



Parasoft UK Ltd.

1st Floor, Orchard Villa
Porters Park Drive
Shenley, Radlett
Herts, WD7 9DS
Tel: +44 1923 858005
Fax: +44 1923 858329

Ensuring SOA ROI: Testing Service-Oriented Architectures with Parasoft SOAtest



Introduction

Service-Oriented Architectures (SOA) have gained much attention as a unifying technical architecture that can be concretely embodied with Web service technologies. SOA is a design model deeply rooted in the concept of encapsulating application logic within services that interact via a common communications framework. A key aspect of the Web Service incarnation of SOA is that the Web Service is viewed as a fundamental building block of a SOA-based application. As more and more Web services integrate into core processing systems, the reliability and performance of these Web services becomes a primary goal for businesses. Companies must ensure that the Web services deployed are as sound as possible. Outages for critical components in customer services or authorizations could cost companies millions of pounds per hour.

Therefore testing, which is important for any Web application, is even more crucial for Web services. Extensive testing of Web services, particularly those that are externally facing and business critical, is essential to securing the enterprise from significant business risks. A down time of an hour can not only cost substantial losses in revenue but, more importantly, the perceived lack of quality and reliability of the company in general. Any mission-critical Web service must be strictly tested and verified as functioning correctly, 24 hours a day, seven days a week – no exceptions. Fortunately, there is a solution in adhering to the strict guidelines necessary for testing your Web service.

By utilizing Parasoft SOAtest throughout the software development lifecycle, you can prevent errors early in development, improve quality and reliability, and accelerate time to market. The sooner you detect a problem, the easier it is to fix it. SOAtest provides a wide variety of tools and testing techniques that immediately and thoroughly exercise Web services and check their reliability. This paper will explain issues specific to Web services and will illustrate how Parasoft SOAtest, an automated tool for testing Web services, can ensure complete Web service functionality, interoperability, and security. By applying SOAtest throughout your team and development process, you establish an environment where reliable Web services can be developed rapidly and efficiently, thereby securing your enterprise from the crippling effects of an error-prone Web service.

Common Web Service Testing Problems

There are many complexities specific to, and inherent in Web services that complicate testing. However, before delving into specific technical issues, it would be wise to first examine the general testing inefficiencies that can occur in any Web service, and how Parasoft SOAtest can prevent these problems from occurring. There are a number of testing inefficiencies that plague companies and cause severe business implications such as extending project timelines and increasing project costs. And for each of these problems, SOAtest provides a solution.

Inefficient Use of Developer Assets

Organizations continuously re-create the same test cases at different points of the development process. Developers create tests to ensure the functionality of their Web services; however these tests are not leveraged in the downstream process. This duplication of effort is not only inefficient, but also introduces an opportunity for errors to be injected.

SOAtest provides improved productivity and labour savings.

With automated generation of test cases, SOAtest saves significant time and resources as opposed to manual processes that may be currently employed.

Heterogeneous Testing Platforms

By re-creating test cases on different platforms there is a significant risk of extending the time it takes to verify functionality. For example, testers that discover errors in one format may go back to the developers who then try to reproduce these errors in their own format. This rework loop is time consuming and inefficient.

SOAtest is a uniform solution for Development, QA and Performance teams.

SOAtest allows test cases to be re-used, combined, and extended across teams, providing seamless transfer of knowledge and test case data. Functional and unit test cases created by development and QA can be used by the performance team to generate scenarios for use in load and performance testing, thereby eliminating the need for extensive script writing and script maintenance. SOAtest saves time and improves accuracy of test creation/execution.

Register for your Free evaluation today!
<http://www.parasoft.com/jsp/products/home.jsp?product=SOAP>

Development and Maintenance of Homegrown Tools

Tools, such as client emulators, are developed in-house in order to facilitate functional testing. At the same time, a load generation tool is purchased that requires proprietary scripts to be written. The creation and use of such homegrown tools and scripts not only adds overhead to the project, but such tools are also impossible to maintain as industry standards and Web service complexity evolve and mature.

SOAtest can automatically generate a suite of test cases using WSDL and HTTP traffic.

In addition, its ability to leverage tests from unit testing through load testing allows more test cases to be generated – ensuring greater coverage and quality of the service while minimizing business risks of production failure or production level error.

Incomplete Testing

The scope of testing provided by in-house applications is generally limited and does not cover all the aspects needed to verify a Web service. Aspects such as security, interoperability, functionality, reliability, and availability are not verified due to these limitations. As a result, businesses face an exponential increase in financial risk.

For example, rather than testing a Web service directly at the code level, organizations may utilize some GUI-based client to consume the Web service. However, dependence on GUI-based testing of a Web service is limited and restricted to only the types of messages the GUI-based client is built to generate. With most Web services, it is impossible to anticipate exactly what types of requests clients will send. Real world partners and consumers of a Web service are certain to send different types of requests, negative inputs, and bad data to the Web service that the GUI-based client has not even considered. In addition, maintaining and expanding a GUI-based testing application would result in numerous and unnecessary productivity losses in order to perform only limited, indirect testing of a Web service. In order to completely and thoroughly test a Web service, you must be able to emulate many of the types of clients that might access the server, and verify that the server will behave as expected in relation to any type of client request.

SOAtest increases the scope of testing by providing the flexibility to manipulate messages in order to directly and thoroughly test the Web service.

SOAtest can perform code based testing of the Web service and isolate and test the data at the API and message layer of the service. By testing the API and services layer, SOAtest allows developers to verify whether the Web service can

handle a wide range of request types and parameters. By checking for the conditions and inputs that are not expected, you enable more thorough tests for what cannot be foreseen. By performing such testing at the unit and application level, you can quickly and easily identify and correct any weaknesses before security breaches have the opportunity to occur.

Total and Holistic Approach to Testing Web Services

As organizations rely more heavily on Web Services to drive their business processes, it is critical for Web services developers to have a proven solution to test the complex layers of Web Services and to create an efficient and bulletproof testing process without being riddled with the common problems discussed earlier. The Parasoft Web Services Solution not only provides clear and practical guidelines for testing, but also provides a cohesive team workflow that makes the most efficient use of project assets across the organization. Parasoft SOAtest goes beyond the simple testing of individual operations of a Web service. SOAtest provides a total and holistic testing strategy for your Service Oriented Architecture.

Web services are usually part of a larger infrastructure that has multiple services, multiple endpoints, and multiple business processes. Merely testing a single service is not representative of testing the whole architecture of the SOA. Because SOAtest can quickly and easily emulate any endpoint (client or server) of a Web service, you have complete control for testing your entire SOA. This tremendous amount of flexibility allows you to create complex test environments that isolate as many points of failure as possible, thereby increasing the scope of testing. For example, to test a proxy, you can use SOAtest as both a client and server, sending traffic back and forth and verifying the policies and transformations taking place within the proxy. Or, to test a business process you can use SOAtest server mode to stub out all external Web service calls and use the client to drive the process.

SOAtest ensures that all facets of the Web Service, including interoperability, functionality, security, reliability, and availability, are extensively tested. Not only does SOAtest improve the breadth of Web services testing, but it also enables you to increase the amount of depth that is covered as well.

Register for your Free evaluation today!

<http://www.parasoft.com/jsp/products/home.jsp?product=SOAP>

Best Practices for Verifying and Testing Web Services

In order for a complete Web service to deliver optimum functionality, both the client and the service must satisfy a number of requirements. Interfaces must be correctly described in a WSDL document. Messages must conform to both the transport protocol specification (such as HTTP 1.1) and the message protocol (such as SOAP 1.1). Messages must also conform to the contract specified in the

WSDL describing the service, both in terms of the message content and the binding to the transport layer. Add to the mix security provisions, interoperability issues, and performance requirements under load, and it is easy to see why Web service testing is not a trivial matter. Fortunately, SOAtest provides the necessary know-how to meet these challenges head on.

SOA Policy Enforcement

Service-Oriented Architectures transform the IT infrastructure to increase business flexibility, connectivity, and control. However, the ability to attain the benefits of service orientation is largely constrained by the ability to manage the various SOA domains: security, management, registry, development, orchestration/composite services and enablement/integration. The lack of a solid SOA policy enforcement strategy throughout the entire service lifecycle can result in an inconsistent and uncontrollable IT infrastructure that compromises the benefits of SOA.

Companies may have any number of developers in an IT group working on different parts of a Web services implementation at any given time. With so many individuals working towards a common goal of a Service Oriented Architecture, the occurrence of errors increases exponentially. For example, when publishing a service to a UDDI, that service must be ready for consumption and reuse across business boundaries. However, various problems and vulnerabilities such as duplication, lack of compliance, and lack of interoperability, can occur if the development group does not take proper measures to follow specific guidelines and best practices. It is crucial for developers to have some sort of control or process for determining what services are published and becomes a part of the SOA being built.

In order to control how Web services are defined, developed, and deployed, companies utilize internal standards and policies, XML schemas, and various Web services specifications such as WS-I Basic Profile and WS-Security. However, it may be difficult for development teams to remain compliant to such standards and policies without anything enforcing such standards on their code. As enterprises extend their implementations of web services and SOA and ramp up their development efforts, establishing a policy enforcement process across the enterprise is paramount to ensuring that uncontrolled and inconsistent Web services development does not compromise the benefits of SOA.

SOAtest provides a complete Policy Enforcement solution, enforcing policies with executable rules that can be applied to WSDLs, Schemas, SOAP messages and any other XML artifact or SOA meta-data component.

Once an organization has defined their policies to guide their SOA deployments, SOAtest recognizes these policies and enforces them throughout the development and QA process. For example, SOAtest verifies schema and semantic validity for W3C and OASIS standards compliance, validates Basic

Profile 1.1 for WS-I Interoperability compliance, and implements rules to enforce various other endorsed WS* Standards. In addition, SOAtest enforces compliance to best practices such as customized company guidelines, security, and maintainability and reusability.

Rules to enforce custom corporate policies and standards can be developed with SOAtest's RuleWizard utility, providing the means to bring all web services development policies and practices under policy enforcement.

Register for your Free evaluation today!
<http://www.parasoft.com/jsp/products/home.jsp?product=SOAP>

WSDL Validation

WSDL validation can be considered as the first step in testing Web Services. Although WSDLs are generally created automatically by various tools, it doesn't necessarily mean that they are correct. When WSDLs are manually altered, WSDL verification becomes even more important. At the minimum, WSDLs should be checked for conformance to the WSDL schema via XML validation. Additional checks include interoperability conformance checks against WS-I's Basic Profile 1.1 and a regression test on the WSDL.

SOAtest automatically generates tests to validate the WSDL definitions, including schema verification.

Additionally, a WS-I test will be created to ensure the interoperability of the business contract. Lastly, SOAtest can maintain a regression of the WSDL in memory that will be compared to the latest version. This will enable teams across the board to ensure that they are building tests for the latest build.

Unit Testing

Following the validation of the WSDL, the next step in testing a Web service is to create unit tests. Each module of a Web service should be developed along with unit tests that verify its functionality. By requiring the development of unit tests before or during, and not after, the development of the code, it can be ensured that the code actually provides the necessary functionality. The span of the unit tests should be such that every functionality requirement of the module is verified. In addition, by using unit tests to drive development, unnecessary code that does not need to be written can be avoided. However, unit testing can be quite labor intensive if performed manually. The key to conducting effective unit testing is automation.

SOAtest automates unit testing and ensures quality throughout the software lifecycle.

SOAtest can also emulate various vendors' implementation of WS-Security, which will allow the user to verify the correctness of his security implementation. Additionally, SOAtest validates the business logic of the server response through various diff, scripting, and rule enforcer tools to create regressions. All artifacts are then saved in a file-based format so they can be easily checked into any source control system and made available to any team member.

Regression Testing

Once a unit test meets a requirement, it is frozen, and we can move on to the next requirement. The way to make sure that a requirement stays frozen is to check the unit tests into source control and run them as regression tests during the nightly test process. This means that once a unit test passes, it should always pass. There will be times when a change to a seemingly unrelated part of the code causes undesirable behavior in another part of the code. By having these unit tests that serve as regression tests, undesirable changes over time can be detected. As the functionality becomes increasingly complex, previous tests should continue to pass and new tests that test new functionality should be added.

SOAtest can automatically create regression controls.

During regression testing, SOAtest runs specified test cases, compares the current outcomes with the previously recorded outcomes, and then alerts you to differences between the current responses and control responses. Subsequent regression test runs will remind you of this discrepancy until the application returns the expected result.

Functional Testing

Unit tests created by the developers can be leveraged and extended into more complicated functional and scenario tests. This is accomplished by combining a series of operations into a complicated scenario that emulates potential client behavior.

SOAtest has the capabilities to re-create stateful transactions to build complicated scenarios for complete functional testing.

Additionally, test flow can be controlled by the defining logic in the test suite. This task is best suited for QA Engineers that work with clients on a daily basis and have the basic knowledge of each operation that is needed to complete a complex scenario.

Register for your Free evaluation today!
<http://www.parasoft.com/jsp/products/home.jsp?product=SOAP>

Performance Testing

Once functional tests have been developed, the Load Generation Team can specify the performance objectives for all the test cases produced by QA and Development and generate the load to the servers, rather having to spend months writing additional scripts to facilitate their testing. Load testing not only monitors the server's response rate with the specified number and mixture of simultaneous requests, but also verifies whether the test loads cause functionality problems.

SOAtest facilitates load testing by allowing you to load test your server functional test suite according to preset or customized load test scenarios.

Preset scenarios include bell curve, buffer test, linear increase, and steady load. You can easily customize these scenarios to use different test cases, load levels, load distributions, and so on. You can also distribute virtual users across remote server machines to simulate extreme loads and/or test from different locations.

In addition, SOAtest can create Windows Perfmon, SNMP, and JMX monitors to collect network information and system performance data during load testing. SOAtest also provides a Detailed Report option in which load test graphs, histograms, and errors can be saved as a Detailed Report that can be viewed after the load test completes. Within the Detailed Report, you also have the ability to select and view individual hits in graphical or table mode. A load test time interval can also be selected through the Detailed Report to view either errors or individual hits within that time frame.

Client Testing

SOAP client developers are responsible for making sure that the client sends requests correctly. If a client sends invalid or improperly formed requests, the server usually cannot deliver the expected results. The process of testing clients is a little different from testing services because clients are the initiators of Web service interactions. This means that from a testing standpoint, there are two main things to verify: whether the client can correctly initiate an interaction by sending a request, and whether the client behaves as expected when it receives a response.

No matter what type of server a client accesses, the same general principle applies: the client sends a request, the server responds, then client success or failure is determined by recording and verifying the request and/or by verifying the server response. The same techniques and tools used to verify server functionality can be used for this purpose.

SOAtest provides a Web service deployment feature that lets your workstation function as the server in your client-server relationship.

SOAtest instantly begins emulating a server based on a WSDL document. This lets you test client functionality and emulate server supplied responses driven by data sources. With SOAtest, you have the full capabilities to manipulate the server-side response. Emulating the server is especially useful when the server is still being implemented, has bugs, or should not be accessed during testing.

BPEL Process Testing

Insuring quality of real life BPEL deployments can be challenging. A non-trivial BPEL process can use multiple business partner links and reference multiple WSDL files that describe external resources. The number of BPEL process components that need to be tested to insure its overall quality can grow quickly as the complexity of the project increases. This makes manual testing of such processes extremely time consuming and inefficient in fast moving IT environments. The fact that BPEL and Web Services technologies are new, complex, and evolving makes the task of testing BPEL processes even harder.

SOAtest ensures the quality of BPEL deployments by allowing users to automatically create test cases from vendor-specific BPEL deployment artifacts and arranging these test cases into suites that reflect different aspects of testing of a BPEL process.

Tests generated fall into four categories: BPEL semantic tests, WSDL tests, BPEL process tests and BPEL Partner tests. Organizations using BPEL to optimize and emulate complex business processes can now more reliably and consistently ensure the quality and completeness of these processes.

Web Service Security

Because of the flexibility and connectivity provided by Web services, proper security measures must be taken to authenticate messages as well as prohibit outside parties from viewing private messages. In addition, Web services can expose customer, financial, and operational systems to security vulnerabilities through common software flaws such as XPath and SQL injections, XML bombs, and parameter fuzzing. Therefore, while being sure to practice the steps necessary for a Web service free of functional errors, developers must also practice the methods necessary to ensure secure Web services.

SOAtest includes security support for testing web services with security layers.

At the transport level, SOAtest supports SSL (both server and client authentication), basic authentication, and cookies for session management. At the wire level, SOAtest allows customizable and highly configurable SOAP Headers for whatever WSSecurity mechanisms you decide to employ. SOAtest has GUI forms for configuring X509, SAML, and Username security tokens and allows you to encrypt and sign your SOAP according to XML Encryption and XML Digital Signature specs.

SOAtest is also the only Web services solution that automatically creates security penetration tests that dynamically exercises and scans the Web service for security vulnerabilities.

By testing the Web service with penetration attacks and analyzing the responses, security vulnerabilities can be discovered and fixed earlier in the software development cycle. SOAtest supports the following penetration tests: Parameter fuzzing, SQL injections, XPath injections, XML bombs, external entities, malformed XML, invalid XML, username harvesting, large XML.

Test Management

The complexity of a Service-Oriented Architecture demands a large number of test cases to ensure and enforce proper functionality. As an organization's services evolve, these tests must run constantly to ensure that what was working in the past will continue to work in the future. The broad scope of test coverage, combined with the need for routine test execution, leads to the question of how to manage all of the testing collateral. Creating test cases for services is only one aspect of the testing solution. Managing dozens of test cases and aggregating the results can be challenging. Automating the execution of test cases and providing visibility to the results is the only way to ensure consistent coverage.

SOAtest provides several features to enable easy and automated maintenance of an organization's test cases.

SOAtest automates the execution of test cases and provides visibility to the results through its own test management and reporting capability. In addition to its own test management capabilities, SOAtest provides direct integrations with Mercury TestDirector and IBM Rational TestManager. Users of these test management platforms can now easily integrate, manage, and orchestrate the execution of SOAtest-generated test assets from within the TestDirector or TestManager environments. A powerful command line interface also allows easy integration of SOAtest and SOAtest test assets with existing test process or automated build environments such as ones supported by Ant and Cruise Control.

Conclusion

As seen throughout this paper, there are countless opportunities for things to go wrong during Web services development—a slight mistake in any component or interface will cause problems that ripple throughout the system. Developers must ensure that each part of the system is reliable, and that all of these parts interact flawlessly and securely.

Parasoft SOAtest is the most comprehensive tool for testing Web services and ensuring secure, reliable, and compliant Service-Oriented Architectures. SOAtest automates all the necessary practices that allow verification of all aspects of a

Web service, including four-tier WSDL validation, unit and functional testing of the client, unit and functional testing of the server, performance testing, as well as the ability to graphically model and test complex, multi-layered transaction scenarios and use cases. SOAtest addresses key SOA issues such as policy enforcement, interoperability, security, change management, and scalability. SOAtest features a collaborative workflow that allows any test asset created by an engineer to be leveraged by QA into scenario-based tests and load tests without the use of scripting.

By utilizing SOAtest throughout the software development lifecycle, users can prevent errors, improve quality and reliability, and accelerate the time to market for their SOA initiatives.

Register for your Free evaluation today!
<http://www.parasoft.com/jsp/products/home.jsp?product=SOAP>

About Parasoft

Parasoft is the leading provider of innovative solutions for automated software test and analysis and the establishment of software error prevention practices as an integrated part of the software development lifecycle. Parasoft's product suite enables software development and IT organizations to significantly reduce costs and delivery delays, ensure application reliability and security, and improve the quality of the software they develop and deploy through the practice of Automated Error Prevention (AEP). Parasoft has more than 10,000 clients worldwide including: Bank of America, Boeing, Cisco, Disney, Ericsson, IBM, Lehman Brothers, Lockheed, Lexis-Nexis, Sabre Holdings, SBC and Yahoo. Founded in 1987, Parasoft is a privately held company headquartered in Monrovia, CA. For more information visit: <http://www.parasoft.com>.

Contacting Parasoft

Tel: +44 (0)1923 858005
Email: sales@parasoft-uk.com

© 2007 Parasoft Corporation
All rights reserved. Parasoft and all Parasoft products and services listed within are trademarks or registered trademarks of Parasoft Corporation. All other products, services, and companies are trademarks, registered trademarks, or servicemarks of their respective holders in the US and/or other countries.