# Ontologies in software testing: A Systematic Literature Review

**Article** · January 2013

CITATIONS
4

READS
177

**3 authors**, including:

Erica Ferreira de Souza
Federal Technological University of Paraná, Brazil
**22** PUBLICATIONS **87** CITATIONS
SEE PROFILE

Ricardo de Almeida Falbo
Universidade Federal do Espírito Santo
**172** PUBLICATIONS **1,661** CITATIONS
SEE PROFILE

# Ontologies in Software Testing: A Systematic Literature Review

**Érica F. Souza[1], Ricardo A. Falbo[2], N. L. Vijaykumar[1]**

[1]Applied Computing – National Institute for Space Research (INPE)
São José dos Campos – São Paulo – SP – Brazil

[2]Department of Computer Science – Federal University of Espírito Santo (UFES)
Vitória – Espírito Santo – ES - Brazil

{erica.souza, vijay}@lac.inpe.br, falbo@inf.ufes.br

***Abstract.*** *Ontologies have been widely recognized as an important instrument for supporting Knowledge Management (KM). In order to look for a domain ontology that can be used in KM in software testing, in this paper, we investigate, by means of a Systematic Literature Review (SLR), ontologies in the software testing domain, including questions related to their coverage of the software testing domain, and how they were developed.*

## 1. Introduction

Verification and Validation (V&V) activities intend to ensure that a software product is being built in conformance with its specification, and that it satisfies its intended use and the user needs [IEEE 2004]. V&V activities can be static or dynamic. Static V&V activities are typically done by means of technical reviews and inspections, and they do not require code execution; dynamic V&V activities involve code execution, and are done by means of testing [Mathur 2012]. Thus, Software Testing consists of dynamic V&V of the behavior of a program on a finite set of test cases, against the expected behavior [SWEBOK 2004].

Software testing processes generate a large volume of information. Thus, it is important to provide computerized support for tasks of acquiring, processing, analyzing and disseminating testing knowledge for reuse [Andrade et al. 2013]. In this context, testing knowledge should be captured and represented in an affordable and manageable way, and therefore, principles of Knowledge Management (KM) can be applied. Ontologies can be used for establishing a common conceptualization to be used in the KM system in order to facilitate communication, integration, search, storage and representation of knowledge [O´Leary 1998]. However, the software testing domain is very complex and building an ontology for it is not a trivial task. One of the main problems in the software testing literature is that there is not uniformity in the vocabulary used. In several cases, authors create and recreate concepts, using different terms. When analyzing different references, it becomes apparent that the terminology used is diverse.

Looking for a domain ontology that can be used in a KM initiative in software testing, in this paper, we investigate, by means of a Systematic Literature Review (SLR) [Kitchenham and Charters 2010], ontologies in the software testing domain. In this

SLR, we consider the following main research questions: (i) What is the coverage of the software testing domain in the existing testing ontologies? (ii) How were these ontologies developed?

This paper is organized as follows. Section 2 discusses briefly the background for this paper. Section 3 presents the SLR we performed. Section 4 discusses important findings identified during data analysis. Finally, Section 5 presents our conclusions.

## 2. Background

The testing process consists of several activities, namely: Test Planning, Test Case Design, Test Execution and Test Result Analysis. Like several other aspects of a project, testing must be planned. Test planning should be documented in a Test Plan. Test Case Design aims at designing the test cases to be run. Test Cases should be documented, and implemented as Test Scripts. During test execution, test cases are run, producing actual results, which should also be documented. Finally, in the Test Result Analysis phase, test results are evaluated to determine whether or not tests have been successful. Testing techniques and test criteria are used to support designing test cases. Moreover, testing usually is performed at different levels. Three important test levels can be distinguished, namely: Unit Testing, Integration Testing and System Testing [SWEBOK 2004, Mathur 2012].

During the testing process, a significant volume of information is generated. Such information may turn into useful knowledge to potentially benefit future projects from experiences gained from past projects [Andrade et al. 2013]. However, converting this information into applicable knowledge is not an easy task. There is a need to properly represent and process the knowledge so that it can be manageable. In this context, principles of Knowledge Management (KM) can be applied. Ontologies are a key technology for KM. They provide a shared and common understanding of a domain that can be communicated between people and application systems. Their use offers an opportunity for improving KM capabilities in large organizations [Davies et al. 2003]. In ontology-based KM systems, ontologies are mainly used for the following three general purposes [Abecker and Elst 2009]: (i) to support knowledge search, retrieval, and personalization; (ii) to serve as basis for knowledge gathering, integration, and organization; and (iii) to support knowledge visualization.

In order to find a domain ontology to be used as basis for a KM initiative in software testing, we conducted a Systematic Literature Review (SLR) aiming at inspecting the existing software testing ontologies. The research method applied was defined based on the guidelines for SLRs given by Kitchenham and Charters (2010). According to them, a SLR is a form of secondary study that uses a well-defined method to identify, analyze and interpret the available evidences in a way that is unbiased and (to a degree) repeatable. A secondary study is a study that reviews primary studies related to specific research questions with the aim of integrating/synthesizing the evidences related to these questions. A SLR involves three main phases: (i) Planning: refers to the pre-review activities, and aims at establishing a review protocol defining the research questions, inclusion and exclusion criteria, sources of studies, search string and mapping procedures; (ii) Conducting: regards searching and selecting the studies, in

order to extract and synthesize data from them; and (iii) Reporting: is the final phase and aims at writing up the results and circulating them to potentially interested parties.

## 3. The Systematic Literature Review

This section presents the Systematic Literature Review (SLR) we perform to investigate existing ontologies in the software testing domain. In Subsection 3.1, we present the main parts of the review protocol. In Subsection 3.2, we briefly describe the selected studies. Finally, in Subsection 3.3, we synthesize data extracted from the studies.

### 3.1. Review Protocol

*Research Questions:* This SLR aims at answering the following research questions:

**RQ1.** What is the coverage of the software testing domain in the existing ontologies about this domain?

**RQ2.** How were they developed?

*Inclusion and Exclusion Criteria:* The selection criteria are organized in one inclusion criterion (IC) and five exclusion criteria (EC). The inclusion criterion is: (IC1) The study presents an ontology about the software testing domain. The exclusion criteria are: (EC1) The study does not have an abstract; (EC2) The study is just published as an abstract; (EC3) The study is not written in English; (EC4) The study is an older version (outdated) of another study already considered; and (EC5) The study is not a primary study, such as editorials, summaries of keynotes, workshops, and tutorials.

*Sources:* Search was done in eight electronic databases that were considered the most relevant according to [Dyba et al. 2007], namely: IEEE Xplore, ACM Digital Library, SpringerLink, Scopus, ISI of Knowledge, DBLP Computer Science Bibliography, Science Direct, and Compendex.

*Search String:* The search string is the following: *("Software Testing" OR "Software Test") AND ("Ontology" OR "Ontologies")*. It was applied in three metadata fields (title, abstract and keywords). The search went through syntactic adaptations according to particularities of each source.

*Assessment*: Before conducting the SLR, we tested the review protocol. This test was conducted in order to verify its feasibility and adequacy, based on a pre-selected set of studies considered relevant to our investigation. The review process was conducted by one of the authors and the other two carried out its validation. They analyzed approximately 35% of the studies using two different samples.

### 3.2. Selected Studies

Using the search string, 396 records were retrieved. The selection process applied on the returned publications was performed in three stages. In the first stage, duplicates were eliminated by examining title and abstract, since several publications are available in more than one source. In the second stage, inclusion and exclusion criteria were applied considering also title and abstract. Finally, in the third stage, the exclusion criteria were applied considering the entire text. After applying the selection criteria, 18 studies remained. Table 1 shows the progressive reduction of the number of studies throughout the selection process for the review.

**Table 1. Result of the Selection Process Stages of the SLR**

| Stage | Criteria | Analyzed Content | Initial Studies | Final Studies | Reduction (%) |
|---|---|---|---|---|---|
| 1st | Eliminating duplication | Title and abstract | 396 | 295 | 25.5% |
| 2nd | IC1, EC1, EC2, EC3, EC4 e EC5 | Title and abstract | 295 | 30 | 89.8% |
| 3rd | IC1, EC4, EC5 e EC6 | Entire Text | 30 | 18 | 40% |

From the 18 studies, 12 different ontologies were identified. This difference comes from the fact that some papers present different parts or evolutions of the same ontology. As the result of this SLR, we ended up in the following testing ontologies: STOWS (Software Testing Ontology for Web Service) [Huo et al. 2003, Zhu and Huo 2005, Hong 2006, Yufeng and Hong 2008, Zhu and Zhang 2012], OntoTest [Barbosa et al. 2006, Nakagawa et al. 2009], TaaS Ontology [Yu et al. 2008, Yu et al. 2009], and the ontologies proposed in [Li and Zhang 2012], [Arnicans et al. 2013], [Guo et al. 2011], [Nasser et al. 2009], [Bai et al. 2008], [Ryu et al. 2011], [Sapna and Mohanty 2011], [Cai et al. 2009] and [Anandaraj et al. 2011]. From the 12 identified ontologies, we analyzed whether there were extensions, evolutions and/or other publications that present the ontologies more completely. It was the case of OntoTest. OntoTest has a testing resource sub-ontology presented in [Barbosa et al. 2008]. However, this study did not return in the SRL, probably because the searched sources do not contain this paper or because they failed to identify it by the search string.

### 3.3. Data Synthesis

After selecting the primary studies, we analyzed each one in order to answer the research questions presented in Subsection 3.1. Next, we present the data synthesis regarding these questions.

**RQ1**. *What is the coverage of the software testing domain in the existing ontologies about this domain?*

Regarding domain coverage, we notice that most of the ontologies have very limited coverage. The ontology presented in [Guo et al. 2011] specifies only the concept of test case. The one in [Li and Zhang 2012] focuses also on test case, but considering some concepts related to test process. Bai et al. (2008) presented an ontology, called Test Ontology Model (TOM), to model only testing artifacts and relationships between them. The ontologies presented in [Arnicans et al 2013], [Cai et al. 2009] and [Anandaraj et al. 2011] are, in fact, taxonomies. These ontologies only present a simple structure of the domain concepts of software testing, and thus they do not qualify as ontologies, or, at most, they are lightweight ontologies.

The ontology presented in [Nasser et al. 2009] is devoted to state machine based testing. The ontology presented in [Sapna and Mohanty 2011] focuses on scenario-based testing, though it captures general testing concepts too. The one presented in [Ryu et al. 2011] is not properly a testing ontology, but it is an OWL implementation of a specific testing maturity model developed by the authors (the Ministry of National Defense-Testing Maturity Model (MND-TMM)).

The ontologies that have higher coverage are: STOWS (Software Testing Ontology for Web Service) [Huo et al. 2003, Zhu and Huo 2005, Hong 2006, Yufeng and Hong 2008, Zhu and Zhang 2012], OntoTest [Barbosa et al. 2006, Barbosa et al. 2008, Nakagawa et al. 2009], TaaS Ontology [Yu et al. 2008, Yu et al. 2009].

STOWS classifies its concepts into three categories: (i) elementary concepts, which are general concepts about computer software and hardware; (ii) basic testing concepts, which include the concepts of Tester, Artifact, Activity, Context, Method, and Environment; and (iii) compound testing concepts, which combine basic testing concepts, giving rise to the concepts of Task and Capability. STOWS presents a set of taxonomies of each basic testing concept, including also some properties and few relations.

The TaaS Ontology has two core concepts (Test Task and Test Capability), which are composite concepts aggregating other concepts. Test Task consists of Test Activity, Test Type, Target Under Test, Test Environment, and Test Schedule. Test Capability, in turn, consists of Test Type, Test Activity, Test Environment, Target Under Test and Quality of Service.

Finally, OntoTest is a modular ontology, built in layers. OntoTest is composed of a "Main Software Testing Ontology", and six sub-ontologies [Barbosa et al. 2006]: Testing Process, Testing Phase, Testing Artifact, Testing Step, Testing Resource, and Testing Procedure sub-ontologies. The Main Software Testing Ontology is presented in [Barbosa et al. 2006]. It is a simple model that includes six concepts. According to this model, a Testing Process is composed of Testing Steps, and it has Testing Phases. A Testing Step requires Testing Resources, adopts Testing Procedures, consumes and generates Testing Artifacts, and depends on other Testing Steps. Testing Artifacts can depend on other Testing Artifacts, and can be composed of other Testing Artifacts. Finally, a Testing Procedure can be supported by Testing Resources, and is adequate to Testing Process. OntoTest Testing Step sub-ontology introduces the concept of Testing Activity, indicating that a Testing Step is composed of Testing Activities, while Testing Activities are not further decomposed. The remainder of this sub-ontology consists of two large taxonomies: a Testing Step taxonomy, and a Testing Activity taxonomy. The Testing Resource sub-ontology [Barbosa et al. 2008] has a taxonomy of types of resources. This taxonomy is organized in two branches: Human Resources (which can be members of Test Teams), and Testing Environment, which is further extended in Software and Hardware Resource. Software Resource is further extended into Testing Tool and Supporting System. Testing Tool can be composed of several types of Testing Modules. We did not find papers presenting the Testing Process, Testing Phase, Testing Artifact, and Testing Procedure sub-ontologies. So, we suppose that OntoTest is a work in progress.

**RQ2.** *How were the testing ontologies developed?*

With respect to this research question, we focused on some aspects related to the way the ontologies were engineered, namely: (i) Do the ontologies try to capture a common (shared) conceptualization of the testing domain, taking into account different references and especially international standards? (ii) Are the ontologies developed following an ontology engineering method (including some sort of evaluation)? (iii) In which abstraction level (conceptual and implementation levels) are the ontologies

developed? Which are the languages used? (iv) Do the ontologies take foundational aspects (foundational ontologies) into account?

The first aspect investigated is if the ontologies try to capture a common conceptualization of the testing domain. Some ontologies take international standards into account: OntoTest is based on 1st edition of ISO/IEC 12207; the ontology presented in [Arnicans et al. 2013] was created based on the glossary "Standard glossary of terms used in Software Testing" of the International Software Testing Qualifications Board – ISTQB; the ontology presented in [Bai et al. 2008] is based on the Unified Modeling Language 2.0 Test Profile (U2TP); and the ontologies presented in [Cai et al. 2009] and [Sapna and Mohanty 2011] are based on the SWEBOK [SWEBOK 2004]. The other studies neither mention the use of international standards as basis for their ontologies, nor which references were used as basis for developing the ontologies. The exception is the ontology presented in [Ryu et al. 2011], which, as said before, is an OWL implementation of the Ministry of National Defense-Testing Maturity Model (MND-TMM). It is worthwhile to point out that, despite some ontologies are based on international standards, generally they take only one standard into account, and thus they do not consider a broad set of testing references to really establish a common (consensual) conceptualization.

Regarding the methods adopted for building the ontologies, Arnicans et al. (2013) propose a method for semi-automatic obtaining lightweight ontologies, which uses the ONTO6 method. In [Sapna and Mohanty 2011], ideas were adapted from two methods for building ontologies: METHONTOLOGY [Juristo et al. 2007] and Ontology Development 101 [Noy and McGuinness 2001]. Cai et al. (2009) used the Uschold and King's skeletal method [Uschold and King 1995] for building their testing ontology. Finally, OntoTest was built using a method that combines guidelines given by SABiO [Falbo et al. 1998] and METHONTOLOGY, with focus on ontology capture and formalization. Finally, Anandaraj et al. (2011) followed a very simple method, comprising four steps, namely: (i) determine domain and scope of the ontology; (ii) define concepts in the ontology; (iii) create a class hierarchy; and (iv) define properties and constraints. The other studies do not mention if a method (or which method) was used for building the proposed ontologies.

Although the aforementioned ontologies have been developed following methods that include activities devoted to ontology evaluation, such as Uschold and King's skeletal method, SABiO and METHONTOLOGY, none of the studies discusses how the ontologies were evaluated, except [Arnicans et al. 2013], which says that a software testing expert has analyzed the ontology fragment related to testing techniques.

Regarding the abstraction level, 7 of the 12 studies (58.3%) present their ontologies as conceptual models, namely: STOWS, OntoTest, TaaS Ontology and the ontologies presented in [Li and Zhang 2012], [Arnicans et al. 2013], [Bai et al. 2008] and [Sapna and Mohanty 2011]. 5 of the 12 studies (41.7%) present the ontologies only as a code artifact (implemented in OWL), namely: the ontologies presented in [Guo et al. 2011], [Nasser et al. 2009], [Ryu et al. 2011], [Cai et al. 2009] and [Anandaraj et al. 2011]. The following ontologies are represented in both conceptual and implementation levels: STOWS, OntoTest, and the ontologies presented in [Arnicans et al. 2013], [Bai et al. 2008] and [Sapna and Mohanty 2011]. It is important to clarify the approach

followed in [Arnicans et al. 2013]. In this study, first the ontology is semi-automatically generated in OWL. The obtained ontology is then transformed to UML class diagram using a tool called OWLGrEd in order to be evaluated by experts.

Concerning the languages used for representing the ontologies, all the studies that present the ontologies in the implementation level used OWL. In the conceptual level, all the ontologies are presented as UML class diagrams. Moreover, two ontologies use first order logics to capture some axioms, namely OntoTest and the ontology presented in [Li and Zhang 2012].

Summarizing, from the 12 ontologies investigated, 2 are represented only as conceptual models presented as UML class diagrams (TaaS Ontology, and the ontology presented in [Li and Zhang 2012]), 5 are represented only as OWL implementations (the ontologies presented in [Guo et al. 2011], [Nasser et al. 2009], [Ryu et al. 2011], [Cai et al. 2009] and [Anandaraj et al. 2011]), and 5 are represented both in the conceptual level (as UML class diagrams) and in the implementation level (as OWL artifacts) (STOWS, OntoTest, and the ontologies presented in [Arnicans et al. 2013], [Bai et al. 2008], and [Sapna and Mohanty 2011]).

Finally, although foundational ontologies have been recognized as an important instrument for improving the quality of conceptual models in general, and more specifically of domain ontologies [Guizzardi 2007], none of the ontologies analyzed in our SLR reuses foundational ontologies.

## 4. Discussion

In this section, we discuss some relevant points that have arisen from the data syntheses done in the SLR and discuss limitations of them.

Currently, software testing is considered a complex process comprising activities, techniques, artifacts, and different types of resources (hardware, software and human resources). Thus, building a complete testing ontology is not a trivial task (if even possible). Although there are a relatively large number of ontologies on software testing published in the literature (at least 12 ontologies), we notice that there are still problems related to the establishment of an explicit common conceptualization regarding this domain. For being applied to KM, a software testing ontology must take some characteristics of good quality ontologies into account.

In an experiment trying mainly to identify good practices in ontology design, D'Aquin and Gangemi [D'Aquin and Gangemi 2011] have identified some characteristics that are presented in what they call "beautiful ontologies". These characteristics were grouped in three dimensions: (i) formal structure, (ii) conceptual coverage and task, and (iii) pragmatic or social sustainability. In order to evaluate the testing ontologies selected by means of the 2nd SLR, we focus on the first dimension, and in part of the second one, namely conceptual coverage. The characteristics included in these dimensions are [D'Aquin and Gangemi 2011]:

> **Structure:** the ontology reuses foundational ontologies; the ontology is designed in a principled way; it is formally rigorous; it also implements non-taxonomic relations; the ontology strictly follows an evaluation method; it is modular, or embedded in a modular framework.

**Conceptual coverage:** the ontology provides important reusable distinctions; it has a good domain coverage; it implements an international standard; the ontology provides an organization to unstructured or poorly structured domains.

Unfortunately, some of these characteristics are difficult to evaluate, since there isn't much information about them in the papers presenting the corresponding ontologies. Thus, in our analysis, we focused on the most easily discernible features, namely: having a good domain coverage; implementing an international standard; being formally rigorous; implementing also non-taxonomic relations; following an evaluation method; and reusing foundational ontologies.

Regarding the first characteristic (having a good domain coverage), we notice that most ontologies have very limited coverage (see Section 3.3). Those that have higher coverage are: STOWS, OntoTest, and TaaS. Some take international standards into account, namely: OntoTest, and the ontologies presented in [Arnicans et al. 2013], [Bai et al. 2008], [Sapna and Mohanty 2011], and [Cai et al. 2009]. Others, on the other hand, do not consider international standards (or at least do not mention them). This is the case of STOWS and TaaS Ontology.

The next two characteristics (being formally rigorous and also implementing non-taxonomic relations) are very important for a reference ontology. As discussed previously, a reference ontology must be a heavyweight ontology, and thus it must comprise conceptual models that include concepts, and relations (of several natures), and also axioms describing constraints and allowing to derive information from the domain models. Taking this perspective into account, we can notice that most of the existing ontologies present problems.

There are five ontologies ([Guo et al. 2011], [Nasser et al. 2009], [Ryu et al. 2011], [Sapna and Mohanty 2011] and [Anandaraj at al. 2011]) that are just OWL artifacts (i.e., operational ontologies), and thus are not enough for the purposes of applying ontologies for KM.

The ontologies presented in [Arnicans et al. 2013] and [Cai et al. 2009] are, in fact, taxonomies, and thus, in our view, they do not qualify as ontologies (or at most, they are lightweight ontologies). STOWS is mainly a set of taxonomies of basic concepts, including some properties and few relations. There are taxonomies of Tester, Context, Testing Activities, Testing Methods, and Testing Artifacts, but there are important relations missing. For instance, which are the artifacts produced and required by a testing activity? Without relations between the concepts, questions such as this one cannot be answered. Moreover, there are two "compound concepts" in STOWS that are defined on the bases of the basic concepts: capability and task. *Capability*, for instance, is modeled as a composite entity, which parts are *Activity*, *Method*, an optionally *Environment*, *Context*, and *Data* (a subtype of *Artifact*). This model is questionable, since it puts together objects and events as part of *Capability*. Objects (or endurants) exist in time; while events (or perdurants) happen in time [Guizzardi 2008]. So what is a Capability? An object or an event? This shows that this ontology presents problems.

TaaS Ontology presents very simple models. UML class diagrams presented in [Yu et al. 2008] and [Yu et al. 2009] do not specify multiplicities of relationships. Moreover, like STOWS, most of the relationships are modeled as aggregations (whole-

part relations in UML). This approach is very questionable from an ontological point of view. For instance, there is a core concept called *Test Task*, which is modeled as composed of *TestActivity*, *TestType*, *TargetUnderTest*, *TestEnvironment*, and *TestSchedule*. Analogously to the analysis on STOWS, the composite object *Test Task* aggregates endurants and perdurants.

Even the most complete ontology among the ones we achieved through the SLR, OntoTest, also presents problems. First, there are sub-ontologies that were not published yet, namely the Testing Process, Testing Phase, Testing Artifact, and Testing Procedure sub-ontologies. Second, OntoTest does not properly link the concepts in the sub-ontologies. For instance, albeit in the Main Software Testing Ontology there is a relationship between Testing Step and Test Resource, there aren't relationships between their subtypes. This is an important part of the software testing conceptualization that needs to be made explicit.

Regarding ontology evaluation, none of the works we investigated in the SLR discusses how the ontologies they propose were evaluated, except the one done by Arnicans et al. (2013), which says that a software testing expert has analyzed the ontology fragment related to testing techniques.

Finally, concerning the reuse of foundational ontologies, none of the ontologies analyzed in our SLR have used one. In our view, this is a problem, because important distinctions made in Formal Ontologies may be disregarded as clearly noticed in the brief analysis we did (as in the aforementioned cases of STOWS and TaaS Ontology). The lack of truly ontological foundations puts in check the truthfulness of those ontologies.

Thus, we concluded that the software testing community has still a lot work to do, in order to advance towards a reference software testing ontology. Once developed a good quality reference testing ontology, an operational version of it should be designed and implemented. With these two artifacts in hand, we can effectively take a step forward in ontology-based KM applied to the software testing domain.

**Limitations of the SRL**

The SLR presented in this paper has some limitations. Due to the fact that the study selection and data extraction steps were performed by just one of the authors, some subjectivity may have been inserted. To reduce this subjectivity, the other two authors performed these same steps in a random sample (including about 35% of the studies). The results of each reviewer were then compared in order to detect possible bias. Moreover, terminological problems in the search strings may have led to missing some primary studies. In order to minimize these problems, we performed previous simulations in the selected databases. We decide not to search any specific conference proceedings, journals, or the grey literature (technical reports and works in progress). Thus, we have just worked with studies indexed by the selected electronic databases. The exclusion of these other sources makes the review more repeatable, but possibly some valuable studies may have been left out of our analysis.

## 5. Conclusions

In this paper, we presented a Systematic Literature Reviews (SLR) in order to investigate ontologies in the software testing domain, including questions related to their coverage of the software testing domain, and how they were developed. We identified 12 ontologies addressing the software testing domain. For analyzing these ontologies, we considered some of the characteristics pointed by D'Aquin and Gangemi (2011) as characteristics that are presented in "beautiful ontologies". In our analysis, we considered the following characteristics: having a good domain coverage; implementing an international standard; being formally rigorous; implementing also non-taxonomic relations; following an evaluation method; and reusing foundational ontologies.

As the main findings obtained from this SLR, we highlight the following conclusions: most ontologies have limited coverage; the studies do not discuss how the ontologies were evaluated; none of the analyzed testing ontologies is truly a reference ontology; and none of them is grounded in a foundational ontology. In sum, we conclude that the software testing community should invest more efforts to get a well-established reference software testing ontology.

As a future work, we intend to develop a KM system for managing testing-related knowledge items. This KM system will be built based on a Reference Ontology on Software Testing (ROoST) that we are now developing [Souza 2013].

**Acknowledgments -** The first author acknowledges FAPESP (Process: 2010/20557-1) for the financial grant. The second author acknowledges FAPES/CNPq (PRONEX Grant 52272362/11) for the financial grant.

## References

Abecker, A., van Elst, L. (2009). Ontologies for Knowledge Management, In: Handbook of Ontologies, Staab, S., Studer, R. (Eds.), Springer, 2nd edition.

Andrade, J., Ares, J., Martínez, M., Pazos, J., Rodríguez, S., Romera, J., Suárez, S. (2013). An architectural model for software testing lesson learned systems, Information and Software Technology. vol. 55, 18-34.

Anandaraj, A., Kalaivani, P., Rameshkumar, V. (2011). Development of Ontology-Based Intelligent System for Software Testing. In. International Journal of Communication, Computation and Innovation, v. 2.

Arnicans, G., Romans, D., Straujums, U. (2013). Semi-automatic Generation of a Software Testing Lightweight Ontology from a Glossary Based on the ONTO6 Methodology," In Frontiers in Artificial Intelligence and Applications, 263-276.

Bai, X., Lee, S., Tsai, W., Chen, Y. (2008). Ontology-Based Test Modeling and Partition Testing of Web Services. International Conf. on Web Services, 465-472.

Barbosa, E. F., Nakagawa, E. Y., Maldonado, J. C. (2006). Towards the establishment of an ontology of software testing. In. International Conference on Software Engineering and Knowledge Engineering (SEKE), v.1, 522-525, San Francisco, CA.

Barbosa, E. F., Nakagawa, E. Y., Riekstin, A. C., Maldonado, J. C. (2008). Ontology-based Development of Testing Related Tools. In International Conference on Software Engineering & Knowledge Engineering (SEKE), San Francisco, CA.

Cai, L., Tong, W., Liu, Z., Zhang, J. (2009). Test Case Reuse Based on Ontology. Pacific Rim International Symposium on Dependable Computing, 103-108.

Davies, J., Fensel, D., Van Harlemen, F. (2003). Towards the Semantic Web: Ontology-driven Knowledge Management, John Wiley & Sons.

Dyba T., Dingsoyr, T., Hanssen, G. (2007). Applying systematic reviews to diverse study types: An experience report. First International Symposium on Empirical Software Engineering and Measurement, Madrid, 225-234.

D'Aquin, M. and Gangemi, A. (2011). Is there beauty in ontologies? Applied Ontology. vol. 6, n.3, 165-175.

Falbo, R. A., Menezes, C. S. and Rocha, A. R. (1998). A systematic approach for building ontologies. In VI Ibero-American Conference on AI (IBERAMIA98), Lisboa, Portugal.

Guizzardi, G. (2007). On Ontology, ontologies, Conceptualizations, Modeling Languages, and (Meta)Models. In: Frontiers in Artificial Intelligence and Applications, Databases and Information Systems IV, 18–39, Amsterdã.

Guizzardi, G., Falbo, R.A., Guizzardi R.S.S. (2008). Grounding software domain ontologies in the Unified Foundational Ontology (UFO): the case of the ODE software process ontology. In XI Iberoamerican Workshop on Requirements Engineering and Software Environments, 244-251.

Guo, S., Zhang, J., Tong, W., Liu, Z. (2011). An Application of Ontology to Test Case Reuse. In. International Conference on Mechatronic Science, Electric Engineering and Computer, 19-22, Jilin, China.

Hong, Z. (2006). A Framework for Service-Oriented Testing of Web Services. Computer Software and Applications Conference, 2006. COMPSAC '06. 30th Annual International, Chicago, IL, 145 - 150.

Huo, Q., Zhu, H., Greenwood, S. (2003). A Multi-Agent Software Environment for Testing Web-based Applications. In 27th International Computer Software and Applications Conference (COMPSAC2003), Dallas, TX, USA, 210-215.

IEEE Std 1012 (2004). IEEE Standard for Software Verification and Validation. New York, NY, USA.

IEEE Computer Society, SWEBOK. (2004). A Guide to the Software Engineering Body of Knowledge. <http://www.computer.org/portal/web/swebok>

Juristo, N., Ferndandez, M., Gomez-Perez, A. (1997). METHONTOLOGY: From Ontological Art Towards Ontological Engineering. In Proceedings of the AAAI97 Spring Symposium. Technical Report SS-97-06, 15(2).

Kitchenham, and Charters, B. S. (2007). EBSE Technical Report, Software Engineering Group, School of Computer Science and Mathematics Keele University and Departament of Computer Science University of Durham, UK, v. 2.3.

Li, X. and Zhang, W. (2012). Ontology-based Testing Platform for Reusing. In. Internet Computing for Science and Engineering (ICICSE 2012), 86 – 89.

Mathur, A. P. (2012). Foundations of Software Testing. 5th ed. Delhi, India: Dorling Kindersley (India), Pearson Education in South Asia.

Nakagawa, E.Y., Barbosa, E.F., Maldonado, J.C. (2009). Exploring ontologies to support the establishment of reference architectures: An example on software testing. Software Architecture. WICSA/ECSA 2009. Joint Working IEEE/IFIP Conference on, Cambridge, 249 - 252.

Nasser, V. H., Du, W., MacIsaac, D. (2009). Knowledge-based software test generation. In. International Conference on Software Engineering and Knowledge Engineering, SEKE'2009, 312 - 317.

Noy, N.F., McGuinness, D.L. (2001). Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI.

O´Leary, D.E. (1998). Enterprise Knowledge Management, Computer, Univ. of Southern California, Los Angeles, CA, v. 31, Issue 3, 54 – 6.

Ryu, H., Ryu, D., Baik, J. (2011). A Strategic Test Process Improvement Approach Using an Ontological Description for MND-TMM. In. International Conference on Computer and Information Science, 561-566.

Sapna, P. G. and Mohanty, H. (2011). An Ontology Based Approach for Test Scenario Management. ICISTM'2011, v. 141, 91–100.

Souza, E. F., Falbo, R. A., Vijaykumar, N. L. (2013). Using Ontology Patterns for Building a Reference Sofware Testing Ontology. In: The 8th International Workshop on Vocabularies, Ontologies and Rules for the Enterprise and Beyond (VORTE2013). The 17th IEEE International EDOC Conference (EDOC2013), Vancouver, BC.

Uschold, M. and King, M. (1995). Towards a Methodology for Building Ontologies.. Presented at the Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI95, AIAI-TR-183, University of Edinburgh, Edinburgh.

Yufeng, Z. and Hong, Z. (2008). Ontology for Service Oriented Testing of Web Services. Symposium on Service-Oriented System Engineering, Jhongli, 129 - 134.

Yu, L., Su, S., Zhao, J. (2008). Performing Unit Testing Based on Testing as a Service (TaaS) Approach. In. International Conference on Service Science, 127-131.

Yu, L., Zhang, L., Xiang, H., Su, Y., Zhao, W., Zhu, J. (2009). A Framework of Testing as a Service. Management and Service Science, International Conf. on, 1- 4.

Zhu, H. and Huo, Q. (2005). Developing A Software Testing Ontology in UML for a Software Growth Environment of Web-Based Applications. Software Evolution with UML and XML, 263-295, IDEA Group.

Zhu, H. and Zhang, Y. (2012). Collaborative Testing of Web Services. In. IEEE Transactions on Service Computing, 116 - 130, v. 5.