# Finding Test Data on the Web

Mustafa Bozkurt and Mark Harman
King's College London, Strand, London WC2R 2LS, UK
Department of Computer Science
{mustafa.bozkurt,mark.harman}@kcl.ac.uk

## Abstract

*This paper proposes an automated solution for generating test cases for a web service using other web services. The proposed system is based on semantic web services that use Ontology Web Language based web service ontology OWL-S. OWL-S provides semantic information for input/output data as well as information on the services behavior. The system uses this information for test case generation.*

## 1. Introduction

In order to generate test cases, testing systems need to generate test data. Generating test data automatically is not necessarily straightforward, especially when it comes to generating real life data, such as valid ISBNs, postcodes and registration numbers. Most of these real life test data are generated using techniques like use of previous test cases or gathered from previously built databases that contain similar data. However, not all the enterprises have access to these kinds of facilities in order to test the service that they want to use.

It has been argued[1,3] that the distributed nature of web services based on multiple protocols such as UDDI and SOAP, together with the limited system behavioral information provided with WSDL specification present new challenges for software testing. Fortunately, we can also derive testing benefit from the existence of web services, as well as new challenges. That is, web services potentially provide a ready source of test data that is, by definition, both realistic and valid.

This paper proposes a system that aims to generate test cases while minimizing the need for previous data and the usage of test data generation algorithms by exploiting the data that could be acquired from existing web services.

## 2. The Proposed Approach

The proposed system generates test cases for a web service using OWL-S process model information and other web services as data providers. The output from these services will be used to construct test input for the Service/System Under Test (SUT). A test input may require a collection from several different web services, together with existing data. Creation of test cases will be handled in three steps:

1. Semantic data for SUT is analyzed by the test case generator and the generator creates test case drafts with the information about required test data for each test case.

2. The semantic matcher analyses the data information of test case drafts and compares it with the output types of known web services

3. When a match is found the system queries the matching web service method and uses its output as input in the generated test case.

## 3. Overall Architecture

As shown in Figure 1, the proposed system consists of four main components; an OWL-S parser, a test case generator, a semantic web service matcher and a web service query system. The OWL-S parser receives information then extracts necessary process information and passes this information to the test case generator. The test case generator analyses process information, then creates draft test cases for each process and passes the required semantic information to the service matcher. The service matcher receives data information and searches for known services that provide similar outputs. When one or more matches are found, the service matcher selects a suitable one to pass on to the web service query system. The selection of service to query based on three criteria:

1. Input requirements for the service method

2. The ease of generating or acquiring the input data

3. Similarity between service functionalities

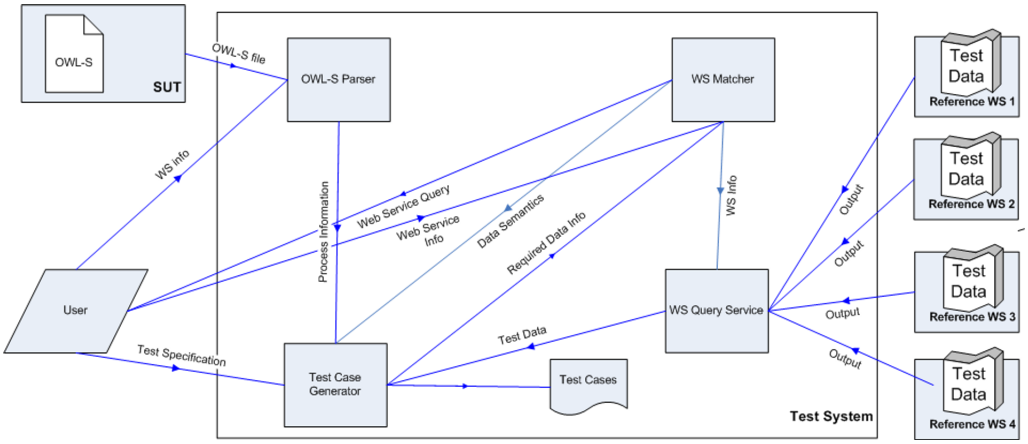


**Figure 1. System Architecture**

The service matcher gives precedence to the services which require no input and are close in functionality. The similarity between functionality is measured by comparing service functions, input/output data types and data semantics. We envisage the application of Search Based Software Engineering (SBSE)[4] to optimize this selection process. The optimization process will be a functionality of the service matcher.

After receiving the required test data, the test case generator combines the gathered data and the draft test cases to create complete test cases.

## 4. Issues and Limitations

There are four issues that the proposed system needs to address.

The **match making** problem is one of the major issues that was partially solved by semantic web services. Classical web services based on UDDI brokers allow only text based search for human use in order to find the required service. The information included in OWL-S specifications allows the matching web service process to be done in an automated way. The recently developed open source tools such as Opossum (semantic web service search engine), OWLSM, OWLS-MX WSMO-MX (Semantic web service matchmakers) have made fully automated service matching a reality. However, more research will be required in order to assess the effectiveness of the matching algorithm for test data generation. The general problem of match making effectiveness is a current topic of research in the services community.

The **Input for output** problem is one of the limitations of this approach. Most of the web services require input(s) in order to get an output. If our tool has to provide an input for an output from each web service, then our tool might end up in an infinite loop. In order to prevent this, the tool favors web services that do not require any input or those with easy-to-acquire input data. If the tool fails to achieve that within a certain number of iterations, it simply asks the user for a web service as the data source or to provide the required data. This required human assistance makes our approach only partly automated. However, it also means that the proposed approach is continually improving.

**Type inconsistencies** are another issue that could limit the efficiency of the proposed system. For example one system might define address data merely as a simple string, while in another system address might be a complex data type consisting of multiple simple or complex data types. This issue could be partially overcome by testing techniques like XML Schema-Based Partition Testing[2].

**Semantic issues** are a major problem in any system that deals with semantic information. This problem might be overcome by the use of tools like WordNet that help with semantics.

## References

[1] X. Bai, W. Dong, W.-T. Tsai, and Y. Chen. Wsdl-based automatic test case generation for web services testing. *Service-Oriented System Engineering, 2005. SOSE 2005. IEEE International Workshop*, pages 207–212, Oct. 2005.

[2] A. Bertolino, J. Gao, E. Marchetti, and A. Polini. Automatic test data generation for xml schema-based partition testing. *Automation of Software Test , 2007. AST '07. Second International Workshop on*, pages 4–4, May 2007.

[3] J. Garcia-Fanjul, C. de la Riva, and J. Tuya. Generation of conformance test suites for compositions of web services using model checking. *Testing: Academic and Industrial Conference - Practice And Research Techniques, 2006. TAIC PART 2006. Proceedings*, pages 127–130, 2006.

[4] M. Harman. The current state and future of search based software engineering. In *FOSE '07: 2007 Future of Software Engineering*, pages 342–357, Washington, DC, USA, 2007. IEEE Computer Society.