

## EXCERPT FROM ZAPTHINK REPORT

### TESTING WEB SERVICES

Report Date: August 23, 2002

Analyst: Jason Bloomberg



**Title:** Report: Testing Web Services

**Last Updated:** August 23, 2002

**Analyst:** Jason Bloomberg

**Practice Area:** Web Services

**Keywords:** Web Services, Testing, Quality Assurance, Software Development

**Size:** 25 Pages

**Figures & Tables:** 1 Figure, 1 table

**Format:** Electronic only: PDF

**Cost:** \$795

**ID:** ZTR-WS105

**Available at:**

<http://www.zapthink.com/reports/ZTR-WS105.html>

All Contents Copyright © 2002 ZapThink, LLC. All rights reserved. Reproduction of this publication in any form without prior written permission is forbidden. The information contained herein has been obtained from sources believed to be reliable. ZapThink disclaims all warranties as to the accuracy, completeness or adequacy of such information. ZapThink shall have no liability for errors, omissions or inadequacies in the information contained herein or for interpretations thereof. The reader assumes sole responsibility for the selection of these materials to achieve its intended results. The opinions expressed herein are subject to change without notice. All trademarks, service marks, and trade names are trademarked by their respective owners and ZapThink makes no claims to these names.



## The Web Services Context for Software Testing

Traditionally, software testing has been the ugly duckling of software development. Long thought a necessary evil to be relegated for the end of a project, testing often gets the budgetary axe in software projects that are experiencing cost overruns. But just like the swan in the story, testing is actually a key element of software quality- and quality, broadly defined, is the ability of a system to do what it's supposed to do. Therefore, quality should be first and foremost in the mind of everyone on a software project.

Two immediate factors are changing this landscape: 1) traditional "waterfall" methodologies of the software development process are giving way to less risky, more proactive iterative processes that implement testing earlier into the process. 2) the software industry is in the early stages of transitioning to the loosely coupled, standards-based environment heralded by Web Services. Today, the primary use for Web Services is to simplify and reduce the cost of integration within the enterprise.

A third, more distant factor, is that Web Services have far more potential than a straightforward simplification of existing component programming techniques. Over the next few years, Web Services promise to fundamentally change the distributed computing landscape. This will occur in three phases:

- *Phase One* – The first phase of Web Services adoption (which describes the current situation) has an internal enterprise focus. Enterprises primarily use Web Services to simplify and reduce the cost of integration inside the firewall.
- *Phase Two* – As new products and services on the market resolve the issues with Web Services security, management, and transactions, companies will be able to exchange Web Service messages with other companies (customers, suppliers, and partners) in a more uninhibited, loosely-coupled manner. Enterprises will find that UDDI registries will become a critical enabler of the dynamic discovery of Web Services within controlled environments. The Services themselves will still typically be static, but Web Service consumers will find and bind to them dynamically.
- *Phase Three* – As Service-oriented architectures mature and Web Service orchestration and business process automation tools become increasingly prevalent, just-in-time (JIT) integration will become the fundamental invocation style for Web Services. In this phase, Web Service consumers dynamically discover Web Services as in phase two, but now those Services may change every time a consumer invokes them. Furthermore, the concept of Web Service "consumer" and "producer" become less meaningful in phase three, because complex orchestrations of Web Services will be the norm, rather than the exception.

The primary question is how will the currently available Web Services testing tools on the market need to change as companies adopt Web Services and move toward dynamic, loosely coupled Service-oriented architectures?

## Today's Software Testing Environment

The current IT environment treats Web Services as a special kind of software component. Therefore, to understand the context of Web Services testing, it is important to understand the current software testing environment. Today's Web Services testing tools address the following range of software testing techniques:

- "Black box" functional testing – verifying that the functionality of a component is consistent with the functional specification for that component.

- “White box” structural testing – analyzing the structure and organization of the programming code within a component in order to verify the quality of that component. White box testing can either be performed at design time (analyzing or profiling source code) or at runtime (evaluating the execution paths as the component runs).
- Load testing – several related testing techniques that include capacity testing, stress testing, and tuning.
- System testing – end-to-end testing of entire systems of components to insure that the system as a whole meets the specifications set out for it. Part of system testing is integration testing, which focuses specifically on the exchange of messages between components.

How, then, do these standard software testing techniques apply to Web Services today? Fundamentally, Web Services are software components that expose their functionality via SOAP interfaces, while WSDL metadata files provide information about the Web Services’ interfaces, including the port type, binding, etc. Black box testing of a Web Service is therefore often a simple matter of exchanging the appropriate SOAP messages with it. Such messages can be “hardwired” into the testing tool, or the testing tool may be able to access the WSDL file in order to dynamically create the SOAP messages necessary for testing the Service.

White box testing of an individual Web Service need be no different from white box testing any software component, because many of today’s Web Services are simply software components with Web Services wrappers. A Web Service may also consist of a packaged application with a SOAP interface or a legacy application with a similar Web Services wrapper. In both these cases, testing the inner workings of the internal application is typically unnecessary, because the application has already been tested and is often in use. Instead, regression testing of the wrapper itself is often sufficient:

- Start with the existing functionality of the application to be Web Service enabled. Ideally, this functionality is clearly defined.
- Use the existing application functionality to build test plans for the Web Services wrapper. For each method call or message exchange, define the expected SOAP response from the wrapper.
- Run the test and compare the expected output with the actual output.

## Parasoft’s Approach

Parasoft takes an interesting approach to white box testing of Web Services. They take the WSDL file and use it to define structural tests for the Web Service, by generating a suite of test cases that cover all possible methods, allowing the tester to drill down in the Web Service code to identify the cause of any problem. This is a logical approach that we expect all testing vendors will eventually adopt.

Their product, SOAPtest, handles the following Phase One Web Services testing capabilities:

- Testing SOAP messages – moving beyond SOAP as the interface to the Web Service to testing the format of the message themselves.
- Testing WSDL files and using them for test plan generation – WSDL files contain metadata about Web Services’ interfaces. Testing tools can use these WSDL files to generate black box test plans automatically. WSDL files can also help with white box testing by providing some information about the structure of a Web Service.

- Web Service consumer and producer emulation – when a testing tool generates test messages that it sends to a Web Service and then subsequently analyzes the results, it is emulating the consumer for that Service. In addition to the Web Service producer, the consumer of the Service also sends and receives SOAP messages. Therefore, testing tools can also emulate the Web Service producer (namely, the Service itself) in order to test the Web Service consumer.

As the market evolves beyond the Phase One of Web Services implementation into Phases Two and Three, the software development cycle will change drastically. The ponderous “design, then develop, then test, then launch” testing processes familiar with the sequential waterfall software development methodology will be entirely inappropriate for the fully realized Service-oriented environment in phase three. Instead software groups will by necessity have to take what is now known as an agile approach to testing and development: work with the customer, tackle just what needs to be done, write a test, code as a team, pass the test, repeat until finished. Testing before coding will move from being “extreme” (as in popular agile development methodology Extreme Programming) to being the only cost-effective method for dealing with the dynamic nature of distributed computing.

### The Web Services Testing Leadership Roadmap

			First Axis			
			Black Box	White Box	Load Testing	System Testing
Second Axis	Testing SOAP Messages	Phase I	Compuware Empirix Mercury Parasoft Rational Red-Gate	Compuware Parasoft Rational Red-Gate Segue	Compuware Empirix <i>Mercury</i> Parasoft Rational Red-Gate Segue	Compuware Parasoft Rational Segue
	Testing WSDL Files & Automatic Test Plan Generation		Empirix Parasoft Red-Gate	Empirix Parasoft Red-Gate	Parasoft Red-Gate	Parasoft
	Consumer & Producer Emulation		Parasoft Rational Red-Gate	Parasoft Red-Gate	Parasoft Red-Gate	Parasoft
	Publish, Find & Bind Testing	Phase II			<i>Mercury</i>	
	Testing Synchronous & Asynchronous Capabilities		Mercury Parasoft		Mercury	
	Testing SOAP Intermediaries					
	SLA and QoS Testing				Compuware <i>Mercury</i>	Compuware <i>Mercury</i>
	Orchestration Testing	Phase III		Rational		
	Versioning and Agile Architecture Testing			<i>Rational</i>	<i>Mercury</i> <i>Rational</i>	<i>Mercury</i> <i>Rational</i>
<p><u>Legend:</u>                      Plain text: currently supports to some extent                      Italics: willing to discuss plans to support</p>						

Source: Copyright © 2002 ZapThink, LLC

**Profile: Parasoft** (August 2002)

Date Founded: 1989  
Headquarters: Monrovia CA  
Funding: Private  
CEO: Adam Kolawa

**ZapThink take**

Of all the Web Services testing vendors covered in this report, Parasoft has the most comprehensive offering for testing Web Services in phase one (SOAP messages, WSDL files, and producer/consumer emulation). Their solid background in Java and C++ testing comes through in their technically oriented product. Parasoft is a small company and cannot afford to apply resources to pushing the Web Services testing envelope too far beyond current customer demand, but nevertheless, they are a leader in Web Services testing today.

## About ZapThink, LLC

Founded in October 2000, ZapThink, LLC (<http://www.zapthink.com>) is an industry research and analysis firm that provides quality, high-value, focused research, analysis, and insight on emerging technologies that will have a high impact on the way business will be run in the future. ZapThink focuses on XML and Web Services technologies that provide open, standards-based, loosely-coupled systems and represent an evolutionary advancement in computing that requires a new way of thinking about computing resources, capabilities, development methodology, and architecture.

ZapThink produces and sells XML-focused research and analysis reports. The company has a comprehensive, thorough, and up-to-date understanding of XML and Web Services technologies, the markets in which they exist, the vendors that operate in various market segments, the growth of the various segments, and applicable ROI and revenue models. ZapThink's research has been often cited and quoted by numerous industry analysts, media, venture capital firms, vendors, and end users. The company is considered to be a reliable, credible, and comprehensive source of research and analysis. The company is more uniquely qualified and better positioned to provide this advice than any other company in the world due to its combination of technical focus and ability to apply detailed and rigorous methodology to research and data sources, while maintaining independence and objectivity.

ZapThink also provides strategic eBusiness consulting services to a select clientele, helping a broad range of companies formulate their business strategies by leveraging XML and Web Services. ZapThink advises hundreds of clients on the usage, adoption, and challenges of applying XML to their business needs. The company focuses on providing its clients advice on strategic planning and business growth as it applies to the use of XML.

**ZAPTHINK CONTACT:**

ZapThink, LLC  
681 Main Street, Suite 2-11  
Waltham, MA 02451  
Phone: +1 (781) 207 8534  
Fax: +1 (786) 524 3186  
[info@zapthink.com](mailto:info@zapthink.com)