Volume 3, Issue 5, May 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Collabrative testing of web service

V.Chanrasekar
B.Tech Student
(Department of Computer Science and Engineering)
Bharath University
Chennai, ,India

S.John arul raj
B.Tech Student
(Department of Computer Science and Engineering)
Bharath University
Chennai, ,India

Abstract— Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs (errors or other defects). Software testers are confronted with great challenges in testing Web Services (WS) especially when integrating to services owned by other vendors.

Keywords: software testing, testing web service.

I. Introduction

The paper presents a prototype implementation of the framework in semantic WS and demonstrates the feasibility of the framework by running examples of building a testing tool as a test service, developing a service for test executions of a WS, and composing existing test services for more complicated testing tasks. Experimental evaluation of the framework has also demonstrated its scalability.

II. Related works:

Data Flow Testing of Service-Oriented Workflow Applications
Lijun Mei

WS-BPEL applications are a kind of service-oriented application. They use XPath extensively to integrate loosely-coupled workflow steps. However, XPath may extract wrong data from the XML messages received, resulting in erroneous results in the integrated process. Surprisingly, although XPath plays a key role in workflow integration, inadequate researches have beenconducted to address the important issues in software testing. This paper tackles the problem. It also demonstrates a novel transformation strategy to construct artifacts. We use the mathematical definitions of XPath constructs as rewriting rules, and propose a data structure called XPath Rewriting Graph (XRG), which not only models how an XPath is conceptually rewritten but also tracks individual rewritings progressively. We treat the mathematical variables in the applied rewriting rules as if they were program variables, and use them to analyze how information may be rewritten in an XPath conceptually.
Service Oriented Architectures Testing: A Survey

Gerardo Canfora and Massimiliano Di Penta

Testing of Service Oriented Architectures (SOA) plays a critical role in ensuring a successful deployment in any enterprise. SOA testing must span several levels, from individual services to inter-enterprise federations of systems, and must cover functional and non-functional aspects. SOA unique combination of features, such as run-time discovery of services, ultra-late binding, QoS aware composition, and SLA automated negotiation, challenge many existing testing techniques. As an example, run-time discovery and ultra-late binding entail that the actual configuration of a system is known only during the execution, and this makes many existing integration testing techniquesinadequate. Similarly, QoS aware composition and SLA automated negotiation means that a service may deliver with different performances in different contexts, thus making most existing performance testing techniques to fail.
An Approach for WSDL-Based Automated Robustness Testing of Web Services

Samer Hanna and Malcolm Munro

Web Services are considered a new paradigm for building software applications that has many advantages over the previous paradigms; however, Web Services are still not widely used because service requesters do not trust services that are built by others. Testing can be used to solve part of this problem because it can be used to assess some of the quality attributes of Web Services. This chapter proposes a framework that can be used to test the robustness quality attribute of a Web Service. This framework is based on analyzing the Web Service Description Language (WSDL) document of Web Services to identify what faults could affect the robustness attribute and then test cases were designed

to detect those faults. A proof of concept tool has been implemented and experiments carried out that show the usefulness of this approach.

Web Services Toolkit Interoperability

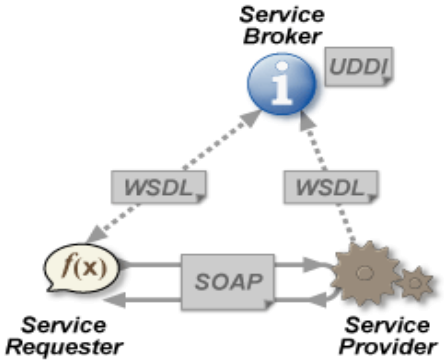
Ivan R. Judson

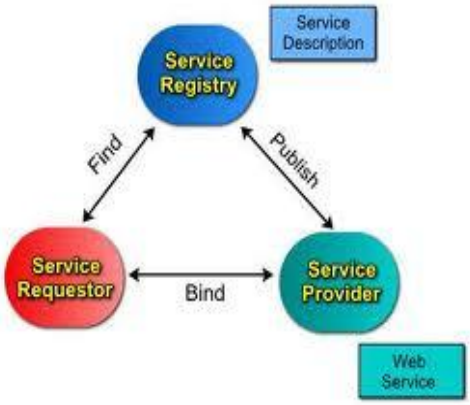
Web services, using WSDL and SOAP and following the WS-I's Basic Profile 1.0, have become the lingua franca for building service-oriented systems. Until recently, the development of tools for Web service took a significant amount of time, hindering the deployment and thus the adoption of Web services as a real technology. With the advent of the Web Services InteroperabilityOrganization (WS-I) and the resulting Basic Profile specifications, which provide guidelines for interoperable Web services, toolkits can now adhere to a set of conventions that can enable interoperability. We examine five Web services toolkits and evaluate their interoperability on simple test cases. Our goal is twofold: o report on the effort needed to produce interoperableservices using these five toolkits, and to be sufficiently rigorous.

Concept:

This paper proposes a framework of collaborative testing in which test tasks are completed through the collaboration of various test services that are registered, discovered, and invoked at runtime using the ontology of software testing STOWS.

Web Service Architecture:





Web service in Cloud Architecture:

Module description:

List of Modules:

1. **Test/Functional Service Generation**
2. **Test ase generation**
3. **Result Checking**
4. **Report Preparation**

Test/Functional Service Generation

F-service should be accompanied with a special T-service so that test executions of the F-service can be performed by the corresponding T-service. Thus, the normal operation of the original F-service is not disturbed by test requests and the cost of testing are not charged as real invocations of the F-service.

The F-service provider can distinguish real requests from the test requests so that no real world effect is caused by test requests. F-service should also provide further support to other test activities. For example, the formal specification of the semantics of the service, the internal design, such as UML diagrams.

III. Test case generation

Besides the service specific T-service that accompanies anF-service, a test service can also be a general purpose testtool that performs various test activities, such as testplanning, test case generation, and test result checking,etc. A general purpose T-service can be specialized incertain testing techniques or methods such as the generationof test cases from WSDL. The test broker TB decomposes the test task into asequence of subtasks and searches for appropriate testersfor each subtask by submitting search requests to theregistry. It then selects one tester for each subtask. In thisexample, we assume two testers TG and TE are selected.

IV. Result Checking

After checking the trustworthy of testerTG, the insurer A's T-service releases its design model toTG. After successfully obtained the design model, TGproduces a set of test cases and returns a test suite to broker TB. The test broker then passes the test cases toTE, requests for the test invocation of the insurer A'sservices using the test cases and requests it to check theoutput correctness and to measure the test coverage. TEperforms these tasks by collaboration with the insurer A'sT-services. The test results are then returned to the testbroker TB.

V. Report Preparation

Finally, TB assembles a test report containinginformation about test output correctness and test adequacy.The test report is sent to CIB, which is used todetermine whether the dynamic link will take place.

VI. 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. Case studies have demonstrated the feasibility of the architecture and illustrated how to wrap up general purpose testing tools and turn them into Tservices and how to develop service specific T-services to support the testing of a WS.

Reference:

[1] F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard, Web Services Architecture, W3C Working Group 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.

