Ontology-Based Web Application Testing

Samad Paydar, Mohsen Kahani
Computer Engineering Department, Ferdowsi University of Mashhad
sa_pa282@stu-mail.um.ac.ir, kahani@um.ac.ir

Abstract- *Testing Web applications is still a challenging work which can greatly benefit from test automation techniques. In this paper, we focus on using ontologies as a means of test automation. Current works that use ontologies for software testing are discussed. Further a theoretical roadmap is presented, with some examples, on ontology-based web application testing.*

Keywords: Ontology, Software testing, Web application, test automation.

1. Introduction

Web applications possess special characteristics, such as multi-tired nature, multiple technologies and programming languages being involved in their development, highly dynamic behavior and lack of full control on user's interaction. This makes the analysis and verification of such systems more challenging than traditional software. Therefore, Web application testing is a labor-intensive and expensive process. In many cases, new testing methods and techniques are required, or at least some adaptations must be applied to testing methods targeted at traditional software [1][2]. Further, with the new trend in web based systems, i.e. using Web Services and SOA-based systems which lead to highly-dynamic and more loosely-coupled distributed systems, the situation gets even more challenging [1].

Test automation, that is, automating the activities involved in testing process, leads to more cost-effective, labor-saving and time-preserving methods. Using methods and techniques for automated testing of web applications can reduce the above mentioned costs and complexities [1].

Generally speaking, there are three main types of automation in software test domain.

1. Writing programs that perform some type of tests on systems. Unit testing is a good example of such automation. In order to test a unit of a system, e.g. a method, a program is written to execute the required tests on the test target. Of course, this is not limited only to unit testing, and for instance, it is possible to write a program to perform functional tests on a Web application using HTTPUnit [3]. This kind of automation, despite its great value, may be expensive for testing web applications, because such systems always grow in size and frequency of modification. We call this type, *manual test generation, automatic test execution.*
2. The second type of automation usually deals with coarse-grained goals, such as functionality testing and acceptance testing. The automation is mainly performed by capture/replay methods [3], relying heavily on human involvement and user interaction. Capture/replay methods, being not real automated methods, are not so cost-effective and scalable, because the capturing phase, which is the main part of the test, needs human intervention and it might be expensive or very hard to capture all interactions and user scenarios [4]. We call this type, *manual test case generation, automatic test execution.*
3. The third type of automations is automatic test generation based on some formal model or specification of the system. This kind of automation, which is called model-based testing, is nearer to real automation. Many works in the literature have been reported using this type of automation [1]. We call this type, *automatic test generation, automatic test execution.*

Beside this categorization, there are some other technologies that can be used for web application testing. For instance, intelligent agents are autonomous and able to live and migrate across the network and adapt to the dynamic and loosely-coupled nature of web applications. Therefore as suggested in [1], they fit better for automating web application testing. Web services can also be considered as another example of such enabler technologies, especially for testing of highly-dynamic and loosely-coupled systems like service-oriented systems [5].

Ideally, to fully automate the testing process, i.e. replacing the human tester with a computer and remove all dependencies on human, all kinds of knowledge that is required for the test process, must be acquired from the human tester and transferred to the computer in a formal and machine understandable format. Ontologies, as a powerful tool for capturing domain knowledge in a machine understandable format, show great potentials for being used to move toward this way.

In our view, ontologies can be assumed as a very powerful infrastructure for real automation of web application testing. Therefore they can be considered in the third category of automation types.

In this paper, we first present current works that have used ontologies in software testing process, and then discuss their benefits, capabilities and potential uses for automating web application testing.

2. Current Works

An ontology is an explicit and formal specification of a conceptualization of a domain of interest [6]. To state it simpler, an ontology defines the basic terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms and relations to define extensions to the vocabulary [7]. The main point about the ontology is its formality and therefore machine-processable format. Ontologies can be used in different phases of software development [8]. Here we are concentrated on current works that have used ontologies for software testing process.

In [9], an agent-based environment for software testing is proposed with the goal of minimizing the tester interferences. There are different kinds of agents in the system, such as interface agent, execution agent, and oracle agent. Each kind of agent is responsible for one part of the testing process. For example, TCG (Test Case Generator) agent has the role of test case generation. In order to enable agents to communicate and understand each others' messages, and also share a common knowledge of the test process, an ontology for software testing is developed and used. This ontology contains concepts like activities, stages, purposes, contexts, methods, artifacts, etc.

TestLixis a project with the goal of developing necessary ontologies for Linux test domains. It focuses on 3 ontologies: OSOnto (Operating System Ontology), SwTO (Software Test Ontology), SwTOi (Software Test Ontology Integrated). This project is registered in 2007/4/14, but there is no information or documentation available on the project homepage [10].

In [11], a work is introduced which is about development and use of ontologies of the fault and failure domains of distributed systems, such as SOA-based system and Grids. The work is said to be in the early stages of the ontology development. It is hoped that in future, this ontology can be used to guide and discover novel testing and evaluation methods for complex systems such as Girds and SOA-based systems. In this work, ontologies are viewed as an intelligent communication media for machines, and also as a means for enabling machines to acquire knowledge necessary to develop their own strategies for testing and evaluating systems.

In [12], ontologies have been used to model Web service composition logics, Web service operational semantics, and test case generation for testing Web services. OWL-S is used to describe the semantic and application logic of the composite Web service process. Then, using the Petri-Net ontology, developed by the authors, a Petri-Net model is constructed to depict the structure and behavior of the taeget composite service. Then, using the Petri-Net model of the composite service, and the ontology, test cases are generated for testing the service.

In [13], an ontology is developed for software metrics and indicators. ISO standards, for instance ISO/IEC 15939 standard[14], and ISO/IEC 9126-1 standard[15], have been used as the main source for development of the ontology. The authors have described the application of this ontology in a cataloging system. This system provides a collaborative mechanism for discussing, agreeing, and adding approved metrics and indicators to a repository. In addition, the system provides semantic-based query functionality, which can be utilized for consultation and reuse. Similar work is also presented in [16].

A SOA-based framework is proposed for automated web service testing in [17] and [18]. The authors have mentioned some technical issues that have to be addressed in order to enable automated online test of web services. For instance, issues like how to describe, publish, and register a testing service in a machine understandable encoding, or how to retrieve a testing service. To resolve these issues, a software testing ontology named STOW (Software Testing Ontology for Web Services) was developed.

In addition to categorization of terms and concepts, they have defined appropriate relations, which can be used to do some reasoning in the testing process. For instance, when a testing service with the capability of testing Java applets is requested, and there is a testing service capable of testing Java programs, it can be reasoned that the later can be used for the required task.

In [8], some examples of ontology applications throughout the whole software development lifecycle are presented. It is claimed that in the testing phase, a non-trivial and expensive task, which demands some degree of domain knowledge, is the task of writing suitable test cases. They propose to use ontologies to encode domain knowledge in a machine processable format. Using ontologies for equivalence partitioning of test data is mentioned as an example. In addition, by storing the domain knowledge in an ontology, it will be possible to reuse this knowledge.

In [19] the main focus is to use ontologies in early software design phases, i.e. specifications, with emphasis on detecting conceptual errors, such as mismatches between system behavior and system specifications. In addition, an architecture and some required tools are presented to support such conceptual error checking.

In [20] it is suggested that ontologies can be used as semantic formal models, and hence MDA (Model-Driven Architecture) can be extended to ODA (Ontology-Driven Architecture). Using ontologies, it will be possible to represent unambiguous domain vocabularies, perform model consistency checking, validation and some levels of functionality testing.

3. Ontology-based software testing requirements

In this section we discuss the required steps to reach the goal of ontology-based web application testing.

The process of using ontologies in software testing can be divided into two phases or activities.

1. The first one is developing the required ontology which captures an appropriate level of required knowledge to perform the testing process. By 'required knowledge' and hence 'required ontology', we mean two different kinds of knowledge and hence ontology:
    - The first kind of knowledge required is the knowledge of the testing process, i.e. different types of tests, their goals, limitations and capabilities, the activities involved in testing, their order and relation. Obviously this kind of knowledge is vital for automating web application testing. Therefore from the point of view of ontology-based software testing, it is required to develop an ontology which captures an appropriate level of this knowledge in a machine processable format.
    - The second kind of knowledge required is the application domain knowledge. It is required to know the concepts, possibilities, limitations, relations, and expected functionalities of the application under test. For instance, testing an online auction web application will require different knowledge from what is needed for testing an e-learning application. One simple reason is that to perform some tests, like functional test, it is required that expected functionalities be known. Therefore, to fully automate the test process, an appropriate level of application domain knowledge is required to be captured and formally expressed through an ontology.
2. The next phase is to develop procedures for utilizing the knowledge embedded in the ontology to automate different tasks in the testing process. Of course the two stages are not necessarily independent or completely sequential. It is possible to start second phase with a reasonable ontology and incrementally improve and enhance the ontology and the testing processes.

3.1 Ontology developing for application testing

Although development of a knowledge-rich ontology is a time-consuming and laborious activity, it seems that it does not possess serious technical problems that need innovative ideas. Currently there are numerous environments for ontology development and also tools and utilities to automate some activities of ontology development. For instance, there are tools that extract basic terms and concepts from a set of technical documents using text-mining methods, though their results need to be verified by an expert [21]. It is worth to note that once an ontology is developed for web application testing, it can be frequently reused and incrementally evolved and improved.

As stated before, an ontology defines the basic terms and relations comprising the vocabulary of a topic area, as well as the rules for combining terms and relations to define extensions to the vocabulary. So the main part of the ontology development is to extract the terms, concepts, relations and rules of the domain. Currently there are good sources available for this purpose. Here, we discuss some of them.

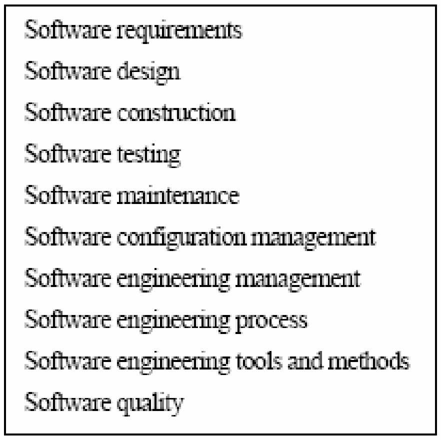
As stated in [22], The Guide to the Software Engineering Body of Knowledge (SWEBOK) is a project of IEEE Computer Society and Professional Practices Committee which aims at providing a consensually validated characterization of the bounds of the software engineering discipline and to provide a topical access to the Body of Knowledge supporting that discipline [23].

The Body of Knowledge is divided into ten software engineering Knowledge Areas (KA) (Figure 1). To promote a consistent view of software engineering worldwide, the guide uses a hierarchical organization to decompose each KA into a set of topics with recognizable labels. A two- or three-level breakdown provides a reasonable way to find topics of interest. The breakdowns of topics do not presume particular application domains, business uses, management philosophies, development methods, and so forth. The extent of each topic's description is only that needed to understand the generally accepted nature of the topics and for the reader to successfully find reference material.

One of the KAs defined in SWEBOK, is the Software Testing KA. This KA is a useful source for developing ontology of software testing. As shown in Figure 2, the number of concepts and facts and relations in the Software Testing KA, is noticeable in comparison to other KAs. Chapter 5 of the guide, which is focused on Software Testing, presents a breakdown of the topics and related concepts in a manner that can be helpful for developing the ontology. Although it does not mainly focus on web application testing, but it can be used as a useful guide to mange and organize the concepts and relations.

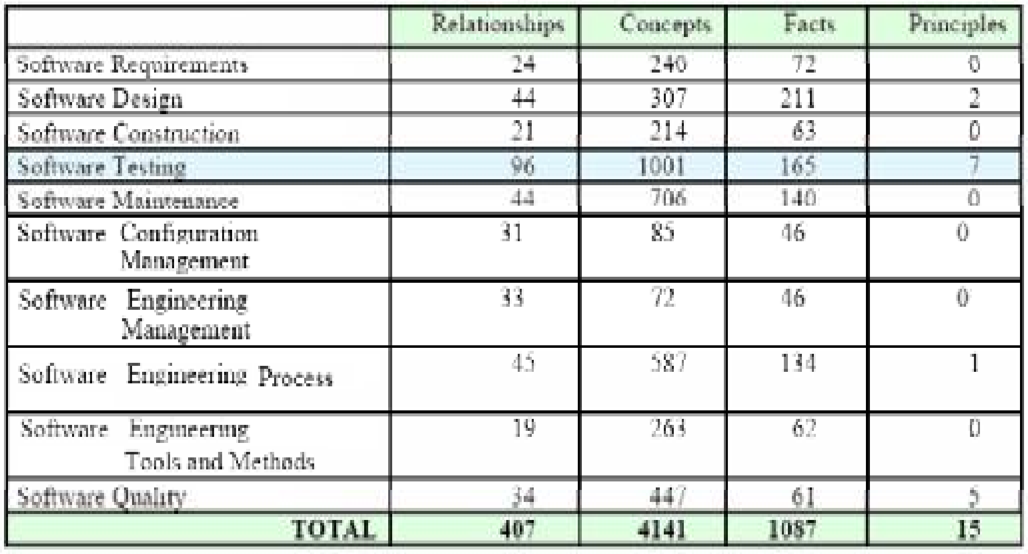
In addition, there are ISO and IEEE standards that can be used to extract the main terminology, concepts, and their relations[14], [15], [24].

Therefore we believe that the first phase, that is, the development of an ontology for web application testing is not theoretically so challenging.



Software requirements
Software design
Software construction
Software testing
Software maintenance
Software configuration management
Software engineering management
Software engineering process
Software engineering tools and methods
Software quality

Figure 1-SWEBOK knowledge areas (KAs)



	Relationships	Concepts	Facts	Principles
Software Requirements	24	240	72	0
Software Design	44	307	211	2
Software Construction	21	214	63	0
Software Testing	96	1001	165	7
Software Maintenance	44	706	140	0
Software Configuration Management	31	85	46	0
Software Engineering Management	33	72	46	0
Software Engineering Process	45	587	134	1
Software Engineering Tools and Methods	19	263	62	0
Software Quality	34	447	61	5
TOTAL	**407**	**4141**	**1087**	**15**

Figure 2- Overview of quantity of elements in the SWEBOK

3.2 Ontology developing for application domain

It is not a good idea to first develop the system completely and then start to develop its ontology separately from the scratch; rather it is desirable to somehow synchronize the development of the system with the development of its ontology. We see two approaches for reaching to this goal.

One approach is to develop the application domain ontology and then start to develop the application. In this approach, supporting tools and environments are required to help the developer use and communicate with the developed ontology, while developing the application. For instance, when designing a HTML form containing a text field, the designer can annotate the text field with the term 'emailAddress' defined in the ontology of the application previously designed. The main difficulty of this approach is of course the development of the application domain ontology. It is worth to note that although it may seem that postponing the development of the system to the completion of the development of the ontology will lengthen the development lifecycle, but it undoubtedly will shorten the testing time and therefore this drawback can be somehow remedied.

The second approach is to use ODA, as to some extent suggested in [20]. In this case, it is required to develop the semantically-rich formal models of the system using ontologies. Then, automatically extract the executables of the system from these models. Although this approach is an open field for future research, but it is worth noting that currently it is possible to use UML and OCL as a language for designing ontologies of the system and then from UML, get executable code, though not 100% complete. Using UML for developing ontologies is used in [18] [25] for example, and significant work has been done to bring together Software Engineering languages and methodologies such as the UML with Semantic Web technologies such as RDF and OWL, exemplified by the OMG's Ontology Definition Metamodel (ODM) [20].

3.3 Developing intelligent methods to utilize the ontology

Once the required ontologies, whether ontology of the testing process or ontology of the application domain, are developed, the main part of the job can be started. That is, to develop intelligent methods and procedures that utilize the available ontologies to minimize the human intervention in the testing process.

Although some works in this direction, has been reported in the literature ([12], [18]), but this is still an open research area and the methods of using ontologies, needs to be improved. For instance, in [9], which is an agent-based testing framework, ontology is used only as a communication media between the agents. Agents run procedures that are exactly hardwired in them, and there is no inference or adaptation.

To further move in this area, it is required to utilize ontologies to enable agents dynamically devise their plans and procedures. This is needed because it can eliminate the need to hardwire all procedures within the agents.

4. Potential applications of ontologies in web application testing

Ontologies are a means of capturing knowledge of a domain in a machine understandable manner. Therefore by using well-developed ontologies, we would be able to write intelligent methods that automate different tasks and activities of the testing process. In this section, we present some examples to show potentials of using ontologies to automate web application testing:

1. Using ontologies for test planning and Test specification: Using an ontology that provides the knowledge of different testing activities and their order and relationships, it is possible to specify the test plan in a machine understandable language. For instance, in the presence of such ontology, by specifying that "system X must be tested using black-box strategy", it can be inferred that what type of tests, in what order, must be performed on this system, and which test criteria and test case generation method should be used.
2. Using ontologies for semantic querying: Using ontology in different testing activities, such as test planning, test specification, test execution and result evaluation, enable automatic generation of the whole test process documents in a machine-understandable format. Therefore it will be possible to retrieve test process information using semantic queries. For instance, after performing code coverage on an application, it would be possible to ask the system which classes or methods have nor been sufficiently tested.
3. Using Ontology as an enabler: Using web services for testing web based application, especially large, distributed ones, seems a good idea because of the interesting properties that they have, such as being loosely coupled, dynamic, and interoperable. In such cases, i.e. using web services for different activities in the testing process, there is a potential for ontology to be utilized for service definition, publication, registration, advertisement and retrieval. In addition to web services, agents are also a good candidate for automating the test process. In this view, ontologies can be used more that just as a communication media, making it possible to share the domain knowledge between agents and make them cooperate with each other. In addition, agents can utilize the ontology to perform their tasks more intelligently.
4. Using Ontology for test case generation: Ontologies show great potentials to be used for test case generation. Here, we just mention some examples. These potentials can be divided into two categories:
   a. *Test case generation based on the software test ontology.* For instance, based on the test type that is to be performed, it might be necessary to use different test generation methods. E.g. when performing security tests on a web application, it is better to use SQL injection or cross site scripting techniques to generate test data, which is used to fill form fields. However, when performing functional testing, other techniques are more appropriate.
   b. *Test case generation based on the domain ontology of the application under test.* For

instance, while testing the registration page of a web forum application, ontology of the application domain- in this case a web forum- can be used to generate appropriate test data for registration form fields. As an example, if a form field is properly annotated with the term "User.Age", it can be used for equivalence partitioning of candidate test data for entering in this field. If a form field is annotated with "Pass.MinLen=6", this information can be used to infer border values for password length, so generating good set of test data. As another example, annotating a form field with term “EmailAddress” and another field with “CountryName”, enables generation of different and specific test data.

5. Ontologies for test oracle: One of the main obstacles in really and fully automating software test process is the test oracle. As mentioned in [26]," *It is little use to execute a test automatically if execution results must be manually inspected to apply a pass/fail criterion. Relying on human intervention to judge test outcomes is not merely expensive, but also unreliable*". Ontologies can be used for test oracle automation. An oracle must judge on the result of a test execution, deciding whether the test is passed or failed. This judgment is based on a set of criteria, which can be categorized and defined formally, and hence can be to some extent embedded in ontologies. Therefore, it is possible to specify the evaluation criteria of each test type in the ontology in order to be used by the automated oracle to judge the test results. For instance, while performing load or performance test on a web application, test results can be judged based on the delay of the HTTP responses. Or, in some cases, test results can be judged by inspecting absence or presence of a special term in the HTTP response. Also, HTTP status codes can be used for this purpose. More complicated judgments may also be automated. For instance, it may be possible to specify in a test specification, that if the test runs successfully, a new record must be inserted [deleted, or changed] in [from, in] table X of database D. Of course, there may be some cases which cannot be satisfied by a non-human oracle, e.g. verifying how user-friendly a system is.

5. Conclusion

In this paper we first presented a brief survey of current works that have used ontology in the software testing process. Then, the possible applications of using ontologies in web application testing were investigated.

It can be concluded that the full potential of using ontologies for web application testing has yet to be explored and it is an open area for research and innovation to develop intelligent methods and procedures for maximize the automation of different activities involved in software testing process.

Acknowledgement

This work has been supported by a grant by Iran’s Telecommunication Research Center (ITRC), which is hereby acknowledged.

References

[1]. G.A. Di Lucca, A.R. Fasolino, "Testing web-based applications: The state of the art and future trends", Information and Software Technology 48:1172-1186, 2006.

[2]. Y. Wu, J. Offutt, “Modeling and testing web-based applications”, GMU ISE Technical Report, ISE-TR-02-08, 2002.

[3]. F. Ricca, P. Tonella, “Web Testing: a Roadmap for the Empirical Research”, WSE:63-70, 2005.

[4]. K. Li, M. Wu, “Effective GUI Test Automation: Developing an Automated GUI testing Tool”, Sybex publications, p20, 2005.

[5]. H. Zhu, “A Framework for Service-Oriented Testing of Web Services”, COMPSAC, 2006.

[6]. T.R. Gruber, "A translation approach to portable ontologies", Knowledge Acquisition, 5(2):199-220, 1993.

[7]. R. Neches, R.E. Fikes, T. Finin, T.R. Gruber, T. Senator, and W.R. Swartout, "Enabling technology for knowledge sharing", AI Magazine, 12(3):36-56, 1991.

[8]. H. J. Happel, S. Seedof, "Applications of Ontologies in Software Engineering", 2nd Int. Workshop on Semantic Web Enabled Software Engineering (SWESE 2006).

[9]. R. Maamri, Z. Sahnoun, “MAEST: Multi-Agent Environment for Software Testing”, Journal of Computer Science, April, 2007.

[10]. TestLix Project: http://projects.semwebcentral.org/projects/testlix/

[11]. The White Rose Grid e-Science Centre, “Developing a Fault Ontology Engine for the Testing and Evaluation of Service-Oriented Architectures”, September, 2006.

[12]. Y. Wang, X. Bai, J. Li, R. Huang, "Ontology-Based Test Case Generation for Testing Web Services", ISADS, March 2007.

[13]. M. de los Angeles Martin, L. Olsina, "Towards an ontology for software metrics and indicators as the foundation for a cataloging web system", LA-WEB, 2003.

[14]. ISO/IEC 15939:2007 – "Systems and Software Engineering - Measurement Process", http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=44344

[15]. ISO/IEC 9126-1:2001 – "Software Engineering – Product Quality - Part 1: Quality Model", http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=22749

[16]. M. Genero, F. Ruiz, M. Piattini, C. Calero, "Towards an Ontology for Software Measurement", SEKE 2003.

[17]. H. Zhu, "A Framework for Service-Oriented Testing of Web Services", COMPSAC 2006.

[18]. H. Zhu et al, "Developing A Software Testing Ontology in UML for A Software Growth Environment of Web-Based Applications", "Software Evolution with UML and XML", 2004, chapter 9.

[19]. Y. Kalfoglou, "Deploying ontologies in Software Design", Ph.D. thesis, Dept. of Artificial Intelligence, University of Edinburgh, 2000.

[20]. W3C Semantic Web Best Practices & Deployment Working Group, "Ontology Driven Architectures and Potential Uses of the Semantic Web in Systems and Software Engineering", 2006.

[21]. C. Calero, F. Ruiz, M. Piattini, "Ontologies for Software Engineering and Software Technology", Springer, 2006, chapter 1.

[22]. "Guide to the Software Engineering Body of Knowledge", www.swebok.org/ironman/pdf/SWEBOK_Guide_2004.pdf

[23]. Guide to the SWEBOK, http://www.swebok.org/

[24]. IEEE Standard for Software Test Documentation, 1998.

[25]. S. Cranefield, "UML and the semantic web", proceedings of International Semantic Web Working Symposium (SWWS), 2001.

[26]. M. Pezz, M. Young, "Software Testing and Analysis: Process, Principles and Techniques", 2008, section 17.5.