Methods for Testing Web Services

Peter Dencker, Rix Groenboom

Parasoft Deutschland GmbH
Bayerstraße 24
80335 München
dencker@parasoft.com
rixg@parasoft.com

Abstract: This paper will discuss software testing, and in particular testing of web services and Service Oriented Architecture (SOA). This topic will be covered from two perspectives: First, from the theory and the current testing needs for SOA and second, we will provide pointers and review practical problems that are not solved yet and would require extra research from universities and research institutes.

1 Introduction

The primary mission of information technology is to improve business processes and increase profits. Companies are constantly rethinking and struggling with how to use IT to a competitive advantage, reduce IT operating and maintenance costs, and reduce the total cost of ownership… all while attempting to deliver increased value.

Most of these problems can be traced to the same source: the struggle to make software work - without incurring unreasonable costs. Most people in the industry would agree that low IT productivity is the culprit here.

Errors are the root cause of this low productivity: Errors that result from mistakes made throughout the software development life cycle. These errors include everything from performance errors, to security errors, to mis-implemented functionality, to errors that crash an entire system. In fact, if you look at virtually any IT team, you will see that its team members spend about 80 percent of their time chasing and fixing bugs, and only about 20 percent of their time on tasks that deliver value and improve the business. This practice is far from productive.

Moreover, Web services' flexibility and connectivity provide an increased opportunity for errors. Problems can be introduced in any of a service's multiple layers, and even the slightest mistake can cause the entire service to fail.

Testing tools and solutions, such as Parasoft's SOAtest, can increase a team's productivity by preventing errors early in development, and in doing so, improve quality, reliability, and accelerate time to market. The sooner you detect a problem, the easier it is to fix it, thereby leaving the team more time to code and be more productive. That is why we need to employ tools and testing techniques that immediately and thoroughly exercise Web services and check their reliability.

This article explains general best practices that developers of Web service servers and/or clients can apply to ensure service functionality and reliability, and how testing solutions providers make research and experts' observations available for them to use. In addition, it discusses interoperability, security, and related issues that affect both Web service producers and consumers. The bulk of the discussion assumes the use of WSDL for describing the service, HTTP for the transport layer, and SOAP for the messaging layer.

2 Best practises for quality

As explained in the introduction, complexity, flexibility, dependency of multiple applications on one service, lack of practices are some of the challenges any Web service development team faces. Adopting and enforcing the industry recommended best practices systematically can help to handle these issues. Why and how should standards be put into practice ?

2.1 Quality in other industries

Many other industries, such as the automobile industry, have also struggled with low quality, high costs, and low productivity as a result of human error. These industries recognized that although mistakes cannot be eliminated, they could, indeed, be controlled. They then modified their production lines to prevent as many errors as possible. By preventing scores of errors from ever entering the products, they addressed their most critical problems and were able to remain viable and productive industries.

The software industry still has not learned this lesson. Many people do not think that error prevention is even possible in this industry; they believe that because each piece of software is different, the lessons learned from working on one piece of software cannot be applied to other pieces. Thus, instead of trying to prevent errors from entering software, the software industry tries to test errors out of software.

In the case of Web services, we build a service, then we attempt to use testing to determine whether the service works and finally, we remove any errors that the testing process exposes. However, a consideration of the number and impact of software errors suggests that this "quality through testing" approach is not yielding the desired results.

The belief that “late” testing can create quality software systems is a fundamental problem in the software industry. We don't think of the whole process of building and deploying software in a way that would prevent errors because we don't believe that it can actually be done. Yet, this error prevention approach is not only possible; it is necessary and efficient, provided that the right best practices, standards, and testing solutions are utilized.

For the same reason, early testing, which is important for any development project, 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.

2.2 Early testing strategy with standards

A driving force behind Web services is the promise of seamless interoperability for disparate programming languages, operating systems, and various runtime environments. Unfortunately, the mere adoption of technologies that promote this idea (XML, SOAP, WSDL, UDDI) does not make the promise a reality.

Ideally, interoperability would be verified by checking that a service adheres to a comprehensive, universally implemented set of standards. However, the existing W3C, WS-I [Ws07] and OASIS [Oa07] recommendations are still evolving. Furthermore, the technologies are flexible enough to provide implementers with a myriad of options (document versus RPC-style, SOAP encoding versus literal encoding, different array representations, different versions of HTTP, SOAP, WSDL, UDDI, etc.) Flexibility is generally beneficial, but if everyone chooses a different way to do things, it not serve the goal of interoperability.

Now, although there is much work to do in this field, sets such as the Basic Profile helped by defining a foundation for Web services development and testing. This obviously allowed adoption of standards and recommendations to spread, and has been driving the current Web services proliferation. Furthermore, well known development and testing methods such as unit testing and iterative development can be utilized in the Web services (messaging) context. If we consider that the guidelines we should use are those that are best established and integrated in testing tools, the only remaining question, is how to enforce those guidelines ensuring that they are consistently and efficiently implemented.

At this point, providers in the industry can bridge the gap between the theory and the practice by facilitating the usage of standards through adapted testing tools and applied methods.

3 Enforce standards and best practices

A testing strategy should actually integrate two approaches to ensure complete covering of Web services: Testing the messaging layer as well as the application layer. Indeed, testing software components ensure that the underlying application component is robust enough to meet the demands of the business service contract. These tests can then be placed into regression suites. The web services test strategy includes 6 main stages which we will discuss below.

3.1 Validation of the WSDL

Interfaces must be correctly described in a WSDL (Web Services Description Language) document. 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. The phase tests the foundation for a SOA implementation.

3.2 Unit testing

Just like with testing of ordinary software components, unit testing is an important step to validate that these building blocks are working correctly. Also within web services, each operation must be exercised in isolation within the environment to confirm request and response. In particular, these tests will cover robustness of the operations by using so-called negative test cases (that check edge conditions and empty values etc).

3.3 Functional testing

The goal of this testing is fairly straightforward: to ensure that the server delivers appropriate responses for the given requests. However, due to the complexity of Web services, this task is far from simple. With most Web services, it is impossible to anticipate exactly what types of requests clients will send. Enumerating all possible requests is not feasible because the space of possible inputs is either unbounded or intractably large. As a result, it is important to verify with realistic business use cases whether the server can handle a wide range of request types and parameters.

3.4 Regression testing

After you have verified the server's functionality, rerun the functional test suite on a regular basis to ensure that modifications do not cause unexpected changes or failures. A common technique is to send various requests, manually confirm the responses, and then save them as a regression control. These regression tests can be incorporated into a regular automated build process. When regression tests are run frequently, regressions are easy to fix because they can be directly attributed to the few changes made since the last time the test was run.

3.5 Load testing

The goal of load testing is to verify the performance and functionality of the service under heavy load. It is a vital steps to make sure the services will delivered the required Quality of Service (QoS) for the customers / users of the system.

The best way to start load testing is to have multiple test clients run the complete functional test, including request submissions and response verifications. When load testing ignores the functionality verification process and focuses solely on load-rate metrics, it risks overlooking critical flaws (such as functionality problems that surface only under certain loads).

3.6 Adherence to standards (governance)

A key criteria of success for web services and SOA is the promise that services can be reused independently of the implementation framework, languages (for example Cobol and CICS, J2EE or .NET), and the technology supplier (e.g. the used Enterprise Services Bus etc).

The most pragmatic approach to achieve this, is the one taken e.g. by the Web Services Interoperability Organization (WS-I, see [Ws07]). The WS-I intends to serve as a standards integrator by developing a core collection of profiles that are a subset of the various Web service technologies. By restricting development to technologies specified in WS-I profiles, developers can increase the odds that their systems will interoperate with other systems. Development and testing solutions editors are already working with the WS-I and similar organizations. They provide tools that automatically check compliance with these profiles and, in the event of noncompliance, pinpoint exactly what needs to be changed to ensure compliance.

5 From Practise back to research

Developing technology and providing it to customers and users raises new questions. Most of these questions are so involved that they require the research community to investigate them in more detail. Based on the consultancy experience obtained by delivering Parasoft tools and solutions, several topics can be formulated in which academia are invited to contribute.

Web services is a recent new programming paradigm and it raised new questions on how to design, develop and maintain web-service and SOA implementations. These questions include:

- How many operations should I cover within a WSDL file?
- How to I make sure that the service upgrades do not invalidate the services that are depending on them ?

- How do I reason asynchronous services ?

- What is the right granularity for my Web services?

It is very interesting to note that these type of questions have been solved for databases, leading to the theory for normal forms [Da86]. Also for the reasoning about services, there is theory available [Ho85]. However, this would require translation of abstract notions like CSP and refinement calculus to current industry standards like BPEL.

In other words, the industry needs similar computer science results as those that have been obtained for the 3rd generation programming languages and database design. We believe that many of the concepts have been studied by research but still have to be mapped to the modern SOAP and XML technologies.

6 Conclusion

You can start implementing the discussed practices at any point in the development process, but if you start testing early, you will maximize your ability to prevent errors as well as detect errors. Typically, the earlier you detect a problem, the easier it is to fix it, and the less chance you and your team members have to inadvertently worsen the problem by building code or components that interact with the problematic element, or by reusing the problematic element for other servers or clients. If you start your testing as early as possible, then continue using the related tests as a regression test suite throughout development, you will not only ensure the client's or server's continued reliability, but also streamline the development process.

This has lead to the development and marketing of tools such as SOAtest. These solutions, built on basis of the industry practices and knowledge, can help to set up and maintain an efficient testing workflow.

Moreover, using state-of-the-art technology promotes innovation and provides new insights for the research community. We have mentioned the development of new theory on the design of web-services (comparable with the database design rules in the 70s), and the need for more assistance in the field requirements engineering.

References

[Da86] Date, C.J: An Introduction to Database Systems. Volume 1 (4th edition). Addison Wesley, 1986.

[Ho85] Hoare, C.A.R: Communicating Sequential Processes. Prentice Hall, 1985.

[Oa07] OASIS: http://www.oasis-open.org/committees/tc_cat.php?cat=ws

[Ws07] WS-I: http://ws-i.org/