ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

Invariant Based Integration of Web Service Interface Testing

V.Alanthuraiyan[1], E.Kruthika[2], R.Devisri[3]

Asst Professor/MCA, Adhiyamaan College of Engineering, Hosur, Tamilnadu, India[1]

Student IIMCA, Adhiyamaan College of Engineering, Hosur, Tamilnadu, India[2, 3]

ABSTRACT: Cloud-based testing is more efficient and effective than traditional methods. Due to the rapid growth in the development of Web Services (WS) techniques and applications in the various domains software testers face the great challenges in Web Services testing. The major issue of testers is to integrate the web services owned by other vendors.

I. CURRENT STATE OF THE ART OF WEB SERVICE AND TEST BED

Most research on testing Web Services fall into the following four classes:
a)Test cases generation: Techniques have been developed to generate test cases from syntax definitions of Web Services in WebServiceDescriptionLanguageb)Business process and behavioral models in BPEL ontology-based descriptions of semantics in OWL-S c)Other formal models of WS finite state machines and labeled transition systems. d)Grammar graphs. These techniques have addressed various WS specific issues like robustness in dealing with invalid inputs, fault tolerance to the failures of other services, broken communication connections, security in the environment that is vulnerable to malicious attacks and so on. In the Web Service unit and Integration testing, the service under test needs to be separated from other services because one service relies on other service. There are various techniques have been developed to generate service stubs (1) or mock services(2) to replace the other services for testing.

1.1 Challenges in Web Service Testing

To check the correctness of service output against specifications of requirements like voting mechanism to compare the output from multiple equivalent services. A number of prototypes and commercial tools have been developed to support various activities in testing.

1.2The lack of software artifacts.

A service oriented application consists of services owned by many different stakeholders. Software developers are one of the stakeholders who are not able to access to the design document, source code and even the executable code of the other parts. This software work of art is crucial to perform test activities efficiently and effectively.

1.3The lack of control over test executions.

A service oriented application is inherently distributed and typically contains parts running on hardware owned by other stakeholders. A tester regularly lacks control over the executions of the other owners' parts mainly observation of internal behavior.

II. RESEARCH OBJECTIVES

It is widely recognized that a testing technology for WS must also meet the following requirements:

2.1Dealing with Assortment.

The distributed and shared ownership of services implies that the parts of a service-oriented application may function with different deployment configurations and delivering services of differing quality. On the other hand different service requesters may have different test requirements to meet their own company purposes. Testing must deal with all such

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

varieties and their combinations. The scenario of service-oriented computing is that a service client searches for a required function in a registry, with dynamism links to the service and invokes it. It is widely assumed that testing before the invocation is necessary especially in mission critical applications. This testing is called testing on-the-fly(ie time of testing is just before the invocation).All parts to be integrated are already in operation.

2.2Need for collaborative Automation

Service provider's point of analysis, the test invocations of a service must be distinguished from the real ones. So that the normal operation of the service is not interrupted by test activities. Client's point of observation is test invocations should also be distinguished from real ones so that they do not actually receive the real services and no need to pay for such test invocations as real services. The requirement of testing on-the-fly is performed automatically. It has been predictable that to address all these issues, testing WS should be a collaborative effort contributed to by all stakeholders. We present a framework for collaborative testing in which testing activities are accomplished through contacts.

III. RELATED WORKS

Modern web interfaces incorporate client-side scripting and user interface manipulation which is increasingly separated from server-side application logic [3]. Although the field of rich web interface testing is mainly unexplored, much knowledge may be derived from two closely related fields.

3.1 Traditional Web Testing

Benedikt et al. [4] present VeriWeb, a tool for automatically exploring paths of multipage websites through a crawler and detector for abnormalities such as navigation and page errors VeriWeb uses Smart Profiles to extract candidate input values for form-based pages. Although VeriWeb's crawling algorithm has some support for client-side scripting execution, the paper provides insufficient detail to determine whether it would be able to cope with modern AJAX web applications. Tools such as WAVES [4] and SecuBat [5] have been proposed for automatically assessing web application security. The general approach is based on a crawler capable of detecting data entry points which can be seen as possible points of security attack. Malicious patterns, e.g., SQL and XSS vulnerabilities, are then injected into these entry points and the response from the server is analyzed to determine vulnerable parts of the web application. Alfaro apply model-checking [5] to web applications using his tool called MCWEB [6]. His work, however, was targeted toward web 1.0 applications.

3.2 Model Bases Testing by Ricca and Tonella

A model-based testing approach was proposed by Ricca and Tonella [6]y introduce ReWeb, a tool for creating a model of the web application in UML, which is used along with defined coverage criteria to generate test cases. Another approach was presented by Andrews et al. [7], who rely on a finite state machine together with constraints defined by the tester. All such model-based testing techniques focus on classical multipage web applications. They mostly use a crawler to infer a navigational model of the web. Unfortunately, traditional web crawlers are not able to crawl AJAX applications [8]. Logging user session data on the server is also used for the purpose of automatic test generation [8], [9]. This approach requires sufficient interaction of real web users with the system to generate the necessary logging data. Session-based testing techniques are merely focused on synchronous requests to the server and lack the complete state information required in AJAX testing. Delta-server messages [10] from the server response are hard to analyze on their own. Most of such delta updates become meaningful after they have been processed by the client-side engine on the browser and injected into the DOM.

3.3 Fault Finding Tool

Exploiting static analysis of server-side implementation logic to abstract the application behavior is another testing approach. Artzi et al. [10] propose a technique and a tool called Apollo for finding faults in PHP web applications that is based on combined concrete and symbolic execution. The tool is able to detect runtime errors and malformed HTML output. Halfond and Orso [6], [7] present their static analysis of server-side Java code to extract web application request parameters and their potential values. They use [10] symbolic execution of server-side code to identify possible interfaces

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

of web applications. Such techniques have limitations in revealing faults that are due to the complex (client-side) runtime behavior of modern rich web applications.

3.4 GUI Application Testing

AJAX applications can be seen as a hybrid of desktop and web applications since the user interface is composed of components and the interaction is event-based [11]. AJAX applications have specific features, such as the asynchronous client/server communication and dynamic DOM-based user interface, which make them different from traditional GUI applications [11], and therefore require other testing tools and techniques.

3.5 Current AJAX Testing Approaches

The server side of AJAX applications can be tested with any conventional testing technique. On the client, testing can be performed at different levels. Unit testing tools such as JsUnit3 can be used to test JAVASCRIPT on a functional level. The most commonly used AJAX testing tools are currently capture/replay tools such as Selenium IDE,4WebKing,5 and Sahi,6 which allow DOM-based testing by capturing events fired by user interaction. Other web application testing tools such as WebDriver7 or Watij8take a different approach: Rather than being a JAVASCRIPT application running within the browser, they use a wrapping mechanism and provide APIs to control the browser. Such tools demand, however, a substantial amount of manual effort on the part of the tester to create and maintain a test suite since every event trail and the corresponding DOM assertions have to be written by the tester.Marchetto et al. [12] discuss traditional web testing techniques (e.g., code coverage testing [12], model-based testing [12], session based testing [13],) have serious limitations when applied to modern Web 2.0 applications. They propose an approach for state-based testing of AJAX applications. In Our approach is the first to exercise automated testing of Web 2.0 applications by simulating real user events on the web user interface and inferring an abstract model automatically.

3.6 Invariants

Invariants are used to assert program behavior at runtime is as old as programming itself [13]. Web applications, domain, performs validation of the HTML output (e.g., [14], [12]) could be considered to be using invariants on the DOM. This paper makes the use of invariants for testing web applications explicit by defining different types of (client side) invariants, providing a mechanism for expressing those invariants, and automatically checking them through dynamic analysis. Automatic detection of invariants is another direction that has gained momentum.. We have also started exploring ways of automatically detecting DOM and JAVASCRIPT invariants in web applications [14].

IV. SEMANTIC WEB SERVICES IMPLEMENTATION

The Web Ontology Language OWL is a semantic markup language for publishing and sharing ontology's on the World Wide Web [15]. It is designed for applications that need to process the content of information. It is a part of the growing stack of W3C recommendations related to the Semantic Web. In this section, we adapt the ontology of software testing developed in [9], [7] and discuss how it is implemented in OWL.

4.1 Software Testing Ontology for Web Services

Concepts in STOWS are Elementary concepts, basic testing concepts, and compound testing concepts. The elementary concepts are computer software and hardware based on which testing concepts are defined. They include the simple objects involved in software testing, such as the types of Hardware and Software artifacts and their Format. These basic concepts are combined together to express compound testing concepts, which include Task and Capability.

4.2 Capability of T-Services and Test Tasks

The capability of a T-service represents its capability of performing test tasks. The class Capability in the ontology defines the aspects that affect the capability of a service to perform tasks, including the activities that the service can do, the test methods that the service uses, the artifacts that the service consumes and produces, the context in which the service performs test activities, and the environment for test activities .The structure of Capability is shown in the UML class diagram given in Fig. 1.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

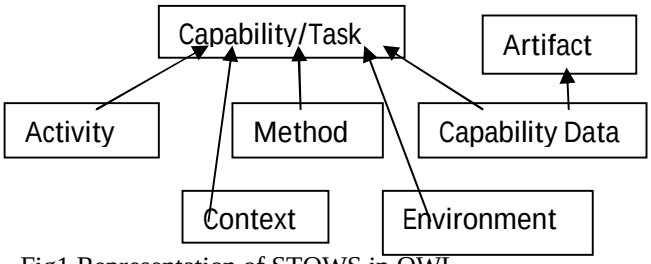
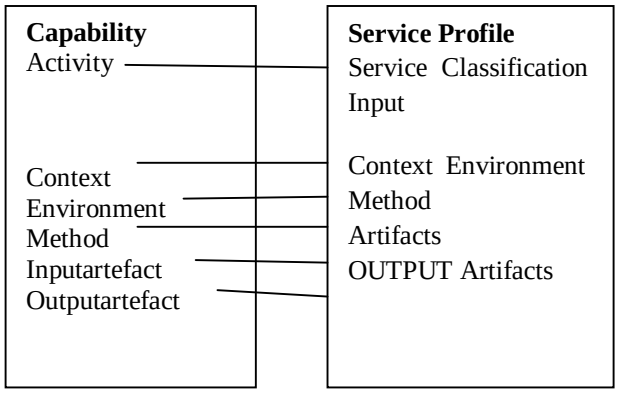


Fig1 Representation of STOWS in OWL

In OWL-S, semantic descriptions are presented in the form of service profiles and used in service registration and discovery. The vocabulary of a subject domain is defined in a data model as classes with subclass relations. To implement the ontology STOWS, we represent the concepts, including elementary, basic, and compound concepts as classes in OWL data model. In OWL-S, a service profile contains the Inputs, Outputs, Preconditions, and Results (IOPR) and a classification of the service [8]. Fig. 2 shows how the concept of capability is represented in a service profile.

In the service profile of T-service, the test context, the environment, and the method aspects are represented as input parameters Context, Environment, and Method. For example,. By representing capability and task concepts in profiles, OWLS/UDDI Matchmaker can be employed to perform semantic- based search of T-services.

Fig2 Mapping between capability and service Profile



4.3T-Service Registration and Discovery

The OWL-S/UDDI Matchmaker (Matchmaker for short) extends UDDI registry with a capability-based service matching engine [15], [16]. It provides three levels of matching between capability and search request.

1. Exact matching: the capabilities in the registry and in the request match exactly.
2. Plug-in matching: the service provided is more general than that in the request.
3. Relaxed matching: there is a similarity between services provided and that in the request.

The Matchmaker also provides filters for users to construct more accurate service discovery: which are namespace filter, domain filter, text filter, I/O type filter, and constraint filter [11]. With these filters, users can construct necessary compound filters to control the precision of matching. The matching engine returns a numeric score for each candidate so that the higher the score, the more similar between the candidate and the request. Therefore, selection from the candidates can be based on the scores that tagged by the Matchmaker on the candidate services.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

4.4 Ontology Management Service

It is impossible to build a complete ontology of software testing due to the rapid development of new testing technique, methods and tools. So, the ontology must be extendable and open to the public for updating. For this purpose ontology management mechanism is used. Using the API, OWL data model stored in OWL files or databases can be loaded, changed and saved, queries be made and reasoning performed using a description logic inference engine [10]. The manipulation of the ontology can be implemented as operations on OWL files. OMS provides a WS interface to read and update the ontology data model, which is open to the public.

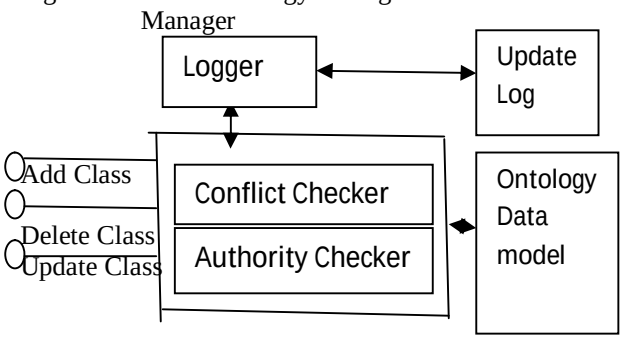
4.5 WS interface in OMS

OMS provides a WS interface to read and update the ontology data model, which is open to the public. It provides three services to users: Add Class, Delete Class, and Update Class to add new concept, delete concept, and revise concept of the ontology. A new test method name "Data Mutation" can be added to the ontology as a subclass of test method. If a new T-service is to be registered that generates test cases from a new formal specification language called FSL, then a new type of software artifacts called "FSL" can be added to the ontology as a subclass of software artifact, rather than a subclass of test method

4.6 Prevent Misuse of OMS

To prevent misuses of the ontology management service two technical solutions are used. First we classify the classes in the ontology into two types' elementary classes and extended classes. Elementary classes are those that form the framework of the ontology STOWS. None of them could be pruned down from the ontology hierarchy to avoid structural damage to the ontology. The extended classes are those classes attached to the framework to populate the ontology with concrete concepts and instances of the concepts. They can be added by the users and deleted from the hierarchy. We have implemented an Authority Checker, which checks delete operations to ensure that the class to be deleted is extended class. Conflict Checker checks the operations on the ontology to ensure that the new class to be added does not exist in the ontology and that the class to be deleted has no subclasses in the hierarchy. If the update is only to add a new concept to the ontology, there is no possibility of high risk errors. If the update changes or deletes an existing concept or relation then errors occur. To prevent such errors and reduce the risk of such errors remains an open question that deserves further research.

Fig3 Structure of Ontology Management Service



4.7 Composition of T-Services Using Brokers

One of the most promising mechanisms of service collaboration is service brokers. A test broker is just a T-service composition and itself is a T-service as well. There may be multiple test brokers owned by different vendors. We have developed a prototype test broker to demonstrate the feasibility of the approach. It receives test tasks from service requesters, decomposes a test task into a sequence of subtasks, sets a test plan and then invokes the T-services according to the plan to carry out the subtasks and passing information between them. Finally, it assembles the results from the services and reports to the service requester. The broker is composed of the following four modules.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

4.7.1 Communication Module

Communication Module provides an interface to the users. It receives test requests in the form of test tasks and sends out test results also in SOAP format. It transfers test tasks to Task Analyzer and gets test results from the Task Execution Module. Failures to fulfill test requests are also reported to the requesters through this module.

4.7.2 Task Analyzer

Task Analyzer decomposes a test task into several subtasks and produces test plans according to codified knowledge of software testing processes. It also keeps the track record of test plan executions for each task so that back tracking can be made when a subtask fails.

4.7.3Tester Search Module

Tester Search Module searches for testers for each subtask in the test plan generated by the Task Analyzer. Task Execution Module executes the test plan by invoking the testers and passing information between them. A failure to carry out a subtask is reported to the Task Analyzer and an alternative tester will be employed if any, or an alternative test plan is generated if possible.

4.7.4 Template in the test task

The knowledge-base of software testing processes plays a central role in the test plan generation. It can be considered as a finite set of templates of test plans with parameters like task, input, and output artifacts. A test task is then checked against the templates one by one and a test plan is

Produced by instantiating the template when a match is found. Each template can be regarded as a collaboration pattern of T-services. They can also be regarded as heuristic rules about how to compose and coordinate T-services. This significantly reduces the size and complexity of the space in which T-services are searched for and combined. Therefore, the complexity of T-service composition and collaboration can be reduced. It is worth noting that test tasks and capabilities have the similar structure and the corresponding semantics so that test requests can be easily transformed into search requests similarly, testers' capabilities can be transformed into test subtasks according to the test plan and submitted to the testers.

V. CONCLUSION

In this paper we presented service oriented architecture for testing WS. In this architecture, various T-services collaborate with each other to complete test tasks. We employ the ontology of software testing STOWS to describe the capabilities of T-services and test tasks for the registration, discovery, and invocation of T-services. The knowledge intensive composition of T-services is realized by the development and employment of test brokers, which are also T-services. We implemented the architecture in Semantic WS technology. Here we specify how to develop service specific T-services to support the testing of a WS. Experimental evaluation also shows the scalability of the approach.

REFERENCES

[1] F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D.Orchard, Web Services Architecture, W3CWorkingGroup Note, http://www.w3.org/TR/ws-arch, 2004.

[2] M. Stal, "Web Services: Beyond Component-Based Computing,"Comm. ACM, vol. 45, no. 10, pp. 71-76, Oct. 2002.

[3] G. Canfora and M. Penta, "Service-Oriented Architectures Testing: A Survey," Software Eng.: Int'l Summer Schools (ISSSE 2006-2008),Revised Tutorial Lectures, A. Lucia and F. Ferrucci, eds., pp. 78-105, Springer-Verlag, 2009.

[4] M. Bozkurt, M. Harman, and Y. Hassoun, Testing Web Services: A Survey," Technical Report TR-10-01, Dept. of Computer Science, King's College London, Jan. 2010.

[5] X. Bai, W. Dong, W. Tsai, and Y. Chen, "WSDL-Based Automatic Test Case Generation for Web Services Testing," Proc. IEEE Int'lWorkshop Service Oriented System Eng. (SOSE '05), pp. 215-220, Oct. 2005.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

[6] W. Tsai, R. Paul, W. Song, and Z. Cao, "Coyote: An XML-Based Framework for Web Services Testing," Proc. IEEE Int'l Symp. High Assurance Systems Eng. (HASE '02), pp. 173-174, Oct. 2002.

[7] N. Looker, M. Munro, and J. Xu, "WS-FIT: A Tool for Dependability Analysis of Web Services," Proc. 28th Ann. Int'l Computer Software and Applications Conf. (COMPSAC '04), pp. 120-123, Sept.2004 [8] J. Offutt and W. Xu, "Generating Test Cases for Web Services Using Data Perturbation," SIGSOFT Software Eng. Notes, vol. 29,no. 5, pp. 1-10, Sept. 2004.

[9] S.C. Lee and J. Offutt, "Generating Test Cases for XML-Based Web Component Interactions Using Mutation Analysis," Proc. 12th Int'l Symp. Software Reliability Eng. (ISSRE '01), pp. 200-209, Nov. 2001.

[10] W. Xu, J. Offutt, and J. Luo, "Testing Web Services by XML Perturbation," Proc. IEEE 16th Int'l Symp. Software Reliability Eng. (ISSRE '05), pp. 257-266, Nov. 2005.

[11] M.P. Emer, S.R. Vergilio, and M. Jino, "A Testing Approach for XML Schemas," Proc. 29th Ann. Int'l Conf. Computer Software and Applications Conf. (COMPSAC '05), pp. 57-62, July 2005.

[12] A. Bertolino, J. Gao, and E. Marchetti, "XML Every-Flavor Testing," Proc. Second Int'l Conf. Web Information Systems and Technologies (WEBIST '06), pp. 268-273, Apr. 2006.

[13] A. Bertolino, J. Gao, E. Marchetti, and A. Polini, "Systematic Generation of XML Instances to Test Complex Software Applications," Rapid Integration of Software Engineering Techniques. Guelfi, et al., eds., pp. 114-129, Springer, 2007.

[14] A. Bertolino, J. Gao, E. Marchetti, and A. Polini, "Automatic Test Data Generation for XML Schema-Based Partition Testing," Proc.Second Int'l Workshop Automation of Software Test (AST '07), p. 4, May 2007.

[15] J.B. Li and J. Miller, "Testing the Semantics of W3C XML Schema,"Proc. 29th Ann. Int'l Computer Software and Applications Conf.(COMPSAC '05), pp. 443-448, July 2005.

[16] L.F. de Almeida and S.R. Vergilio, "Exploring Perturbation Based Testing for Web Services," Proc. IEEE Int'l Conf. Web Services (ICWS '06), pp. 717-726, Sept. 2006.