The Journal of Systems and Software 162 (2020) 110485

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

A systematic literature review on semantic web enabled software testing

Mahboubeh Dadkhah, Saeed Araban*, Samad Paydar

Department of Computer Engineering, Faculty of Engineering, Ferdowsi University of Mashhad, Mashhad, Iran

ARTICLE INFO

Article history:
Received 2 November 2018
Revised 24 September 2019
Accepted 25 November 2019
Available online 26 November 2019

Keywords:
Software testing
Test generation
Semantic web
Ontology
Systematic literature review

ABSTRACT

Software testing, as a major verification and validation activity which revolves around quality tests, is a knowledge-intensive activity. Hence, it is reasonable to expect that it can be improved by effective application of semantic web technologies, e.g., ontologies, which have been frequently used in knowledge engineering activities.

The objective of this work is to investigate and provide a better understanding of how semantic web enabled techniques, i.e., the techniques that are based on the effective application of the semantic web technologies, have been used to support software testing activities. For this purpose, a Systematic Literature Review based on a predefined procedure is conducted. A total of 52 primary studies were identified as relevant, which have undergone a thorough meta-analysis with regards to our posed research questions.

This study indicates the benefits of semantic web enabled software testing in both industry and academia. It also identifies main software testing activities that can benefit from the semantic web enabled techniques. Furthermore, contributions of such techniques to the testing process are thoroughly examined. Finally, potentials and difficulties of applying these techniques to software testing, along with the promising research directions are discussed.

1. Introduction

Software development process consists of many complex and error-prone activities. One of the most important of these activities is Quality Assurance (QA). In order to ensure the quality of software products, performing Verification & Validation (V&V) activities is essential throughout the software development and maintenance process. The purpose of V&V activities is to ensure that software is built in conformance with its specification and that it satisfies its user's needs (Ammann and Offutt, 2016). Software testing is a major V&V activity that consists of dynamic V&V of the behavior of software on a finite set of test cases, against the expected behavior (Ammann and Offutt, 2016). Advances in technology and the emergence of increasingly complex and critical applications require test strategies, in order to achieve high quality and reliable software products. A wide range of testing techniques are proposed for software products. However, software testing is usually performed under tight resource and time constraints, and hence, researchers are continuously seeking to develop new approaches to address this issue.

Knowledge management principles and techniques have been applied in different phases of the software development process (Rus and Lindvall, 2002; Bjørnson and Dingsøyr, 2008; Vasanthapriyan et al., 2015). As a sub-area of software engineering, software testing is also a knowledge-intensive process, and it is essential to provide automation support for capturing, sharing, analyzing, retrieving, and representing testing knowledge (Andrade et al., 2013). In this context, testing knowledge should be captured and represented in an affordable and manageable way, and therefore, could make use of principles of knowledge management. The software testing community has recognized the need for managing knowledge and that it could learn much from the knowledge management community (de Souza et al., 2014). So, different aspects of software testing have been the subject of knowledge management initiatives (de Souza et al., 2014). Test generation using available system knowledge is an application of knowledge-based software testing. Knowledge of the application domain and the software testing domain, as well as a tester's personal knowledge, can be used to generate tests and to recognize failures (Itkonen et al., 2013).

* Corresponding author.
E-mail addresses: mah.dadkhah@mail.um.ac.ir (M. Dadkhah), araban@um.ac.ir (S. Araban), s-paydar@um.ac.ir (S. Paydar).

https://doi.org/10.1016/j.jss.2019.110485
0164-1212/© 2019 Elsevier Inc. All rights reserved.

With the emergence of semantic web technologies, new approaches have been proposed for integrating software and knowledge engineering. There is a range of studies that utilize knowledge management in software test generation activities, for instance, automated test generation, which benefits from semantic web technologies. One of the main challenges in knowledge based software testing approaches is how to provide a reasonably formal specification of the test process data so that it is possible to increase test automation (Gutiérrez et al., 2015). Semantic web data models and ontologies, due to their logic based nature, inference capability and machine understandability, are good candidates for providing this formalism and improving test automation. For example, an ontology can represent requirements from a software requirements specification, and the inference rules can describe strategies for deriving test cases from that ontology (Tarasov et al., 2016). The meta-model of a requirement in this ontology consists of requirement conditions, requirement parameters, results, and actions. Semantic web technologies are also supporting other software testing activities, e.g., test data generation, test oracle, and test reuse. For example, the Web of Data, a global dataset containing billions of interconnected and machine-processable statements represented in RDF triples, can be exploited for generating test data (Mariani et al., 2014).

Given the great importance of automatic software testing and the potential benefits of using semantic web technology to manage testing knowledge, this review aims to identify the state of the art on using semantic web technology in software testing. Using semantic web technologies, it is possible to separate knowledge related to an application domain from the business logic.

Hence, the objective of this study is to conduct a systematic review of the literature to find out how semantic web technologies support software testing process. It is also essential to investigate whether there is evidence of the improvements on exploiting semantic web technologies to support test generation activities. Moreover, we also need to investigate: which activities of the test generation process are amenable to the use of semantic web technologies; if semantic web technologies have been used to support testing both functional and non-functional requirements; what levels of testing and which application domains have been addressed by semantic web technologies; which languages, tools, and methodologies have been used in developing ontologies for software testing; and if these test ontologies have been reused and how they have been reused. In this paper, we use the systematic literature review (SLR) method (Kitchenham and Charters, 2007) to identify, evaluate, interpret, and analyze the available studies to answer particular research questions on the symbiosis of semantic web technologies and software testing and to establish the state of evidence with in-depth analysis.

2. Background

This section briefly presents the two main concepts related to our review, namely: software testing and the Semantic Web.

2.1. Software testing

In the context of software engineering domain, Verification and Validation (V&V) techniques are used to ensure the quality of software products. The purpose of V&V is to help the development of quality software systems. Software testing is a significant V&V activity that checks the behavior of a software system on a finite set of test cases against the expected behavior.

Increasingly complex and critical software systems have made software testing an extremely necessary activity (de Souza et al., 2014). Software testing is conducted through the software development and maintenance life cycle and should be supported by a well-defined and controlled testing process. Software testing process consists of several activities, typically including planning, test generation, test environment development, test execution, test result evaluation, test logs, and defect tracking (Bourque et al., 2014). There are also some key issues and practical considerations in the test process, such as the oracle problem, testability, and test reuse. Testing is usually performed at different levels. Low-level testing (e.g., unit or component testing) focus on testing each program unit or component in isolation from the rest of the system. Integration testing to ensure proper handling of interfaces among the components. High-level testing (e.g., system or acceptance testing) for validating the behavior of the entire system.

Software testing is a knowledge-intensive process, and thus Knowledge Management (KM) principles and techniques can support managing software testing knowledge. KM activities, such as capturing, processing, analyzing, sharing, and reusing knowledge is applicable to software testing. Therefore, provided knowledge, together with the observed actual behavior of the system under testm can be used to create better tests during exploratory testing (de Souza et al., 2014). For example, where test cases are not defined in advance, KM techniques can be used to dynamically design, execute tests, and analyze the results. KM systems are further discussed in 5.2.

Ontologies are considered as an enabling technology for knowledge management in software testing (de Souza et al., 2014). Some researchers in software testing have used ontologies mainly for knowledge representation. Tonjes et al. (2015) used upper ontologies (e.g., the Suggested Upper Merged Ontology (SUMO)(Niles and Pease, 2001)) to represent knowledge about the context of a parameter. Moreover, ontologies have strong formal and reasoning foundation that can support software testing (Gašević et al., 2009).

2.2. The semantic web

The term semantic web represents both semantic web technologies as a stack of technologies for data representation and processing, and the Semantic Web as a large repository of machine-processable datasets published based on those technologies.

The Semantic Web has a layered architecture where each layer exploits and uses capabilities of the layers below. The architecture of the Semantic Web is known as Semantic Web Stack, Semantic Web Cake or Semantic Web Layer Cake. The Semantic Web stack illustrates the hierarchy of technologies that are standardized for Semantic Web and how they are organized to make the Semantic Web possible. Some of these layers and especially middle layers contain technologies standardized by W3C [1] to enable building semantic web applications (i.e., RDF, RDFS, OWL, SPARQL, RIF).

Ontologies are the central part of the Semantic Web technologies and facilitate representation of the real-world domain knowledge. Gruber et al. (1993) defines an ontology as an explicit specification of a conceptualization, where a conceptualization illustrates an abstract, simplified picture of the world used for representation and designation. More precisely, an ontology is a data model that represents a set of concepts within a domain and their relationships. For example, Fig. 1 shows a sub-ontology in software testing domain. This sub-ontology is a part of the ROoST's ontology (Souza et al., 2017). It depicts the concepts and their relationships in the software testing techniques domain using a UML class diagram. In this sub-ontology, it is stated that there are different types of testing techniques: black-box, which-box, defect-based, and model-based. The ROoST (Souza et al., 2017) will be further discussed in Section 8.

[1] https://www.w3.org.

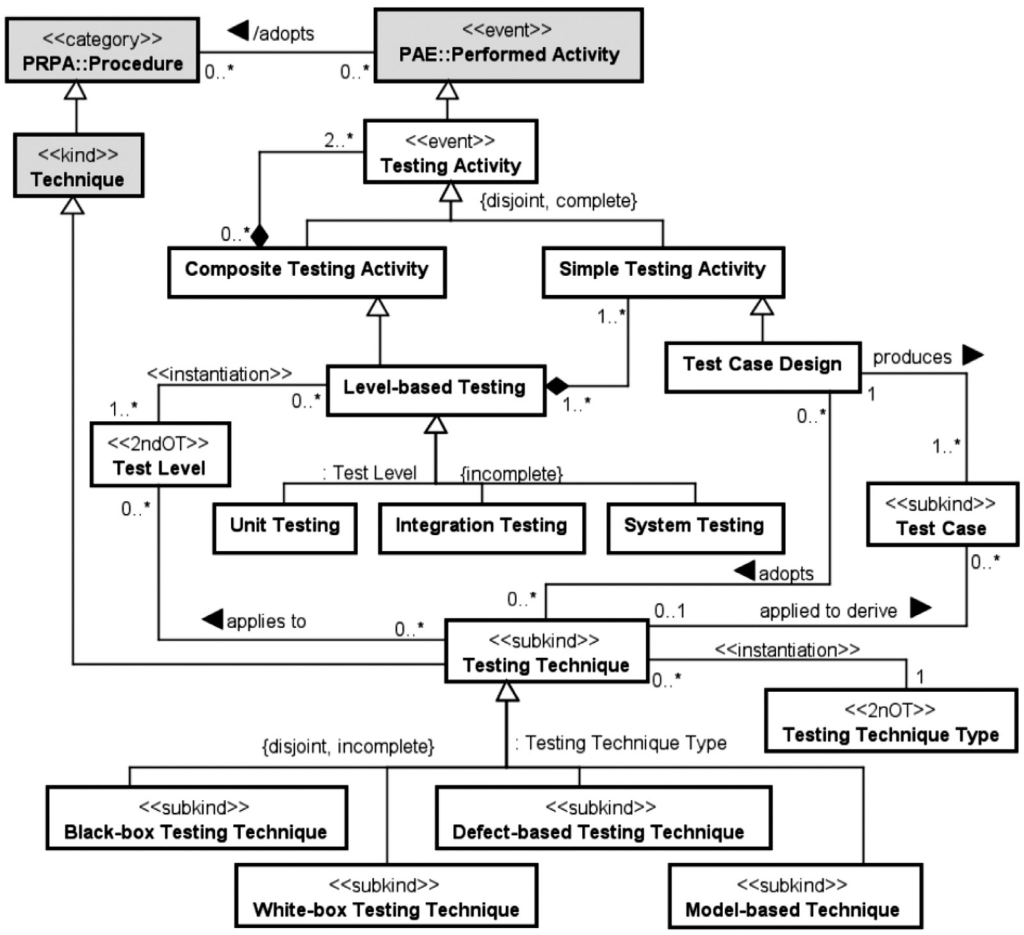

Fig. 1. ROoST's Testing Techniques sub-ontology (Souza et al., 2017).

Semantic reasoning is also expected to play an important role. Ontologies and semantic reasoners can provide a better representation format and improving analysis and processing of data. Reasoning can derive implicit statements by inference based on ontological knowledge and a set of statements.

The Semantic Web is an extension (not replacement) of classical hypertext web. It represents an effective means of data representation in the form of a machine-understandable web of data. The Semantic Web provides access to different sources of linked databases which publish data based on the Semantic Web data model (e.g., ontology repositories or Linked Data (Bizer et al., 2009) sources).

The Semantic Web and Semantic Web technologies offer a new approach to manage information and processes (Davies et al., 2006). Semantic Web technologies have been investigated in many disciplines where information reuse and integration promises significant added value, e.g., in the life sciences, in geographic information science, digital humanities research, for data infrastructures and web services, as well as in software engineering. The formalism, inference capability, and machine-understandability that are provided by the Semantic Web data model is frequently utilized in the software engineering domain for different goals. Gašević et al. (2009) defined a framework that identifies different phases of software lifecycle which ontologies can support and improve the software development process. They apply the framework to analyze the use of ontologies in different phases of the software life cycle. Software testing, as an important part of the V&V process is no exception. The potential applications of the semantic web technologies in software testing domain, along with their advantages and challenges are investigated in various publications (Wnuk and Garrepalli, 2018; Paydar and Kahani, 2010; Bhatia et al., 2016; Souza et al., 2013a; 2013b; Palacios et al., 2014). Semantic Web sources also can be used in software testing for different purposes, such as, generating test data inputs (Mariani et al., 2014). In this paper, we tried to investigate applications of both semantic web technologies and the Semantic Web data sources in software testing.

3. Research method

This systematic review was conducted following the procedure outlined by Kitchenham and Charters (2007). We followed the particula recommendations for Ph.D. students[2] where applicable.

We first define our research questions to make our goals more specific (see Section 3.1). The search terms are presented in Section 3.2. Then, we describe in Section 3.3 how we designed a strategy for searching relevant studies in the selected data sources. The specific steps taken during the study selection process are described in Section 3.4. The quality of the studies was assessed as suggested by the SLR methodology. The defined quality assessment criteria are described in Section 3.5. Finally, we report in Section 3.6 on the process of information extraction from the selected articles.

[2] One of the authors (Mahboubeh Dadkhah) is a Ph.D. student.

Table 1
Research questions and motivations.

Research question	Description and motivation
RQ1. What are the theoretical foundations of semantic web enabled software testing?	To investigate the studies that present new points of views on the use and the potential applications of semantic web technologies in software testing, but do not provide a concrete realization, e.g., in terms of a specific method or tool, of those potentials.
RQ2. What concrete approaches realized the semantic web enabled software testing? What are the supported test activities?	To investigate specifications of the proposed concrete approaches and to identify the potentials of using semantic web technologies in different software testing activities.
RQ3. Which semantic web technologies have been used in the software testing process?	To identify the most popular semantic web technologies in the software testing.
RQ4. What are the available ontologies in the software testing domain and how frequently are they reused?	To identify the existing test ontologies and those that have been frequently reused by researchers and hence can be considered to be of good quality.
RQ5. In what application domains has semantic web enabled testing been used?	To investigate the general and specific application domains in which the capabilities of semantic web technologies is utilized for software testing.
RQ6. How semantic web technologies have improved software testing?	To identify which quality attributes involved in the testing process have been improved by the use of semantic web enabled approaches.

Table 2
Search terms.

First round	Population	Software Testing, Automated Testing, Test Case Generation, Test Case Creation, Test Data, Test Planning.
	Intervention	Semantic, Semantic-based, Semantic Web, Semantic Annotation, Ontology, Ontology-based, OWL, RDF, XML.
Final round	Population	Software Testing, Automated Testing, Model-based Testing, Test Case Generation, Test Data Generation, Test Case Reuse, Traceability Matrix, Oracle.
	Intervention	Semantic Web, Semantic Model, Semantic Annotation, Semantic-based, Ontology.

3.1. Research questions

Defining research questions is an essential part of a systematic review as they drive the entire review methodology (Kitchenham and Charters, 2007). The purpose of this systematic review is to better understand how semantic web technologies support software testing and identify to what extent they have been applied to this field. To identify the existing semantic web enabled software testing approaches, research questions, their descriptions, and motivations are described in Table 1.

3.2. Search terms

We used a nine-step strategy to obtain our search terms:

1. Derive major terms from the research questions and the SWEBOK (Bourque et al., 2014).
2. Identify keywords in already known primary studies found by conducting an initial mapping study based on Petersen et al. (2015).
3. Identify alternative spellings, plurals, related terms, and synonyms for major terms using Microsoft Academic Search.[3]
4. When allowed by the database, use Boolean "OR" to incorporate alternative spellings and synonyms.
5. When database allows, use Boolean "AND" to link the major terms from population, intervention, and outcome.
6. Do the pilot search with different combinations of search terms.
7. Check pilot search results.
8. Refining search terms based on discussion among the authors.

Search terms used in the pilot and final round of search were identified in Table 2. The pilot search resulted in a huge number of hits for general search terms like "semantic" and almost no result for very specified search terms like "OWL" and "RDF". After analyzing the results of the pilot search, we performed some refinement of search terms being used for the second and final round of search. The search terms were used with quotation marks for searching exact phrases. The search string was constructed as follows (Kitchenham and Charters, 2007):

$(P_1$ OR P_2 ... OR $P_n)$ AND $(I_1$ OR I_2 ... OR $I_n)$

Where P_n refer to population terms and I_n refer to intervention terms connected using the Boolean operators AND and OR.

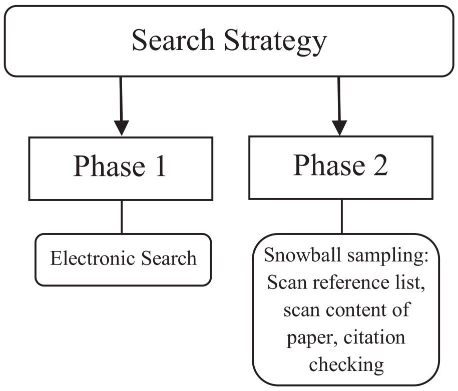
Search Strategy
Phase 1
Phase 2
Electronic Search
Snowball sampling: Scan reference list, scan content of paper, citation checking

Fig. 2. Search strategy.

3.3. Search strategy and data sources

The search was divided into two main phases (see Fig. 2):

1. *Phase one.Direct database search.* In this phase, we automatically searched electronic databases. The search terms were used to search the following five well-known and widely used digital libraries and databases: ACM Digital library[4], IEEE Xplore[5], ScienceDirect[6], SpringerLink[7], Scopus[8].
The search was conducted on July 2019 and was limited to studies published from 2000 until that date. The reason we chose year 2000 as the starting year for our search, is that

[3] http://academic.research.microsoft.com

[4] http://dl.acm.org.
[5] http://ieeexplore.ieee.org.
[6] http://www.sciencedirect.com.
[7] www.springerlink.com.
[8] www.scopus.com.

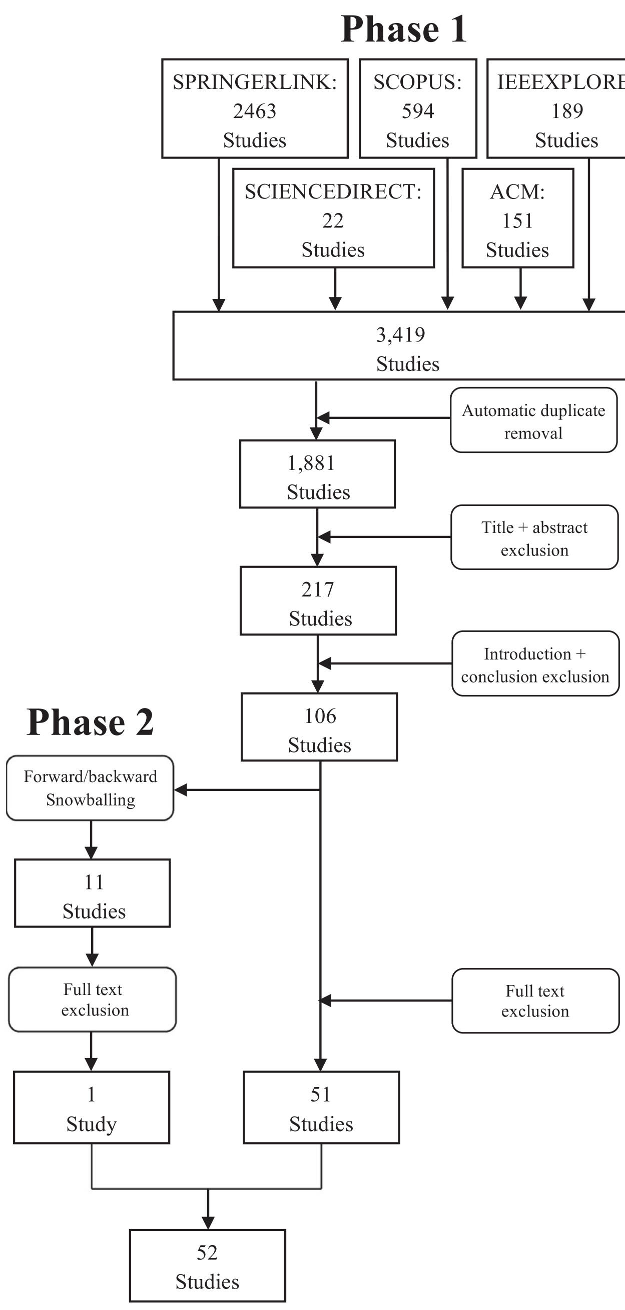

Fig. 3. Study selection process.

it predates the first ontology language for the web, i.e., the DARPA Agent Markup Language (DAML) (Horrocks et al., 2002). In Springer Link, a limitation to search only in 'title', 'abstract' and 'keywords' were not possible. Therefore, we searched in full-text while for all other databases we searched in 'title', 'abstract', and 'keywords'. So, the search resulted in a vast number of hits for Springer Link. Fig. 3 outlines the numeric results of the electronic search in databases.

2. *Phase two. Forward and backward snowballing.* In this phase, both backward and forward snowballing techniques are applied. These techniques use respectively the list of the references and the citations of a study to identify new relevant publications (Wohlin, 2014). For forward snowballing we considered checking citations of the selected studies in Google scholar [9]. As a result of this phase, 11 other studies are identified. After applying the full-text exclusion criteria over these studies, one study remained.

As a final result, we got to 52 studies to be analyzed (51 from the sources, and 1 from snowballing).

3.4. Study selection process

The direct database search (phase 1) resulted in a total of 3419 studies. After eliminating duplicates, the number of results reduced to 1881 (see Fig. 3). Then, the exclusion process was performed by two authors of this paper independently. In each step, after reading that part, authors chose one out of the three possible remarks for each study 'yes' (for inclusion) or 'maybe' (for further investigation in the next study selection step) and 'no' (for exclusion due to irrelevance to the research question). The exclusion steps are:

1. Title and abstract exclusion. In this step, the authors agreed to exclude 1664 studies.
2. Introduction and conclusion exclusion. The researchers agreed to exclude 111 studies and to include 106 studies.
3. Full-text exclusion. The full text of the remaining 106 studies was read, and a further set of 55 studies were excluded by consensus. These 106 studies were also considered as the basis for conducting Phase 2.

The detailed inclusion and exclusion criteria are presented in Tables 3 and 4 summarizes the phases and their results.

3.5. Study quality assessment

Selected studies were evaluated against a set of 11 quality criteria, nine of them were adapted from SLRs published in a high reputation venue, the remaining two questions were proposed according to the scope and research questions of this SLR. Q1, Q2, Q3, Q4, Q5, Q7, Q8, Q9, and Q10 were adopted from the literature, while Q6 and Q11 were proposed. The assessment questions used are presented in Table 5.

The scores of questions Q2, Q6, and Q7 were determined using a two-grade scale score (Yes/No). If the answer was Yes, the study received 1 point in this question; otherwise, it received 0. Besides these alternatives, the questions Q1, Q3, Q4, Q5, Q8, Q9 also allowed a third one. If the contribution was not so substantial, the study received 0.5=" to some extent", consisting of a three-grade scale score to these questions. Q10 receives 1 point if the study is applied in industry and 0.5 if its setting is the academy. Q11 receives 1 point if the proposed solution is generally applicable in software testing and 0.5 if the solution is proposed for domain-specific software. The study quality score is computed by finding the sum of all its scores of the answers to the questions. Each selected study was assessed independently by the authors. All discrepancies on the scores were discussed among the authors, and the study was reevaluated in cases of non-agreement with the aim of reaching consensus.

3.6. Data extraction and synthesis

The data extraction process was performed by full-text reading for each one of the selected studies. In order to guide this data extraction, the data collection process from Kitchenham and Charters (2007) was adopted. Data were extracted according to a predefined extraction form (see Table A.24 in 'Appendix A'). This form enabled us to record full details of the studies under review and to be specific about how each of them addressed our research questions.

[9] https://scholar.google.com.

Table 3
Detailed inclusion and exclusion criteria.

Inclusion criterion		Exclusion criterion	
I1	Primary studies	E1	Secondary studies
I2	Peer-reviewed studies	E2	Non-peer-reviewed studies
I3	All studies published in English language	E3	Short studies (< 4 pages)
I4	Satisfies the minimum quality threshold	E4	Knowledge engineering exclusive studies
I5	Studies published from 2000 to July 2019	E5	Duplicate studies (only the most complete, recent and improved one is included).
I6	Studies that use the semantic web to support testing process	E6	Studies that do not use Semantic Web in software testing process
I7	All published studies that have the potential of answering at least, one research question	E7	Gray literature; i.e., editorial, abstract, keynote, opinion, studies without bibliographic information e.g., publication date/type, volume and issue numbers

Table 4
Study selection results and reductions.

Phase	Criteria	Analyzed content	Initial number of studies	Final number of studies	Reduction (%)
1st	Duplication removal	Title, abstract, keyword	3,419	1,881	44.9
1st	I3, E4, E6, E7	Title and abstract	1,881	217	88.4
1st	I1-I7 and E1-E7	Introduction and conclusion	217	106	56.7
1st	I6, I7, E4, E6	Full text	106	51	51.1
2nd	I1-I7 and E1-E7	Full text	11 (From snowballing)	1	90.9
		Final result	3419 + 11 = 3430	51 + 1 = 52	98.48

Table 5
Quality assessment criteria.

	Questions	Possible answers		
1	Is there a rationale for why the study was undertaken? (Dermeval et al., 2015; Mahdavi-Hezavehi et al., 2013)	Y=1	N=0	P=0.5
2	Is the study based on research (or is it merely a “lessons learned” report based on expert opinion)?(Dermeval et al., 2015; Dybå and Dingsøyr, 2008)	Y=1	N=0	
3	Are the aims and the objectives of the research clearly articulated?(Dermeval et al., 2015; Tosi and Morasca, 2015; Tahir et al., 2013; Achimugu et al., 2014; Dybå and Dingsøyr, 2008)	Y=1	N=0	P=0.5
4	Is the proposed semantic web enabled testing technique clearly described?(Tahir et al., 2013; Achimugu et al., 2014)	Y=1	N=0	P=0.5
5	Is there an adequate description of the context (industry, laboratory setting, products used and so on) in which the research was carried out?(Dermeval et al., 2015; Tosi and Morasca, 2015; Mahdavi-Hezavehi et al., 2013; Dybå and Dingsøyr, 2008; Tahir et al., 2013)	Y=1	N=0	P=0.5
6	Is the study supported by a semantic web enabled tool?	Y=1	N=0	
7	Does the study have an empirical evaluation?(Tosi and Morasca, 2015; Dermeval et al., 2015; Tahir et al., 2013)	Y=1	N=0	
8	Is there a discussion about the results of the study?(Dermeval et al., 2015)	Y=1	N=0	P=0.5
9	Do the authors discuss the credibility and limitations of their findings explicitly?(Dermeval et al., 2015; Mahdavi-Hezavehi et al., 2013; Ding et al., 2014)	Y=1	N=0	P=0.5
10	Does the study provide value for research or practice.(Dybå and Dingsøyr, 2008; Achimugu et al., 2014)	Y=1		P=0.5
11	Does the study propose a general or domain-specific solution?	Y=1		P=0.5

4. Overview of the studies

A total of 52 studies met the inclusion criteria, and their data were extracted. A complete list of these studies is presented in Table 6. The first column divides the studies into three main categories based on their major contribution. Each category corresponds to one or more of the research questions. The first category, theoretical foundations, includes studies that correspond to the RQ1. The second category, concrete approaches, includes studies that correspond to the RQ2, RQ3, RQ5, and RQ6. The last category, test ontologies, includes studies that correspond to the RQ4.

In the second column of Table 6, sub-categories of the main categories are identified. Sub-categories of the theoretical foundations include: studies that propose a semantic web enabled test process or knowledge management systems (See Section 5). There are seven sub-categories for the concrete approaches (See Section 6). Semantic web technologies that have been used in the proposed concrete approaches are identified in Section 7. The proposed test ontologies are categorized into reference and application ontologies and then investigated based on their specifications in Section 8. The application domains that concrete approaches have been proposed for are presented in Section 9. Finally, improvements provided by semantic web enabled software testing are discussed in Section 10. Each research question will be answered in a separate section.

4.1. Publication year

The reviewed studies were published between 2005 and 2019. From a temporal point of view (Fig. 4), an increasing number of publications in the context of this review is observed since 2009. Most of the studies have been published in 2011 (15.3%), 2017 (13.4%), and 2015 (11.5%) followed by 2019, 2014 and 2009 (0.09%).

Table 6
Final list of selected studies.

Category	Sub-category	Studies	Count	Total count	%
Theoretical foundations	semantic web enabled test process	Paydar and Kahani (2010), Nasser et al. (2010), Nakagawa et al. (2011), Bueno et al. (2018), Çiflikli and Co\cskunçay (2018), Eckhart et al. (2019)	6	10	19.2
	semantic web enabled KMS	Vasanthapriyan et al. (2017), Palacios et al. (2014), Liu et al. (2009), Hilera et al. (2018)	4		
Concrete approaches	Test generation	Tseng and Fan (2013), Sinha et al. (2015), Tarasov et al. (2016), Silva et al. (2017), Tonjes et al. (2015), Moser et al. (2010), Nguyen et al. (2009), Hajiabadi and Kahani (2011), Li et al. (2011), Naseer and Rauf (2012), Rauf et al. (2010), Tao et al. (2019), Moitra et al. (2019), Ul Haq and Qamar (2019), Mekruksavanich (2017), Silva et al. (2019)	16	33	63.4
	Test data generation	Mariani et al. (2014), Szatmári et al. (2011), Li and Ma (2015a)	3		
	Test oracle	Bai et al. (2011)	1		
	Test reuse	Dalal et al. (2015), Li and Ma (2015b), Li and Zhang (2012), Guo et al. (2011), Cai et al. (2009)	5		
	Traceability	Guo et al. (2009), de Almeida Falbo et al. (2014), Alqahtani et al. (2017), Bicchierai et al. (2013)	4		
	Consistency checking	Feldmann et al. (2016), Harmse et al. (2014)	2		
	Test optimization	Sapna and Mohanty (2011), De Campos H.S. et al. (2017)	2		
Test ontologies	Reference ontologies	Souza et al. (2017), Barbosa et al. (2006), Zhu and Huo (2005), Engström et al. (2017)	4	9	17.3
	Application ontologies	Freitas and Vieira (2014), Arnicans and Straujums (2015), Bezerra et al. (2009), Anandaraj et al. (2011), Duarte et al. (2018)	5		
Total				52	

We can observe that researchers are currently concerned with the topic.

4.2. Publication sources

The studies included in this review may be of a journal, conference, workshop, or book chapter publications (see Fig. 5). The majority of studies are conference publications (69.2%; 36 studies), followed by journal publications (17.3%; 9 studies), book chapter publications (9.61%; 5 studies) and workshop publications (3.84%; 2 studies). This indicates the immaturity of the proposed techniques and incomprehensiveness of the studies on using semantic web technologies to test industrial software. Table B.25 (in 'Appendix B') presents the distribution of selected studies over publication sources, including the publication name, type, count (i.e., the number of selected studies from each source), and the percentage of selected studies. The 52 selected studies are distributed over 48 publication sources, suggesting that the use of Semantic Web in testing process has been widespread concern in the research community. As shown in Table B.25, the leading venue in this study topic is the International Conference on Software Engineering and Knowledge Engineering (SEKE). This venue indicates the presence of sources of software and knowledge engineering areas.

5. Theoretical foundations

RQ1 investigates the theoretical foundations for semantic web enabled software testing. Studies addressing this research question are classified into two sub-categories. The first sub-category includes studies that propose to support test process with semantic web technologies and define a roadmap, framework or

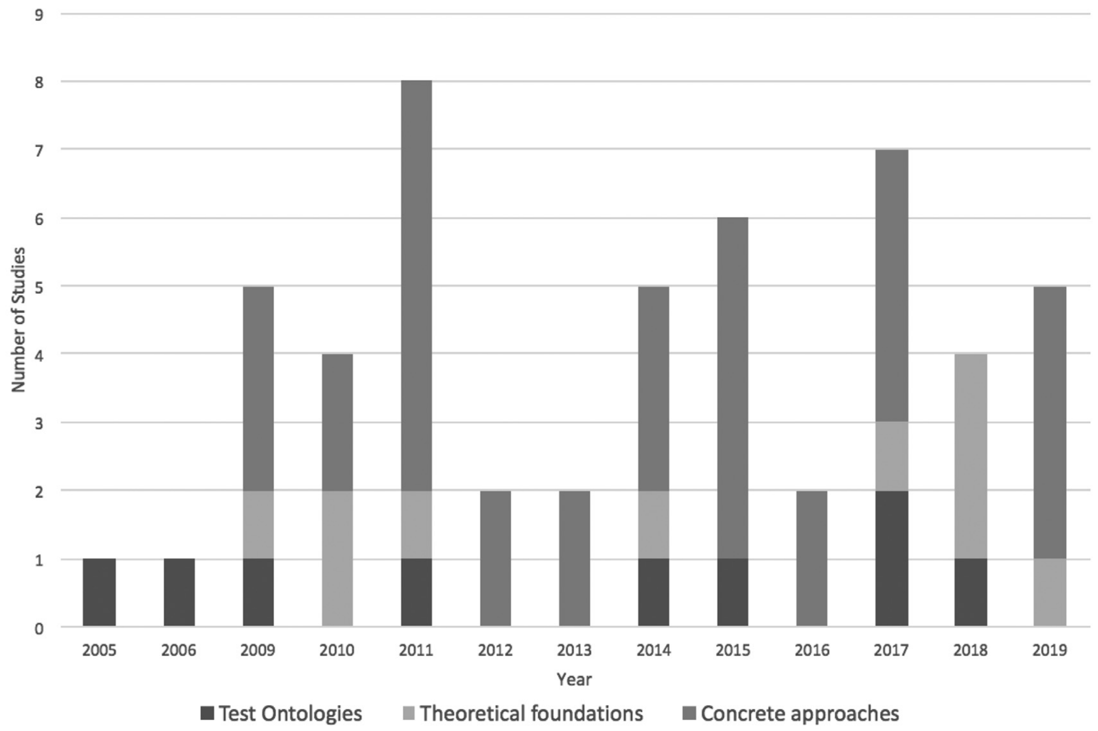

Fig. 4. Temporal view of the studies.

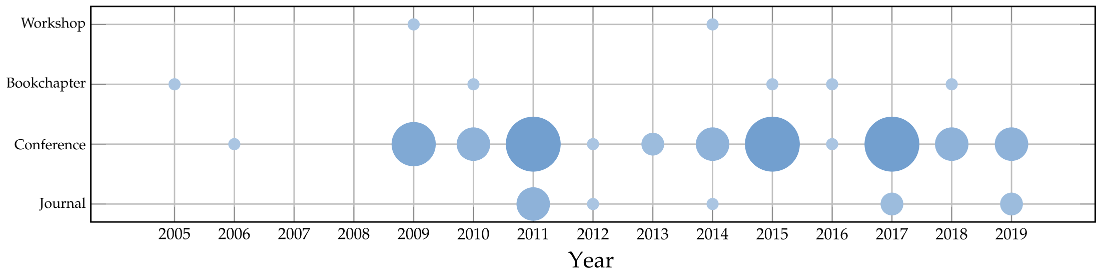

Fig. 5. Bubble plot with year and source of publication.

reference architecture to develop concrete approaches for realizing semantic web enabled software testing (Bueno et al., 2018; Nakagawa et al., 2011; Nasser et al., 2010; Paydar and Kahani, 2010; Çiflikli and Co\cskunçay, 2018; Eckhart et al., 2019). These studies don't propose a test ontology or ontology-based testing approach themselves. Instead, they investigate different knowledge management activities in software testing that might benefit from the semantic web technologies, e.g., ontologies. The second sub-category includes studies that propose Knowledge Management Systems (KMSs) based on semantic web technologies for sharing software testing knowledge and collaboration within the software testing community (Hilera et al., 2018; Liu et al., 2009; Palacios et al., 2014; Vasanthapriyan et al., 2017). Further analysis of studies in each of the sub-categories are as follows:

5.1. Semantic web enabled test process

There are six studies in this category that investigated theoretical foundations of using semantic web technologies in software testing by supporting the test process (Bueno et al., 2018; Nakagawa et al., 2011; Nasser et al., 2010; Paydar and Kahani, 2010; Çiflikli and Co\cskunçay, 2018; Eckhart et al., 2019). A separate section, C.1.1, is dedicated to summarizing the studies.

These studies suggest ways of utilizing test ontologies and semantic web technologies for automating various activities in software testing process:

1. Test specification (Paydar and Kahani, 2010; Nasser et al., 2010): Developing test ontologies for representing the knowledge of different testing activities, along with relationships and order amongst them.
2. Test generation (Paydar and Kahani, 2010; Nasser et al., 2010): Utilizing ontologies for automated test generation in the following two phases:
   (a) Abstract test generation (Nasser et al., 2010): Abstract tests are generated without considering a programming language or other implementation constraints.
   (b) Executable test generation (Nasser et al., 2010): For every abstract test, one or more executable test is generated using the implementation knowledge.
3. Test data generation: Generate test data based on domain ontologies (Paydar and Kahani, 2010). The web of data can be used as a source for finding appropriate test data(Mariani et al., 2014).
4. Test oracle (Paydar and Kahani, 2010): An ontology-based mechanism for deciding on the pass or fail of tests.

Table 7
Layers of knowledge management systems based on semantic web technologies.

Study	Knowledge management Layers					
	Ontology	Reasoning layer	Sharing layer	Enrichment layer	Retrieval layer	Storage layer
Vasanthapriyan et al. (2017)	✓	✓	✓	✓	✓	✓
Palacios et al. (2014)	✓	✗	✓	✓	✓	✓
Liu et al. (2009)	✓	✗	✓	✓	✓	✓
Hilera et al. (2018)	✓	✓	✗	✗	✓	✓

5. Test objectives (Nasser et al., 2010): Representing what needs to be tested.
6. Test planning (Nasser et al., 2010): Based on a test ontology and test objectives, the required tests and their order are inferred.
7. Traceability (Nasser et al., 2010): For every test objective, rules are defined to check whether it has been satisfied by a test or not.
8. Test optimization (Nasser et al., 2010): Ontology-based test representation makes it possible to identify and reason about redundant tests. It can also facilitate defining test coverage criteria and test selection rules.
9. Test process design and assessment (Bueno et al., 2018; Çiflikli and Co\cskunçay, 2018; Eckhart et al., 2019): Utilizing ontologies for supporting the test process.

These studies also suggest developing various ontologies for describing different types and levels of knowledge that can be used in automated software testing:

1. Test ontology (Nakagawa et al., 2011; Nasser et al., 2010; Paydar and Kahani, 2010): Representing knowledge within a software testing domain.
2. Domain ontology (Paydar and Kahani, 2010): Representing specific application domain knowledge.
3. Expert knowledge ontology (Nasser et al., 2010): Representing test experts mental models that can be used for defining the system or domain-specific coverage criteria, as well as identifying test objectives.
4. Behavioral Model Ontology (Nasser et al., 2010): Representing software artifacts, which are used for test generation (e.g., UML models ontology).
5. Implementation Knowledge Ontology (Nasser et al., 2010): Representing knowledge used in the automatic generation of executable tests for each abstract test.
6. Test process ontology (Bueno et al., 2018; Çiflikli and Co\cskunçay, 2018; Eckhart et al., 2019): Representing testing processes knowledge sources (e.g., TMMi (van Veenendaal, 0000), ISO/IEC/IEEE 29119) that can be used for test process design and assessment.

Three studies in this category are intensely focused on the test process itself and also proposed ontologies for representing test process knowledge (Bueno et al., 2018; Çiflikli and Co\cskunçay, 2018; Eckhart et al., 2019). Two of these studies proposed test process ontologies for test process evaluation (Bueno et al., 2018; Çiflikli and Co\cskunçay, 2018). Bueno et al. (2018) proposed an ontology-based test process to evaluate the security characteristics of IT systems. Çiflikli and Co\cskunçay (2018) presented an ontology-based assessment infrastructure to check test process maturity based on the TMMi reference model. The main contribution of these studies is to define conceptual models, including required ontologies and their main concepts for representing software testing process. They also illustrated example usage scenarios of how the proposed ontologies can be applied. Both of these two studies used TMMi (van Veenendaal, 0000) as their knowledge source of the test process for ontology development. Furthermore, Bueno et al. (2018) also used ISO/IEC/IEEE 29119-2010 for this purpose. The third study (Eckhart et al., 2019), presented a framework for supporting semi-automatic security analysis of software testing process. The proposed framework is based on the VDI/VDE 2182-1 (2011) guideline and utilizes ontologies for modeling background knowledge. These ontologies are used to model data flows within the software testing process (including concepts like Process, DataFlow, DataStore), and attack-defense trees (including concepts like AttackNode, DefenseNode, Threat).

5.2. Semantic web enabled knowledge management systems

Knowledge Management Systems (KMSs) store and retrieve the knowledge that improves collaboration between users. Users can create and share knowledge through a knowledge base. Reusing the shared knowledge is possible through searching and retrieving from the knowledge base. There are four studies that proposed a software testing KMS based on semantic web technologies (Hilera et al., 2018; Liu et al., 2009; Palacios et al., 2014; Vasanthapriyan et al., 2017). A separate section, C.1.2, is dedicated to summarizing the studies.

Here, we will investigate these studies from two points of view. First, we will check the Knowledge Management Layers (KMLs) of each proposed KMS. Second, we will study how each KMS provides collaboration between their users. Knowledge management layers of proposed KMSs are presented in Table 7. We have identified six layers in the proposed KMSs.

1. Ontology layer: In this layer, the semantic web data model supports the representation of the knowledge in software testing domain.
2. Reasoning layer: Specifies logical consequences that are inferred from existing facts to enrich the knowledge base.
3. Sharing layer: Provides sharing test knowledge amongst testers and project managers that may also include testers specialty and, or competence level.
4. Enrichment layer: Enables testers to annotate the knowledge base with test ontologies. Such enrichment can improve knowledge search and retrieval.
5. Retrieval layer: Supports search and retrieval process in the knowledge base.
6. Storage layer: Provides other layers with permanent storage of data for knowledge maintenance.

KMSs encourage collaboration amongst stakeholders through sharing, enriching, and retrieving knowledge from test knowledge bases. Each of the proposed KMSs has targeted its groups of users. Table 8 summarizes the group of users and the knowledge they can share, enrich, and retrieve.

Here is a list of our findings regarding RQ1:

1. As it is presented in Table 7, all of the proposed KMSs included ontology, storage, and retrieval layers, and most of them included sharing and enrichment layers. Two KMSs support reasoning layer (Hilera et al., 2018; Vasanthapriyan et al., 2017).
2. As it is presented in Table 8, proposed software testing KMSs usually have two primary goals. The first is to share and

Table 8
Knowledge sharing and retrieval in KMSs proposed based on semantic web technologies.

Study	Users	Share/Enrich	Retrieval
Vasanthapriyan et al. (2017) Palacios et al. (2014)	Testers Contractors	- Annotate test knowledge - Define test process and it's required competences and competence level - Rate tester	- Search test knowledge - Searching testers based on competences and competence level
	Testers	- Rate the process and contractor - Feedback on test process - Feedback on the test results	- Searching processes based on competencies and competence level
Liu et al. (2009)	Testers	- Documents and questions - Evaluate knowledge level of documents	- Retrieve Knowledge documents
	Managers	- Evaluate knowledge level of documents	- Search tester and specialist
	Knowledge Analyst	- Submit knowledge document - Evaluate the knowledge level of staff - Evaluate knowledge level of documents	
Hilera et al. (2018)	Testers	- Load and combine test reports	- Search final test result

reuse testing knowledge, which is supported by most of the KMSs. Semantic web technologies can be used for managing knowledge of testers, test process, and test documents, e.g., test reports. The second is to facilitate collaboration between users, that is only supported by Palacios et al. (2014) and Liu et al. (2009). In (Palacios et al., 2014), on one hand, contractors can define test process and search for appropriate testers based on their competences, and ratings. On the other hand, testers can search for test processes based on the required competence level. In Liu et al. (2009), only managers can search for testers based on their level of knowledge.

3. Only Liu et al. (2009) considers knowledge analysts as the third group of users that can share and evaluate knowledge. The knowledge analysts also set the knowledge level of testers.

6. Concrete approaches

RQ2 investigates concrete approaches for realizing the semantic web enabled software testing. There are 33 studies in this category that cover software testing activities (see Table 6).

The distribution of the above studies based on the software testing activities that they have addressed is shown in Fig. 6. The main concern of each study is considered as its primary topic (shown in a darker shade). If a study has also mentioned other test activities, besides its primary topic, that activity has been considered as a secondary topic (shown in the gray shade). For example, Nguyen et al. (2009) have proposed a semantic web enabled test generation approach (its primary topic) and just mentioned the oracle problem (its secondary topic). Although the secondary topics are not investigated as well as the primary ones in the studies, if the authors mentioned them, we have counted them. Distribution of the studies shows that test generation has received the most attention, whereas, test oracle followed by test optimization, and consistency checking have received the least attention as the primary topics.

In the following sections, we will first investigate studies that proposed a test generation approach using semantic web technologies in Section 6.1. Then we will present results on studies proposing a test data generation approach in Section 6.2. Studies addressing test oracle and test reuse are presented in Section 6.3 and Section 6.4, respectively. Finally, we will investigate studies that utilized semantic web technologies for improving software testing in three activities, i.e., traceability, consistency checking, and test optimization in Section 6.5.

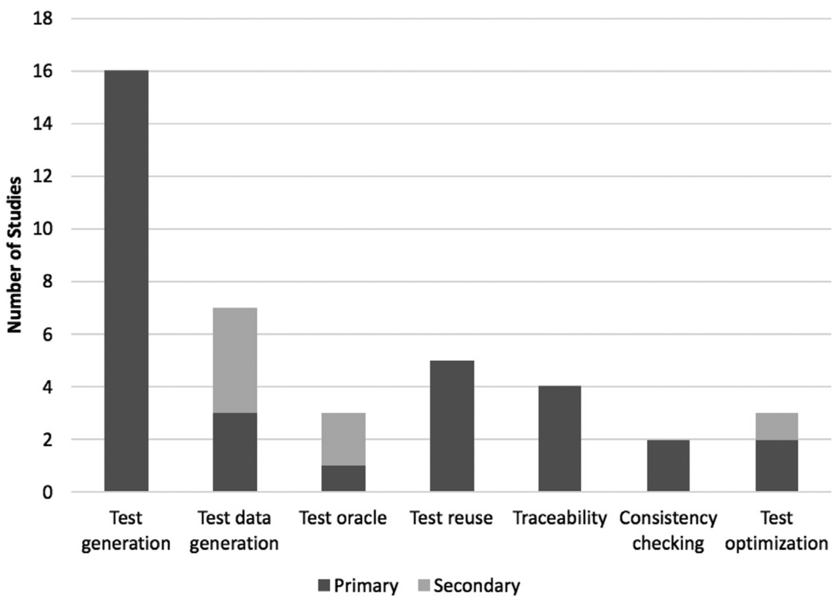

Fig. 6. Distribution of the primary and secondary topics of studies.

Table 9
Specifications of proposed test generation approaches.

Study	Requirement type	Test level	SUT
Silva et al. (2017)	Functional	Higher levels	A flight tickets e-commerce application
Hajiabadi and Kahani (2011)	Functional	Higher levels	WebCalendar application
Li et al. (2011)	Functional	Higher levels	A communication application
Naseer and Rauf (2012)	Functional	Higher levels	Notepad application
Rauf et al. (2010)	Functional	Higher levels	-
Tseng and Fan (2013)	Functional	Higher levels	A general nuclear plant simulator PCTran
Sinha et al. (2015)	Functional	All levels	An industry-strength water process system
Tarasov et al. (2016)	Functional	Higher levels	An embedded system provided by Saab Avionics
Moser et al. (2010)	Functional	Higher levels	A production automation system (an extension of MAST Vrba, 2003)
Nguyen et al. (2009)	Functional	Lower levels	A book-trading multi-agent system
Tonjes et al. (2015)	Functional	-	A generic service
Tao et al. (2019)	Functional	Higher levels	An Autonomous Emergency Braking System Function (AEB)
Moitra et al. (2019)	Functional	Higher levels	Several projects within General Electric Company
Ul Haq and Qamar (2019)	Functional	Higher levels	-
Mekruksavanich (2017)	Functional	Higher levels	CommonCLI v1.0 and JUNIT v1.3.6
Silva et al. (2019)	Functional	Higher levels	A web system for booking business trips

-: not mentioned.
http://webcalendar.sourceforge.net
www.microsimtech.com/pctran/

6.1. Test generation approaches

In this section, we will investigate the studies that propose test generation approaches. A separate section, C.2.1, is dedicated to summarizing the studies. Tables 9 and 10 present details of the proposed approaches. Table 9 shows the test generation source, requirements type, test levels, and application domain. Table 10 shows the test generation methods and artifacts, and test evaluation methods and criteria as well as the System/Software Under Test (SUT).

The following is a list of our findings on test generation approaches based on the information presented in Tables 9 and 10:

1. All of the studies generate tests for functional requirements. Around 87% of the studies support higher levels of testing (e.g., acceptance testing, system testing).
2. These approaches have been applied on a wide range of systems including avionics and control systems, nuclear plant simulators, automation systems, agent-based systems, and several projects within General Electric (GE) company. Therefore, it is observed that semantic web technologies have been used for testing industrial software, even in safety-critical systems.
3. Around one third (31%) of the test generation methods used by the proposed approaches are rule-based, around one quarter (18%) of them are model-based, and (18%) behavior-driven.
4. Different artifacts and behavior models (e.g., sequence diagrams and state-charts) have been used in the proposed approaches. They have also used GUI elements, event flow graph, requirements, and user stories.
5. Most of the proposed approaches (81%) have provided an evaluation of their generated tests. Among these studies, 53% have compared their results with another work, and 46% have analyzed the results of applying their approach to some case studies. The studies that have used comparison for evaluation have compared their results mostly with manual testing (29%), and random testing (29%). There is not a consensus on using a specific approach for comparison in the field. They have not also compared their approaches with each other. Most of the studies (75%) have evaluated their approach using test coverage criteria (e.g., path coverage, condition coverage, action coverage, and scenario coverage). Although various coverage criteria have been used, most of them are high level and requirement driven (e.g., scenario coverage, requirement coverage, and model coverage). Only one of the studies has used failure detection rate and execution time as evaluation criteria.

6.2. Test data generation

While test generation generally includes generating test data, there are studies that have considered this step as a separate activity. Three studies have addressed the test data generation as their primary concern (Mariani et al., 2014; Szatmári et al., 2011; Li and Ma, 2015a). Four studies (Tonjes et al., 2015; Nguyen et al., 2009; Rauf et al., 2010; Hajiabadi and Kahani, 2011) that have proposed test generation approaches (their primary topic) have also addressed test data generation (their secondary topic). In the following, we will investigate the specifications of all these seven approaches (Table 11). A separate section, C.2.2, is dedicated to summarizing the studies. The following is a list of our findings concerning test data generation:

1. More than half of the proposed approaches (57%) have used ontology mapping for test generation. In these studies, ontology mapping has been used to generate a mapping from 1) a GUI model to the Web of Data (Mariani et al., 2014), 2) a context ontology to domain meta-model (Szatmári et al., 2011), 3) a GUI ontology to database ontology (Hajiabadi and Kahani, 2011), and 4) a Test Atom to interface elements (Li and Ma, 2015a). Some of the studies (29%) have used semantic annotation for test data generation. In these studies, Annotations have been applied on parameters (Tonjes et al., 2015), and event flow graph (Rauf et al., 2010). Only one of the studies, (Nguyen et al., 2009) have used rule-based techniques and boundary value analysis for test data generation.
2. All of the proposed approaches generate valid data, and only one of them generates both valid and invalid data. In the software testing domain, testing software systems with invalid data is considered as necessary as testing with valid data.
3. Most of the studies (57%) have not mentioned their data source. In one of the studies reported by Hajiabadi and Kahani (2011), a relational database has been used for storing test data. One study, (Nguyen et al., 2009), has used domain ontology instances as its data source, and one study, (Mariani et al., 2014), has used the Web of data for its test data generation, where a data set from the Linked Open Data (LOD)[10] cloud, i.e., DBpedia, has been used. Hence, we see that the opportunity to utilize the Semantic Web sources has been neglected by most of these studies.

[10] http://linkeddata.org/.

Table 10
Test generation and evaluation methods of the proposed approaches.

Study	Test generation		Test evaluation	
	Method	Artifact/Model	Method	Criteria
Silva et al. (2017)	Behavior-driven	State machine	Case study results (generated test cases)	-
Hajiabadi and Kahani (2011)	Model-based	GUI elements	Comparison with manual testing	Coverage
Li et al. (2011)	Rule-based	GUI element tree	Comparison with GUI Ripping (Memon et al., 2003)	Component sequence coverage
Naseer and Rauf (2012)	Rule-based	Event Flow Graph	-	Path coverage Efficiency
Rauf et al. (2010)	Semantic annotations	Event Flow Graph	-	Coverage of GUI events
Tseng and Fan (2013)	Model-based	UML sequence diagrams	Comparison with Random testing	Equivalence partition coverage, Condition coverage, Action coverage, Scenario coverage
Sinha et al. (2015)	Model-based	UML state charts	Case study results (test execution results of output values)	-
Tarasov et al. (2016)	Rule-based	Ontology-based component structure	Comparison with Industry results (generated test cases)	Requirement coverage, Code coverage
Moser et al. (2010)	Scenario-based	Ontology-based GUI specification	Comparison with traditional static approach	Test coverage, Efforts for test description and implementing test case parameters
Nguyen et al. (2009)	Rule-based	Ontology-based agent interaction model	Comparison with manually derived tests	Ontology coverage, Revealed faults
Tonjes et al. (2015)	Behavior-driven	FSM-based Application Behaviour Model (ABM)	Comparison with Random Selection	Failure detection rate, Computation time
Tao et al. (2019)	Scenario-based	Ontology-based combinatorial testing input models	Case study results from simulation platform	-
Moitra et al. (2019)	requirements-based	requirements written in structured natural language	Case study results from simulation	Model coverage, Structural coverage
Ul Haq and Qamar (2019)	Learning-based	Ontology-based requirement specification	-	-
Mekruksavanich (2017)	Rule-based	Ontology of design flaws	Case study results	Precision, false positive
Silva et al. (2019)	Behavior-driven	Ontology-based user stories	Case study results	passed and failed tests

-: not mentioned.

Table 11
Specifications of proposed test data generation approaches.

Study	Data generation technique	Input type	Data source	Software artifact/model	Evaluation method	Evaluation criteria
Mariani et al. (2014)	Ontology mapping	Valid	The Web of Data i.e., DBpedia	GUI elements	Comparison with regular expressions	Branch coverage
Szatmári et al. (2011)	Ontology mapping	Valid	-	Context model	Case study	Context patterns coverage
Li and Ma (2015a)	Ontology mapping	Valid	-	GUI elements	Case study	User experience, Execution time
Tonjes et al. (2015) Hajiabadi and Kahani (2011)	Semantic annotation	Valid	-	GUI elements	Indirect	Indirect
Rauf et al. (2010)	Semantic annotation	Valid	-	GUI elements	Indirect	Indirect
Nguyen et al. (2009)	Rule-based and boundary value analysis	Valid and Invalid	Domain ontology	Agent interaction Model	Indirect	Indirect

-: not mentioned.

4. Most of the proposed approaches (71%) have used GUI elements in test data generation. Two studies (Szatmári et al., 2011; Nguyen et al., 2009) have proposed a test data generation approach using context model and an agent interaction model, respectively.
5. Four of the studies that have addressed test data generation as their secondary topic have not evaluated the results separately. They have just evaluated the results of test generation in general (see Section 6.1). They have presumed that evaluating the test generation process includes test data generation as well. From the other three studies, two of them (Szatmári et al., 2011; Li and Ma, 2015a) have analyzed their results based on case studies, and one (Mariani et al., 2014) has compared results with regular expressions. Two of these three studies (Mariani et al., 2014; Szatmári et al., 2011) have used coverage criteria for evaluation, and one (Li and Ma, 2015a) has considered user experience and execution time.
6. An important issue that none of the approaches has considered in their test data generation is the test type. For instance, in web applications, security and performance testing are usually done with different test data (e.g., testing against SQL injection vulnerabilities requires using string values with special characters and sequences, while testing against performance issues might need to enter very long strings).

Table 12
Specifications of the proposed test oracles.

Study	Output type	Technique	Evaluation method	SUT
Bai et al. (2011)	OS interface services outputs, error messages, and effects on environment variables	Rule-based	Productivity (estimated average cost of each test oracle), Quality (errors in test oracle)	Operating systems
Nguyen et al. (2009)	Response message of the agent	Rule-based	Indirect	agent-based systems
Tonjes et al. (2015)	Parameters output values	Semantic annotation	Indirect	context-aware applications

-: not mentioned.

6.3. *Test oracle*

Automatic software testing requires an automated test oracle, i.e., a procedure that distinguishes between the correct and incorrect behaviors of the SUT. Given an input for a system, the challenge of distinguishing the corresponding desired, correct behavior from potentially incorrect behavior is called the test oracle problem (Barr et al., 2015). However, compared to other test activities, the problem of automating the test oracle has received significantly less attention and remained comparatively less well-solved (Barr et al., 2015). This current open problem represents a significant bottleneck in test automation. Only one study (Bai et al., 2011) has considered test oracle as its primary topic. Its proposed approach requires hard-coding all test oracles into test scripts. This means that maintaining the test scripts is expensive and requires a thorough analysis of the test programs. Another issue with this approach is that the test designers need to have expertise in a rule-based specification language, which is usually logic-based and harder to use than procedural programming languages.

Two other studies (Tonjes et al., 2015; Nguyen et al., 2009) have addressed test oracles as their secondary topic. The specifications of all these three studies are presented in Table 12. A separate section, C.2.3, is dedicated to summarizing the studies. The following is a list of our findings concerning test oracle:

1. The proposed test oracles have supported a variation of outputs. Based on the corresponding SUT, proposed oracles can handle diverse types of outputs.
2. Two of the studies (Bai et al., 2011; Nguyen et al., 2009) have used rule-based techniques for test oracle generation. They have modeled the SUT with ontology rules. The other study (Tonjes et al., 2015) has used semantic annotations to augment parameters with data about output value.
3. Two studies that have addressed test oracle as their secondary topic haven't evaluated the results separately. They have evaluated the results of test generation in general, which have been investigated in Section 6.1. It is presumed that with evaluating the test generation process, test oracle is evaluated as well. Only one of the studies (Bai et al., 2011) have evaluated the proposed test oracle directly in terms of productivity and quality.
4. All of these studies tried to exploit the idea of ontology-based test oracles in their specific domain (i.e., context-aware applications and agent-based systems). However, considering the scale of the experimental evaluations (or the size and complexity of the system that is used for evaluation), we can conclude that the idea is not explored sufficiently.

6.4. *Test reuse*

In the software engineering domain, software reuse has been a well-known strategy for reducing development costs and improving quality. Ontologies can provide a flexible approach for capturing and retrieving reusable software artifacts. Keivanloo and Rilling (2014) have used semantic web technology for representing and analyzing large global source code corpora. They have introduced SeCold, the first online linked data source code dataset which is publicly available for software engineering researchers and practitioners. Although source code is the most commonly reusable asset in the software domain, other types of assets can also be reused. To represent, share, and retrieve reusable artifacts, reuse ontologies have been proposed in the literature (Da Silva et al., 2014). In (Da Silva et al., 2014), ONTO-ResAsset, an ontology of reusable assets specification and management has been proposed. It's estimated that almost 60% of the total test time and cost were spent on the design of test cases (Li and Ma, 2015b). Therefore, if one can generate reusable test cases and then store them after software testing is performed successfully, it can reduce testing cost and time and also improve the quality of the software. Ontologies can provide a more flexible approach for representing reusable test cases and can handle user queries for retrieving them.

The most important objective of using test ontology is for test reuse. When a quality test is generated, it is important to store it in a machine understandable format, define retrieval measures for querying test repository and reuse those tests. Five studies (Dalal et al., 2015; Li and Ma, 2015b; Li and Zhang, 2012; Guo et al., 2011; Cai et al., 2009) have addressed test reuse based on semantic web technologies. Four of them (Li and Ma, 2015b; Li and Zhang, 2012; Guo et al., 2011; Cai et al., 2009) have also proposed test ontologies which have been investigated in Section 8. A separate section, C.2.4, is dedicated to summarizing the studies. In the following, we will investigate the proposed approaches from three aspects: test storage, retrieval, and adaptation. The specifications of these approaches are presented in Table 13. Here is a list of our findings concerning test reuse:

1. Only two studies (Li and Zhang, 2012; Guo et al., 2011) (40%) have mentioned their test storage. They have used ontology or ontology-based KMS for storage.
2. Three studies (Dalal et al., 2015; Li and Ma, 2015b; Cai et al., 2009) (60%) have mentioned their retrieval technique. One of those (Dalal et al., 2015) has used a matching technique for test case retrieval. It has matched the ontology of application under test with the ontology of applications which their test cases are going to be reused. Further, the data type properties have been retrieved for which its related test cases can be reused. Two studies (Li and Ma, 2015b; Cai et al., 2009) have used semantic similarity for finding appropriate reusable test cases. In (Li and Ma, 2015b), the semantic similarity between test cases and test requirements has been used while (Cai et al., 2009) has used semantic similarity between concepts of ontologies. In (Cai et al., 2009), semantic annotations have also been used to augment test cases with meta-data and to retrieve reusable test cases.
3. Only two studies (Li and Ma, 2015b; Cai et al., 2009) (40%) have mentioned test adaptation, and both of them have used semantic rules.

Table 13
Specifications of the proposed test reuse approaches.

Study	Storage	Retrieval	Adaptation
Dalal et al. (2015)	-	Ontology matching	-
Li and Ma (2015b)	-	Semantic similarity	Rule-based
Li and Zhang (2012)	Knowledge management system	-	-
Guo et al. (2011)	Ontology	-	-
Cai et al. (2009)	-	Semantic similarity, Semantic annotation on test cases	Rule-based

-: not mentioned.

Table 14
Specifications of the proposed test optimization approaches.

Study	Optimization technique	Optimization goal	Optimization criteria
Sapna and Mohanty (2011)	Selection	Maximizing requirement coverage	Test coverage
De Campos H.S. et al. (2017)	Selection, Prioritization	Source code coverage, Failure detection rate	Provenance data about past executions of regression tests
Tonjes et al. (2015)	Selection	Maximizing test case diversity	Semantic similarity between tests

6.5. Subsidiary activities: traceability, consistency checking, test optimization

There are eight studies (Guo et al., 2009; de Almeida Falbo et al., 2014; Alqahtani et al., 2017; Bicchierai et al., 2013; Feldmann et al., 2016; Harmse et al., 2014; Sapna and Mohanty, 2011; De Campos H.S. et al., 2017) that cover activities related to semantic web enabled software testing and support accomplishing optimized testing. In the following paragraphs, these studies and the activities that they have addressed are investigated. A separate section, C.3, is dedicated to summarizing the studies.

Four studies (Guo et al., 2009; de Almeida Falbo et al., 2014; Alqahtani et al., 2017; Bicchierai et al., 2013) have utilized semantic web technologies to provide traceability for improving test quality. Generally, traceability is classified into horizontal and vertical traceability. The former includes relations between different models, while the latter includes relations between elements of the same model (Lindvall and Sandahl, 1996). Vertical and horizontal traceability can help testers in test generation specifications or test coverage evaluations. Semantic web technologies such as SPARQL queries and rules have the potential to find traceability between different software artifacts, especially between requirements, tests, and failures. This information can improve the quality of generated test cases and define new coverage criteria.

Two studies (Feldmann et al., 2016; Harmse et al., 2014) have used semantic web technologies for consistency checking between tests and requirements. Requirements consistency is one of the problems that have also been addressed by the majority of studies in ontology-driven requirement engineering (Dermeval et al., 2015).

Two studies (Sapna and Mohanty, 2011; De Campos H.S. et al., 2017), have utilized semantic web technologies for test optimization as their primary topic. One study, (Tonjes et al., 2015), has also addressed this concern as its secondary topic. Specifications of these three studies are presented in Table 14. With the increasing size of software systems, it is necessary to manage test scenarios and to optimize test suites, i.e. minimization, selection, and prioritization (Yoo and Harman, 2012). The test scenario management also involves ordering and selecting test scenarios for fulfilling criteria like maximum coverage or defect discovery (Sapna and Mohanty, 2011).

Here are the findings on the benefits of using ontology-based traceability, consistency checking, and test optimization:

1. Ontology-based traceability between requirements and test cases has been used to measure the requirements coverage of each test case (Guo et al., 2009; de Almeida Falbo et al., 2014; Alqahtani et al., 2017; Bicchierai et al., 2013). It has also been used to identify failure events by finding the associations between failures, requirements, and tests (Bicchierai et al., 2013). Traceability information provided by semantic web technologies has been used for test optimization. For example, test cases can be selected or prioritized based on their requirements coverage (de Almeida Falbo et al., 2014) or based on their failure discovery (Bicchierai et al., 2013). Traceability at a cross-project boundary can also help to trace vulnerable codes in APIs and provide information about them (Alqahtani et al., 2017).
2. An ontology-based approach (Feldmann et al., 2016) has been used to formulate the requirements of an industrial mechatronic system in early design phases and generate test cases to check whether the imposed requirements are fulfilled. Reasoning mechanisms have been applied to ensure consistency between requirements and test cases. An ontology-based scenario testing approach has been used to validate the consistency of a UML class diagram and the business requirements (Harmse et al., 2014).
3. All of the studies that have used semantic web technologies for test optimization techniques have proposed approaches for test selection. Only one study has proposed a test prioritization technique (De Campos H.S. et al., 2017). These studies have addressed different optimization criteria and goals. Sapna and Mohanty (2011) have used these technologies to select test cases that maximize requirement coverage. De Campos H.S. et al. (2017) have proposed a provenance ontology to maintain data about the results of test executions in regression testing. Inference in ontologies and querying it with SPARQL can help to select or prioritize test cases based on past results. This will help testers to reorder the execution sequence of the test cases, such that those test cases that might reveal a failure are executed first. Tonjes et al. (2015) have used semantic similarity between test cases to select the most diverse set of test cases.

7. Semantic web technologies in the software testing process

RQ3 explores semantic web technologies used in the software testing process. There are 33 studies that applied the W3C standard layers of semantic web stack (e.g., RDF, OWL, SPARQL, RIF,

Table 15
Semantic web technologies used in studies.

Study	Data exchange	Ontology language	Rule /Reason	Query	Tool	Technique
Tseng and Fan (2013); Fan and Wang (2012)	-	XML	-	-	-	-
Sinha et al. (2015)	XML	OWL	-	-	-	-
Tarasov et al. (2016)	-	OWL	-	-	-	-
Silva et al. (2017)	-	OWL	-	-	protégé	-
Tonjes et al. (2015)	XML	-	-	-	-	Annotation
Moser et al. (2010)	XML	-	-	-	-	-
Nguyen et al. (2009)	XML	-	OWL rules	-	protégé	-
iHajiabadi and Kahani (2011)	-	OWL	-	-	-	-
Li et al. (2011)	-	OWL	✓	-	-	-
Naseer and Rauf (2012)	-	OWL	✓	-	protégé	-
Rauf et al. (2010)	RDF	OWL	-	-	-	Annotation
Tao et al. (2019)	XML	-	-	-	-	-
Moitra et al. (2019)	XML	OWL	-	-	-	-
Ul Haq and Qamar (2019)	-	OWL	✓	✓	protégé	-
Mekruksavanich (2017)	-	OWL	SWRL	-	-	-
Silva et al. (2019)	XML	-	-	-	-	-
Mariani et al. (2014)	RDF	-	-	SPARQL	-	Web of data (LOD)
Szatmári et al. (2011)	-	OWL	✓	-	-	-
Li and Ma (2015a)	-	OWL	✓	-	-	-
Bai et al. (2011)	RDF	OWL	SWRL	-	protégé	-
Dalal et al. (2015)	RDF/XML	OWL	-	-	protégé	-
Li and Ma (2015b)	-	OWL	SWRL	-	-	-
Li and Zhang (2012)	-	✓	-	-	-	-
Guo et al. (2011)	RDF	OWL	-	-	protégé	-
Cai et al. (2009)	RDF/XML	OWL	RDQL	-	protégé	-
Guo et al. (2009)	-	OWL	-	-	protégé	-
de Almeida Falbo et al. (2014)	RDF	OWL	-	SPARQL	-	Annotation
Alqahtani et al. (2017)	RDF	OWL	SWRL	SPARQL	-	Linking to other repositories
Bicchierai et al. (2013)	RDF	OWL	SWRL	SPARQL	-	-
Feldmann et al. (2016)	-	OWL	-	-	-	-
Harmse et al. (2014)	-	OWL	-	-	protégé	-
Sapna and Mohanty (2011)	XML	OWL	-	-	protégé, Pellet, FACT+	-
De Campos H.S. et al. (2017)	XML	-	✓	SPARQL	protégé	-

✓: used but not specified.
-: not mentioned.

SWRL) to various activities of the software testing process. Table 15 presents the semantic web technologies used in the studies. Many of the studies did not mention all the semantic web technologies they have utilized (shown as '-'). For instance, rule defining, reasoning, and querying capabilities were not mentioned by most of the studies. Some studies indicated a capability but did not clearly specify the used technology (shown as '✓'). For instance, Li and Zhang (2012) did not specify the ontology language they have used.

Here is a list of our findings regarding RQ3:

1. Around 54% of the studies have specified their data exchange notation. Of those, 50% used RDF and 50% XML.
2. OWL is the dominant language for ontology definition among these studies (92%).
3. Around 39% of the studies have indicated rule definition or reasoning capabilities; 38% of those used SWRL for rule definition and reasoning.
4. Only 18% of the studies have specified their query capability, all used SPARQL.
5. Around 36% of the studies have specified their used semantic web tool. Almost all of them have used protégé, a well-known ontology development IDE in the Semantic Web community. Two other tools, Pellet and FACT++, which are inference engines providing reasoning capability have only been used in Sapna and Mohanty (2011).
6. Only three studies have specified using semantic annotation for augmenting data (Rauf et al., 2010; de Almeida Falbo et al., 2014; Tonjes et al., 2015). Ontologies provide the capability of annotating additional machine-processable information to existing data. Most of the knowledge bases used in these studies are domain-specific (see Table 21). Alqahtani et al. (2017) is the only study mentioned linking data to other repositories, and Mariani et al. (2014) is the only one used the Web of Data (i.e., LOD). However, an increasing number of datasets are getting published according to the principles of Linked Data, a huge source of knowledge in various domains is becoming available.

8. Test ontologies

RQ4 investigates the availability of high-quality test ontologies suitable for reuse. The first phase in the semantic web enabled testing is to develop the required ontologies (Paydar and Kahani, 2010). An ontology is an explicit and formal specification of a conceptualization of a domain of interest (Gruber et al., 1993). In other words, an ontology defines the basic concepts and relations that form a vocabulary of a specific domain along with the rules for defining extensions to the vocabulary. Test ontology defines the concepts of testing such as the tester, testing environment, available testing mechanisms, testing artifacts as well as testing techniques and test levels (Bhatia et al., 2016).

Designing and developing an ontology for the software testing domain has been the subject of some studies. These studies have used different sources and methods for ontology development which leads to different types of ontologies (i.e., reference ontologies and application ontologies). As was stated by Menzel (2003), reference ontologies (foundational ontologies) are rich, axiomatic theories. The focus of reference ontologies is to clarify the intended meanings of terms used in specific domains, while application ontologies (lightweight ontologies), by contrast, provide a minimal terminological structure to fit the needs of a specific community. Developing a reference (foundational) ontology (Menzel, 2003) which is accurate and comprehensive requires appropriate methods and knowledgeable experts. Once it is designed and developed, it could be reused in various research and applications.

Four studies have proposed a reference test ontology (Barbosa et al., 2006; Engström et al., 2017; Souza et al., 2017; Zhu and Huo, 2005). A separate section, C.4.1, is dedicated to summarizing these studies. Five studies proposed an application test ontology (Freitas and Vieira, 2014; Arnicans and Straujums, 2015; Bezerra et al., 2009; Anandaraj et al., 2011; Duarte et al., 2018). A separate section, C.4.2, is dedicated to summarizing the studies. Six studies that have proposed a concrete semantic web enabled test approach have also developed a test ontology to be applied in their approach (Vasanthapriyan et al., 2017; Cai et al., 2009; Sapna and Mohanty, 2011; Guo et al., 2011; Li and Ma, 2015b; Li and Zhang, 2012). All of these 15 test ontologies have been investigated, and their specifications are presented in Section 8.1.

One of the studies in this category did not exactly propose an ontology but a taxonomy in the area of software testing (Engström et al., 2017). Although, ontologies and taxonomies are different and taxonomies do not cover all that ontologies can represent (especially constraints and relationships), but the taxonomy proposed by Engström et al., 2017 can be considered as a high-quality one providing a detail hierarchical specification of the concepts in the software testing domain. Therefore, the authors decided to include this study despite differences between ontologies and taxonomies.

Some of the studies that have proposed ontologies in software domain, in general, have also represented concepts or relations related to the test domain (e.g., ONTO-ResAsset ontology proposed by Da Silva et al. (2014)). The ontology of software product quality attributes (SWQAs) have presented by Kayed et al. (2009) is one of these studies. This ontology is the result of several experiments to extract the main concepts for SWQAs. The main goal of this study is to identify common software quality attributes and extract the relevant concepts and relationships, along with their frequency of use. For instance, testability is one of the attributes specified in this study. Another ontology is proposed by García-Castro et al. (2010) to support the automated evaluation of the software. It presents an extensible model for representing software evaluations and evaluation campaigns. Test data is one of the main concepts in this ontology. Palacios et al. (2014) have developed an ontology based on the SABUMO ontology (López-Cuadrado et al., 2012). The SABUMO ontology allows experts to represent and share their knowledge with other experts utilizing semantic annotations and ratings. Ratings include rates provided by other users, taking into account the individual rating of each user. Palacios et al. have extended the SABUMO ontology with software testing concepts such as Test, Tester, and Test Element. Although these ontologies don't address the testing domain specifically, their related concepts and relations between them could help to design more applicable test ontologies. Some studies have defined test case sub-ontologies for the purpose of requirement traceability (Guo et al., 2009; de Almeida Falbo et al., 2014; Bicchierai et al., 2013). Test case concept and sub-concepts are related to concepts of software requirement ontology to provide traceability through the software development process. These sub-ontologies are not comprehensive from the software testing perspective. Therefore, they are not investigated in this SLR.

One of the characteristics of high-quality ontologies identified by d'Aquin and Gangemi (2011) is to reuse foundational ontologies. Therefore, we have investigated the reused test ontologies. As it is shown in Table 16, all of the proposed reference ontologies have been reused. However, the studies that reused these ontologies have the same authors as the study that introduced the ontology. It indicates that the studies that have proposed a concrete semantic web enabled software testing approach (investigated in Section 6) have not considered reusing existing test ontologies.

Table 16
Test ontologies that have been reused.

	Test ontology	Studies reused them
1	ROoST ontology by Souza et al. (2017)	de Almeida Falbo et al. (2014)
2	OntoTest ontology by Barbosa et al. (2006)	Nakagawa et al. (2011)
3	STOWS ontology by Zhu and Huo (2005)	Zhu (2006), Zhang and Zhu (2008), Zhu and Zhang (2012), Zhu and Zhang (2014)

As an example of reference test ontology reuse, a semantic document management platform to the requirements domain has been extended by de Almeida Falbo et al. (2014), and the conceptualization established by the Software Requirements Reference Ontology (SRRO) has been explored in order to support the Requirement Engineering Process. This study has extended the previous version of SRRO by including some properties of requirements and integrating this ontology with the Reference Ontology on Software Testing (ROoST) (Souza et al., 2017). In Nakagawa et al., 2009, 2011, OntoTest has been used to establish the reference architecture for the software testing domain. Studies introducing and improving STOWS were initiated by Huo et al. (2003); Zhu and Huo (2005) by proposing a test ontology and agent-based approach for testing web-based applications. The framework presented in Zhu and Zhang (2014) has its inception in Zhu (2006). A preliminary implementation and case study of the framework has been reported in Zhang and Zhu (2008). The STOWS ontology, which was proposed in Zhu and Zhang (2012) is based on the ontology developed in Huo et al. (2003); Zhu and Huo (2005).

8.1. Test ontology specifications

In this section, we investigate the proposed test ontologies from different perspectives. Our goal is to investigate the specifications of existing test ontologies to identify high-quality ones (Burton-Jones et al., 2005). For this purpose, we have collected and reported specifications of the proposed test ontologies in the studies. The source and specific domain of the proposed test ontologies are shown in Table 17. The most used languages, tools, and methods for developing these ontologies are presented in Table 18. The evaluation approaches of ontologies described in the studies are reported in Table 19. The proposed test ontologies are presented in Table 20 in order of publication date. There are 15 studies investigated in this section. Four reference ontologies and five application ontologies are introduced in the previous section. Six other studies that developed a test ontology for their ontology-based approaches are also investigated.

Assessing the quality of an ontology would help developers to reuse these test ontologies (Brank et al., 2005) in their applications. Ontology evaluation requires assessing a given ontology from different points of view. Various approaches and techniques for ontology evaluation have been proposed in the literature (Brank et al., 2005; Burton-Jones et al., 2005; Vrandečić, 2009). For instance, (Brank et al., 2005) identifies four categories of ontology evaluation approaches which have been commonly used: 1) comparing the ontology to a golden standard, 2) using the ontology in an application and evaluating its effect on the application's output, 3) comparing the ontology with another source of data, and 4) human assessment. As it is shown in Table 19, 57% of the studies

Table 17
Test ontology sources.

	Ontology	Ontology domain	Source
Reference ontologies	Souza et al. (ROoST) (Souza et al., 2017)	Software testing ontology	ISO/IEC/IEEE 29119-2013, SWEBOK (Bourque et al., 2014) IEEE 829–2008 (Software and Systems Engineering Standards Committee, 2008), SP-OPL (de Almeida Falbo et al., 2013)
	Barbosa et al. (OntoTest) (Barbosa et al., 2006)	Software testing ontology	ISO/IEC 12207 (Singh, 1996), SPO (de Almeida Falbo and Bertollo, 2005)
	Zhu et al. (STOWS) (Huo et al., 2003; Zhu and Huo, 2005)	Software testing ontology	Not mentioned
	Engström et al. (SERP-test) (Engström et al., 2017)	A taxonomy in the area of software testing	Literature reviews and interviews with practitioners and researchers
Application ontologies	Freitas and Vieira (Freitas and Vieira, 2014)	Performance testing ontology	SWEBOK (Abran et al., 2004), IEEE 829-1998 (S.E.T. Committee, 1998) IEEE 610.12-1990 (Radatz et al., 1990),
	Arnicans et al. (Arnicans et al., 2013; Arnicans and Straujums, 2015)	Software testing ontology	ISTQB-2010 (Van Veenendaal, 2010)
	Bezerra et al. (SWTO) (Bezerra et al., 2009)	Software testing ontology	SWEBOK (Abran et al., 2004)
	Anandraj et al. (Anandaraj et al., 2011)	Software testing ontology	Not mentioned
	Duarte et al. (OSDEF) (Duarte et al., 2018)	ontologies errors and failures	CMMI (Team, 2010), SWEBOK (Bourque et al., 2014) IEEE 1044 (ISDW Group and others, 2010), IEEE 1012 (IEEE Computer Society, 2016)
	Vasanthapriyan et al. (Vasanthapriyan et al., 2017)	Software testing ontology	IEEE 829–2008 (Software and Systems Engineering Standards Committee, 2008), ISTQB-2008 (Graham et al., 2008)
	Cai et al. (Cai et al., 2009)	Software testing ontology	SWEBOK (Abran et al., 2004), ISO 9126 (Iso, 2001)
	Sapna and Mohanty (Sapna and Mohanty, 2011)	Software testing ontology	SWEBOK (Abran et al., 2004)
	Guo et al.-1 (Guo et al., 2011)	Test case ontology for reuse	Not mentioned
	Li and Ma-2 (Li and Ma, 2015b)	Test case ontology for reuse	Not mentioned
	Li and Zhang (Li and Zhang, 2012)	Reusable test case ontology	Not mentioned

Table 18
Test ontology language, tool and development method.

	Ontology	Language	Tool	Development method
Reference ontologies	Souza et al. (ROoST) (Souza et al., 2017)	OWL	protégé	SABiO method (Falbo, 2014)
	Barbosa et al. (OntoTest) (Barbosa et al., 2006)	OWL, UML	Not mentioned	Capture and formalization(de Almeida Falbo et al., 1998)
	Zhu et al. (STOWS) (Huo et al., 2003; Zhu and Huo, 2005)	UML, XML	Not mentioned	Not mentioned
	Engström et al., 2017	Not mentioned	Not mentioned	Oreé et al.s (Bayona-Oré et al., 2014) method for taxonomy development
Application ontologies	Freitas and Vieira (2014)	OWL	protégé, Pellet	Noy and McGuinness (Noy and McGuinness, 2001)
	Arnicans et al. (2013); Arnicans and Straujums (2015)	OWL	protégé, OWLGrEd	ONTO6, Noy and McGuinness (Noy and McGuinness, 2001)
	Bezerra et al. (SWTO)(Bezerra et al., 2009)	OWL	protégé, Racer	Not mentioned
	Anandaraj et al. (2011)	OWL	protégé	Not mentioned
	Duarte et al. (OSDEF) (Duarte et al., 2018)	Not mentioned	Not mentioned	SABiO method (Falbo, 2014)
	Vasanthapriyan et al. (2017)	OWL	protégé	Grüninger and Fox methodology Grüninger and Fox, 1995
	Cai et al. (2009)	OWL	protégé	Skeletal (Uschold and Gruninger, 1996)
	Sapna and Mohanty (2011)	OWL	protégé	A self-defined method
	Guo et al. (2011)	OWL	protégé	Skeletal (Uschold and Gruninger, 1996)
	Li and Ma (2015b)	OWL	Not mentioned	Not mentioned
	Li and Zhang (2012)	Not mentioned	Not mentioned	Not mentioned

that have evaluated their proposed ontologies, have followed the method presented in Brank et al. (2005).

The important problem in investigating test ontologies is that there is not enough information in the associated studies for investigating them. Sometimes even the source and method of developing the ontology is not mentioned in the study. Not all of the discussed ontologies are publicly available for download and we can't get a public version of them.

Here is a list of our findings regarding RQ4:

1. Most of the proposed ontologies are general test ontologies (60%), three of them (20%) are test case ontologies developed for reuse purposes, only one of the ontologies (Freitas and Vieira, 2014) (6%) is specialized for a specific type of testing, i.e., performance testing, and one ontology (Duarte et al., 2018) (6%) is dedicated to software defects, errors, and failures.
2. More than half of the ontologies (66%) have mentioned specific sources for verifying their concepts and relations with standards or domain knowledge documents. ROoST (Souza et al., 2017) and OSDEF (Duarte et al., 2018), have used four different standards and glossaries. The most popular source of test ontology development is SWEBOK (Bourque et al., 2014; Abran et al., 2004) (60% of ontologies) followed by IEEE 829 standard (S.E.T. Committee, 1998; Software and Systems

Table 19
Test ontology evaluation.

Ontology	Evaluation method
Souza et al. (ROoST) (Souza et al., 2017)	Assessment by human approach to ontology evaluation based on the method presented in (Brank et al., 2005) Data-driven approach to ontology evaluation based on the method presented in (Brank et al., 2005) Ontology testing approach to ontology evaluation based on the method presented in (Vrandečić and Gangemi, 2006) Application-based approach to ontology evaluation based on the method presented in (Brank et al., 2005)
Barbosa et al. (OntoTest) (Barbosa et al., 2006)	Not mentioned
Zhu et al. (STOWS) (Huo et al., 2003; Zhu and Huo, 2005)	Not mentioned
Engström et al. (SERP-test) (Engström et al., 2017)	Evaluated by utilizing it in an industry–academia collaboration project (EASE)
(Freitas and Vieira, 2014)	Comparison with OntoTest(Barbosa et al., 2006) and SwTO(Bezerra et al., 2009) ontologies based on the method presented in (Brank et al., 2005)
(Arnicans et al., 2013; Arnicans and Straujums, 2015)	Evaluation by domain experts based on the method presented in Brank et al. (2005)
Bezerra et al. (SWTO)(Bezerra et al., 2009)	Quantitative (ontology structure) based on the method presented in Ning and Shihan (2006) and Qualitative (consistency, completeness, and conciseness) based on the method presented in (Uschold and Gruninger, 1996)
(Anandaraj et al., 2011)	Not mentioned
Duarte et al. (OSDEF) (Duarte et al., 2018)	Answering to competency questions suggested by SABiO (Falbo, 2014)
(Vasanthapriyan et al., 2017)	Internal consistency and inferences with FaCT+ and HermiT Evaluation with online ontology evaluator OOPS! Evaluation by ontology experts based on the method presented in Vrandečić (2009) Evaluation by software testing experts based on the method presented in Brank et al. (2005)
Cai et al. (2009)	Not mentioned
Sapna and Mohanty (2011)	Not mentioned
Guo et al. (2011)	Not mentioned
Li and Ma (2015b)	Not mentioned
Li and Zhang (2012)	Not mentioned

http://ease.cs.lth.se/. http://oops.linkeddata.es/

Table 20
Test ontology evolution temporal order.

Ontology	# of Ontology sources	Development methodology	# of Evaluation methods	Year of publication
Duarte et al. (OSDEF) (Duarte et al., 2018)	4	✓	1	2018
Engström et al., 2017	10	✓	1	2018
Souza et al. (ROoST) (Souza et al., 2017)	4	✓	4	2017
Vasanthapriyan et al. (2017)	2	✓	4	2017
Arnicans et al. (2013); Arnicans and Straujums (2015)	1	✓	1	2015
Li and Ma (2015b)	0	✗	0	2015
Freitas and Vieira (2014)	3	✓	1	2014
Li and Zhang (2012)	0	✗	0	2012
Anandaraj et al. (2011)	0	✗	0	2011
Sapna and Mohanty (2011)	1	✓	0	2011
Guo et al. (2011)	0	✓	0	2011
Cai et al. (2009)	2	✓	0	2009
Bezerra et al. (2009)	1	✗	2	2009
Barbosa et al. (2006)	2	✓	0	2006
Huo et al. (2003); Zhu and Huo (2005)	0	✗	0	2005

Engineering Standards Committee, 2008) (30%), and different versions of ISTQB (Graham et al., 2008; Van Veenendaal, 2010) (20%).

3. Most of the proposed ontologies have been defined in OWL language (73%) and using protégé (60%) tool.
4. Most of the proposed ontologies (66%), especially recent ones, have been developed based on a predefined methodology. It can be attributed to the maturity of the semantic web field and understanding the importance of quality in ontology development. There is not a common methodology, but three of the methodologies have been used by at least two studies including the Skeletal methodology (Uschold and Gruninger, 1996), the methodology introduced by Noy and McGuinness (2001), and SABiO methodology (Falbo, 2014).
5. Nearly half (46%) of the reference and application test ontologies have been evaluated. The ROoSt ontology(Souza et al., 2017) and the ontology developed by Vasanthapriyan et al. (2017) (both were proposed in 2017), each has been evaluated by four evaluation methods. Unfortunately, test ontologies that have been developed for a proposed ontology-based approach (Vasanthapriyan et al., 2017; Cai et al., 2009; Sapna and Mohanty, 2011; Guo et al., 2011; Li and Ma, 2015b; Li and Zhang, 2012) have not been evaluated.
6. Our investigation shows that high-quality test ontologies have been developed in recent years in both reference and application categories (Souza et al., 2017; Freitas and Vieira, 2014; Vasanthapriyan et al., 2017; Arnicans and Straujums, 2015; Duarte et al., 2018). Six of the above ontologies (Arnicans and Strau-

Table 21
Domain-specific approaches and their domain ontologies.

Domain	Study	Existing/Proposed domain ontology	Ontology development source
GUI-based systems	Rauf et al. (2010)	Ontology of GUI events	Set of events, GUI elements, Expert opinion
	Li et al. (2011)	Ontology of GUI elements	Source code of GUI
	Hajiabadi and Kahani (2011)	GUI elements	from web forms, System database
	Naseer and Rauf (2012)	Ontology of GUI events	Hierarchy of GUI components including File and Edit menus
	Mariani et al. (2014)	Existing ontologies in the Web of Data	✗
	Silva et al. (2017)	Ontology of GUI elements	Camaleon (Calvary et al., 2003), UsiXML (Limbourg et al., 2004), W3C MBUI (Pullmann, 2016)
	Li and Ma (2015a)	Ontology of GUI elements	Style description documents
	Silva et al. (2019)	Ontology of user behaviors	✗
Safety critical systems	Tseng and Fan (2013); Fan and Wang (2012)	Ontology of safety analysis report	Chapter 15 of the Standard Review Plan (SRP)
	Sinha et al. (2015)	Ontology of industrial cyber-physical systems	CESAR European project (Rajan and Wahl, 2013)
	Tarasov et al. (2016)	Ontology of avionics systems	Requirements specification
	Bicchierai et al. (2013)	Ontology of safety-critical systems	Structural, functional, and process perspectives
	Tao et al. (2019)	Ontology of Autonomous Emergency Braking (AEB)	EuroNcap protocol (Euro, 2013)
Agent-based systems	Moser et al. (2010)	Ontology of complex production automation systems	Simulation of Assembly Workshop (SAW) (Merdan et al., 2008b)
	Nguyen et al. (2009)	Ontology of agent interaction	Not mentioned
	Szatmári et al. (2011)	Ontology of agent context	Context model of agents
Context-aware applications	Tonjes et al. (2015)	Existing upper ontologies (e.g., the SUMO ontology)	✗
Manufacturing mechatronic systems	Feldmann et al. (2016)	Ontology of a mechatronic plant	System's requirements
Domain-specific operating systems	Bai et al. (2008)	Ontology of domain-specific operating system	Interface standard of real-time embedded OS

jums, 2015; Duarte et al., 2018; Engström et al., 2017; Freitas and Vieira, 2014; Souza et al., 2017; Vasanthapriyan et al., 2017) have used guidelines for all of the aspects presented in Table 20. If we look at the big picture of reference ontology development for the software testing domain, it seems like an evolutionary path. Each one of the above reference ontologies is an improved version of the previous one (with respect to their order based on the date of first publication). Further, each one tried to use the recent and more complete version of the same source and method for developing their ontology. For instance, OntoTest ontology is based on SPO (de Almeida Falbo and Bertollo, 2005), while ROoST reused SP-OPL (de Almeida Falbo et al., 2013). The development method of OntoTest is based on the capture and formalization introduced in de Almeida Falbo et al. (1998), while ROoST is developed based on the improved version of that method presented in Falbo (2014). Although these ontologies have few differences, especially in defining relationships, they are based on the same set of concepts.

9. Application domains

RQ5 investigates application domains in which semantic web enabled concrete test approaches have been applied. Some of these approaches are general purpose and have not been proposed for a specific domain (Dalal et al., 2015; Li and Ma, 2015b; Li and Zhang, 2012; Guo et al., 2011; Cai et al., 2009; Guo et al., 2009; de Almeida Falbo et al., 2014; Alqahtani et al., 2017; Harmse et al., 2014; Sapna and Mohanty, 2011; De Campos H.S. et al., 2017; Moitra et al., 2019; Ul Haq and Qamar, 2019; Mekruksavanich, 2017). For other studies that proposed a domain-specific approach, their underlying domain and the domain ontology that has been used by them are presented in Table 21.

Here is a list of our findings regarding RQ5:

1. Most of the approaches (42%) have been applied to GUI-based systems followed by safety-critical systems (26%) and agent-based systems (16%). It is a surprising observation that ontology has been used in reliable software domains such as safety-critical systems. This can be attributed to the fact that formal requirements specification is very common in these domains.
2. Studies in GUI-based, safety-critical, and agent-based systems have used different sources and developed ontologies specifically for their SUT. It seems that in such domains a general ontology can't be applied for all systems. Only one study (Mariani et al., 2014) in these domains reused existing ontologies.
3. There are three studies that proposed their approaches in other domains (context-aware applications, manufacturing mechatronic systems, and operating systems). Only one of these studies reused existing ontologies (Tonjes et al., 2015). It seems that reusing domain ontologies is neglected in most of the studies.

10. Impact of semantic web technologies on software testing

RQ6 investigates the impact of semantic web technologies on software testing. In this section, we explore improvements in the software testing domain due to utilizing semantic web technologies. The purpose is to find out which test quality criteria are improved by using these technologies. The 33 studies which proposed

Table 22
Improvements provided by semantic web technologies in software testing.

Activity	Study	Improvements									
		Coverage	Fault detection	Cost	Automation	Reuse	Maintenance	Knowledge representation	Knowledge sharing	Knowledge discovery	Other
Test generation	Tseng and Fan (2013)	Exp	-	-	Exp	-	-	Imp	-	-	Adequacy of tests
	Sinha et al. (2015)	-	-	-	Exp	-	-	Imp	-	-	-
	Tarasov et al. (2016)	-	Imp	-	Exp	-	-	Exp	-	Exp	Correctness of tests
	Silva et al. (2017)	-	-	Imp	Imp	Imp	-	Exp	-	-	Teamwork
	Tonjes et al. (2015)	Imp	Exp	Imp	Imp	-	-	Imp	-	-	Higher level of test description
	Moser et al. (2010)	Exp	-	Exp	Exp	Imp	-	Imp	-	-	-
	Nguyen et al. (2009)	Exp	Exp	-	Exp	Imp	-	Imp	-	Imp	Wider exploration of the input space
	iHajiabadi and Kahani (2011)	Exp	-	-	Exp	-	-	Exp	-	-	-
	Li et al. (2011)	Exp	-	-	Exp	-	-	Imp	-	Imp	-
	Naseer and Rauf (2012)	Exp	Imp	-	Exp	Imp	-	Imp	-	Imp	Test efficiency (coverage/test cases)
	Rauf et al. (2010)	-	-	-	Imp	-	-	Imp	-	-	-
	Tao et al. (2019)	-	-	-	Exp	-	-	Exp	-	Imp	-
	Moitra et al. (2019)	Exp	-	Imp	Exp	-	-	Exp	-	Imp	-
	Ul Haq and Qamar (2019)	-	-	Imp	Imp	-	-	Imp	-	Imp	-
	Mekruksavanich (2017)	-	-	-	Exp	-	-	Imp	-	Exp	Precision, false positive
	Silva et al. (2019)	Imp	-	-	Exp	Imp	Imp	Imp	-	-	-
Test data generation	Mariani et al. (2014)	Exp	Exp	-	Exp	-	-	Imp	-	Exp	-
	Szatmári et al. (2011)	Imp	-	Imp	Imp	-	-	Imp	-	-	-
	Li and Ma (2015a)	-	-	Exp	Exp	-	-	Imp	-	Imp	-
Oracle development	Bai et al. (2011)	-	-	Exp	Exp	Imp	-	Imp	Imp	Exp	correctness of test oracles
Test reuse	Dalal et al. (2015)	-	-	-	-	Imp	-	Imp	-	Imp	-
	Li and Ma (2015b)	-	-	Exp	Imp	Exp	-	Imp	-	Exp	Validity of tests
	Li and Zhang (2012)	-	-	Exp	-	Exp	-	Imp	Imp	Imp	Teamwork, Efficiency (test cases/effort)
	Guo et al. (2011)	-	-	-	-	Imp	-	Imp	-	-	-
	Cai et al. (2009)	-	-	-	-	Imp	-	Imp	Imp	Imp	-
Traceability management	Guo et al. (2009)	-	-	-	Imp	-	Imp	Imp	-	-	-
	de Almeida Falbo et al. (2014)	-	-	-	Imp	-	Imp	Imp	-	Imp	Documentation
	Alqahtani et al. (2017)	-	-	-	Imp	-	Exp	Imp	Imp	Exp	Extensibility
	Bicchierai et al. (2013)	-	Imp	-	-	Imp	-	Imp	-	Imp	Documentation
Consistency checking	Feldmann et al. (2016)	-	-	-	Imp	-	-	Imp	-	Imp	Test management
	Harmse et al. (2014)	-	-	-	-	-	Imp	Imp	-	Imp	-
Test optimization	Sapna and Mohanty (2011)	Imp	-	-	-	-	-	Imp	-	Imp	Test management
	De Campos H.S. et al. (2017)	-	-	-	Imp	-	-	Imp	-	Imp	Test management

Exp: Explicit improvement.
Imp: Implicit improvement.

semantic web enabled concrete testing approaches have been investigated, and their possible improvements have been collected. Based on how explicitly the improvements are mentioned in the studies, and hence how reliable they are, we have distinguished two levels of improvements:

- Exp: Explicit Improvements that have been explicitly demonstrated the contributions of the semantic web technologies by evaluating the proposed approach and reporting the results, demonstrating that the use of semantic web technologies has improved some desired quality attribute or criterion. In some works, the evaluation also includes a comparison with other approaches. These improvements are the most confident since they are evaluated and explicitly reported by the corresponding authors.
- Imp: Implicit improvements are those that have been only mentioned by the authors in the study publication, but they have not been evaluated independently. In these cases, the authors didn't provide any evaluation or comparison results of these claimed improvements. In some cases, the authors mentioned that after applying the proposed approach, some criterion had been improved according to the subjective judgment from users (e.g., test engineers). As these improvements have not been evaluated explicitly, they have a lower confidence level, compared to explicit improvements.

Various activities in software test generation have been subject to the above-mentioned improvements. The results of analyzing the studied studies with regards to these aspects are shown in Table 22.

Table 23
List of studies included in the review along with their quality scores.

Study	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Total Score	Qual.(%)
Tseng and Fan (2013)	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.5	0.5	0.5	0.5	7	63.6
Sinha et al. (2015)	1.0	0.0	0.5	0.5	0.5	1.0	0.0	0.5	0.5	0.5	0.5	5.5	50
Tarasov et al. (2016)	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.5	0.5	1.0	0.5	7.5	68.1
Silva et al. (2017)	1.0	1.0	1.0	1.0	0.5	1.0	0.0	0.5	0.5	1.0	1.0	8.5	77.1
Tonjes et al. (2015)	1.0	1.0	0.5	0.5	0.5	0.0	0.0	0.5	0.5	1.0	0.5	6	54.5
Moser et al. (2010)	1.0	1.0	0.5	0.5	0.5	0.0	1.0	0.5	0.5	0.5	0.5	6.5	59
Nguyen et al. (2009)	1.0	1.0	0.5	0.5	0.5	0.0	0.0	0.5	0.0	0.5	0.5	5.5	50
Hajiabadi and Kahani (2011)	1.0	1.0	0.5	1.0	1.0	0.0	1.0	0.5	0.5	0.5	0.5	7.5	68.1
Li et al. (2011)	1.0	1.0	1.0	1.0	0.5	0.0	1.0	1.0	0.5	1.0	1.0	9	81.8
Naseer and Rauf (2012)	0.5	1.0	0.5	1.0	0.5	0.0	1.0	0.0	0.0	0.5	1.0	6	54.5
Rauf et al. (2010)	1.0	0.0	1.0	0.5	0.5	0.0	0.0	0.5	0.5	0.5	1.0	5.5	50
Tao et al. (2019)	1.0	1.0	0.5	0.5	1.0	0.0	1.0	0.5	0.5	1.0	0.5	7.5	68.1
Moitra et al. (2019)	1.0	1.0	1.0	0.5	1.0	1.0	1.0	0.5	0.5	1.0	1.0	9.5	86.3
Ul Haq and Qamar (2019)	1.0	1.0	1.0	0.5	0.5	0.0	0.0	0.0	0.5	0.5	1.0	6	54.5
Mekruksavanich (2017)	1.0	1.0	1.0	0.5	1.0	0.0	0.0	1.0	0.5	0.5	1.0	7.5	68.1
Silva et al. (2019)	1.0	1.0	1.0	0.0	0.5	0.0	0.0	1.0	1.0	1.0	1.0	7.5	68.1
Mariani et al. (2014)	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5	1.0	0.5	1.0	10	90.9
Szatmári et al. (2011)	1.0	1.0	0.5	0.5	0.5	0.0	0.0	0.5	0.5	0.5	0.5	5.5	50
Li and Ma (2015a)	1.0	1.0	0.5	0.5	0.5	0.0	0.0	0.5	0.5	0.5	1.0	6	54.5
Bai et al. (2011)	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	0.5	0.5	9	81.8
Dalal et al. (2015)	1.0	1.0	0.5	0.5	0.5	0.0	0.0	0.5	0.0	0.5	1.0	5.5	50
Li and Ma (2015b)	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	0.5	0.5	1.0	9	81.8
Li and Zhang (2012)	0.5	1.0	1.0	0.5	0.5	0.0	0.0	0.0	0.5	0.5	1.0	5.5	50
Guo et al. (2011)	1.0	1.0	1.0	0.5	0.5	0.0	0.0	0.0	0.0	0.5	1.0	5.5	50
Cai et al. (2009)	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.5	0.0	0.5	1.0	7	63.6
Guo et al. (2009)	1.0	1.0	0.5	1.0	0.5	0.0	1.0	1.0	0.5	0.5	1.0	8	72.7
de Almeida Falbo et al. (2014)	1.0	1.0	0.5	0.5	0.5	1.0	0.0	0.0	0.5	0.5	1.0	6.5	59
Alqahtani et al. (2017)	1.0	1.0	1.0	1.0	0.5	1.0	1.0	1.0	1.0	1.0	1.0	10.5	95.4
Bicchierai et al. (2013)	1.0	1.0	1.0	0.5	0.5	1.0	0.0	0.5	0.5	1.0	0.5	7.5	68.1
Feldmann et al. (2016)	1.0	1.0	1.0	0.5	1.0	0.0	1.0	1.0	0.5	1.0	0.5	8.5	77.1
Harmse et al. (2014)	1.0	1.0	0.5	0.5	1.0	0.0	0.0	0.0	0.5	0.5	1.0	6	54.5
Sapna and Mohanty (2011)	0.5	1.0	0.5	0.5	1.0	0.0	0.0	0.0	0.5	0.5	1.0	5.5	50
De Campos H.S. et al. (2017)	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.5	1.0	1.0	8.5	77.1

Here is a list of our findings regarding RQ6:

1. More than half of the studies focused on test generation activity have reported improvements in test coverage. Fault detection rate and cost are two other criteria that have been reported by many studies. Since these criteria are generally popular for test quality measurement, therefore, one can conclude that semantic web technologies can improve the quality of the generated tests. More than half of the studies (56%) that have reported coverage improvements and half of the studies (50%) with fault detection improvements have evaluated the results explicitly.
2. All of the studies have reported improved knowledge representation in their work (15% explicit, 85% implicit).Improvements in automation have been reported in most of the studies (78%), and more than half of them (57%) have reported this explicitly. Some of the studies reported that using semantic web technologies may increase the initial setup cost but decrease the overall cost in the long run (Moser et al., 2010). Knowledge discovery improvements have also been reported in many of the studies (66%), where 27% of them evaluated it explicitly.
3. Improvements in reusability and cost have been reported by less than half of the studies. About 36% of the studies have reported improvements in reusability, but only two of them evaluated this explicitly. About 30% of the studies have reported improvements in cost, and half of them evaluated it explicitly (50%). This cost includes the time or the effort needed for testing. Reported improvements in maintenance and knowledge sharing are less than others (15% and 12% respectively), which are mostly evaluated implicitly.
4. There are other improvements explicitly reported by studies such as efficiency, test management, and documentation. A few studies have reported these improvements, they are presented under ”Other” column in Table 22.

11. Quality assessment results

In this section, we depict the quality assessment results for studies which proposed a semantic web enabled testing approach. The quality assessment of the selected studies is useful to increase the accuracy of the data extraction results. This evaluation helped to determine the validity of the inferences proffered and in ascertaining the credibility and coherent synthesis of results.

The quality assessment results are shown in Table 23, according to the assessment questions described in Table 5. The scores of all studies are no less than 50% and the average score is 7.16. The overall quality of the selected studies is acceptable. Taken together, these 11 criteria provided a measure of the extent to which we could be confident that a particular study's findings could make a valuable contribution to this review.

12. Discussion and future directions

We will discuss the findings and future directions from two points of view. First, the potential value of semantic web technologies to testing is discussed in Section 12.1. Then we will investigate the difficulties that hinder its practical applications and how to overcome the difficulties in Section 12.2. Research directions for semantic web enabled software testing are presented in Section 12.3.

12.1. Potential value of semantic web technologies to software testing

From the point of view of test management, it is crucial to be able to store, retrieve, and analyze different kinds of data related to the test process. These include: detailed data about the test target (e.g., unit, module, system); the test case design technique (e.g., boundary value analysis, random testing techniques, etc.); the tools used in different phases of the test, along with the settings used for each tool; the resources involved in the test (both human and software/hardware resources); the test outcome; the test execution timestamp.

The semantic web technologies provide a flexible data representation layer with a main benefit that the semantics of data and its relation to other data is explicitly expressed in a machine-processable format. This makes the semantic web technologies a promising enabler for better test management in a software development organization, improving activities like test tractability analysis and test reuse which are crucial to regression testing.

Another potential value of semantic web technologies to software testing is that they can provide a more powerful mechanism for sharing test assets that are less application-dependent and hence more reusable. For instance, it would be very interesting for a tester who is testing a web page against SQL injection or Cross-Site Scripting (XSS) vulnerabilities, to be able to retrieve the test cases previously developed by a security testing expert for testing the same vulnerabilities. As another example, it would be possible for a test expert to share his carefully crafted test cases for testing a unit that converts Gregorian dates to Chinese, regardless of the fact that the conversion unit is part of a financial system or an online e-learning system. The semantic web technologies help establish the required infrastructure for sharing such self-describing test assets. This can also provide potentials for crowdsourcing some test activities, e.g., test input generation or test output verification.

12.2. Difficulties in applying semantic web technologies to software testing

While interesting potentials can be considered for application of the semantic web technologies in software testing, there are some impediments that might describe why the realization of those potentials is less evident. We believe that a reason is the lack of tool support for developers and testers. Actually, it is needed that semantic web enabled techniques are realized not in terms of independent research prototypes, but as plugins which are integrated with full-featured development IDEs or test tools. For instance, translation of the test case information into an ontology-based representation should be streamlined with the regular activities that a tester performs and inside the tools that he/she uses. Otherwise, it is conceived as an extra activity added to the testers responsibilities that might reduce his focus without immediate results.

If the semantic web technologies are promised to be able to provide a mechanism for sharing data about test assets, a very first requirement of this is the availability of well-known, expressive and rich ontologies for describing different details about the test assets. As this study demonstrates, the existing reference ontologies for software testing are not yet at the level to appropriately address this requirement. This is evident from the fact that many researchers have developed their own ontologies instead of reusing existing ones, whether because those ontologies have not had the required characteristics or they have been yet far from being well-known or credited among the researchers.

Considering the fact that software systems are rapidly growing both in terms of size and complexity and the diversity of the technologies involved, it is essential for the semantic web enabled techniques to be built-in throughout the whole development and maintenance lifecycle, not only for a small part of the process. For instance, the semantic web enabled tools should be integrated with the continuous development and continuous integration infrastructure. If they are considered to be available only during the early unit testing steps, their added value might be hindered by the overhead of their application in the first place. The current state of research does not provide evidence for the presence of such support.

We believe that alleviating the problems discussed above mainly requires a shift in the point of view of the software testing community, instead of specific progress or innovation on the technical side. Actually, we do believe that the problems are attributed to the lack of knowledge and mutual collaboration between the semantic web and the software testing communities. For instance, the development of high-quality reference ontologies for the software testing domain is not actually hindered by the technical difficulties, but by the lack of awareness about its importance and the benefits it provides.

12.3. Research directions for semantic web enabled software testing

In order to improve the realization of the potentials of semantic web enabled software testing, a research direction is to investigate a framework for semantic test management that effectively integrates different data of the test process and represents this data using the semantic web data model. This framework aims to support test engineers in their daily activities by answering their information needs. For instance, due to the importance and implications of effective regression testing, different algorithms and tools have been proposed in the literature for test case selection, test case prioritization, and test suite minimization. A semantic web enabled framework is capable of employing ontologies to create a semantic layer over data generated by different algorithms and tools, and integrate this data with other test-related data. Through such a semantic layer, it would be possible for the test engineers to access, analyze, and retrieve the underlying data in a flexible way through semantic web query languages like SPARQL.

A main research question in this respect is how to provide this semantic representation so that it is 1) flexible enough to cope with the continuous changes in the software under test, 2) maintainable to be easy to understand and update, and 3) scalable so that it does not introduce new bottlenecks as the software evolves.

Another research direction is to investigate how it is possible to use ontological representation for implementation-independent test specification. This means that the tests are specified in a high level of abstraction, regardless of the specific implementation details, so that it is possible to share these specifications to be adapted to concrete test specifications for a specific implementation. This realizes test reuse over different implementations. For instance, an abstract test script defined for testing the login functionality can be used to test that functionality on different web applications. The use of ontologies is expected to enable annotation of the abstract test specifications with the required semantic so that the adaptation phase can be automated to some extent.

Considering the integration of semantic web enabled testing with the software development process, a main research agenda is to identify the required interface and integration points between these two domains. In other words, the question is that in order to support different types and levels of testing during different phases of software development, which parts of the development process need to be under the umbrella of the semantic representation? What is the granularity of the semantic layer that should be created over data? For instance, is it necessary to include information of each commit in the semantic representation? Addressing this research question provides a clear understanding of the requirements for realizing this integration.

13. Threats to validity

In this section, we describe validity concerns in software engineering studies. These concerns must be taken into account in order to generalize the results of the SLR performed in this work. This section is organized by classification of the threats to validity in four class of: Internal, External, Construct, and Conclusion categories (Wohlin et al., 2000).

Internal validity is the extent to which the design and conduct of the study are likely to prevent systematic error (Kitchenham and Charters, 2007). We are here concerned about the study selection process and data extraction. Some subjective decisions may have occurred during these two phases since some studies did not provide a clear description or proper objectives and results. This leads to difficulty in the objective application of the inclusion/exclusion criteria. Also, some required data were missing in a few primary studies that may pose a threat to internal validity. In order to minimize mistakes in the selection and extraction process, the selection process was performed iteratively, and the data extraction was realized collaboratively by reviewers to mitigate the threats due to personal bias. It is also worth noting that the first author is a Ph.D. student in software engineering with a background on the Semantic Web, and the other two authors are experienced researchers with expertise in software engineering.

External validity is the extent to which the effects observed in the study are applicable outside of the study (Kitchenham and Charters, 2007). The generalizability of the SLR outcomes is related to the degree the studies are representative for the review topic. We provided the coverage and representativeness of retrieved studies using automatic database search and references scan. Also, this SLR was focused on research questions and quality assessment items to mitigate the risk of generalizability of the results.

Construct validity is provided assuring that the conduction of this SLR matches its objectives. The search process and search terms as the main concerns. The search terms used in this review were obtained from a nine-step strategy considering research questions and well-known sources like SWEBOK. Then, they were tried against a list of already known primary studies and iteratively adjusted. However, the completeness and the comprehensiveness of the terms used are not guaranteed. To reduce this risk, we used forward and backward snowballing. Additionally, there were articles not in English which have been filtered in the exclusion process. This may present a threat to the construct validity. A complementary manual search was not performed in the SLR due to the fact there are no conferences and journals specifically focused on the semantic web enabled software testing. Although five well known digital libraries were searched for relevant studies, other sources searched with different keywords may return relevant studies that have not been taken into consideration in this work.

As a threat to conclusion validity, some studies may have been excluded in this review that should have been included. To mitigate this threat, the selection process and the inclusion and exclusion criteria were carefully designed and discussed by authors to minimize the risk of exclusion of relevant studies.

14. Conclusion

The systematic literature review reported in this work was carried out to acquire knowledge on the state of the art in the area of semantic web enabled software testing. Our goal was to improve understanding of how semantic web technologies support software testing as well as to identify evidence of its use in this field. Finally, 52 studies were included that addressed three main research questions of this review. Ten studies that addressed the theoretical foundations of semantic web enabled software testing have been investigated, and potential values of semantic web technologies to software testing were identified. Nine test ontologies have been investigated with their specifications. Additionally, 33 studies that proposed concrete semantic web enabled testing approaches have been reviewed. The results presented in this systematic review can be useful to the software testing community since it gathers evidence from the studies included in the review regarding the use of semantic web technologies in software testing. As future work, we suggest to further investigate some of the research directions presented in this review, especially semantic web enabled approaches that have not received sufficient attention from the studies.

Appendix A. Data extraction form

See Table A.24.

Table A.24
Data extraction form.

No.	Study data	Description	Relevant RQ
1	Study identifier	Unique id for the study	Study overview
2	Application context	Industrial, academic	Study overview
3	Article source		Study overview
4	Type of article	Journal, conference, workshop, book chapter	Study overview
5	Authors, Year, Title, Country		Study overview
6	Date of data extraction		Study overview
7	Theoretical foundations	What are the theoretical foundations of the semantic web enabled software test generation?	RQ1
8	Concrete approaches	What concrete approaches are proposed and what are their specifications?	RQ2
8.1	Test activities	Which test activities are supported?	RQ2
8.2	Test levels	Which level of testing are supported? (unit, integration, system, acceptance)	RQ2
8.3	Type of requirements	Which type of requirements is addressed? (functional and/or nonfunctional)	RQ2
8.4	Test method	Which test generation methods are supported? (model-based, scenario-based, ...)	RQ2
9	Semantic web technologies	Which semantic web technologies and tools have been used in the proposed concrete approaches?	RQ3
10	Test ontologies	What test ontologies are designed and developed?	RQ4
10.1	Ontology specifications	What are the specifications of these test ontologies?	RQ4
10.2	Reused test ontologies	Which ontologies have been reused?	RQ4
11	Domain ontologies	What domain ontologies are developed in the proposed concrete approaches?	RQ5
12	Improvements	What are the improvements provided by the proposed concrete approaches?	RQ6

Appendix B. Publication sources

See Table B.25.

Appendix C. Summarizing the studies

C1. High level solutions

C1.1. Semantic web enabled test process

Nasser et al. (2010), proposed a framework that considers test objectives and customized coverage criteria. The framework

Table B.25
Distribution of studies over publication sources .

Publication source	Type	Studies	Count
Journal of Systems and Software	Journal	(Nakagawa et al., 2011)	1
Information and Software Technology	Journal	(Tseng and Fan, 2013)	1
Software Quality Journal	Journal	(Engström et al., 2017)	1
Requirements Engineering	Journal	(Moitra et al., 2019)	1
Computers & Security	Journal	(Eckhart et al., 2019)	1
Applied Ontology	Journal	(Souza et al., 2017)	1
International Journal of Computer Applications	Journal	(Li et al., 2011)	1
Computer Science and Information Systems (ComSIS)	Journal	(Palacios et al., 2014)	1
International Journal of Communication, Computation and Innovation (IJCCI)	Journal	(Anandaraj et al., 2011)	1
International Conference on Software Engineering and Knowledge Engineering (SEKE)	Conference	(Nasser et al., 2010; Barbosa et al., 2006) (Moser et al., 2010; Vasanthapriyan et al., 2017)	4
International Conference on Software Testing, Verification and Validation (ICST)	Conference	(Alqahtani et al., 2017; Tao et al., 2019)	2
International Symposium on Software Testing and Analysis (ISSTA)	Conference	(Mariani et al., 2014)	1
IEEE Conference on Computer Software and Applications Conference (COMPSAC)	Conference	(Bai et al., 2011)	1
International Conference on Information Technology and Computer Science (ITCS)	Conference	(Liu et al., 2009)	1
International Conference on Engineering of Complex Computer Systems (ICECCS)	Conference	(Sinha et al., 2015)	1
International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)	Conference	(Freitas and Vieira, 2014)	1
Agent-Oriented Software Engineering (AOSE)	Conference	(Nguyen et al., 2009)	1
IEEE Conference on Open Systems (ICOS)	Conference	(Hajiabadi and Kahani, 2011)	1
International Multitopic Conference (INMIC)	Conference	(Naseer and Rauf, 2012)	1
International Conference on Educational and Information Technology (ICEIT)	Conference	(Ul Haq and Qamar, 2019)	1
International Conference on Informatics in Control, Automation and Robotics (ICINCO)	Conference	(Szatmári et al., 2011)	1
ACS/IEEE International Conference on Computer Systems and Applications (AICCSA)	Conference	(Bueno et al., 2018)	1
International Conference on Software Process Improvement and Capability Determination(SPICE)	Conference	(Çiflikli and Co\cskunçay, 2018)	1
International Conference on Conceptual Modeling (ER)	Conference	(Duarte et al., 2018)	1
International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP)	Conference	(Mekruksavanich, 2017)	1
International Conference on Computational Science and Applications (ICCSA)	Conference	(Silva et al., 2019)	1
Intelligent Computing, Communication and Devices (ICCD)	Conference	(Dalal et al., 2015)	1
International Conference on Artificial Intelligence and Industrial Engineering (AIIE)	Conference	(Li and Ma, 2015b)	1
International Conference on Internet Computing for Science and Engineering (ICICSE)	Conference	(Li and Zhang, 2012)	1
International Conference on Mechatronic Science, Electric Engineering and Computer (MEC)	Conference	(Guo et al., 2011)	1
IEEE Pacific Rim International Symposium on Dependable Computing (PRDC)	Conference	(Cai et al., 2009)	1
International Symposium on Knowledge Acquisition and Modeling (KAM)	Conference	(Guo et al., 2009)	1
International Conference on Emerging Technologies (ICET)	Conference	(Rauf et al., 2010)	1
International Conference on Formal Ontology in Information Systems (FOIS)	Conference	(Harmse et al., 2014)	1
International Conference on Information Intelligence, Systems, Technology and Management (ICISTM)	Conference	(Sapna and Mohanty, 2011)	1
Brazilian Symposium on Systematic and Automated Software Testing (SAST)	Conference	(De Campos H.S. et al., 2017)	1
International Conference on Reliable Software Technologies	Conference	(Bicchierai et al., 2013)	1
OWL: Experiences and Directions–Reasoner Evaluation	Conference	(Tarasov et al., 2016)	1
The International Conference on Semantic Computing (ICSC)	Conference	(Silva et al., 2017)	1
Vehicular Technology Conference (VTC Fall)	Conference	(Tonjes et al., 2015)	1
International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)	Conference	(Li and Ma, 2015a)	1
Semantic Web Technologies for Intelligent Engineering Applications	Book chapter	(Feldmann et al., 2016)	1
Innovations and Advances in Computing, Informatics, Systems Sciences, Networking and Engineering	Book chapter	(Arnicans and Straujums, 2015)	1
Software Evolution with UML and XML	Book chapter	(Zhu and Huo, 2005)	1
Novel Algorithms and Techniques in Telecommunications and Networking	Book chapter	(Paydar and Kahani, 2010)	1
Advances in Information Systems Development	Book chapter	(Hilera et al., 2018)	1
International Workshop on Ontology, Conceptualization and Epistemology for Information Systems, Software Engineering and Service Science	Workshop	(Bezerra et al., 2009)	1
Workshop on Requirements Engineering (WER)	Workshop	(de Almeida Falbo et al., 2014)	1

consists of four phases which generate selected executable test cases from test objectives. Testers use ontologies and rules to specify what needs to be tested which includes: behavioral model specifications, expert knowledge, and coverage criteria rules (Nasser et al., 2009).

Paydar and Kahani (2010) presented a theoretical roadmap. They divided the process into two phases. The first one is developing the required ontologies which consist of testing and domain ontologies. These two types of ontologies should capture an appropriate level of required knowledge to perform the testing process. The second phase is developing intelligent methods and procedures that utilize the available ontologies to automate the testing process. They also presented some examples on ontology-based web application testing.

Nakagawa et al., 2011, investigated the impact of using ontologies as a central element for building reference architectures. They illustrate the idea in software testing domain using ProSA-RA(Nakagawa et al., 2011) as a process to develop, describe and evaluate reference architectures and OntoTest (Barbosa et al., 2006) ontology. A reference architecture, named RefTEST (Reference Architecture for Testing Tools), was designed based on the architectural requirements. Although, only the module view of RefTEST (Nakagawa et al., 2009) is presented in the study and there is no evaluation on that, the authors claimed that the results obtained provide a preliminary evidence on the practical use of ontologies. They also propose to use ontologies even if the ontology is not complete. They believe that its use is relevant, since at least the main and more important concepts are considered in the ontology. Thus, they believe that the same results could be achieved for other domains that have mature and complete ontologies.

Bueno et al. (2018), proposed STEP- ONE, the Security TEst Process supported by ONtology Environment. The foundations of this process is based on security testing standards and is aimed at evaluating the security characteristics of IT systems. The security test process includes a list of ordered Sub-Processes (SP). There are seven predefined SPs in this study proposed by authors. Each SP is composed of Activities of Test (AT) which are described as tasks and has a specific goal with respect to the security assessment objective. The STEP process can be applied to IT systems that are already developed or Systems that still will be developed. They also proposed a Security Testing Process Ontology (STPO) to define the main concepts of the domain explicitly. This ontology covers concepts in both security domain and test domains.

Çiflikli and Co\cskunçay (2018) presented a test process assessment infrastructure based on ontologies. This study used TMMi as the process reference model and tried to track conformance of test process to it. The authors also proposed a TMMi Assessment Ontology which is composed on three separate ontologies.

Eckhart et al. (2019), proposed a framework including a default testing process model. This framework supports the semi-automated security analysis of software testing process in organizations. Users can adapt the default process to their software testing environment. The proposed framework is based on the VDI/VDE 2182 guideline and utilizes ontologies for representing background knowledge, including, e.g., data flows, threats, assets, entities. The default testing process is modeled based on best practices and aligned with the ISO/IEC/IEEE 29,119 series of software testing standards.

C1.2. Knowledge management systems based on semantic web technologies

Vasanthapriyan et al. (2017), proposed a KMS for sharing software testing domain knowledge. They also developed a software testing ontology for testers to annotate their testing knowledge in proposed KMS. Specifications of the proposed ontology will be investigated in following sections 8.1.

Palacios et al. (2014), presented SABUMO-dTest, a KMS for collaboration between contractors and testers which help them test a software remotely. Each tester is related to a number of concepts according to their expertise and interests that help contractors find the best processes and testers. Testers follow the workflow of the testing processes defined by the developer, execute the testing process and provide feedback on test results and rating test process. The rating engine allows the rating of testers, contractors and testing processes. It also automatically updates the rating of each element according to the evaluation of the users involved. The architecture of SABUMO-dTest is based on a well-established framework which allows the semantic definition of business processes (López-Cuadrado et al., 2012).

Liu et al. (2009), have used ontology for knowledge representation of software testing concepts and relations in their proposed software testing KMS. They proposed this KMS with the purpose of learning practices in software testing domain by sharing and retrieving test knowledge.

Hilera et al., 2018, proposed a process that can automatically combinate a set of test reports obtained from different testing tools. This study also proposed applying different standards on Web accessibility based on reports about evaluating accessibility of Web applications. The proposed knowledge base includes description of the accessibility standards, relationship between them, and rules that apply when evaluating the website. They also developed a software prototype for combining multiple accessibility evaluation reports of the same website based on the proposed knowledge base. Reasoners are used to infer new knowledge based on defined rules.

C2. Concrete approaches

C2.1. Test generation approaches

Semantic web technologies have been used in testing safety critical systems. Tseng and Fan (2013) and Fan and Wang (2012) developed a systematic validation test schema that is effective for large nuclear industry systems. The user's original safety needs are addressed by Preliminary Safety Analysis Report (PSAR), which is written in natural language. They extracted testing related concepts and relations from the PSAR using the Standard Review Plan (SRP) (of Nuclear Reactor Regulation, 1981), which is the regulatory guide for reviewing Safety Analysis Reports. Then, they developed a domain ontology using the method proposed by Gruber (1995). The ontology is used to tag (annotate) major concepts and relations in a PSAR. Test scenarios described in natural language and tagged based on the proposed ontology are converted to UML sequence diagrams. The testing scenarios generated from UML sequence diagram use the initial environment test set as the testing input data. The more important limitation of this approach is that tagging the textual requirements needs to be done manually. It is also an expensive and time consuming process that also needs to be done by a domain expert.

Sinha et al. (2015) proposed an ontology-based approach for automatically generating test cases from high-level functional requirements in the model-driven engineering process. The approach is proposed for industrial cyber-physical and safety-critical systems. The CESAR European project (Rajan and Wahl, 2013) is used to formalize requirements and developed a requirements ontology during the requirements engineering phase. In the next phases, such as design, development, and testing, the initial ontology is extended to contain more concrete concepts. The main contribution of this study is defining relationship between ontologies, requirements, and test cases and using the first two to generate the third. One of the limitations of this approach is that ontologies refined through the software development phases must be linked manually. Although it is stated that user efforts are reduced, the degree of automation provided by the proposed approach should be investigated.

Tarasov et al. (2016) proposed an approach for test case generation based on ontologies in embedded avionic systems. Although, the requirements of the system are presented in an ontology, the ontology is translated from OWL to Prolog syntax. The inference rules are also implemented in Prolog. Inference rules are constructed based on the existing test cases and then are refined by experienced software testers.

Another type of testing which has been the subject of using semantic web technologies is GUI testing. In recent years, lots of techniques and models have been proposed in literature to facilitate and automate GUI-based testing. GUI testing is also a knowledge-intensive process and knowledge-based approaches have been proposed in this domain. The idea of using ontology to assist GUI testing have been investigated in different studies.

Silva et al. (2017) proposed an ontology-based behavior-driven approach for automated testing of functional requirements in interactive systems. They also proposed an ontology based on the Behavior-Driven Development (BDD) principles. The ontology describes interactions between user and User Interface elements in a Scenario-based approach. They also developed tools to implement the proposed ontology-based approach. These tools utilize the proposed ontology to test prototypes and Web Final User Interfaces.

Tonjes et al. (2015) addressed the particular requirements of context-aware applications. Their approach relies on ontologies to provide context information and utilizes semantic annotations to generate specific test data. One of the advantages of using ontology in this approach is that it separates the context provisioning of the real world from the application, i.e., system under test. They proposed a semi-automatic test case generation approach using application description documents and its behaviour. Test cases are automatically extracted from an application behaviour model. Each input parameter or return value can be annotated by an upper ontology e.g., SUMO ontology(Niles and Pease, 2001). The knowledge in the ontology can be used to generate test data for inputs or to evaluate the value of outputs. The proposed approach also includes a similarity-based test case reduction methodology to identify the most diverse test cases for test execution based on a pairwise similarity between all test cases.

Another area is where semantic web technologies are used in testing agent-based systems. Moser et al. (2010), proposed an approach to generate a suite of test cases based on an ontology data model of the testing knowledge. The authors proposed a test case sub-ontology as a layer of the SAW project ontology (Merdan et al., 2008a; 2008b). The proposed approach is empirically evaluated with a use case from the production automation domain using Manufacturing Agent Simulation Tool (MAST) (Vrba, 2003) simulator.

The proposed approach provides users with possibility to choose from all test case parameters to define the parameter setting at any time in test process. A tool is implemented to receive parameter settings from user and adapt correspond to the ontology at runtime. It is also possible to add new parameters to the ontology with tool support. Test cases are generated and run with respect to the parameters chosen by the user. The evaluation results showed that the high-level test description of the ontology-based approach takes more initial effort for setup, but increases the usability and reduces the risk of errors during the test case generation process (Moser et al., 2010). It also improved the changeability of the test case generation approach due to lower efforts to implement new test case parameters using ontology.

Nguyen et al. (2009) proposed an ontology-based approach for automated test case generation in the context of multi-agent systems. Test case generation in Multi-Agent Systems (MAS) is a process able to build sequences of messages that exercise the agent under test. They proposed an agent interaction ontology, which define the semantics of agent interactions. They then integrated this approach with their previously proposed MAS testing framework, called eCAT (Nguyen et al., 2008a), which supports continuous testing and automated test case generation. The test case generation process is to generate a full message the Tester Agent is going to send to the agent under test. The messages are understood by both agents which is achieved by means of interaction ontology. The approach is able to generate both valid and invalid test inputs for messages.

The proposed approach is evaluated experimentally on two different-size MAS applications: a book-trading system and a system that supports bibliography research. Experimental results show that whenever the interaction ontology has non-trivial size, the proposed method achieves a higher coverage of the ontology classes than manual test case derivation. The importance of developing quality ontologies is shown in the results. One of the limitations of the proposed approach is that it is able to test single agents. Testing a team of agents is not supported in this study. The other limitation is that this approach is only capable of revealing faults resulting from agent interaction.

Hajiabadi and Kahani (2011) proposed a test data generation technique for web applications based on the elements in the GUI. They exploited Web Form ontology for filling forms of web applications and evaluating dynamic features of the web applications. At first, the structural model of the web application is constructed to demonstrate static aspects of the web application. Then using ontology and mappings, test inputs for filling forms are automatically generated to model and evaluate dynamic features of the web application. An ontology is developed from labels of text box elements of about one hundred web forms. Boundary coverage and equivalent partitioning criterion is used for generating test data based on the ontology. Test data inputs are extracted from the data stored in a database.

Li et al. (2011), exploited the ontology to generate user-centric GUI test cases by defining test case generation rules. They first developed a GUI ontology by analyzing relations among GUI components. Any visible GUI component which can be directly manipulated by users is defined as an interactive component. A test case is equivalent to a sequence of interactive components. Test case generation rules are defined in ontology and used to simplify test case generation process (Li et al., 2009). A case study on a general communication application with resulted number of extracted relations and generated test cases is presented in the study publication.

Naseer and Rauf (2012), proposed an ontology based approach for testing GUI applications. The ontology is designed for an example GUI application (i.e., Notepad) using OWL language and protégé tool. One of the limitations of this study is that the ontology developed based on a special application not a comprehensive source which define characteristics of GUI-based applications in general. Therefore, with increasing the number of events in the model, the complexity of rule definition is increased. They evaluate the proposed approach with calculating efficiency which is the ratio of coverage provided by the generated test cases.

Rauf et al. (2010), proposed an approach for automating GUI testing based on ontology and semantic annotations. Test cases are generated based on application's Event Flow Graph (EFG) and ontology of GUI events. Semantic annotations have been used to generate test data and oracle.

Tao et al. (2019), proposed an ontology based method for testing automated and autonomous driving functions. Automatic test case generation is performed using Combinatorial Testing. The ontology of SUT is converted to its corresponding CT input model which then will be used by the proposed CT-ONT and CT-ONT2 algorithms to recursively compute the combinatorial input models for the different concepts. In order to improve test case generation, constraints are added to the automatically generated input models. The authors reported on the application of the method at the industrial level. The proposed method is applied on a detailed case study based on an Autonomous Emergency Braking System Function (AEB).

Moitra et al. (2019), proposed a tool called Analysis of Semantic Specifications and Efficient generation of Requirements-based Tests (ASSERT). ASSERT has a formal requirements analysis engine and helps capturing requirements. As requirements are captured, formal analysis is applied and errors are identified using an automated theorem prover. ASSERT also automatically generates a complete set of test cases based on those requirements and thus provides clear and measurable productivity gains in system development. Then, it performs the test optimization process by analyzing generated test cases, removing invalid test cases and combining test cases with intersections.

Ul Haq and Qamar (2019), proposed a test case generation framework by integrating learning based methods and ontology-based requirement specification for conducting black box testing. The authors used learning based testing to improve the process of specification based black box testing by adding feed back loop to testing process. Learning based testing is applied to execute existing test cases derived from formal requirements and infer the model of system under test. This learned model is a representation of black box SUT.

Mekruksavanich (2017), proposed an ontology-based design flaw detection for object oriented software. An ontology of flaw structures is proposed to describe the knowledge in the flaw domains. In order to develop this ontology, an explicit description of the knowledge involved in the identification of flaw is required. Therefore, this method focused on a number of design flaws which are already well-documented. This ontology is sufficient to describe and to generate detection rules of such flaw. To perform the detection algorithm, the source code is transformed to first order logic facts and pattern matching mechanism is applied between facts and rules.

Silva et al. (2019), proposed an ontology-based approach for automated acceptance testing that ensures consistency of user requirements. This approach is based on Behavior- Driven Development (BDD) and provide automated assessment of web GUIs. The proposed approach also provides reusability through predefining a set of interactive behaviors on GUIs which could be implemented once and then automatically reused to generate tests. This set only includes behaviors that indicate steps performing actions directly on the GUI through interaction elements and is not subject to particular business characteristics. Therefore, behaviors can be easily reused to build different scenarios in different business domains. A flexible architecture is also presented to provide GUI automated testing for systems developed under whatever technology for designing the presentation layer of web pages.

C2.2. Test data generation

Mariani et al. (2014) presented Link, a technique to automatically generate test data that satisfy the semantic constraints that arise among interrelated fields. The idea of this study is to exploit the Web of Data i.e., Linked Open Data datasets to generate realistic test data. Generating realistic test data is a technique that has been used in other test data generation studies like (McMinn et al., 2012), (Bozkurt and Harman, 2011). Bozkurt and Harman (2011) exploited existing web services as sources of realistic test data based on tester-specified constraints. Their results showed that generating realistic data using service compositions achieved more success rates than random test data generation. McMinn et al. (2012) used the Internet as part of test input generation source for string inputs. They reformulated program identifiers into web queries.

Mariani et al. (2014) extract labels from application form and map them to the classes and predicates of the Web of Data with SPARQL queries. In fact, this study is the only one that utilized the Semantic Web (i.e., Web of Data) for improving test process. Web of Data is a source of huge data in various domains that can be used along with domain ontologies. DBpedia[11], which is one of the most famous datasets in the Web of Data have been used as source of generating test data in this study. Using results of SPARQL query (i.e., concepts and relations) on DBpedia, an RDF graph is generated. The resulted graph should be refined and then translated into a SPARQL query for test data generation. The refinement procedure is an iterative one which continues until all components generate data from the Web of Data. They evaluate their approach with comparing it with regular expressions approach in testing six applications. Although, results are promising, they are applications in domains that have rich data on the web of data. It seems that the complexity of these applications and the number of interrelated fields are not high enough for judging the results. The proposed technique (i.e., Link) also generates many normal test cases which can be time consuming and expensive. Analysis of the results of running test cases with normal test data is an expensive work as the oracle part is not automatic.

Szatmári et al. (2011) used ontology for testing data dependent behavior of autonomous software agents. The ontology is used to represent the context model. The hierarchy and relations of objects and changes in the environment can be precisely formulated with ontology. These relations can be directly utilized when defining and computing context coverage as test coverage metric. On the basis of the ontology of context model, semantic constraints that are included in the functional specification of the domain are determining the valid context configuration. The goal of testing autonomous agents can be expressed as testing the behavior in case of various configurations of the context. Considering defined test goal, test data which is the input data for the agent program is generated through the generation and manipulation of the agent's context. In other word, test data are specific configurations and changes in the agent's context.

Li and Ma (2015a) proposed an ontology-based automated GUI testing of spacecraft systems. For automating the process of spacecraft GUI testing, it is necessary to generate a lot of test parameters setting. It is a complex and repetitive task and needs to provide parameters in accordance to the different test environment. They utilized ontology to improve the scalability of the automatic testing by seperating the test atom from the test content. A common description method for the user interface is designed based on the ontology. Then, the rules are used to establish the mapping relationship between test atom attributes and interface elements.

C2.3. Test oracle

Bai et al. (2011), is the only study found that focused on using semantic web technologies for test oracle generation. They proposed a rule-based method to represent and calculate test oracle. The system under test is a domain-specific operating system (OS) that conforms to interface standard of real-time embedded OS. Ontology provides well-defined domain knowledge of service data, functionalities and constraints. Rules are created to model the expected behavior of the system. Test oracles are specified as 70 semantic rules, using standard rule language i.e., SWRL. The inputs of a test are matched to rule's antecedents and expected results are obtained by reasoning on the ontology. Oracles specified using semantic rules are independent of SUT implementations and can be reused across different systems conforming to the same interface standards.

[11] http://dbpedia.org.

An experiment is carried out for conformance testing on the example SUT of ARINC 653 OS (Specification, 2003). A number of 20 services out of 56 services defined in APEX interface were selected from process and partition management service categories for the experiment. Altogether 114 ontology classes, 231 individuals and 70 rules are defined for testing process management (Bai et al., 2011). The performance of the proposed approach was analyzed in terms of productivity and quality. Productivity measures how fast it can develop test oracles following the proposed approach, compared with traditional approaches. By quality, it evaluates how much the proposed approach can avoid errors in test design.

Nguyen et al. (2009), proposed an ontology- based test generation approach for multi-agent systems. An interaction ontology Nguyen et al. (2008b) is proposed in the study which define the semantics of agent interactions and is the base for test case generation process. In this study, the expected behavior of the agent under test is checked with a set of constraints automatically derived from the interaction ontology. The output of the system which is content of the message sent by the agents under test, should comply to the rules and data types specified in the interaction ontology. If the Tester Agent receives a message content that is invalid according to the interaction ontology, a fault is notified.

C2.4. Test reuse

Dalal et al. (2015) proposed a test case reuse approach based on ontology matching, which consists of four steps. In first step, application ontology is developed from scratch or reusing existing ontologies if any. This ontology defines the software artifact for which test cases are to be built and reused (Dalal et al., 2015). In the next two steps, other ontologies are searched to find concepts and properties similar to the concept and properties in the software under test ontology. The idea is that if there is concepts similar to the concept of software under test, their related test cases could be reused. Searching for similar concepts is done in the entire inheritance relationship. After finding a concept match, search for data type properties match in the matching concepts will begin. The last step is checking the range of matched properties. In this process, if any of the steps fail to match, the process is repeated with other ontologies or concepts. The output produced by this system is the data type properties for which its related test cases can be reuse. In order to evaluate the proposed approach, one test ontology and five different example ontologies were developed, but there is no evaluation of results of the proposed approach. This study presented an abstract idea which is considerable but it is not supported with proper results. One of the abilities of semantic based approaches is that they could be beneficial in automatic testing through their machine understandable format. There is no discussion about automation of the proposed approach in this study. The relation between test ontology and example software ontologies is not clarified. If the proposed approach is just matching ontologies and finding matched concepts and relations, evaluations should be done to show the effectiveness of this specific matching. There are ontology mapping techniques proposed in ontology engineering filed. In this study though, the approach for matching concepts and properties is not clear.

Li and Ma (2015b), proposed an ontology based approach to generate the reusable test case from test case library. They also proposed a simple approach for calculating the semantic similarity between test case and test requirement based on the WordNet. Two important assumptions are considered in this study. First one is that the same or similar test requirements can use the same or similar test case. The second one is that the same or similar test requirements will be repeated. After finding similar test cases, the retrieved test cases are adapted to the new test requirements based on the rules defined in the ontology. These rules establish the relationship between the test case and test requirement to modify the retrieved test cases and generate the final test case sequence. The test case ontology is developed which define a test case as a 7-tuple consisting Test Id, Test Purpose, Precondition, Test Environment, Test Input, Operation, Expected Result. Test requirement is regarded as query case for searching the test case library. The reusable test case generation approach consists of two phases: searching the cases which has the highest degree of matching with query case from case base, analyzing and adapting the cases according to the actual conditions in test requirement. An initial implementation is done using a test case library consists of 200 test case and 23 rules. OWL language is used to describe the test case and SWRL to describe the adaptation rules. Results showed that the reusable test case generation method proposed in this study is feasible in practical applications.

Li and Zhang (2012), proposed a reusable test case knowledge management model to provide test engineers with retrieve and reuse of test cases flexibly. Ontology is used for representation of reusable test cases. Along with the transformation of test case design knowledge, ontology is used to select appropriate test engineers for the specific testing projects based on their knowledge. A test ontology is developed which is aspired by OntoTest(Barbosa et al., 2006). In this ontology, a reusable test case is the sub-class of a traditional test case. The only difference in this ontology is that test data is abstracted and is separated. Each test data group has attribute, value and expected testing result. The proposed knowledge management model for reusable test cases is composed of three main parts. In the data layer, there is a testing knowledge warehouse which represents reusable test case repository and use the ontology presented in the logic layer. In presentation layer, there are knowledge management capture and retrieval to help test engineers retrieve appropriate reusable test cases. Test engineers also can find the right experts based on their knowledge for a specific test according to the knowledge map. The presented model has been applied in a testing center and a reusable test case repository with more than 12,000 test cases is constructed. In the provided case study, documentation testing, functional testing and usability testing are conducted and efficiency of using the proposed model is evaluated by the effort needed for designing a test case. Although it is stated in the study that the efficiency and productivity of test case design has improved obviously, results are not expressive enough.

Guo et al. (2011) proposed a minimum ontology for reusable test cases. They adopt Skeletal Methodology proposed by Uschold and Gruninger Uschold and Gruninger (1996) for developing their ontology. As it is stated in the study, the authors analyzed a large number of test cases for identifying reusable test case properties and simplified it as a 6-tuple consisting Id, Name, Precondition, Input, Operation, Expectation.

The only concept introduced in the study is the reusable test case and the authors didn't mention any concept hierarchy. One of the most important parts in reuse, is searching and retrieving a reusable asset. In this study the query process is not clear. Considering that most of the property types are string, natural language processing techniques will be required for querying the test case library. There is no particular evaluation for presenting results in the study. Only a case study of a simple ATM is presented to describe the test cases. Ontology is used only to represent the test cases and other capabilities of ontologies like SPARQL queries or reasoning are ignored in this study.

Cai et al. (2009) proposed a test case representation and retrieval based on ontology for test case reuse. In this study two ontologies are developed using the skeletal methodology proposed by Uschold and Gruninger (1996). Both ontologies are implemented with OWL language in protégé tool. The first ontology is software testing ontology which is based on the Guide to the Software Engineering Body of Knowledge (SWEBOK)(Abran et al., 2004). The software testing ontology represents three main concepts i.e., test case, test process and test techniques. The other ontology is software testing classification ontology which is build based on ISO 9126 software quality characteristics (Iso, 2001) and with the help of domain experts. The core part of the study discusses the management and retrieval of test cases based on the semantic similarity of two test concepts in two ontologies according to difference sets of super concept, sub concept, extension and intension. The super concept set consists of all the concept's ancestors. The sub concept set consists of all the concept's descendants. The extension of a concept consists of all the instances. The intension can be calculated through the difference set of data property and the value difference of data property (Cai et al., 2009). The semantic distance of testing concept will be computed by weighted sum of these four difference sets.

C3. Subsidiary activities: traceability, consistency checking, test optimization

C3.1. Traceability

Guo et al. (2009) proposed an improved Requirement Traceability Matrix (RTM) based on ontology which could trace not only vertical traceability from functional requirements to the other software products but also traceability between functional requirements. An ontology of RTM is developed in which define the classes and their relations of RTM. The proposed approach can support three kind of traceabilities, including dual-direction traceability between user requirements and functional requirements, dual-direction traceability between functional requirements and test cases, and traceability between functional requirements. One of the main concepts of this ontology is TestCase which have four sub concepts each one indicating one test level (i.e., unit, integration, system, acceptance). The ontology is implemented with protégé and instances from a banking business system are inserted in the ontology. Using queries in protégé it can be found out which functional requirements are related with the given test case. This can help to measure the requirement coverage criteria of test cases.

de Almeida Falbo et al. (2014), proposed a semantic document management platform to the requirements domain which can support tracing requirements through traceability matrices. They extended Infrastructure for Managing Semantic Documents (IMSD) (de Oliveira Arantes and de Almeida Falbo, 2010) to provide specific features supporting the requirements engineering process. IMSD is an infrastructure for managing semantic annotations on document templates. The proposed platform can generate traceability matrices as well as evaluating consistency of requirement priorities, supporting requirements change and verifying requirements. Traceability matrices are generated manually using SPARQL queries against IMSD. The requirements engineer has to define several SPARQL queries to generate traceability matrices. The platform is based on the new version of Software Requirements Reference Ontology (SRRO) which integrated the old version with ROoST ontology (Souza et al., 2013b). This platform supports both vertical and horizontal traceability by generating several types of traceability matrices using a relationship named ‘depends on’ and related axioms defined in the ontology. One of these types of traceability matrices is the requirement to test traceability matrix which can be used for coverage of requirements by test cases.

Traceability at a cross-project boundary (global) scale can also help improve the software testing process.Alqahtani et al. (2017) proposed an ontological approach for tracing source code vulnerabilities at the API level across project boundaries. They took advantage of the Semantic Web to share and represent information about vulnerabilities in APIs. The purpose is to provide additional analysis knowledge for tracing the use of vulnerable code in APIs and provide information about vulnerabilities found in APIs. This will help developers to find the potentially useful APIs and to reduce development and testing time.

Bicchierai et al. (2013), proposed a general framework that addressed the exploitation of ontology and semantic technology to support cohesion across different phases of software development life-cycle. They proposed an ontological model which formalized concepts and data involved in the development process of safety-critical systems. The formalized model was integrated in a web application, called RAMSES. The ontology integrated in the application can verify the consistency of documents produced along the development life-cycle. A class named ‘Usage Degree’ is defined to identify instances of the association between requirement and software components. The association between test class and requirement class in the ontology, is identified by an object property. A plug-in module is implemented to do the process of tracing requirements and obtain instances of the association between requirements and software components (i.e., extract traceability matrix information to verify it). The tools ontological architecture brings about a number of benefits like implementing the process of tracing requirements. It can also help the analyst in the identification of failure events and accomplishment of testing activities by finding the associations between failures, requirements and tests. The tool also recommends the execution or re-execution of tests or the accomplishment of testing activities. Although the approach takes advantage of knowledge management in the testing process by tracing the faults to the requirements, it doesn't have a direct impact on test generation.

C3.2. Consistency checking

Feldmann et al. (2016) used the concept in the field of testing software within machine and plant manufacturing domain. Semantic web technologies are used to ensure consistency between requirements and test cases in the mechatronic systems domain. Although, this study supports consistency between requirements and test cases in early phases and thus is more related to the requirement engineering domain, the proposed ontology has a TestCase concept which represents test data values. Requirements of the system are formulated in early design phases and test cases are used to fulfill the imposed requirements. The proposed modeling approach (Feldmann, S and Rosch, Susanne and Legat, C and Vogel-Heuser, 2014) formulates requirements in early design phases and generates test cases to check whether the imposed requirements are fulfilled. The proposed ontology has three main concepts: feature, requirement and test case. Features are functionalities of the plant components which is the capability of a plant to transport a work piece or to detect a certain work piece type. Features are needed during requirements and test case management. The parameter concept represents properties that a feature required to formulate requirements on features and generate test Cases. Such parameters can represent sensors and actuators of a mechatronic system. So, the mechatronic view and a software view on the system to be tested are integrated to model the system. Reasoning mechanisms are applied to ensure consistency between requirements as well as between requirements and test cases. The applicability of the approach was discussed only for a bench-scale application example and various aspects remained open.

Harmse et al. (2014) proposed an ontology-based scenario testing approach ensuring that a UML class diagram represents business requirements accurately. This approach supports the development of accurate requirements. The main objective of the proposed approach is to detect ambiguous and incomplete requirements before system development. First part of this research is providing translations between UML class diagrams and OWL to check consistency of models. This issue have been addressed by other researches before (Berardi et al., 2005; Gasevic et al., 2004; Zedlitz et al., 2012). Yet, even when a UML class diagram is consistent, it may not represent the business requirements accurately (Harmse et al., 2014). The contribution of this study is to present a scenario testing approach based on ontologies to validate that a UML class diagram represents the business requirements accurately.

C3.3. Test optimization

The use of ontology in test scenario management is investigated by Sapna and Mohanty Sapna and Mohanty (2011) to maximize test coverage. It is assumed that system requirements are modeled by UML use case diagram and activity diagrams. Ontology is exploited to present and analyze the relationship between requirements represented by UML diagrams. Test scenarios are generated from activity diagrams and use case diagram. Rules are written for defining relationships between concepts in an ontology to infer new facts. Test coverage criteria are used to define queries for identifying more valuable test scenarios.

De Campos H.S. et al. (2017), have proposed a regression test optimization technique based on the use of a data provenance ontology. They developed an ontology named Regression Test Execution Ontology (RTE-Ontology) by extending the PROV-O ontology (Lebo et al., 2013). PROV-O is the ontology that describes the data provenance model. They also presented a high-level service-oriented architecture, named RegressIon Test prOvenance data management (RITO). The proposed ontology-based approach is implemented by this architecture is capable of capturing and providing information about past executions of regression tests. The first step of this approach is to collect data from past regression test execution and then discover knowledge through inference over collected data. Using semantic web technologies like SPARQL queries, access to collected and inferred data is provided. These data can also be used as input for regression test optimization techniques (e.g., prioritizing test cases that fail often) or to prevent problems that happened during regression test execution in the past. The objective of this approach is to improve regression test process through a continuous cycle of feedback.

C4. Test ontologies

C4.1. Reference ontologies

Souza et al. (2017) developed a Reference Ontology on Software Testing (ROoST). ROoST defines a shared vocabulary for testing domain which can be used in knowledge management systems to facilitate communication, integration, search, and representation of test knowledge. They tried to preserve two important characteristics of quality ontologies which are being formally rigorous and also implementing non-taxonomic relations. They developed a reference ontology that adhere to the one defined by Guizzardi (2007). ROoST is developed in a modular way and it has four modules (sub-ontologies) names: Testing Process and Activities, Testing Artifacts, Testing Techniques, and Testing Environment. In order to develop ROoST, Souza et al. adopted SABiO (Systematic Approach for Building Ontologies) method (Falbo, 2014).

ROoST has several commonalities with OntoTest, since they share the same basis. OntoTest is inspired by the Software Process Ontology (SPO) version proposed in de Almeida Falbo and Bertollo (2005) while ROoST reused the most recent one Software Process Ontology Pattern Language (SP-OPL) (de Almeida Falbo et al., 2013). In ROoST, the main artifacts used and produced during the testing process and in each activity are modeled. Although it seems that ROoST presents a good coverage, there are still some points ignored, including competences that characterize testers. Competences are included in the ontology developed by Palacios et al. (2014) and represent the skills, attitudes and knowledge that a tester needs to perform different testing activities.

One of the first ontologies developed for software testing is the one proposed by Huo et al. (2003); Zhu and Huo (2005). STOWS (Software Testing Ontology for Web Service) is a taxonomy of concepts which includes two groups of concepts: the basic and the compound. Although this taxonomy provides a high coverage of concepts in software testing domain, description of compound concepts is in general. Most of the relations between concepts are of type of UML composition and inheritance. As is stated in Falbo (2014), the ontology concepts and relations must be necessary and sufficient to answer the competency questions. These few relations defined in this ontology cannot represent the true complex relationships that exist between concepts of this domain.

Barbosa et al. (2006) presented OntoTest (the ontology of software testing) used in the architectural specialization of RefTEST(Nakagawa et al., 2009) to the testing domain. Their ontology is based on ISO/IEC 12207 standard. They also explored aspects from definition and evaluation of testing criteria and also theoretical and empirical studies involving testing. It is notable that most of the testing concepts considered in OntoTest are in agreement with STOWS ontology developed by Huo et al. (2003) and Zhu and Huo (2005).

They have adopted a layered approach (de Almeida Falbo et al., 1998) to the development of OntoTest. OntoTest is a modular ontology and consists of two levels. In the ontology level, the Main Software Testing Ontology defines the main concepts and relations associated to testing. In the sub-ontology level, specific concepts from the Main Software Testing Ontology are refined in details. The sub-ontology layer includes six sub-ontologies: Testing Process, Testing Phase, Testing Artifact, Testing Step, Testing Procedure, and Testing Resource. OntoTest encompasses 115 concepts and for each basic concept represented in the Main Software Testing Ontology, there is a number of sub-concepts (Nakagawa et al., 2009). The ontology is represented in UML, at a high level of abstraction based on de Almeida Falbo et al. (1998) with few axioms defined in first order logic and also implemented in OWL (Souza et al., 2013b). The information provided in the study about sub-ontologies is not in detail enough. Testing Process, Testing Phase, Testing Artifact, and Testing Procedure sub-ontologies are just introduced and are not presented in any other study.

Engström et al., 2017, proposed SERP-test, a taxonomy of concepts in the area of software testing. The aim of constructing this taxonomy is to improve communication between researchers and practitioners. SERP-test comprises four facets for classifying research contributions and practical challenges including: Intervention, Scope, Effect target and Context constraints. This taxonomy has been evaluated by utilizing it in an industry-academia collaboration (the EASE project). The authors have used an online survey designed to let researchers and practitioners classify research results and practical challenges.

C4.2. Application ontologies

Freitas and Vieira (2014) developed a performance test ontology. They studied the possibilities of representing performance testing knowledge and supporting tester's decisions with ontologies. There is not any ontology beside this one covering specifically the domain of performance testing (Freitas and Vieira, 2014). The test activities represent important concepts in this ontology. One of the advantages of this study is that the authors demonstrated different applications of the proposed ontology. The applications aim at supporting tester's decisions with domain knowledge of technologies used in performance testing, validating and recommending options according with the test environment, goals and activities that tests might present.

Arnicans and Straujums (2015) proposed a semi-automatic methodology for creating software testing ontology from a glossary. They introduced a method for extracting the most significant words from a text document in the form of a glossary. The source that is used for developing this ontology is the Version 2.1 of "Standard glossary of terms used in Software Testing" (ISTQB) issued on April 2010 (Van Veenendaal, 2010). They used the ONTO6 (Arnicans et al., 2013) methodology for ontology development. Their development methodology is also based on popular basic methodology described by Noy and McGuinness (2001). Most important concepts are extracted and relationships between them are discovered. To do that, they assign a weight to each word from the glossary and then select top words that have higher weight or word count. One point about this ontology is that the types of relations have to be added.

Bezerra et al. (2009) developed three ontologies: OSOnto (Operating System Ontology) which represents concepts of the operating systems domain, SwTO (Software Test Ontology) which deals with the software testing domain, and SwTOI (SwTO Integrated) which represents concepts of both domains in an integrated way. Although, it is mentioned in the study that SWEBOK is the more significant source in software testing domain, it's not clear that they develop the ontology based on it. So, the source and also methodology of ontology development is not explicitly mentioned in the study.

Anandaraj et al. (2011) proposed an ontology for software testing techniques. The main concept in this ontology is testing technique class. The main goal of using this ontology is integrated teaching of programming foundations and testing in software industry. They used a four-step process to design the ontology which consists of determining domain and scope of ontology, defining concepts in the ontology, creating a class hierarchy and defining properties and constraints. The ontology is developed with OWL language in protégé tool.

Duarte et al. (OSDEF) Duarte et al. (2018) proposed an ontology of Software Defects, Errors and Failures in an ecosystem of software artifacts (OSDEF). This ontology is based on well-known standards (e.g., IEEE 1044 (ISDW Group and others, 2010), IEEE 1012 (IEEE Computer Society, 2016)) and guidlines (e.g., CMMI (Team, 2010), SWEBOK(Bourque et al., 2014)). This ontology is also one of the ontologies that reused existing foundational ontologies. It is grounded on the Unified Foundational Ontology (UFO) which is believed to be among the most used foundational ontologies that has the fastest growing rate of adoption (Duarte et al., 2018).

References

Abran, A., Moore, J., Bourque, P., Dupuis, R., Tripp, L., 2004. Software Engineering Body of Knowledge. IEEE Computer Society, Angela Burgess.

Achimugu, P., Selamat, A., Ibrahim, R., Mahrin, M.N., 2014. A systematic literature review of software requirements prioritization research. Inf. Softw. Technol. 56 (6), 568–585. doi:10.1016/j.infsof.2014.02.001.

de Almeida Falbo, R., Barcellos, M.P., Nardi, J.C., Guizzardi, G., 2013. Organizing ontology design patterns as ontology pattern languages. In: Proceedings of the Extended Semantic Web Conference. Springer, pp. 61–75.

de Almeida Falbo, R., de Menezes, C.S., da Rocha, A.R.C., 1998. A systematic approach for building ontologies. In: Proceedings of the Ibero-American Conference on Artificial Intelligence. Springer, pp. 349–360.

Alqahtani, S.S., Eghan, E.E., Rilling, J., 2017. Recovering semantic traceability links between apis and security vulnerabilities: an ontological modeling approach. In: Proceedings of the 10th IEEE International Conference on Software Testing, Verification and Validation, ICST 2017, pp. 80–91. doi:10.1109/ICST.2017.15.

Ammann, P., Offutt, J., 2016. Introduction to Software Testing. Cambridge University Press.

Anandaraj, A., Kalaivani, P., Rameshkumar, V., 2011. Development of ontology-Based intelligent system for software testing. Int. J. Commun. Comput. Innov. 2 (2), 157–161.

Andrade, J., Ares, J., Martínez, M.-A., Pazos, J., Rodríguez, S., Romera, J., Suárez, S., 2013. An architectural model for software testing lesson learned systems. Inf. Softw. Technol. 55 (1), 18–34.

Arnicans, G., Romans, D., Straujums, U., 2013. Semi-automatic generation of a software testing lightweight ontology from a glossary based on the ONTO6 methodology. Front. Artif. Intell. Appl. 249, 263–276. doi:10.3233/978-1-61499-161-8-263.

Arnicans, G., Straujums, U., 2015. Transformation of the Software Testing Glossary into a Browsable Concept Map, pp. 349–356. doi:10.1007/978-3-319-06773-5_47.

Bai, X., Hou, K., Lu, H., Zhang, Y., Hu, L., Ye, H., 2011. Semantic-based test oracles. In: Proceedings of the International Computer Software and Applications Conference, pp. 640–649. doi:10.1109/COMPSAC.2011.89.

Bai, X., Lee, S., Tsai, W., Chen, Y., 2008. Ontology-based test modeling and partition testing of web services. In: Proceedings of the IEEE International Conference on Web Services, ICWS 2008, pp. 465–472. doi:10.1109/ICWS.2008.111.

Barbosa, E.F., Nakagawa, E.Y., Maldonado, J.C., 2006. Towards the establishment of an ontology of software testing. In: Proceedings of the 18th International Conference on Software Engineering and Knowledge Engineering, SEKE 2006, pp. 522–525.

Barr, E.T., Harman, M., Mcminn, P., Shahbaz, M., Yoo, S., 2015. The oracle problem in software testing : A Survey. IEEE Trans. Softw. Eng. 41 (5), 507–525.

Bayona-Oré, S., Calvo-Manzano, J.A., Cuevas, G., San-Feliu, T., 2014. Critical success factors taxonomy for software process deployment. Soft. Q. J. 22 (1), 21–48.

Berardi, D., Calvanese, D., De Giacomo, G., 2005. Reasoning on UML class diagrams. Artif. Intell. 168 (1), 70–118.

Bezerra, D., Costa, A., Okada, K., 2009. SwTOI (Software Test Ontology Integrated) and its application in Linux test. In: Proceedings of the CEUR Workshop Proceedings, 460, pp. 25–36.

Bhatia, M., Kumar, A., Beniwal, R., 2016. Ontologies for software engineering: past, present and future. Indian J. Sci. Technol. 9 (9).

Bicchierai, I., Bucci, G., Nocentini, C., Vicario, E., 2013. Using Ontologies in the Integration of Structural, Functional, and Process Perspectives in the Development of Safety Critical Systems, pp. 95–108. doi:10.1007/978-3-642-38601-5_7.

Bizer, C., Heath, T., Berners-Lee, T., 2009. Linked data-the story so far. Int. J. Semant. Web. Inf. Syst. 5 (3), 1–22.

Bjørnson, F.O., Dingsøyr, T., 2008. Knowledge management in software engineering: a systematic review of studied concepts, findings and research methods used. Inf. Softw. Technol. 50 (11), 1055–1068.

Bourque, P., Fairley, R.E., et al., 2014. Guide to the software engineering body of knowledge (SWEBOK (r)): Version 3.0. IEEE Computer Society Press.

Bozkurt, M., Harman, M., 2011. Automatically generating realistic test input from web services. In: Proceedings of the IEEE 6th International Symposium on Service Oriented System (SOSE), pp. 13–24. doi:10.1109/SOSE.2011.6139088.

Brank, J., Grobelnik, M., Mladenić, D., 2005. A survey of ontology evaluation techniques. In: Proceedings of the International Multi-Conference Information Society, pp. 166–169.

Bueno, P.M.S., de Franco Rosa, F., Jino, M., Bonacin, R., 2018. A security testing process supported by an ontology environment: a conceptual proposal. In: Proceedings of the IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA), pp. 1–8. doi:10.1109/AICCSA.2018.8612820.

Burton-Jones, A., Storey, V.C., Sugumaran, V., Ahluwalia, P., 2005. A semiotic metrics suite for assessing the quality of ontologies. Data Knowl. Eng. 55 (1), 84–102.

Cai, L., Tong, W., Liu, Z., Zhang, J., 2009. Test case reuse based on ontology. In: Proceedings of the 15th IEEE Pacific Rim International Symposium on Dependable Computing, pp. 103–108. doi:10.1109/PRDC.2009.25.

Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J., 2003. A unifying reference framework for multi-target user interfaces. Interact. Comput. 15 (3), 289–308.

Çiflikli, E.G., Co\cskunçay, A., 2018. An ontology to support TMMi-Based test process assessment. In: Proceedings of the International Conference on Software Process Improvement and Capability Determination. Springer, pp. 345–354.

Da Silva, L.É.P., Paiva, D.M.B., Barbosa, E.F., Braga, R.T.V., Cagnin, M.I., 2014. ONTO-ResAsset development: an ontology for reusable assets specification and management. In: Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE, 2014-Janua, pp. 459–462. doi:10.13140/RG.2.1.3376.3285.

Dalal, S., Kumar, S., Baliyan, N., 2015. An ontology-Based approach for test case reuse. Adv. Intell. Syst. Comput. 309 AISC (VOLUME 2), 361–366. doi:10.1007/978-81-322-2009-1_41.

d'Aquin, M., Gangemi, A., 2011. Is there beauty in ontologies? Appl. Ontol. 6 (3), 165–175.

Davies, J., Studer, R., Warren, P., 2006. Semantic Web Technologies: Trends and Research in Ontology-Based Systems. John Wiley & Sons.

de Almeida Falbo, R., Bertollo, G., 2005. Establishing a common vocabulary for software organizations understand software processes. In: Proceedings of the EDOC International Workshop on Vocabularies, Ontologies and Rules for The Enterprise, VORTE.

de Almeida Falbo, R., Braga, C.E.C., Machado, B.N., 2014. Semantic documentation in requirements engineering. In: CIBSE 2014: Proceedings of the 17th Ibero-American Conference Software Engineering, pp. 506–519.

De Campos H.S., J., De Paiva, C.A., Braga, R., Araujo, M.A.P., David, J.M.N., Campos, F., 2017. Regression tests provenance data in the continuous so ware engineering context. In: Proceedings of the 2nd Brazilian Symposium on Systematic and Automated Software Testing, Part F1306 doi:10.1145/3128473.3128483.

Dermeval, D., Vilela, J., Bittencourt, I.I., Castro, J., Isotani, S., Brito, P., Silva, A., 2015. Applications of ontologies in requirements engineering: a systematic review of the literature. Require. Eng. doi:10.1007/s00766-015-0222-6.

Ding, W., Liang, P., Tang, A., Van Vliet, H., 2014. Knowledge-based approaches in software documentation: a systematic literature review. Inf. Softw. Technol. 56 (6), 545–567. doi:10.1016/j.infsof.2014.01.008.

Duarte, B.B., Falbo, R.A., Guizzardi, G., Guizzardi, R.S.S., Souza, V.E.S., 2018. Towards an ontology of software defects, errors and failures. In: Proceedings of the International Conference on Conceptual Modeling. Springer, pp. 349–362.

Dybå, T., Dingsøyr, T., 2008. Empirical studies of agile software development: a systematic review. Inf. Softw. Technol. 50 (9–10), 833–859. doi:10.1016/j.infsof.2008.01.006.

Eckhart, M., Meixner, K., Winkler, D., Ekelhart, A., 2019. Securing the testing process for industrial automation software. Comput. Secur. 85, 156–180.

Engström, E., Petersen, K., bin Ali, N., Bjarnason, E., 2017. SERP-test: a taxonomy for supporting industryacademia communication. Software Quality Journal, 25. Springer, pp. 1269–1305.

Euro, N., 2013. Test Protocolaeb Systems. Brussels, Belgium: Eur. New Car Assess. Programme (Euro NCAP).

Falbo, R.D.A., 2014. SABiO : Systematic Approach for Building Ontologies An Overview of SABiO. In: Proceedings of the 1st Joint Workshop ONTO.COM / ODISE on Ontologies in Conceptual Modeling and Information Systems Engineering,.

Fan, C.-F., Wang, W.-S., 2012. Validation test case generation based on safety analysis ontology. Ann. Nucl. Energy Elsevier 45, 46–58. doi:10.1016/j.anucene.2012.02.001.

Feldmann, S., Kernschmidt, K., Vogel-Heuser, B., 2016. Applications of semantic web technologies for the engineering of automated production systemsthree use cases. In: Semantic Web Technologies for Intelligent Engineering Applications. Springer International Publishing, Cham, pp. 353–382. doi:10.1007/978-3-319-41490-4_14.

FeldmannS and Rosch, Susanne and Legat, C and Vogel-Heuser, B., 2014. Keeping Requirements and Test Cases Consistent : Towards an Ontology-based Approach. In: Proceedings of the 12th IEEE International Conference on Industrial Informatics (INDIN). Ieee, p. 726—-732.

Freitas, A., Vieira, R., 2014. An ontology for guiding performance testing. In: Proceedings of the IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Workshops, WI-IAT 2014, 1, pp. 25–36. doi:10.1109/WI-IAT.2014.62.

García-Castro, R., Esteban-Gutíerrez, M., Kerrigan, M., Grimm, S., 2010. An ontology model to support the automated evaluation of software. In: Proceedings of the 22nd International Conference on Software Engineering and Knowledge Engineering, pp. 129–134.

Gasevic, D., Djuric, D., Devedzic, V., Damjanovi, V., 2004. Converting uml to owl ontologies. In: Proceedings of the 13th international World Wide Web Conference on Alternate Track Papers & Posters. ACM, pp. 488–489.

Gašević, D., Kaviani, N., Milanović, M., 2009. Ontologies and Software Engineering. In: Handbook on Ontologies. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 593–615. doi:10.1007/978-3-540-92673-3_27.

Graham, D., Van Veenendaal, E., Evans, I., 2008. Foundations of Software Testing: ISTQB Certification. Cengage Learning EMEA.

Gruber, T.R., 1995. Toward principles for the design of ontologies used for knowledge sharing? Int. J. Hum. Comput. Stud. 43 (5), 907–928.

Gruber, T.R., et al., 1993. A translation approach to portable ontology specifications. Knowl. Acquis. 5 (2), 199–220.

Guizzardi, G., 2007. On ontology, ontologies, conceptualizations, modeling languages, and (meta) models. Front. Artif. Intell. Applic. 155, 18.

Guo, S.b., Zhang, J., Tong, W., Liu, Z., 2011. An application of ontology to test case reuse. In: Proceedings of the International Conference on Mechatronic Science, Electric Engineering and Computer, MEC 2011, pp. 775–778. doi:10.1109/MEC.2011.6025579.

Guo, Y., Yang, M., Wang, J., Yang, P., Li, F., 2009. An ontology based improved software requirement traceability matrix. In: Proceedings of the Second International Symposium on Knowledge Acquisition and Modeling, 1, pp. 160–163. doi:10.1109/KAM.2009.63.

Gutiérrez, J., Escalona, M., Mejías, M., 2015. A model-Driven approach for functional test case generation. J. Syst. Softw. 109, 214–228. doi:10.1016/j.jss.2015.08.001.

Hajiabadi, H., Kahani, M., 2011. An automated model based approach to test web application using ontology. In: Proceedings of the IEEE Conference on Open Systems, ICOS 2011, pp. 354–359. doi:10.1109/ICOS.2011.6079282.

Harmse, H.F., Britz, K., Gerber, A., Moodley, D., 2014. Scenario testing using formal ontologies. In: Proceedings of the CEUR Workshop, 1301.

Hilera, J.R., Otón, S., Timbi-Sisalima, C., Aguado-Delgado, J., Estrada-Mart'\inez, F.J., Amado-Salvatierra, H.R., 2018. Combining Multiple Web Accessibility Evaluation Reports Using Semantic Web Technologies. Advances in Information Systems Development. Springer, pp. 65–78.

Horrocks, I., et al., 2002. Daml+oil: a description logic for the semantic web. IEEE Data Eng. Bull. 25 (1), 4–9.

Huo, Q., Zhu, H., Greenwood, S., 2003. A multi-agent software environment for testing web-based application. In: Proceedings of the IEEE Computer Society's International Computer Software and Applications Conference, pp. 210–215.

IEEE Computer Society, 2016. 1012-IEEE Standard for System, Software, and Hardware Verification and Validation. Institute of Electrical and Electronics Engineers.

ISDW Group and others, 2010. 1044-2009-IEEE Standard Classification for Software Anomalies. IEEE, New York.

Iso, I., 2001. IEC 9126-1: Software Engineering-Product Quality-Part 1: Quality Model. Geneva, Switzerland: International Organization for Standardization, p. 27.

Itkonen, J., Mäntylä, M.V., Lassenius, C., 2013. The role of the tester's knowledge in exploratory software testing. IEEE Trans. Softw. Eng. 39 (5), 707–724.

Kayed, A., Hirzalla, N., Samhan, A.A., Alfayoumi, M., 2009. Towards an ontology for software product quality attributes. In: Proceedings of the Fourth International Conference on Internet and Web Applications and Services. IEEE, pp. 200–204.

Keivanloo, I., Rilling, J., 2014. Software trustworthiness 2.0A semantic web enabled global source code analysis approach. J. Syst. Softw. 89, 33–50. doi:10.1016/j.jss.2013.08.030.

Kitchenham, B., Charters, S., 2007. Guidelines for performing systematic literature reviews in software engineering. Technical Report. Keele University and 1556 Durham University doi:10.1145/1134285.1134500.

Lebo, T., Sahoo, S., McGuinness, D., Belhajjame, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S., Zhao, J., 2013. Prov-o: the prov ontology.

Li, H., Chen, F., Yang, H., Guo, H., Chu, W.C.-C., Yang, Y., 2009. An ontology-based approach for GUI testing. In: Proceedings of the 33rd Annual IEEE International Computer Software and Applications Conference, 1. IEEE, pp. 632–633.

Li, H., Guo, H., Chen, F., Yang, H., Yang, Y., 2011. Using ontology to generate test cases for GUI testing. Int. J. Comput. Appl. Technol. 42 (2–3), 213–224. doi:10.1504/IJCAT.2011.045407.

Li, R., Ma, S., 2015a. The implementation of user interface autogenerate for spacecraft automatic tests based on ontology. In: Proceedings of the International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2015, pp. 2676–2681. doi:10.1109/FSKD.2015.7382380.

Li, R., Ma, S., 2015b. The use of ontology in case based reasoning for reusable test case generation. In: Proceedings of the International Conference on Artificial Intelligence and Industrial Engineering, pp. 369–374.

Li, X., Zhang, W., 2012. Ontology-based testing platform for reusing. In: Proceedings of the Sixth International Conference on Internet Computing for Science and Engineering. IEEE, pp. 86–89. doi:10.1109/ICICSE.2012.18.

Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., López-Jaquero, V., 2004. UsiXML: a language supporting multi-path development of user interfaces. In: Proceedings of the International Workshop on Design, Specification, and Verification of Interactive Systems. Springer, pp. 200–220.

Lindvall, M., Sandahl, K., 1996. Practical implications of traceability. Softw. Pract. Exp. 26 (10), 1161–1180.

Liu, Y., Wu, J., Liu, X., Gu, G., 2009. Investigation of knowledge management methods in software testing process. In: Proceedings of the International Conference on Information Technology and Computer Science, ITCS 2009, 2, pp. 90–94. doi:10.1109/ITCS.2009.157.

López-Cuadrado, J.L., Colomo-Palacios, R., González-Carrasco, I., García-Crespo, Á., Ruiz-Mezcua, B., 2012. SABUMO: Towards a collaborative and semantic framework for knowledge sharing. Expert Syst. Appl. 39 (10), 8671–8680. doi:10.1016/j.eswa.2012.01.198.

Mahdavi-Hezavehi, S., Galster, M., Avgeriou, P., 2013. Variability in quality attributes of service-based software systems: a systematic literature review. Inf. Softw. Technol. 55 (2), 320–343. doi:10.1016/j.infsof.2012.08.010.

Mariani, L., Pezz{\'e}, M., Riganelli, O., Santoro, M., 2014. Link : exploiting the web of data to generate test inputs. In: Proceedings of the International Symposium on Software Testing and Analysis (ISSTA), pp. 373–384.

McMinn, P., Shahbaz, M., Stevenson, M., 2012. Search-based test input generation for string data types using the results of web queries. In: Proceedings of the IEEE 5th International Conference on Software Testing, Verification and Validation, ICST 2012, pp. 141–150. doi:10.1109/ICST.2012.94.

Mekruksavanich, S., 2017. Ontology-assisted structural design flaw detection of object-oriented software. In: Proceedings of the Joint International Symposium on Artificial Intelligence and Natural Language Processing. Springer, pp. 119–128.

Memon, A.M., Banerjee, I., Nagarajan, A., 2003. GUI ripping: reverse engineering of graphical user interfaces for testing.. In: Proceedings of the WCRE, 3, p. 260.

Menzel, C., 2003. Reference Ontologies Application Ontologies : Either / Or or Both / And ? In: Proceedings of the KI Workshop on Reference Ontologies and Application Ontologies.

Merdan, M., Moser, T., Wahyudin, D., Biffl, S., 2008a. Performance evaluation of workflow scheduling strategies considering transportation times and conveyor failures. In: Proceedings of the IEEE International Conference on Industrial Engineering and Engineering Management. IEEE, pp. 389–394.

Merdan, M., Moser, T., Wahyudin, D., Biffl, S., Vrba, P., 2008b. Simulation of workflow scheduling strategies using the mast test management system. In: Proceedings of the 10th International Conference on Control, Automation, Robotics and Vision. IEEE, pp. 1172–1177.

Moitra, A., Siu, K., Crapo, A.W., Durling, M., Li, M., Manolios, P., Meiners, M., McMillan, C., 2019. Automating requirements analysis and test case generation. Requirements Engineering 1–24.

Moser, T., Dürr, G., Biffl, S., 2010. Ontology-based test case generation for simulating complex production automation systems. In: Proceedings of the 22nd International Conference on Software Engineering and Knowledge Engineering, pp. 478–482.

Nakagawa, E., Barbosa, E., Fioravanti, M., Maldonado, J., 2011a. PROSA-RA: a process for the design, representation, and evaluation of aspect-oriented reference architectures. J. Syst. Softw. 84, 1–40.

Nakagawa, E.Y., Barbosa, E.F., Maldonado, J.C., 2009. Exploring ontologies to support the establishment of reference architectures: An example on software testing. In: Proceedings of the Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture, WICSA/ECSA 2009, pp. 249–252. doi:10.1109/WICSA.2009.5290812.

Nakagawa, E.Y., Ferrari, F.C., Sasaki, M.M., Maldonado, J.C., 2011b. An aspect-oriented reference architecture for software engineering environments. J. Syst. Soft. 84 (10), 1670–1684.

Naseer, H., Rauf, A., 2012. Validation of ontology based test case generation for graphical user interface. In: Proceedings of the 15th International Multitopic Conference, INMIC 2012, pp. 465–469. doi:10.1109/INMIC.2012.6511486.

Nasser, V., Du, W., MacIsaac, D., 2010. An ontology-based software test generation framework. In: Proceedings of the Proceedings of the 22nd International Conference on Software Engineering and Knowledge Engineering, pp. 192–197.

Nasser, V.H., Du, W., MacIsaac, D., 2009. Knowledge-based software test generation.. In: Proceedings of the 21st International Conference on Software Engineering and Knowledge Engineering, SEKE 2009, pp. 312–317.

Nguyen, C.D., Perini, A., Tonella, P., 2008a. ECAT: a tool for automating test cases generation and execution in testing multi-agent systems. In: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS, 3, pp. 1623–1624.

Nguyen, C.D., Perini, A., Tonella, P., 2008b. Ontology-based Test Generation for Multiagent Systems. In: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 3. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, pp. 1315–1320. doi:10.1109/ANSS-41.2008.13.

Nguyen, C.D., Perini, A., Tonella, P., 2009. Experimental evaluation of ontology-based test generation for multi-agent systems. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 5386, pp. 60–73. doi:10.1007/978-3-642-01338-6_5.

Niles, I., Pease, A., 2001. Towards a standard upper ontology. In: Proceedings of the International Conference on Formal Ontology in Information Systems-Volume 2001. ACM, pp. 2–9.

Ning, H., Shihan, D., 2006. Structure-based ontology evaluation. In: Proceedings of the IEEE International Conference on e-Business Engineering (ICEBE'06). IEEE, pp. 132–137.

Noy, N., McGuinness, D., 2001. Ontology development 101: a guide to creating your first ontology. Technical Report doi:10.1016/j.artmed.2004.01.014.

Grüninger, M., Fox, M. S., 1995. Methodology for the design and evaluation of ontologies.

of Nuclear Reactor Regulation, U. N. R. C. O., 1981. Standard review plan for the review of safety analysis reports for nuclear power plants. US Nuclear Regulatory Commission, Office of Nuclear Reactor Regulation.

de Oliveira Arantes, L., de Almeida Falbo, R., 2010. An infrastructure for managing semantic documents. In: Proceedings of the 14th IEEE International Enterprise Distributed Object Computing Conference Workshops. IEEE, pp. 235–244.

Palacios, R.C., Cuadrado, J.L.L., González-Carrasco, I., Peñalvo, F.J.G., 2014. SABUMO-Dtest: design and evaluation of an intelligent collaborative distributed testing framework.. Comput. Sci. Inf. Syst. 11 (1), 29–45.

Paydar, S., Kahani, M., 2010. Ontology-based web application testing. In: Novel Algorithms and Techniques in Telecommunications and Networking. Springer Netherlands, Dordrecht, pp. 23–27. doi:10.1007/978-90-481-3662-9_4.

Petersen, K., Vakkalanka, S., Kuzniarz, L., 2015. Guidelines for conducting systematic mapping studies in software engineering: an update. Inf. Softw. Technol. 64, 1–18.

Pullmann, J., 2016. Mbui-Glossary-w3c. Fraunhofer FIT.

Rajan, A., Wahl, T., 2013. CESAR: Cost-Efficient Methods and Processes for Safety-Relevant Embedded Systems. Springer.

Rauf, A., Anwar, S., Ramzan, M., Ur Rehman, S., Shahid, A.A., 2010. Ontology driven semantic annotation based GUI testing. In: Proceedings of the 6th International Conference on Emerging Technologies, ICET 2010, pp. 261–264. doi:10.1109/ICET.2010.5638479.

Rus, I., Lindvall, M., 2002. Knowledge management in software engineering. IEEE Softw. 19 (3), 26.

Sapna, P.G., Mohanty, H., 2011. An ontology based approach for test scenario management, pp. 91–100. doi:10.1007/978-3-642-19423-8_10.

S.E.T. Committee, et al., 1998. IEEE standard for software test. IEEE standard IEEE Std 829. IEEE Comput. Soc.

Silva, T.R., Hak, J.-L., Winckler, M., 2017. A behavior-based ontology for supporting automated assessment of interactive systems. In: Proceedings of the IEEE 11th International Conference on Semantic Computing, ICSC 2017, pp. 250–257. doi:10.1109/ICSC.2017.73.

Silva, T.R., Winckler, M., Trætteberg, H., 2019. Ensuring the consistency between user requirements and graphical user interfaces: a behavior-based automated approach. In: Proceedings of the International Conference on Computational Science and Its Applications. Springer, pp. 616–632.

Singh, R., 1996. International standard iso/iec 12207 software life cycle processes. Softw. Process Improv. Pract. 2 (1), 35–50.

Sinha, R., Pang, C., Martinez, G.S., Kuronen, J., Vyatkin, V., 2015. Requirements-aided automatic test case generation for industrial cyber-physical systems. In: Proceedings of the 20th International Conference on Engineering of Complex Computer Systems (ICECCS), pp. 198–201. doi:10.1109/ICECCS.2015.32.

Software and Systems Engineering Standards Committee, 2008. IEEE Std 829–2008, IEEE Standard for Software and System Test Documentation. IEEE.

Souza, E., Falbo, R., Vijaykumar, N., 2013a. Ontologies in software testing: a Systematic Literature Review. VI Seminar on Ontology Research in Brazil, 1041, pp. 71–82.

Souza, E., Falbo, R., Vijaykumar, N., 2013b. Using ontology patterns for building a reference sofware testing ontology. In: Proceedings of the IEEE International Enterprise Distributed Object Computing Workshop, EDOC, pp. 21–30. doi:10.1109/EDOCW.2013.10.

de Souza, É.F., Falbo, R.D.A., Vijaykumar, N.L., 2014. Knowledge management initiatives in software testing: a mapping study. Inf. Softw. Technol. 57, 378–391. doi:10.1016/j.infsof.2014.05.016.

Souza, É.F.d., Falbo, R.d.A., Vijaykumar, N.L., 2017. Roost: reference ontology on software testing. Appl. Ontol. 12 (1), 59–90.

Specification, A., 2003. 653-1. Avionics Application Standard Interface. Aeronautical Radio Inc Software.

Szatmári, Z., Oláh, J., Majzik, I., 2011. Ontology-based test data generation using metaheuristics. In: Proceedings of the 8th International Conference on Informatics in Control, Automation and Robotics, 2, pp. 217–222.

Tahir, A., Tosi, D., Morasca, S., 2013. A systematic review on the functional testing of semantic web services. J. Syst. Softw. 86 (11), 2877–2889. doi:10.1016/j.jss.2013.06.064.

Tao, J., Li, Y., Wotawa, F., Felbinger, H., Nica, M., 2019. On the industrial application of combinatorial testing for autonomous driving functions. In: Proceedings of the IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), pp. 234–240. doi:10.1109/ICSTW.2019.00058.

Tarasov, V., Tan, H., Ismail, M., Adlemo, A., Johansson, M., 2016. Application of inference rules to a software requirements ontology to generate software test cases. In: OWL: Experiences and Directions–Reasoner Evaluation. Springer, pp. 82–94.

Team, C.P., 2010. Improving processes for developing better products and services, CMMI R G for Development. Technical Report. Version 1.3. Technical report, Carnegie Mellon University.

Tonjes, R., Reetz, E.S., Fischer, M., Kuemper, D., 2015. Automated Testing of Context-Aware Applications. In: Proceedings of the IEEE 82nd Vehicular Technology Conference (VTC2015-Fall). IEEE, pp. 1–5. doi:10.1109/VTCFall.2015.7390847.

Tosi, D., Morasca, S., 2015. Supporting the semi-automatic semantic annotation of web services: a systematic literature review. Inf. Softw. Technol. 61, 16–32. doi:10.1016/j.infsof.2015.01.007.

Tseng, W.-H., Fan, C.-F., 2013. Systematic scenario test case generation for nuclear safety systems. Inf. Softw. Technol. 55 (2), 344–356. doi:10.1016/j.infsof.2012.08.016.

Ul Haq, S., Qamar, U., 2019. Ontology Based Test Case Generation for Black Box Testing. In: Proceedings of the 8th International Conference on Educational and Information Technology. ACM, pp. 236–241.

Uschold, M., Gruninger, M., 1996. Ontologies: principles, methods and applications. Knowl. Eng. Rev. 11 (02), 93–136.

Van Veenendaal, E., 2010. Standard glossary of terms used in software testing. Int. Softw. Test. Qualif. Board 1–51.

Vasanthapriyan, S., Tian, J., Xiang, J., 2015. A survey on knowledge management in software engineering. In: Proceedings of the IEEE International Conference on Software Quality, Reliability and Security-Companion (QRS-C). IEEE, pp. 237–244.

Vasanthapriyan, S., Tian, J., Zhao, D., Xiong, S., Xiang, J., 2017. An ontology-based knowledge management system for software testing. In: Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE, pp. 230–235. doi:10.18293/SEKE2017-020.

Radatz, J., Geraci, A., Katki, F., 1990. IEEE standard glossary of software engineering terminology.

van Veenendaal, E.,. Test maturity model integration (tmmi), version 2.0, TMMI foundation.

Vrandečić, D., 2009. Ontology Evaluation. In: Handbook on Ontologies. Springer, pp. 293–313.

Vrandečić, D., Gangemi, A., 2006. Unit tests for ontologies. In: Proceedings of the OTM Confederated International Conferences" On the Move to Meaningful Internet Systems". Springer, pp. 1012–1020.

Vrba, P., 2003. Mast: manufacturing agent simulation tool. In: Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation, 1. IEEE, pp. 282–287.

Wnuk, K., Garrepalli, T., 2018. Knowledge management in software testing: a systematic snowball literature review. e-Informatica Softw. Eng. J. 12 (1), 51–78.

Wohlin, C., 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering. Citeseer, p. 38.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A., 2000. Experimentation in software engineering: an introduction. The Kluwer International Series In Software Engineering.

Yoo, S., Harman, M., 2012. Regression testing minimization, selection and prioritization : a survey. Softw. Test. Verif. Reliab. (March 2010) 67–120. doi:10.1002/stvr.

Zedlitz, J., Jörke, J., Luttenberger, N., 2012. From Uml to Owl 2. In: Knowledge Technology. Springer, pp. 154–163.

Zhang, Y., Zhu, H., 2008. Ontology for service oriented testing of Web Services. In: Proceedings of the 4th IEEE International Symposium on Service-Oriented System Engineering, SOSE 2008, pp. 129–134. doi:10.1109/SOSE.2008.35.

Zhu, H., 2006. A Framework for Service-Oriented Testing of Web Services. In: Proceedings of the 30th Annual International Computer Software and Applications Conference, 2, pp. 145–150. doi:10.1109/COMPSAC.2006.95.

Zhu, H., Huo, Q., 2005. Developing a software testing ontology in UML for a software growth environment of web-Based applications. In: Software Evolution with UML and XML, 1060, pp. 263–295. doi:10.4018/978-1-59140-462-0.

Zhu, H., Zhang, Y., 2012. Collaborative testing of web services. IEEE Trans. Serv. Comput. 5 (1), 116–130. doi:10.1109/TSC.2010.54.

Zhu, H.H., Zhang, Y.Y., 2014. A Test Automation Framework for Collaborative Testing of Web Service Dynamic Compositions. In: Advanced Web Services, 9781461475. Springer New York, New York, NY, pp. 171–197. doi:10.1007/978-1-4614-7535-4_8.

M. Dadkhah is currently a Ph.D. candidate in the Department of Computer Engineering at Ferdowsi University of Mashhad (FUM), Iran. She has received her BSc in 2007, and her MSc in 2011, both in Software Engineering from the FUM, Iran. Her research interests include Semantic Web, Software Testing and Software Engineering. Her recent works focused on using semantic web technologies in Software Testing.

S. Araban has received his Ph.D. in Software Engineering from the University of Melbourne, Australia. He is currently an Assistant Professor in the Department of Computer Engineering at Ferdowsi University of Mashhad (FUM), Iran. His research interests include Software Quality Assessment and Engineering, Empirical Software Engineering and Service-Oriented Enterprise Architecture.

S. Paydar is an Assistant Professor in the Department of Computer Engineering at Ferdowsi University of Mashhad (FUM), Iran. He has received his Ph.D. on Software Engineering from the FUM in 2014, in which he has worked on semantic web enabled techniques for improving model reuse in software development. Currently, his research interests include Semantic Web and Software Testing.