that have a certain expressivity type; it also shows the percentage of increase from one test suite to the next one, i.e., the percentage of those ontologies that didn't have the expressivity type and that have it once the corresponding method is applied.

As can be seen, the method for increasing expressivity using existing ontologies is more effective for some types of expressivity ($\mathcal{O}$, $\mathcal{R}_+$, and $\mathcal{D}$) but for other types there is no increase ($\mathcal{R}$, $\mathcal{Q}$, and $\mathcal{H}$). Moreover, in some cases expressivity decreases, this is because the method produced an inconsistent ontology and axioms that produced that expressivity had to be removed.

Regarding the expressivity increase obtained in the OCPFETS through the application of the method for maximizing expressivity (up to $\mathcal{SROIQ}(\mathcal{D})$), the table shows that most of the times the method had to add two types of expressivity ($\mathcal{R}$ and $\mathcal{Q}$) that were not present either in the original ontologies or in the ontologies in the Web.

7 Conclusions and Future Work

This paper has presented an automated process for generating test data for semantic technology conformance evaluations. This process takes as input a set of ontologies and increases their expressivity using constructs from ontologies available in the Web.

This process is implemented in the OWL2EG tool and, using the ontologies of an Ontology Design Pattern repository, we have generated test suites with ontologies covering different degrees of expressivity up to the maximum expressivity allowed in OWL 2.

These test suites will be used in the second SEALS Yardsticks for Ontology Management evaluation campaign; the results of this campaign will be published in the scope of the SEALS European project[11].

We must highlight that the method is concerned with increasing ontology expressivity and we disregard its effects in other aspects such as their usability (e.g., that the generated ontologies make sense in a concrete domain); our only requirements were that the resulting ontologies are syntactically valid and semantically consistent.

Related to the previous point and to the real-world representativeness of the ontologies available in the Web, not every ontology returned by Watson can be considered as a real-world ontology; some of these ontologies are inconsistent, some have been produced as examples for tutorials, etc.

From the results presented in section 6.3, we can see that using the ontologies available in the Web helps increasing expressivity but it is not a complete solution, since not every type of expressivity is found.

Future improvements could be related to enhancing the method to improve the amount of expressivity obtained and to use it with other groups of ontologies and compare the expressivity increase obtained.

Furthermore, the method effectiveness clearly depends on the expressivity of the input ontologies and of existing ontologies in the Web. The detailed effect of these expressivity types in the method outcomes requires further analysis.

Finally, currently we search the Web for all the classes in the ontology. One alternative to take into account would be to search only for the key classes in the ontology (using an approach such as the one presented in [19]).

Acknowledgments

This work has been supported by the SEALS European project (FP7-238975) and by the EspOnt project (CCG10-UPM/TIC-5794) co-funded by the Universidad Politécnica de Madrid and the Comunidad de Madrid.

References

1. García-Castro, R., Gómez-Pérez, A.: Perspectives in semantic interoperability. In: Proceedings of the 1st International Workshop on Semantic Interoperability, Rome, Italy (2011) 13–22
2. García-Castro, R., Gómez-Pérez, A.: Interoperability results for Semantic Web technologies using OWL as the interchange language. Web Semantics: Science, Services and Agents on the World Wide Web **8** (2010) 278–291

[11] `http://www.seals-project.eu/`

3. Grant, J., Beckett, D.: RDF Test Cases. Technical report, W3C Recommendation 10 February 2004 (2004)
4. Carroll, J., Roo, J.D.: OWL Web Ontology Language Test Cases. Technical report, W3C Recommendation 10 February 2004 (2004)
5. Smith, M., Horrocks, I., Krötzsch, M., Glimm, B.: OWL 2 Web Ontology Language Conformance. Technical report, W3C Recommendation 27 October 2009 (2009)
6. Schneider, M., Mainzer, K.: A Conformance Test Suite for the OWL 2 RL/RDF Rules Language and the OWL 2 RDF-based Semantics. In: Proceedings of the OWL: Experiences and Directions Workshop 2009 (OWLED 2009), Chantilly, VA, USA (2009)
7. García-Castro, R., Gómez-Pérez, A.: RDF(S) interoperability results for Semantic Web technologies. International Journal of Software Engineering and Knowledge Engineering **19** (2009) 1083
8. García-Castro, R., Toma, I., Marte, A., Schneider, M., Bock, J., Grimm, S.: D10.2. Services for the automatic evaluation of ontology engineering tools v1. Technical report, SEALS Project (2010)
9. Guo, Y., Pan, Z., Heflin, J.: LUBM: A Benchmark for OWL Knowledge Base Systems. Journal of Web Semantics **3** (2005) 158–182
10. Chillarege, R.: Software testing best practices. Technical Report RC 21457, IBM Research (1999)
11. Motik, B., Patel-Schneider, P., Parsia, B., Bock, C., Fokoue, A., Haase, P., Hoekstra, R., Horrocks, I., Ruttenberg, A., Sattler, U., Smith, M.: OWL 2 Web Ontology Language: Structural specification and functional-style syntax. W3C Recommendation. 27 October 2009. Technical report, W3C (2009)
12. d'Aquin, M., Sabou, M., Motta, E., Angeletou, S., Gridinoc, L., Lopez, V., Zablith, F.: What can be done with the Semantic Web? an overview of Watson-based applications. In: Proceedings of the 5th Workshop on Semantic Web Applications and Perspectives (SWAP 2008). Volume 426., Rome, Italy, CEUR-WS (2008)
13. d'Aquin, M., Sabou, M., Motta, E.: Modularization: a key for the dynamic selection of relevant knowledge components. In: Proceedings of the 1st International Workshop on Modular Ontologies (WoMO 2006), Athens, GA, USA (2006)
14. Doran, P., Palmisano, I., Tamma, V.: SOMET: Algorithm and tool for SPARQL based ontology module extraction. In: Proceedings of the International Workshop on Ontologies: Reasoning and Modularity (WORM 2008), Tenerife, Spain (2008)
15. Cuenca-Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Just the right amount: extracting modules from ontologies. In: Proceedings of the 16th International World Wide Web Conference (WWW 2007), Banff, AB, Canada, ACM (2007) 717–727
16. García-Castro, R., Esteban-Gutiérrez, M., Gómez-Pérez, A.: Towards an infrastructure for the evaluation of semantic technologies. In: Proceedings of the eChallenges 2010 Conference, Warsaw, Poland (2010) 1–8
17. Gamma, E., Helm, R., Johnson, R.E., Vlissides, J.: Design Patterns. Elements of Reusable Object-Oriented Software. Addison- Wesley (1995)
18. Gangemi, A., Presutti, V.: Ontology Design Patterns. International Handbooks on Information Systems. In: Handbook on Ontologies. Second edition. Springer (2009) 221–243
19. Peroni, S., Motta, E., d'Aquin, M.: Identifying key concepts in an ontology, through the integration of cognitive principles with statistical and topological measures. In: Proceedings of the 3rd Asian Semantic Web Conference (ASWC 2008), Bangkok, Thailand (2008) 242–256