A pattern for WSDL-Based Testing of Web Service Compositions

DESSISLAVA PETROVA-ANTONOVA, Sofia University
SYLVIA ILIEVA, IICT-BAS
VERA STOYANOVA, Sofia University

The wide adoption of Service-Oriented Architecture (SOA) and its web service based implementation brings new research challenges related to development of complex business processes and their subsequent testing. The common language for definition of business processes is Business Process Execution Language for Web Services (WS-BPEL), which provides a set of activities describing the composition of web services. Since WS-BPEL processes are consumed through their interface, described with Web Service Description Language (WSDL), a logical assumption is that that the mature testing tools for web services could be used for their automation testing. This paper proposes a pattern for WSDL-based testing of both single and composite web services. The usefulness of the pattern is proved through implementation of a software tool that automates the testing activities prescribed by the pattern.

Categories and Subject Descriptors: **D.2.5 [Software Engineering]**: Testing and Debugging —*Testing Tools*; **D.2.9 [Software Engineering]**: Management—*Software Quality Assurance*

General Terms: Software Testing

Additional Key Words and Phrases: Test automation, WSDL-based testing, WS-BPEL processes

ACM Reference Format: Petrova-Antonova D., Ilieva S. and Stoyanova V. 2015. A pattern for WSDL-Based Testing of Web Service Compositions. In proceedings of 20th European Conference on Pattern Languages of Programs, EuroPLoP 2015. 6 pages.

1. INTRODUCTION

Testing of Service-Oriented Architecture (SOA) implementations brings various challenges at infrastructure, service as well as orchestration level (Morris et al. 2010). Since the web services provide most common implementation for SOA, the existing testing approaches and software tools are focused mainly to provide testing solutions for them. Usually, these solutions use the interface of the web service under test, described with Web Service Description Language (WSDL) to generate appropriate requests for invocation of web service's operations. Such testing scenario can be applied to single as well as composite web services, since they both provide a WSDL-based interface. It is realized by various software tools like SoapSonar (Esposito 2006), SoapUI (SmartBear), Parasoft SOAtest (Parasoft Corporation), etc. On the other hand, composite web services, known as business processes, require more exhaustive testing due to their complex behavior caused by orchestration of multiple single web services.

The paper proposes a pattern for testing of WSDL-based web services. Its usefulness is proved through design and development of new testing tool that is integrated in TASSA framework (Pavlov et al. 2010) – a platform for end-to-end testing of composite web services, described with Business Process Execution Language for Web Services (WS-BPEL). The tool provides test case generation and execution for functional testing of WSDL-based web services. Its implementation avoids some limitations of the current WSDL-based testing tools concerning defect allocation in case of web service composition testing.

Author's address: D. Petrova-Antonova, 125 Tsarigradsko shoes Blvd., Bl. 2, 1113 Sofia; email: d.petrova@fmi.uni-sofia.bg; S. Ilieva, Acad. Georgi Bontchev Str., Bl. 25A, Sofia; email: sylvia@acad.bg; V. Stoyanova 125 Tsarigradsko shoes Blvd., Bl. 2, 1113 Sofia; email: vera.stoyanova@yahoo.com

EuroPLoP '15, July 08-12, 2015, Kaufbeuren, Germany
 ISBN 978-1-4503-3847-9/15/07…$15.00
DOI: http://dx.doi.org/10.1145/2855321.2855324

2. PATTERN: WSDL-BASED TESTING OF WEB SERVICE COMPOSITIONS

2.1 Context

Single as well as composite web services provide WSDL-based interfaces allowing client applications to invoke their operations. The testing activities for both type of web services share similar issues. Therefore, a unified approach could be applied during their testing.

2.2 Problem definition

SOA allows software applications to interoperate in a new way in distributed environment. Currently, web services are the most widely adopted technology for implementation of SOA. In order to achieve a particular business objective, they are composed in complex business processes following WS-BPEL standard.
The problem of testing business processes is complicated since the composed web services are built and deployed on heterogeneous platforms. These services are outside organization boundaries and are very hard to be tested. Furthermore, they could be unavailable for a given period of time or in the worst case could be undeployed by their provider. This in turn complicates the testing of the business processes due to the necessity of emulation of the missing or unavailable web services. Additional efforts are needed for generation of appropriate message data, which will replace the actual ones expected by the business process. Therefore, the following open questions arise:

- How to **design test cases** in order to support testing of both single and composite web services?
- How to **generate test cases** to achieve better test coverage especially when business processes are tested?
- How to **execute test cases** and manage follow-up activities?

2.3 Forces

Testing web services, both single and composite, is more difficult than traditional software testing due to the following reasons:

- Missing of graphical user interface of business process;
- Invocation of services which are external for business process under test;
- Need of additional efforts and tools for testing third party/external web services of the business process in order to validate their quality;
- Possibility for usage of particular web service by multiple business processes;
- Security issues established in distributed environment such as authentication, authorization, data integrity and privacy, etc.

Current software tools are focused on generation of SOAP requests by analyzing the WSDL descriptions. They support sending of SOAP request to the web service under test and validation of SOAP response according to the preliminary defined test assertions. The majority of testing activities, such as population of SOAP requests with test data, are left to the testers. In addition, the presented tools support only black-box testing, giving information about the final result from the execution of the business process. If a particular test case fails, the testers are able to analyze only the fault data in the SOAP response having no idea what actually happens with the data flow of the orchestrated web services. The possible cause for failure is difficult to be identified and located.

2.4 Solution

The concept of the proposed pattern is presented in Fig. 1. The testing activities can be split in three main groups as follows:

- Test case design (1);
- Test case generation (2);
- Test case execution (3).

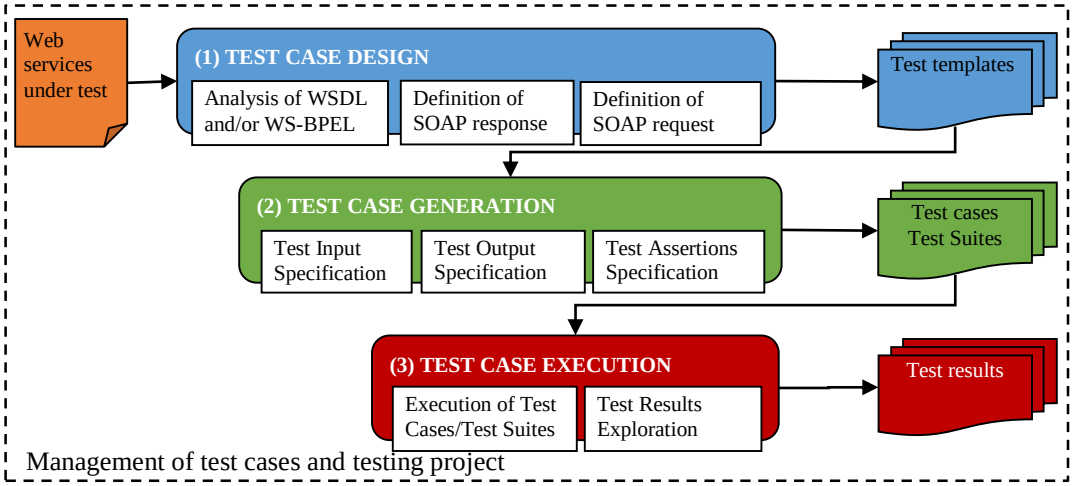

Fig. 1. *Concept of the pattern.*

All testing activities are performed within a given testing project, which is related to a certain WSDL file that corresponds to single or composite web service under test. During creation of a new project the WSDL file is extracted from a specified location and an empty project with a single element, named WSDL, is created. Next, the following activities related to **test case design** are performed:

– Performing WSDL and/or WS-BPEL analysis in order to identify the web service operations as well BPEL variables in case of web service composition;
– Definition of SOAP request templates;
– Definition of SOAP response templates.

After analysis of the WSDL file, the WSDL element is populated with all available operations of the web service. In order to create a test case, the tester should select an operation and to specify a name for the test case itself and for a test suite, where the test case will be placed. The newly created test case is populated with an empty SOAP request template relevant to the selected operation. In order to facilitate test data definition, the generated SOAP request is supplied with hints for the data types of the SOAP body elements.

The **generation of test cases** includes the following activities:

– Generation of input and output test data for the defined SOAP templates;
– Preparation of external data sources in order to perform data-driven testing;
– Definition of assertions at HTTP, SOAP and BPEL variable level.

The tester is able to generate two types of test cases:

(1) **Simple test case**, including a SOAP request and a list of assertions that should be satisfied after receiving of SOAP response from the web service under test;
(2) **Data-driven test case** that is similar to the ordinary test case but the data related to the SOAP request and the assertions is obtained from external data source.

The following types of assertions are can be specified:

(3) **HTTP Status Code** – verifies the status code of the HTTP response;
(4) **Response Time** – verifies that the response is received in a given period of time;
(5) **SOAP Fault** – verifies that the HTTP response contains a valid SOAP Fault message;
(6) **SOAP response valid** – verifies that the HTTP response contains a valid SOAP message;
(7) **XPath Equals** – verifies that when a certain XPath expression applied to a SOAP body or a BPEL variable, its result is equal to a particular value. It is possible to make it case insensitive and/or match it with regular expression.
(8) **XPath Exists** – verifies if a certain XPath expression applied to a SOAP body or a BPEL variable is not empty;
(9) **Contains** – verifies that a certain string exists within a SOAP body or a BPEL variable.

In addition, a **Negated assertion** can be specified, which can be applied to all assertions defined above. Finally, **execution of test cases** requires the following activities to be performed:

- Execution of single test case, all test cases in a test suite or all test cases in a test project;
- Obtaining test results and determination of the test verdict (Success, Failure, Error);

The simple test case execution includes two steps: (1) sending of SOAP request to the proper web service address defined in the WSDL file of the business process, and (2) receiving and assessment of obtained result according to the assertions defined in the test case. On the first step the information stored in the test case is used to retrieve the web service address, where the SOAP request should be sent and the message parts such as HTTP headers, user name and password if HTTP authentication is needed, and data of SOAP request body. The obtained result from test case execution consists of HTTP headers of the received response, data of SOAP response body, values of process variables and execution time.

When a data-driven test case is defined, the actual data of the SOAP request body and the expected data in the SOAP response body are replaced with variables. Its execution differs in two points from normal test case execution. First, a connection to the data source needs to be established. Second, for each record in the data source a separate test case is created, where the variables are replaced with concrete values. Finally, the generated test cases are executed ordinarily.

Activities related to management of test cases and file organization of the testing project are related to the whole testing process and could be part of all defined above three groups of activities.

2.5 Consequences

Some consequences of the pattern application are:

- A grey-box testing of web service compositions is possible using XPath assertions on BPEL variables;
- The communication of the business process with its partner web services is possible to be tested using XPath assertions on BPEL variables;
- Functionality of the current testing tools for WSDL-based services can be improved by supporting testing of both single and composite web services;
- Client applications are provided with higher quality services;
- Since the pattern is focused on functional testing, the security issues are not considered.

2.6 Known uses

The widely adopted software tools for WSDL-based testing of web services have the same key features closely related to the proposed pattern. The main testing activities are summarized in (Bertolini 2008) and include WSDL analysis, SOAP envelop derivation, message parts definition, envelopes composition, message sending and results analysis.

SoapUI is a powerful tool for automated functional testing of web services. It generates SOAP requests that will be used during the execution of the test cases. Its interface allows selection of web service operation to which the SOAP request will be sent from an automatically populated list. The test data can be generated manually or obtained from an external data source in case of data-driven testing. *SoapUI* provides a useful feature called service mocking, which can be used for simulation of the response from a given web service when the last one is not available or is under development. Thus, the SOA environment could be tested before there are services in it. *SoapUI* supports variety of assertions for validation of SOAP responses. TAXI is a software tool, which is in charge of the actual SOAP message definition extracting the XML Schema data from a WSDL description (Bertolini 2009). It works closely with *SoapUI* in order to support all mentioned so far testing activities. Similar issues to that addressed by TAXI are presented in (Bai 2009).

SoapSonar is a testing tool providing functional, performance, security and compliance testing of web services (Esposito 2006). It simulates web service interaction with clients by automated generation of client requests. The free version of the tool supports only functional testing. Features such data driven tests, capture of dynamic response HTTP header value and conditional test execution are available only in licensed versions.

Parasoft *SOAtest* is an integrated solution for end-to-end testing of SOA implementations (Parasoft Corporation). Along with this, the *SOAtest* performs checking of the reliability, security, and maintainability of web services developed against WS-I, W3C, OASIS, and WS- standards. However, since it is distributed with proprietary software license, which is quite expensive, only the big software vendors take its advantages.

WebInject and *TestMaker* are software tools that can be used for testing both web services and web applications. *TestMaker* provides an Agent Wizard that reads a WSDL description and creates scripts written in Python, called test agents. The test agent drives the web service under test showing how it performs under simulates production situation (Brunner 2003).

WebInject is a free tool for testing software components that have HTTP interfaces and therefore SOAP-based web services. It can be used as a test harness in order to execute tests written in XML files and to collect the obtained results including test verdicts, errors, response times and so on (WebInject).

3. IMPLEMENTATION OF THE PATTERN

The software tool following the proposed pattern is implemented using Java language as a plug-in for NetBeans IDE. Its graphical user interface (GUI) is built upon Swing API. The Web Service Description Language for Java Toolkit (WSDL4J) is used for generation, manipulation and object representation of WSDL documents. XML Schema Object Model (XSOM) and XMLBeans are chosen for processing of XSD and XML files. The tool consists of nine modules depicted with a component diagram in Fig. 2.

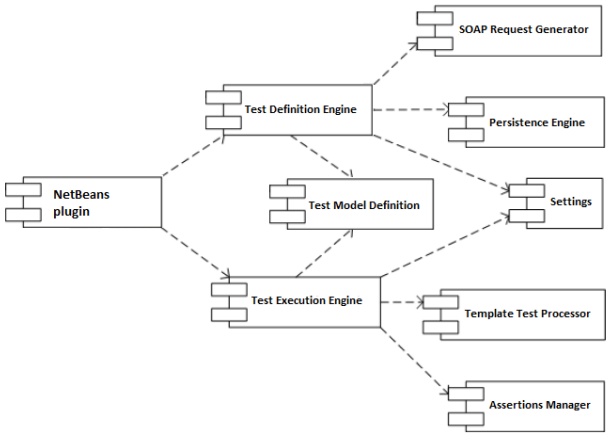

Fig. 2. Architecture of the tool.

The *Test Model Definition* module implements the main interfaces and classes of the domain model, namely TestCase, TestSuite, BpelOperation, etc. The *Test Definition Engine* module manages test cases and test suites. The *Persistence Engine* module performs serialization and deserialization of test cases. The *SOAP Request Generator* module handles the generation of template SOAP requests based on a selected operation from WSDL file. The *Settings* module is responsible for the tool configuration related to connection timeout, request charset, request content type, proxy server address and port, address of OpenESB server and port, user name and password. The *Test Execution Engine* module performs execution of test cases. The *Template Test Processor* module provides support for generation data driven tests. The *Assertion Manager* module implements different types of assertions, described in the next section, and provides GUI for definition and modification of an assertion in a test case. The *NetBeans plug-in* packages the functionality of the tool in a NetBeans plug-in.

4. CONCLUSION

The present work proposes a pattern for WSDL-based testing of web service compositions. The feasibility of the pattern is proved through implementation of a software tool, which functionality is similar to one supported by SoapUI tool, but enhanced with new features for validation of BPEL variables. Thus, following the pattern prescriptions, the tool provides detection of defects caused by wrong business logic or incorrect behavior of the partner web services of the business process under test.

5. ACKNOWLEDGEMENTS

We thank our shepherd Christopher Preschern for his feedback that helped us significantly in bringing the workshop version of this paper into shape. We also want to thank the participants of the VikingPLoP workshop for their thoughtful comments.
This work is partially supported by the ERDF under agreement n. BG161PO003-1.1.06.-0003-C0001 and Scientific Research fund of Sofia University “St. Kliment Ohridski” under agreement n. 143/17.04.2015.

REFERENCES

Morris, E., Anderson, W., Bala S., Carney, D., Morley, J., Place, P., Simanta, S. 2010. Testing in Service-Oriented Environments. Technical Report CMU/SEI-2010-TR-011, Carnegie Mellon University.

Esposito, D. 2006. SOAPSonar™ Measures the Distance Between Your Web Services and the Real World. http://www.codeproject.com/ Articles/15182/SOAPSonar-Measures-the-Distance-Between-Your-Web-S.

SmartBear. soapUI, http://www.soapui.org/.

Parasoft Corporation. SOAtest http://www.parasoft.com/soatest.

Pavlov, V. Borisov, B. Ilieva, S., Petrova-Antonova, D. 2010. Framework for Testing Service Compositions. *In Proceeding of the 12th International Symposium on* Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, Romania, 557-560.

Bertolini, C., Bertolino, A., Marchetti, E., Polini, A. 2008. Towards Automated WSDL-Based Testing of Web Services. *In Proceeding of the International Conference on* Service Oriented Computing, LNCS 5364, 524-529.

Bartolini C., Bertolino A., Marchetti E., Polini A. 2009. WS-TAXI: A WSDL-based testing tool for web services, *In Proceeding of the International Conference on* Software Testing Verification and Validation, Denver, Colorado, USA, 326–335.

Bai, X., Dong, W., Tsai, W.-T., Chen, Y. 2005. WSDL-based automatic test case generation for web services testing. In *Proceeding of* IEEE Int. Work. SOSE, Washington, DC, USA, IEEE Computer Society, Los Alamitos, 215–220.

Brunner R. et al. 2003. Java Web Services Unleashed, Sams Publishing, ISBN-10: 0-672-32363-X.

WebInject, http://webinject.org/.