Semantic Mutation Test to OWL Ontologies

Alex Mateus Porn and Leticia Mara Peres
Informatics Department, Federal University of Paraná, Av. Cel. Francisco H. dos Santos, Curitiba, Brazil

Keywords: Ontology, Semantic, Mutation.

Abstract: Ontologies are structures used to represent a specific knowledge domain. There is not a right way of defining an ontology, because its definition depends on its purpose, domain, abstraction level and a number of ontology engineer choices. Therefore, a domain can be represented by distinct ontologies in distinct structures and, consequently, they can have distinct results when classifying and querying information. In light of this, faults can be accidentally inserted during its development, causing unexpected results. In this context, we propose semantic mutation operators and apply a semantic mutation test method to OWL ontologies. Our objective is to reveal semantic fault caused by poor axiom definition automatically generating test data. Our method showed semantic errors which occurred in OWL ontology constraints. Eight semantic mutation operators were used and we observe that is necessary to generate new semantic mutation operators to address all OWL language features.

1 INTRODUCTION

Ontologies are considered one of the semantic web support. They describe and represent concepts of a domain and relations between them. They have a fundamental role to describe data semantics and act like a backbone in knowledge based systems.

In information systems, an ontology is an engineering artefact. They introduce a vocabulary to define concepts, classify terms and relationships, and define constraints (Gruber, 1995). They are used to describe and represent a knowledge domain, and provides an explicit specification of this vocabulary (Horrocks, 2008). Ontologies can be very complex, with several thousands of terms or very simple, describing one or two concepts only.

The World Wide Web Consortium (W3C) developed the Ontology Web Language (OWL) standard (McGuinness and Harmelen, 2004). OWL is a semantic web language designed to represent rich and complex knowledge about things, groups of things, and relations between things. It is a computational logic-based language such that knowledge expressed in OWL can be exploited by computer programs, e.g., to verify the consistency of that knowledge or to make implicit knowledge explicit (OWL Working Group, 2012).

OWL formalism adopts an object-oriented model in which the domain is described in terms of individuals, classes and properties (Horrocks, 2008). A key feature of OWL is it is based in a very expressive Description Logics (DL), and in light of this, an OWL ontology consists of a set of axioms which are defined to represent a specific knowledge domain (Horrocks, 2008).

However, a knowledge domain can be represented by several distinct ontologies, which can result in distinct structures, axioms and consequently, they can have distinct results when classifying and querying information. Therefore, ontology evaluation is an important ontology engineer process to identify whether an ontology meet its goals and whether it is free of faults.

In light of this, we propose apply semantic mutation test to OWL ontologies and generate automatically test data, with objective to reveal semantic faults in OWL ontology constraints defined for any knowledge domain.

In the following, we describe related works in section two, a briefly semantic mutation test overview, semantic mutation operator definition and semantic mutation test application in section three. Next, we showing our experiment setup in section four, and last, our conclusions.

Porn, A. and Peres, L.
Semantic Mutation Test to OWL Ontologies.
DOI: 10.5220/0006335204340441
In *Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS 2017) - Volume 2*, pages 434-441
ISBN: 978-989-758-248-6

2 RELATED WORKS

In (Grüninger and Fox, 1995) is provided a mechanism to guide evaluation of design and adequacy of ontologies. Firstly, informal competency questions are defined. Then, using a first-order logic terminology, they are converted in formal competency questions. These formal competency questions are used as axioms in ontology evaluation.

With objective to guarantee that an ontology is well-verified, (Gómez-Pérez, 1996) presents a framework which evaluates correctness of ontology definitions. Using design criteria, Gómez-Pérez analyses architecture, lexicon and syntax, and content. The focus of this work is ontology evaluation, which consists on verification, validation and assessment.

Several authors propose methods to semantic evaluation. (Poveda Villalón et al., 2012), proposes a web based tool to improve ontology quality by automatically detecting potential pitfalls which could lead to modelling error.

(Batet and Sanchez, 2014) propose a score of the accuracy evaluation which is dependent of the degree of semantic dispersion of concepts in a given ontology. This work is based on (Fernández et al., 2009), which identified how taxonomic depth and breadth variance can be used to reasonably predict ontologies semantic accuracy.

In this same context, (Hlomani and Stacey, 2014) propose apply a data-driven ontology evaluation, using among others, clarity metric to measure the number of word meanings, to evaluate quality and correctness of the ontology.

According (Porn et al., 2016), ontology testing as a specific ontology evaluation process is little explored in the literature. In this sense, (Vrandečić and Gangemi, 2009) propose apply unit testing in ontologies like in software engineer.

In (García-Ramos et al., 2009) is proposed a method to dynamically test ontologies. An automated tool allows the user to define a set of tests to check the functional requirements of ontology, to execute them, and to inspect the results of execution.

In (Blomqvist et al., 2012), is proposed to find errors verifying ontology inference through error provocation and competency questions verification.

Some previous works concerning OWL mutation testing can be saw in (Lee et al., 2008), (Porn and Peres, 2014) and (Bartolini, 2016). Those three works are similar to our proposal in this paper.

In (Lee et al., 2008) mutation operators are applied in the mutation test to OWL-S, a standard XML-based language for specifying workflows and semantic integration among Web services (WS). In this work, they analyse fault patterns of specific OWL-S and their workflows, proposes an ontology-based mutation analysis method, and applies specification-based mutation techniques for WS simulation and testing.

In (Porn and Peres, 2014) is proposed apply mutation test exclusively to OWL ontologies, and 19 syntactic mutation operators on classes and relations structures are defined with goal to find OWL ontology pitfalls according to faults found in the literature. However, in this work is not applied mutation on OWL ontology DL axioms and is not generate automatically test data. Although these 19 operators are defined to syntactic mutation, 5 of them can be also used to semantic mutation.

Similar process is used in (Bartolini, 2016). In this work are presented 22 mutation operators to semantic mutation test, but 9 of them are similar to proposed in (Porn and Peres, 2014). Others proposed syntactic operators are applied in OWL annotations and OWL structure, not being possible apply them in DL axioms to evaluate OWL ontology semantic. Results do not show the mutation score to analyse the test data quality.

Small change in a semantic definition can produce, in knowledge-based systems like ontologies, a semantic meaning which is completely distinct from the original axiom. In this sense, just syntactic analysis is not enough to test an ontology.

Therefore, in this paper, we propose semantic mutation operators and apply semantic mutation test method to OWL ontologies. These semantic operators are defined to make syntactic changes in DL constraints of OWL ontology. They are applied with aim to reveal semantic fault caused by poor axiom definition and automatically generate test data.

3 OWL SEMANTIC MUTATION

We define OWL semantic mutation test as an error-based technique where syntactic changes are introduced in a set C of DL constraints or DL axioms of an OWL ontology O. These syntactic changes are made through predefined mutation operators, and each change generates a new set C' called mutant of C.

Thus, semantic mutation test in OWL ontologies consists in make changes in OWL ontology constraints Q of a set C, replacing, removing or adding logic operators, generating new constraints Q' of a set C' and which can give another meaning to original constraint Q, according to predefined mutation operators. In this sense, it is possible consider as a test case set *T*, a set of competency questions proposed in (Grüninger and Fox, 1995).

For this comparison is necessary to execute *O* and *C*, and *O* and *C'* with the same test case set *T*. After these executions, the results are compared to analyse if the results of distinct executions, with the same set *T*, are distinct for the execution of *C'* in *O*. A set *T* is used to distinguish the results of *C'* from *C*, where each *C'* is executed with the same set *T* applied to *C* in *O* (Delamaro et al., 2007).

Similar to program mutation test, if after executing all test cases *T*, there are still *Q'* in *C'* of *O* that generate the same output of *Q* in *C* of the same *O*, and it is not possible generate new test cases that differentiate *Q* from *Q'*, the mutant constraint *Q'* is considered similar to *Q*, it means that, or *Q* is correct, or it has errors unlikely to occur (DeMillo, 1978).

In the same way, as in software engineer, test case set *T* is suitable to *C* concerning *C'*, whether for each constraint *Q* belonging to *C'*, or *Q'* is similar to *Q* or *Q'* is distinct from *Q* in at least one test data (Delamaro et al., 2007).

Deciding whether a mutant is equivalent to an original constraint, is made by the engineer, because determine whether two programs compute the same function is an undecidable question (Budd, 1981).

Although mutation operators apply syntactic changes, they are considered semantic operators because they allow semantic analysis of results.

With the objective of to analyse adequacy of *T* executed in *C* and C', the mutation score is calculated. This score ranges from 0 to 1 and provides an objective measure of how much *T* is considered appropriate (DeMillo, 1978 and Delamaro et al., 2007). For an ontology *O*, a set of constraints *C'* and a set of test cases *T*, the mutation score *S* is obtained as follows (DeMillo, 1978):

$$S(T) = \frac{C'm}{C'g - C'e} \qquad (1)$$

The mutation score is obtained through the total of dead mutant constraints (*C'm*) of OWL ontology *O*, over the generated mutant constraints (*C'g*) of OWL ontology *O*, minus equivalent mutant constraints (*C'e*) of OWL ontology *O*.

3.1 OWL Semantic Mutation Operators

In (Porn and Peres, 2014), 19 mutation operators were defined to introduce syntactic changes in OWL ontologies. Those operators generate variations like change hierarchical structure of a class, add or remove a disjunction definition between classes, add or remove a class equivalence definition, remove “AND” and “OR” operators in an equivalence definition, among others.

Some of those operators can be used to produce semantic mutation in OWL ontologies, but they are not sufficient to test all OWL possibilities.

According (Horrocks et al., 2000), a description logic knowledge base is made up of a terminological part (Tbox) and an assertional part (Abox), each part consisting of a set of DL axioms. Tbox asserts facts about concepts and roles (binary relations), usually in the form of inclusion axioms, while Abox asserts facts about individuals (single objects), usually in the form of instantiation axioms.

OWL comprises Tbox and Abox, and it is based on a very expressive DL called $\mathcal{SHOIN}$, a sort of acronym derived from several language features. The symbol $\mathcal{S}$ is an abbreviation to $\mathcal{ALC}$, a DL Alternative Language whit add-ons. It depicts a basic set of features, like data types, constraints as intersection, union and complement, as well as, existential and universal quantifiers. The symbol $\mathcal{H}$ corresponds to properties hierarchies, symbol $\mathcal{O}$ to enumerated classes and axioms, like disjunction and equivalence, symbol $\mathcal{I}$ stands inverse properties and symbol $\mathcal{N}$ cardinality restrictions.

How description logic is composed by several formal knowledge representation languages, and OWL ontologies implementation is based on DLs, each semantic mutation operator should consider Tbox or Abox axioms defined in DL $\mathcal{ALCON}$, in other words, axioms defined in Attributive Language ($\mathcal{AL}$), which allow atomic negation, concept intersection, universal restrictions and limited existential quantification. They should support complex concept negation ($\mathcal{C}$), enumerated classes of objects value restrictions ($\mathcal{O}$) or cardinality restrictions ($\mathcal{N}$).

Therefore, to apply this semantic mutation test method to OWL ontologies, we select 5 mutation operators (CEUA, CEUO, ACOTA, ACATO and ACSTA) proposed in earlier works (Porn and Peres, 2014), and we defined 3 new operators to generate semantic mutants and generate automatically test data. These 8 operators accomplish at least one of the DL $\mathcal{ALCON}$ feature.

In this context, we propose the following semantic mutation operators applied over DL constraints defined in OWL classes, OWL object properties and OWL data properties. The semantic mutation operators are presented as follow:

- CEUA removes each AND operator in a given OWL DL constraint, generating two mutants, one with the left side of the AND operator and another with the right side.
- CEUO removes each OR operator in a given OWL DL constraint, generating two mutants, one with the left side of the OR operator and another with the right side.
- ACOTA replaces each OR operator in a given OWL DL constraint by one AND operator, generating one mutant for each OR operator replaced.
- ACATO replaces each AND operator in a given OWL DL constraint by one OR operator, generating one mutant for each AND operator replaced.
- ACSTA replaces each ∃ (Existential) operator in a given OWL DL constraint by one ∀ (Universal) operator, generating one mutant for each Existential operator replaced.
- ACATS replaces each ∀ (Universal) operator in a given OWL DL constraint by one ∃ (Existential) operator, generating one mutant for each Universal operator replaced.
- AEDN adds one negation operator for each AND, OR, ∃ or ∀ operator in a given OWL DL constraint, generating one mutant to each operator.
- AEUN removes one negation operator in a given OWL DL constraint, generating one mutant for each *not* operator removed.

Some considerations about these operators are which they are applied only in logical axioms, because we considered in this analysis to apply semantic mutation test on OWL constraints defined according to at least on DL $\mathcal{ALCON}$ feature.

These types of axioms do not include annotations or labels, they include axioms like disjunction between class, cardinalities constraints, domain and range definitions of classes, object properties and data type properties, as well as, axioms which are not addressed by the proposed mutation operators, like an equivalence axiom defined by only one class, similar to say which *disease* class is equivalent to the *pathology* class, or remove a disjunction definition between these two classes.

Table 1 shows an example of the semantic mutation test applied in the *people* OWL ontology (Bechhofer et al., 2003), where an equivalence axiom defined on the class "haulage truck driver" is mutated with ACATO semantic mutation operator, generating three new mutant OWL ontology constraint. Each one of these constraints is also used as a test data.

Table 1: Example of semantic mutation operator ACATO applied in an original OWL ontology constraint.

Class	Constraint	Mutant constraint
haulage truck driver	Person **and** (drives some truck) **and** (works for some (part of some 'haulage company'))	person **or** (drives some truck) **and** (works_for some (part_of some 'haulage company'))
		person **and** (drives some truck) **or** (works_for some (part_of some 'haulage company'))
		person **or** (drives some truck) **or** (works_for some (part_of some 'haulage company'))

3.2 Semantic Mutation Application

We propose to execute semantic mutation test in OWL ontologies in a similar process as (Porn and Peres, 2014):

1. Mutation operator selection: the first step is select or define appropriate semantic mutation operators which address at least one DL $\mathcal{ALCON}$ feature, and realize semantic mutations on DL constraints of OWL ontologies.
2. Mutants DL constraints generation: each selected semantic mutation operator is applied on original DL constraints *Q* in *C* of original OWL ontology *O* based on its specification. They generate an arbitrary number of mutant constraints *Q'* with the same error type applied in distinct constraints.
3. Test data generation: each original DL constraint *Q* and each new mutant DL constraint *Q'* in *C'* are selected as new test data, generating a set of test case *T* to be executed in the original ontology DL constraints *C* and each *C'*.
4. Original ontology constraint execution: after define *T*, *C* must be executed with *T*. Each test data in *T* is interpreted by a reasoner, which analyse test data according to *C* defined in the ontology, giving back a result based on super classes, subclasses and individual instantiation.
5. Mutants ontology constraint execution: each test data *T* executed in *C* should be executed in each mutant constraint *Q'*, and result of *C'*

should be compared with result of original ontology *O*.

6. Result analysis: whether a mutant *Q'* presents a distinct result of *Q* after a test data *T* execution, *Q'* is considered killed. Otherwise, whether *Q'* presents the same result of *Q* and is not possible generate new test data to be used, *Q'* can be considered equivalent to *Q*, or a fault was revealed in *C*. This analysis is determined by the engineer. The mutation score can be calculated after execute all mutants, to set suitability degree of used test data.

The main differences between these steps in relation to process proposed by (Porn and Peres, 2014), are the test data set automatically generated and the application of mutation operators in constraints, which satisfy at least one DL $\mathcal{ALCON}$ feature. In light of generate automatically test data, each mutated constraint is considered a new test data to be executed over DL mutant constraint generated of the OWL ontology.

About the application of mutation operators just in semantic context, it is an excellent alternative to produce mutants containing significant faults, and reduce the large number of inconsistent mutants and with obvious faults, like circulatory faults or partition faults (Gómez-Pérez, 2004) or unlikely to occur.

4 EVALUATION

4.1 Objectives

Executing semantic mutation test in OWL ontologies, and analyse in detail the application adequacy of proposed mutation operators, as well as, validate them and analyse the test data quality automatically generated.

4.2 Hypothesis

Mutation test is the most effective to reveal faults, but also the most expensive (Wong et al., 1995). According to this, for our evaluation we infer two hypotheses, identified as H1 and H2:

H1: Semantic mutation operators to OWL ontologies reduce application cost of mutation test and allows to reveal faults which are not identified with syntactic operators.

H2: After applying each semantic mutation operator in OWL ontology constraints, these mutated OWL constraints are effective test data to find faults in original OWL ontology.

4.3 Activities and Instruments

In order to execute and validate the steps of semantic mutation test presented in section 3.2, we used 8 semantic mutation operators presented in section 3.1.

Steps 2 and 3 were made using Protégé tool (Musen, 2015). Each semantic mutation operator was applied over all DL constraints, each operator at a time, generating mutants according to the number of existing operators. According to (Delamaro et al., 2007), it is possible generate mutants applying more than one mutation operator at once. However, this situation has a high cost of mutant generation and implementation, and it does not contribute to generate better test cases (Budd, T. A. et al., 1980).

For steps 4 and 5, each generated test data during step 3 is firstly executed on the original OWL ontology constraint *C* and next on each mutant OWL ontology constraint *C'*, using in these steps the Protégé DL query.

For step 6, the last one, the results of *C* and each *C'* are compared. This analysis is to verify results which are composed by instantiation of super classes, subclasses and individuals. If results of *C* and *C'* are distinct with the same test data, *C'* is considered killed and a fault revealed. However, if results are similar, *C'* can be considered equivalent to *C*, and in other words, or *C* is correct, or *C* has errors unlikely to occur.

Table 2 presents results example from an original OWL ontology constraint after executing a test data.

Table 2: Example of results from an original OWL ontology constraint.

Class	Original Constraint *C*	Test data *T*	Results
driver	`person` **`and`** `(drives some vehicle)`	`person` **`and`** `(drives some vehicle)`	Superclasses - adult - animal - grownup - person Subclasses - bus driver - haulage truck driver - lorry driver - mad cow - van driver - white van man Instances - Mick

After execute a test data it was obtained 4 super classes, 6 subclasses and 1 individual according to Table 2. On the other hand, according to Table 3, when executing the same test data in the OWL ontology mutant constraint, it was obtained 5 super classes, 7 subclasses and 1 individual. In this example, the test data revealed the inserted fault and the mutant was considered killed and discarded.

Table 3: Example of results from a mutant OWL ontology constraint.

Class	Mutant *C'*	Test data *T*	Results
driver	`person` **`or`** `(drives some vehicle)`	`person` **`and`** `(drives some vehicle)`	Superclasses - adult - animal **- driver** - grownup - person Subclasses - bus driver - haulage truck driver - lorry driver - mad cow - van driver - white van man **- kid** Instances - Mick

Defining if *C'* is equivalent to *C* is an undecidable question. Decide whether the test keep going while *C'* presents the same result from *C* is an engineer decision. The mutation score is used to evaluate generated test data and as a metric to decide whether it is necessary generates new test data or close the test.

4.4 Data Set

We selected as data set of our evaluation the OWL ontology called *people*, from (Bechhofer et al., 2003). This is a simple ontology which describes people, links between them, pets and things they do. This ontology has DL expressivity $\mathcal{ALCHOIN}$. It contains 372 axioms and 108 logical axioms. We consider to this setup only logical axioms, because they do not include annotations, which do not represent logical concepts.

Next, we presenting a summary of features of *people* OWL ontology: 372 axioms, 108 logical axioms, 60 classes, 14 object properties, 22 individuals, 33 subclasses; and 21 equivalent classes.

We consider only axioms which can be mutated and generated at the same time as a new test data, with at least one DL $\mathcal{ALCON}$ feature. Due to this, in *people* ontology we consider 68 logical DL $\mathcal{ALCON}$ axioms which can be mutated with these 8 semantic mutation operators.

4.5 Results

After execute semantic mutation test in *people* OWL ontology, we obtained 434 mutants for 8 semantic mutation operators and the mutation score 0,94. Table 4 presents the results, showing the total of generated, killed, inconsistent and live mutants.

Table 4: Semantic mutation test results.

Mutation operator	Generated Mutants	Killed Mutants	Inconsistent Mutants	Live Mutants
CEUA	77	73	0	4
CEUO	18	7	10	1
ACOTA	12	7	5	0
ACATO	45	43	2	0
ACSTA	66	57	0	9
ACATS	13	13	0	0
AEDN	200	152	38	10
AEUN	3	3	0	0
Total	434	355	55	24
Mutation score				0,94

Live mutants can be a fault or equivalent mutant. In this analysis, we considered live mutants as equivalent to original DL constraints.

Inconsistent mutant is a mutant that can not be executed by a reasoner, because it has in its definition inconsistent axioms. This fault type is immediately revealed in which reasoner is executed. In this case, no test data needs to be executed. However, this mutant type is considered a mutant killed, but it does not allow to evaluate the set of test case.

Therefore, to calculation of mutation score, we considered the total of mutants minus the total of inconsistent mutants as the total of generated mutants.

Table 5 shows the number of the test data used to kill each mutant DL constraint of *people* OWL ontology.

Table 5: Number of test data used to kill mutants.

Mutant operator	Total of mutants	Original test data	Mutated test data	Other test data
CEUA	77	25	46	1
CEUO	18	---	---	7
ACOTA	12	1	---	6
ACATO	45	18	25	---
ACSTA	66	34	5	18
ACATS	13	13	---	---
AEDN	200	62	25	65
AEUN	3	3	---	---

About Table 5, we considered on the "Original test data" column the same constraints used by mutation operator to generate mutant DL constraints. The column "Mutated test data" refers to the number of mutated constraints by the same mutation operator which generated mutant DL constraints. The column "Other test data" refers to the number of test data generated by a given mutation operator and used to kill a mutant DL constraint generated by other mutation operator.

4.6 Discussion

With these results was possible observe that, to kill some mutant DL constraints of *people* OWL ontology, generated by five mutation operators (CEUA, CEUO, ACOTA, ACSTA and AEDN) was necessary carry out test data generated by another mutation operators. This occurred because the original test data and mutated test data produced the same semantic result, but in contrast with another distinct constraint, the fault was revealed.

In other cases, like ACATS and AEUN mutation operators, original test data were sufficient to kill and reveal all inserted faults.

Some mutant DL constraints could not be killed in this experiment. We did not define them as equivalents or fault, because we consider that is necessary define new semantic mutation operators which approach other OWL language features. Thus, with new test data it will be possible that more mutants will be killed.

This result presents operators effectiveness, showing a large amount of errors which can be inserted in OWL ontologies by developer, due to the large number of generated mutants.

According (DeMillo, 1978), mutation score is better the closer to 1. Our test case set proved to be efficient to revealed fault, because the mutation score is 0,94. This score was found during the first execution of mutation test.

In a general context, according to hypothesis H1, this method of semantic mutation test produced a large number of mutants, but a smaller amount than the syntactic mutation test, decreasing the cost of its application and improving the quality of mutants, because few equivalent mutants have been generated, which provides a more accurate evaluation of the test data used in accordance with hypothesis H2.

5 CONCLUSIONS

OWL ontologies are knowledge representation model of a specific domain. However, a domain can be defined by several ways, because its definition depends on features as desired abstraction level, purpose and a number of developer choices, among others.

In this sense, there is not a right way to develop OWL ontologies. Methods to evaluate them are useful to guide developers and testers to develop ontologies which are correct or close to reality.

The semantic mutation test proposed in this paper showed be an alternative to OWL ontology testing. The generated mutants revealed faults with efficiency based on mutation score of 0,94.

According to this, the results of this method present a large amount of errors which can occur in OWL ontologies, based on a large number of generated mutants. In this context, the generated test cases proved are very efficient to revealed these errors, in accordance with few mutants remained alive, as mentioned in Table 4 and Table 5.

Our semantic mutation operators have shown that they do not cover all OWL language implementation possibilities. So, it is necessary generate new semantic mutation operators addressing other OWL language features, like add, remove or replace disjunction between class, cardinality restrictions, object properties and data type properties domain and range, as well as, semantic mutation operators to individuals, property characteristics, among others.

Therefore, it is still necessary to develop an automated tool to facilitate OWL semantic mutation test application, because Protégé tool aids to generate mutants and execute test data, but it does not provide a mechanism to execute this process automatically.

ACKNOWLEDGEMENTS

We thank the support provided by the Federal University of Paraná Graduate Program in Computer Science (PPGInf / UFPR), CAPES and Federal Institute of Paraná.

This work was conducted using Protégé resource, which is supported by grant GM10331601 from the National Institute of General Medical Sciences of the United States National Institutes of Health.

REFERENCES

Gruber, T. R., 1995. Toward principles for the design of ontologies used for knowledge sharing. In *Int. j. hum-comput. st.*, Vol. 43, No. 5-6, pp. 907-928.

Grüninger, M., Fox, M. S., 1995. Methodology for the Design and Evaluation of Ontologies. In *Workshop on Basic Ontological Issues in Knowledge Sharing in Int. Joint Conf. on Artificial Intelligence*. Toronto, Canada.

Gómez-Pérez, A., 1996. Towards a framework to verify knowledge sharing technology, Expert Systems with Applications, Vol 11, No. 4, pp. 519-529.

Poveda-Villalón, M., Suárez-Figueroa, M. C., Gómez-Pérez, A., 2012. Validating ontologies with OOPS!, In *18th Int. Conf. in Knowledge Eng. and Knowledge Management*. Galway City, Ireland, pp. 267-281.

Batet, M., Sanchez, D., 2014. A Semantic Approach for Ontology Evaluation. In *26th Int. Conf. on Tools with Artificial Intelligence*. Limassol, Cyprus, pp. 138-145.

Fernández, M., Overbeeke, C., Sabou, M., Motta, E., 2009. What Makes a Good Ontology? A Case-Study in Fine-Grained Knowledge Reuse. In *4th Asian Conference on The Semantic Web*, pp. 61-75.

Porzel, R., Malaka, R., 2004. A task-based approach for ontology evaluation. In *Proc. of the Workshop on Ontology Learning and Population at the 16th European Conf. on Artificial Intel*. Valencia, Spain.

Brewster, C., Alani, H., Dasmahapatra, S., Wilks, Y., 2004. Data driven ontology evaluation, In *Int. Conf. on Language Resources and Evaluation*. Lisbon, Portugal.

Porn, A. M., Huve, C. G., Peres, L. M. Direne, A. I., 2016. A Systematic Literature Review of OWL Ontology Evaluation. In *15th International Conference on WWW/Internet*. Mannheim, Germany, pp. 67-74.

Vrandečić, D., Gangemi, A., 2006. Unit tests for ontologies. In *On the Move to Meaningful Internet Systems*. Montpellier, France, pp. 1012-1020.

García-Ramos, S, Otero, A., Fernández-López, M., 2009. OntologyTest: A Tool to Evaluate Ontologies through Tests Defined by the User. In *10th Int. Work-Conf. on Art. Neural Networks*. Salamanca, Spain, pp. 91-98.

Blomqvist, E., Sepour, A. S., Presutti, V., 2012. Ontology testing-methodology and tool, In *18th International Conference in Knowledge Engineering and Knowledge Management*. Galway City, Ireland, pp. 216-226.

Lee S, Bai X, Chen Y., 2008. Automatic Mutation Testing and Simulation on OWL-S Specified Web Services. In *Simulation Symposium*, Annual, pp. 149-156.

Bartolini, C., 2016. Mutation OWLs: semantic mutation testing for ontologies. In *Proc. of the International Workshop on domain specific Model-based approaches to verification and validation*. Rome, Italy, pp. 43-53.

Porn, A. M., Peres, L. M., 2014. Mutation Test to OWL Ontologies. In *13th International Conference on WWW/Internet*. Porto, Portugal, pp. 123-130.

Delamaro, M. E., Maldonado, J. C., Jino, M., 2007. Introdução ao Teste de Software, *in Portuguese*, volume 394 of Campus. Elsevier, 2007.

DeMillo, R. A., Lipton, R. J., Sayward, F. G., 1978. Hints on Test Data Selection: Help for the Practicing Programmer. *Computer*. pp. 34-41.

Budd. T. A., 1981. Mutation Analysis: Ideas, Example, Problems and Prospects. In *Proceedings of the Summer School on Computer Program Testing held at SOGESTA*, Urbino, Italy, pp. 129-148.

Horrocks, I., Sattler, U., Tobies, S., 2000. Reasoning with Individuals for the Description Logic SHIQ. In *Proceedings of the 17th International Conference on Automated Deduction*. Pittsburgh, USA, pp. 482-496.

Musen, M. A., 2015. The Protégé project: A look back and a look forward. *AI Matters. Association of Computing Machinery Specific Interest Group in Art. Intelligence*.

Bechhofer, S., Horrocks, I., Patel-Schneider, P., 2003. Tutorial on OWL. In *2nd International Semantic Web Conference*. Sanibel Island, Florida, USA. Available in: <http://owl.man.ac.uk/2006/07/sssw/people.owl>. Accessed: November 10th, 2016.

Horrocks, I., 2008. Ontologies and the semantic web. In *Magazine Communications of the ACM – Surviving the data deluge*. Vol. 51, Issue 12, pp. 58-67.

McGuinness, D. L., Harmelen, F. van, 2004. OWL Web Ontology Language Overview. Available in: <http://www.w3.org/TR/owl-features/>. Accessed: November 11th, 2016.

OWL Working Group, 2012. Web Ontology Language (OWL). Available in: <https://www.w3.org/2001/sw/wiki/OWL>. Accessed: November 11th, 2016.

Budd, T. A., DeMillo, R. A., Lipton, R. J., Sayward, F. G., 1980. Theoretical and Empirical Studies on Using Programa Mutation to Test the Functional Correctness of Programs. In *Proc. of the 7th ACM Symposium of Principles of Programming Languages*. pp. 220-233, New York, NY, USA.

Gómez-Pérez, A., 2004. Ontology Evaluation. *Handbook on Ontologies, Springer*, pp. 251-273.

Burton-Jones, A., Storey, V. C., Sugumaran, V., Ahluwalia, P., 2005. A semiotic metrics suite for assessing the quality of ontologies, Data & Knowledge Engineering, Vol. 55, No. 1, pp. 84-102.

Hlomani, H., Stacey, D., 2014. An extension to the data-driven ontology evaluation. In *15th Int. Conf. on Info. Reuse and Integration*. Redwood, USA, pp. 845-849.

Wong, W. E., Mathur, A. P., Maldonado, J. C., 1995. Mutation Versus All-uses: An Empirical Evaluation of Cost, Strength and Effectiveness. In *Software Quality and Productivity: Theory, Practice and Training*. London, UK, pp. 258-265.