# A FRAMEWORK FOR TESTING SOA APPLICATIONS

Samia Oussena[1], Balbir Barn[2] and Dan Sparks[1]
[1]*Thames Valley University, U.K.*
[2]*Middlesex University, U.K.*

Keywords: Testing, Test-driven Development, Distributed Architecture, Framework, SOA, BPEL, WSDL, XML.

Abstract: Test driven development (TDD) is emerging as one of the most successful developer productivity aids. A key feature of TDD is the construction of a robust test harness before implementing code, enabling the creation of a "virtual contract". The semantics of that contract are the fully enumerated set of test conditions for the system under construction. Service Oriented Architecture (SOA) raises a particular challenge in that there exists no unified method for testing an SOA application, which not only looks at individual artefact of the SOA application but also the complete application. Further, in SOA, the flexibility and connectivity provided by loosely coupled services increases both the opportunity for errors and the complexity of the testing. Given this situation, this paper describes a unified test-driven approach to a ground-up implementation of an SOA application where testing is seen as central to the development process. The paper proposes a framework that focuses on process-, configuration-, and service-oriented testing that provides relatively complete and flexible viewpoints of an SOA artefact's health. A critical evaluation of our approach is presented in the context of the development of SOA applications that support core Higher Education business processes.

## 1 INTRODUCTION

SOA is gaining industry-wide acceptance and usage. Typically a system developed in SOA will consist of a collection of finer grained services put together through an orchestration. The services themselves may consist of even more finely-grained web services. A number of client applications will be developed to consume the service and interact with the end user, for example; a web portal, web application or a fragment of a workflow application. This loose coupling of SOA services presents additional challenges to testing.

SOA's architecture of loosely coupled services provides increased opportunities for errors and therefore the complexity of the testing increases dramatically. Execution patterns are dynamic, distributed, and by their nature are not easily repeatable. The orchestration of services within an SOA system is itself also dynamic i.e. the selection of services can be done at the run time and therefore adds further complexity to the testing process.

Few practical attempts have been made to address the problem of SOA testing. Those that have been made have mainly addressed Web Service functionality. Tsai et al (Tsai 2002) proposed extensions to WSDL to allow for the testing of the services described by the use of invocation sequences, input-output dependencies, hierarchical functional description, and sequence specification. Other approaches have included multi-phase, iterative testing (Tsai 2004, Bloomberg 2002), and so-called rapid testing, involving regression testing, pattern verification, and group testing (Onoma 98, Tsai 2003). Lenz et al have proposed a model driven testing to SOA application, but here as well the main focus in on the web services (Lenz 2007).

All these investigations, however, have taken a more focussed approach to individual parts of the SOA, or even components within those parts, with a particular focus on the Web Service interface. It is recognised that there are many parts to the testing of a Web Service; however, this is still only concerned with the web services, and each service is only looked at in isolation. Integration testing is rarely considered. Orchestrations and how services are integrated to provide the component parts of an application should also be taken into account. To develop a comprehensive testing solution, one must take into consideration all aspects of SOA. To fail to consider deployment variables, for instance, is to only partially address the issues involved even in testing only a Web Service, and as such we believe that only a holistic approach can achieve this. Tool vendors, such as Parasoft and Empirix (Parsoft 2009, Empirix 2009) have started to address the issues of SOA testing. Persisting problems of solutions proposed by tool vendors are, in our opinion, either a lack of integration with open-source IDEs, or too much integration; tying developers into one IDE for all development work. Although for example, Parasoft's approach is relatively exhaustive, again there is a focus on Web Service testing only; BPEL testing is not covered. The reason may be that a lack of maturity of the technology coupled with the problems associated with working with constantly evolving standards and protocols make testing tool development difficult.

In this paper, we propose a testing framework for SOA that will provide a holistic approach. Successful testing needs to have as near to total coverage as possible of all the artefacts that are produced in the development of an SOA system. This framework will be the basis for combined automated and methodological support in testing such a system.

Most recent development trends incorporate some level of unit testing and test first approaches. Therefore any framework would need to be aligned with the test-driven development process. The framework will need to make testing an integral part of the development. One reason that test-driven development process has become so widely accepted is that in addition to the natural integration of the development and testing, there is provision for automation support for unit testing. With continuous integration, as requested by most agile processes (Beck 2000, Cockburn 2001), automation of unit tests is a must. The framework has therefore to provide and identify the expected level of automation support

## 2 THE AGILE APPROACH TO TESTING

Traditionally testing has been done separately from the development work, by a separate team. Systems today tend to be developed using iterative development methodologies, which make the traditional model of testing costly, and increasingly ineffective. Agile approaches to testing have strong links with SOA, coming, as they do, from a heavily OO-influenced sector. The separate services that the SOA artefact utilises can be viewed in much the same way as a software package, module, or component, and consequently, we feel, should reap the same benefits from an agile approach. It is interesting to note how well the technique of test-first coding, now considered to belong to the realm of smaller projects using agile development methods, lends itself to an SOA approach. The most likely reason for this is that although the domain is a vast canvas, breaking the domain into service components implies a modular approach, similar to a number of smaller projects. The overall management of the project can only be undertaken with safeguards, and in particular tests, in place at the service level.

Such a comprehensive approach to testing may not be a problem in what is viewed as the 'target audience' for SOA. Much discussion about SOA concerns the reuse and exposure of Legacy applications as services. These are systems which already work. The difficulty in SOA-enabling them is in wrapping and exposing them via a WSDL specification, and then orchestrating that via BPEL. Hopefully, the general programming logic has been cleared of most (if not all) bugs.

Developing an SOA application from the ground up, there are many more factors to consider, and therefore a more comprehensive approach to testing is required. It is vital to monitor and constantly assess the health of all newly written artefacts. Presumably in any reasonably-sized development there will be teams taking responsibility for different services, for the exposure via WSDL, and for the orchestration. Even so, those teams need to ensure they have testing that covers the entire range of functionality for their code. All these parts feed into the orchestration process to identify the bugs with a minimum of effort and frustration.

Test design, where possible, needs to be addressed at an early a stage as possible; ideally at the modelling stage. The automation of some of the development activities means that it is not always possible to do this. For example, WSDL creation must wait until XML schemas have been generated. This means it is not possible to create full test cases at the modelling stage.

In summary, there is a parallel between SOA development and agile development processes. The need for comprehensive testing in the development of ground-up web services, should, we feel, be able to be adequately addressed by an agile, unit-test-driven approach. This approach, however, must be integrated with the SOA development process to take into account the automation of artefact creation.

## 3 REQUIREMENTS FOR A SOA TESTING FRAMEWORK

The framework has been developed as a result of the difficulty we faced when developing an SOA artefact (Barn 2006) and was then refined and used in two further projects. We have so far identified a number of requirements due to the nature of SOA applications. These are:

*R1) Our development using SOA involved working with constantly evolving standards and protocols. Our framework will therefore require a modular approach, as well as implementing different levels of abstraction.*

*R2) In order to be integrated in a test-driven development process, the framework needs to provide and identify the expected level of automation support.*

In the rest of this section we discuss the requirements of the framework with reference to the problems we have encountered, and needs we have identified during our development. We will examine the testing activities from two perspectives; the development of the application, and the application deployment.

### 3.1 Development Requirements

It is most likely that a variety of Integrated Development Environments (IDE) will be used. The different artefacts that make up an SOA application can often require a specific IDE. In our case, we needed to work with XML Schema, BPMN (Business process management notation) and BPEL processes, WSDL creation and code generation from the WSDL and Database development

*R3) The framework needs to accommodate all the artefacts (or components) that will be a part of the SOA application's development and deployment.*

To successfully deploy our BPEL processes, we had to use an IDE that supported deployment to the Oracle BPEL Process Manager (Juric 2006), which in this case, was Oracle JDeveloper in two projects. In another project we used Intalio Designer and Intalio Server of the development and the deployment of BPMN processes (Barn 2008). Similarly, we needed to support UML and XML Schema generation and WSDL to Java code generation. These capabilities were provided by IBM Rational Software Architect (Quatrani 2006). The framework therefore needs to address the fact that different and indeed incompatible IDEs will be required during the development.

*R4) The design of the framework requires an IDE-independent approach to be taken. This includes the automation of test, and test data, creation, which needs to reach a level congruent with that expected by developers.*

### 3.2 Deployment Requirements

With distributed elements, deployment of services will almost certainly not be uniform for an SOA artefact. As with IDEs, certain services will require specific server environments. All our web services have been deployed to a Tomcat 5.5 application server, running behind an Apache 1.3 web server (Tomcat 2009), using the mod_jk Tomcat/Apache module (Mod_jk 2009) as a connector between the two servers. The advantage of this approach is that the Apache server routes all requests for web services to the appropriate application server. It is consequently possible to change the port or application server that contains any or all web services without any client needing to be aware of such a change. The successful deployment of a web service in this environment relies not only on correctness of the component but also each server's correct configuration.

*R5) The configuration of server environments therefore also needs to be factored into our testing framework.*

Another key consideration when deploying an application is the classloading. This is typically done by either adjusting the server's startup classpath, or putting the shared libraries in a special library directory. Our database was an implementation of Oracle 10g XML DB (Scardina 2004). The services that used this needed specific libraries – in addition to standard JDBC libraries, they needed the XML DB libraries installed in their server environment

*R6) The server's classpath and libraries configuration needs consideration within the testing framework.*

Physical deployment of the application components to the servers might be server specific. For example, developed BPEL processes are deployed to local instances of the Oracle BPEL

Process Manager. This provides a web interface similar to Tomcat's where a BPEL suitcase (also known as a suitecase) JAR (Java Archive) file containing the packaged BPEL specification and related code can be uploaded. We found though that this method of deployment did not deploy the necessary JSP files needed for User Tasks; this was achievable by deploying directly from Oracle's JDeveloper, which became our IDE of choice for BPEL development. This was similar for the Intalio XForms.

*R7) Because technologies may require certain environments, we must be sure to accommodate them all. Our testing framework must have a flexible, environment-independent approach.*

Having distributed resources and components require that we test the accessibility of each resource used by a number of artefacts, including WSDL, database and BPEL. In our case the schema resides on the same server as our services, however, this is not to say that they could not be deployed on a different server, or servers. Ensuring the availability of schema is another element of the framework's deployment requirements.

As another example, most business processes will require some human intervention, which is represented in BPEL by User Tasks, presented to users in the form of a workflow application. Supposedly this could be deployed to any server, though in our case we made use of the Oracle workflow application provided by the same server that hosted our BPEL processes.

*R8) The accessibility of resources (particularly where distributed) needs to factored into the testing framework.*

Even in this relatively simple scenario, the deployment requirements are not trivial. If not taken care of as seriously as the development requirements, a simple problem in one part of the deployment configuration may jeopardize the reliability of the entire application.

We next present elements of the proposed framework. Although we present in a pseudo-chronological order, it is important to take into account that the testing is iterative, for each stage, and also for the entire process.

## 4 THE FRAMEWORK

We have identified that there are a number of testing requirements that need to be considered to achieve our holistic testing approach. We now present our proposed framework (Figure 1) as a series of elements that can meet these requirements.

Our framework contains the identified testing elements, contextualised to facilitate understanding and cohesion of the test-driven development process. Each element appears individually, but is also grouped with other elements by criteria modelled on that used in the Zachman Framework (Zachaman). Elements are grouped by different architectural areas. These areas typically require different and specific types of testing such as data, function and user interaction.

Elements are further contextualised by their place within the development lifecycle. While the process in the framework seems largely linear, we have allowed for iterations within the process, as well as the possibility of iterations of the entire process. The testing 'checkpoints', or phases, are pre-implementation, post-implementation and post-deployment. These contain the testing elements of the framework that drive the development of the next stage. In this way, we lay a foundation for a comprehensive test-driven SOA development process.

We show in our diagram that Method Unit Test Cases should take place at the pre-implementation phase. They are directly informed by the outputs of the conceptual and logical phases (in this case a service or component model), and they in turn drive the development of the physical phase's artefacts, such as service or interface implementation, WSDL, and generated code. We have also made it clear within the framework that the purpose of these particular tests is to meet the functional requirements of the project.

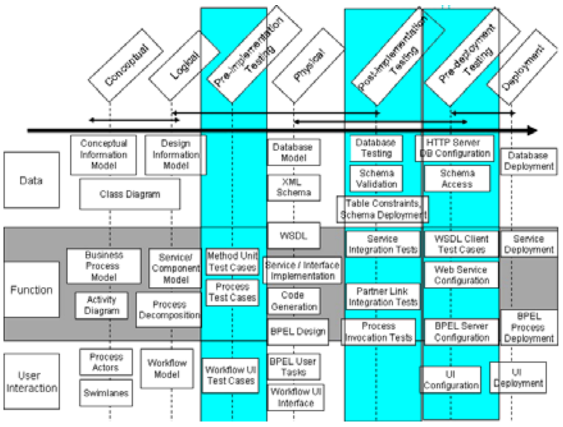

Figure 1: Framework for testing SOA applications.

Our identified user interaction configuration tests, on the other hand, need to be written at pre-deployment testing status, and this time the framework indicates that these tests relate to the project's user interaction requirements.

Each test element can be similarly interpreted. The framework provides a clear indication of the purpose, and function of each test element, and from this information it will be possible to identify and allocate resources to, and responsibility for, each particular area of testing.

## 4.1 WEB Service Testing

Unit tests were written for each service, testing the exposed web service methods, but with more of a focus on internal functionality. Appendix 1 shows a fragment of code from such a test. This meant that any changes to code on a service could easily be checked. On identification of a problem, the involved services can have their unit tests run against their code; if the tests pass, the problem is elsewhere. If a test fails, the problem is easily located and solved.

Client 'applications' were also developed for each service, this time with a focus on the exposed methods of the web service. These applications could be viewed as 'external test suites', and, when packaged into an overall test with the internal web service unit tests, provide a robust and helpful method of testing a web service. This approach has been taken further, by integrating the 'client' tests with the WSDL. A schema for a collection of test results is used to return test results via a WSDL method. By extending the JUnit framework [JUnit], we can run the unit tests built for the service whilst in its deployment environment.

## 4.2 BPEL Testing

The Oracle BPEL Process Manager provides a browser-based tool for testing BPEL processes. It is possible to run individual tests of a process, or to run a 'stress test', with varying data, to see how the process holds up under demand. Feedback from the tests comes in summaries of process states, visual flows of the processes, identifying places where the process has failed, and full audit trails of XML messages and data objects as they are created and passed back and forth.

This tool is very useful, but not without limitations. As our scenarios contain User Tasks, the processes tested need user interaction. From a testing perspective this is problematic: not only is it time consuming to have a tester physically involved, there is also the greatly increased risk of user error. A mistyped field can have serious consequences on the output of a given test, resulting in more time lost to testing, or the false identification of a bug that does not exist.

During our development process, we examined the Oracle BPEL API and the Intalio BPEL API with the aim of mechanising as many of our test cases as possible. This had the advantage of being less time consuming, and more error-free than human-operated testing. The approach was quite primitive, largely due to the amount of time and effort needed to ensure that sufficient test data was in place, and the time and effort required to maintain the integrity of that test data. We did not invest too much time as we were aware that in the latest release of the Oracle Business Process Manager (10.1.3) a BPEL test framework would be available. This now includes the automation of some process unit testing.

### 4.2.1 BPEL Test Framework

The newest version of the Oracle Business Process Manager (10.1.3) contains the BPEL test framework. It is possible to build an entire test suite within JBuilder. The results of these tests can be created in JUnit format, useful if other parts of an SOA application's testing utilises JUnit.

Creating a test suite is, on a basic level, as simple as following a click-through wizard. In practice, it is a little more complicated. External service interactions are spoofed by the Process Manager when running the tests, so it is important to ensure there have been adequate 'dummy responses' created for each interaction with an external service. Of course, these can be created to provide different responses depending on the data you want to start the test with.

Advantages to this approach are that the only thing being tested is the process itself – there is no confusion over an ambiguous problem being the fault of either a process or an external service. The logic of the process can be tested thoroughly without having to consider any influence from anything not contained within the BPEL code.

Disadvantages are that the data that one uses to start a test might not produce anything other than an error from an external service – the disconnect between external services and the process being tested means that it might be the case that the process is being tested to completion with completely useless data. Similarly, testing in this way does not take into consideration such things as invoking the correct service method, or even the correct service – such environmental issues are not examined by this method and will need to be explored elsewhere.

The BPEL Process Manager also provides a means for automated load testing of a process. The tester completes a small browser-based form specifying maximum concurrent threads, number of loops (repetitions of the concurrent thread execution), and the delay between each invocation, in milliseconds. Once run, detailed results of the load test are available. If this interface were to be extended it could provide a more comprehensive collection of testing approaches; for example, the sending of a range of input parameters, rather than just one. The recently released test framework for BPEL Process Manager builds upon this functionality. It is still not perfect – there are issues where complex processes that loop may fail to generate successful test cases. The testing is constantly being refined, however, and if it continues to do so, BPEL testing will become much easier, and a great deal less time-consuming.

## 4.3 UI Testing

Workflow applications, like most web applications, follow the Model View Controller (MVC) approach. Our views were programmed as Java Server Pages (JSPs) and Xforms, The UI testing therefore did not focus on logical behaviour, instead testing code which accessed and displayed attributes, loop iterations for dynamic creation of user input forms and the dynamic, event-based generation of pages. At this point, we can use HttpUnit [Httpunit] or a similar tool to validate HTML pages. Further testing of web-based interaction can be done through such applications as AutomationAnywhere (Automation Anywhere), which allows for the recording and subsequent replay of browser-based interaction. This kind of automation software is sparse on detail when recording the results of testing, and should be used in conjunction with logging software in order to identify causes of failure. A detailed description of web application testing is beyond the scope of this paper. There are many sources including (Dallaway, Link 2003) which discuss unit testing of web applications.

## 4.4 Database Testing

Persistency plays an important role in most applications and this is still true for SOA applications. The creation of unit tests for persistence mechanisms is often accompanied by major problems, because both the execution speed and the large number of dependencies make the testing approach difficult. It is therefore important to adopt an architecture that will not hinder testing; i.e. the persistency layer should not have unnecessary external dependencies and the ones that remain should easily be replicated in a test environment. Our persistency layer implementation has first followed the DAO pattern (DAO) and later used Hibernate framework (Hibernate), which are particularly good at eliminating unnecessary dependencies.

The other step that we have taken in our testing is to isolate each test by clearing the database before each test is run. This has been facilitated by the use of Dbunit framework (DBunit). Dbunit is an extension of JUnit that helps simplify testing database applications. Within this framework the database can be created and populated from an XML file, making it easy to manage the test environment.

Here as well a full detailed discussion of persistency unit testing is beyond the scope of this paper. There are many sources including (DBUnit, Freeman) which discuss unit testing the persistency layer.

## 4.5 Validation

Any schema or WSDL can also be checked for well-formedness, however, this will ideally be integrated within the development environment(s), and additionally automated. This will ensure that each time a change is made, any error is identified and a chance is given to quickly and easily address it. This is similar to functionality provided by many IDEs, where code is constantly compiled, and errors are pointed out to the developer at source.

## 4.6 Deployment Configuration Testing

While testing of artefacts within their deployed environment can give an overall picture of application health, the isolation of the deployment environment is extremely useful as it allows for focused examination of deployment issues. Testing should always identify problems quickly and efficiently. Assessing deployment configurations lets us quickly spot problems, or otherwise disregard the environment as being a cause of a problem. This has the beneficial effect of narrowing our investigation scope and guiding us to the cause of the fault we are investigating.

Within an SOA artefact, there are a great deal of factors to be tested relating to deployment. We have already identified these in our requirements analysis, and here we provide our approaches to the testing of two of these; the need for a server to contain the required libraries, and resource naming and lookup.

### 4.6.1 Library Configuration Testing

The machines running certain servers, database implementations, and services, require specific libraries. These libraries are in turn required by specific server such as database, or service implementations, though any combination of these may share libraries. While this may appear to be an administrative nightmare, the unit test approach provides a simple and manageable solution.

For each library, tests can be written that call a method or methods from the library. If these tests pass, it can be assumed that the library is installed correctly in the environment. Each test can then be integrated into a deployment test suite for a specific server, database, or service. This may provide some redundancy, but in this case this is not necessarily a bad thing, as it removes the burden of test reconfiguration with each deployment of a new service, database or server to an environment. Instead, we can provide specific test cases which can be combined as required, allowing for a greater degree of flexibility.

The automation of this approach, for example some kind of linking between an artefact's classpath and the deployment tests required for it, will add more value to the process in terms of flexibility, ease of use, and reduced test configuration and implementation time. We intend to explore this further in future work.

### 4.6.2 Resource Naming and Lookup Configuration Testing

We have identified that, in addition to library requirements, deployment environment testing needs to address correct resource naming and lookup. Similarly to library testing, resource testing is specific to the environment as a whole, and to individual service, database, and server implementations, and again, a unit testing approach can achieve the flexible and comprehensive testing solution required.

Again, the approach adopted for library testing can be used. This approach is modular not only in terms of one aspect of the deployment testing, be it library or resource configuration testing, but also for the deployment testing overall. The test suites, in addition, can be combined to provide views of overall artefact deployment health, and again these views can be combined with previously discussed approaches to provide the needed holistic, multi-layered, multi view solution to SOA testing.

## 5 CONCLUSIONS

Testing methods for SOA are underdeveloped. In this paper, we have proposed a unified approach to SOA testing. It is our opinion that such a testing framework could have a significant impact on the SOA development community.

Providing a framework with divisional foci allows for analysis at varying levels. Because we can take different views, it is possible to quickly identify and isolate causes of problems. This also helps the development process; the test cases provide requirements, which inform the development of artefacts. The integration of artefacts is supported because tests for each artefact already exist, and can be used in varying combinations for different integrations. Additionally, the iterative SOA development process is made easier.

Recognising that we have only looked at the functionality of SOA applications, our future work will consider expanding the framework to take into account non-functional aspects such as quality of service, security, and performance. We are also considering implementing the framework and providing a model driven approach to testing.

## REFERENCES

Apache Server http://httpd.apache.org

AutomationAnywhere http://www.automationanywhere.com

Barn B.S Oussena S (2008) "BPMN, Toolsets and Methodology: A case study of business process management in higher education" In: ISD'08 conference

Barn, B.S., Dexter, H., Oussena, S. Petch, J.: An Approach to Creating Reference Models for SOA from Multiple Processes. In: IADIS Conference on Applied Computing, Spain 2006.

Beck, K. *Extreme Programming Explained: Embrace Change*, Boston: Addison-Wesley, 2000

Beck, K., *Test-Driven Development: By Example*. Boston: Addison-Wesley, 2003

Bloomberg, J., Web services testing: Beyond SOAP, ZapThink LLC, Sep 2002

Cockburn, A. *Agile Software Development*, Boston: Addison-Wesley 2001

Dallaway, R., JSP test http://www.dallaway.com/jsptest/

DAO Pattern http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html

DBUnit http://sourceforge.net/projects/dbunit/

Empirix, http://www.empirix.com/

Freeman,S., Developing Jdbc Applications Test-First http://www.mockobjects.com/DevelopingJdbcApplicationsTestFirst.html

Hibernate: http://docs.jboss.org/ejb3/app-server/Hibernate3/reference/en/pdf/hibernate_reference.pdf

HttpUnit http://httpunit.sourceforge.net/

JUnit http://www.junit.org

Juric., M. *Business Process Execution Language for Web Service, 2nd ed.*, Birmingham, Packt, 2006.

Lenz C., Chimiak-Opoka,, J. Breu R.. “Model driven testing of soa-based software” In Daniel Luebke, editor, *Proceedings of the* SEMSOA Workshop 2007 *on Software Engineering Methods for Service-Oriented Architecture*, volume 244 of *CEUR Workshop Proceedings (ISSN 1613-0073)*, pages 99-110, Hannover, Germany, May 2007

Link, J. and Frohlich, P., Unit Testing in Java: How Tests Drive the Code, Morgan Kaufmann Publishers, 2003

mod_jk Tomcat Connector http://tomcat.apache.org/connectors-doc/

Onoma A.K., Tasi W.T., Poonawala M. and Sugama H. Regression testing in an Industrial Environment, Communication of ACM, Vol. 41, No 5, 1998, 81-86

Parasoft, www.parasoft.com

Tsai, W. T., Paul, R., Wang, Y., Fan C., and Wang D., “Extending WSDL to Facilitate Web Services Testing”, Proc. of IEEE HASE, 2002, pp. 171-172.

Quatrani, T. and Palistrant, J. *Visual Modeling with IBM Rational Software Architect and UML,* Pearson Education, 2006

Scardina, M., Chang, B., and Wang, J., *Oracle Database 10g XML & SQL: Design, Build, and Manage XML Applications in Java, C, C++ & PL/SQL*, McGraw-Hill, 2004

Tsai W. T., Zhang D., Chen Y., Huang H., Paul R., and Liao N., “A Software Reliability Model for Web Services”, the 8th IASTED International Conference on Software Engineering and Applications, Cambridge, MA, November 2004, pp. 144-149.

Tsai W. T., Paul R., Yu L., Saimi A., and Cao Z., “Scenario-Based Web Service Testing with Distributed Agents”, IEICE Transactions on Information and Systems, 2003, Vol. E86-D, No. 10, 2003, pp. 2130-2144

Tomcat Server http://tomcat.apache.org

Zachman Institute for Framework Advancement http://www.zifa.com