Proceedings of the 2014 Federated Conference on Computer Science and Information Systems pp. 1553–1558
DOI: 10.15439/2014F93
ACSIS, Vol. 2

Tool for Automatic Testing of Web Services

Ilona Bluemke, Michał Kurek, Małgorzata Purwin
Institute of Computer Science Warsaw University of Technology
Nowowiejska 15/19, 00-665 Warsaw, Poland
Email: I.Bluemke@ii.pw.edu.pl.

***Abstract*— The Web Services technology has received a significant amount of attention in recent years because it allows to easily utilize and integrate existing software applications to create new business services. With the increase of interest and popularity of Web Services, web applications are developed. This way of software development causes new issues for Web Service testing to ensure the quality of service that are published. This paper describes the tool (named WSDLTest) for automatic testing of web services. The tool can be used for testing of web services for which WSDL 1.1 or WSDL 2.0 document are available. WSDLTest parses the WSDL document, and based on, it tests the web service by sending automatically generated messages. Some examples of the usage of our tool are given.**

I. Introduction

Web services are very important in providing software in the World Wide Web. They have emerged as the next generation of Web-based technology for exchanging information. They are modular, self - descriptive, self - contained applications that are based on open standard. A Web Service is a kind of Internet application based on SOAP [1] and XML [2] technology. The W3C [3] (World Wide Web Consortium) defines web service as "a software system designed to support interoperable machine-to-machine interaction over a network". Web Service relies on a family of protocols to describe, deliver, and interact with each other, such as the Web Service Description Language (WSDL) [4], the Universal Description, Discovery and Integration (UDDI) [5] protocol or the Web Service Inspection Language (WSIL) [6], and the Simple Object Access Protocol (SOAP) [1]. WSDL, UDDI, WSIL, and SOAP are all based on XML [2]. Due to the nature of standards-based architecture and XML-based messaging, Web Service is vendor-neutral, language-agnostic, and platform-independent. Therefore, developers of a Web Service are not able to assume which type of clients will use the Web Service and the developers at the client side, are not aware of which language and platform are used at the server side. A Web Service and its clients can be developed in totally different programming languages and on different platforms.

Web service can be described by a Web Service Description Language [4] document. WSDL was established as a standard by the W3C [3]. The WSDL document describes web service in terms of its interfaces with operations, types of data the web service takes or returns. Other systems interact with the Web service in a manner prescribed by its description using SOAP [1] messages.

Testing effort is often a major cost factor during software development, which sometimes consumes more than 50 % of the overall development effort [7] . Consequently, one major goal is often to reduce the testing effort e.g. by providing a tool facilitating the testing. With services the difficulty of testing increases because of the changes that this architectural style induces on both the system and the software business/organization models.

The purpose of our research was to determine the possibility and methods of testing web service based on its WSDL document. An application, named WSDLTest, has been designed and implemented at the Institute of Computer Science Warsaw University of Technology. This application can test web services for which WSDL 1.1 or WSDL 2.0 document are available. WSDLTest parses the WSDL document, and based on it, tests the web service by sending automatically generated messages.

The paper is organized as follows. Section II identifies key features of web services testing, also WSDL is briefly presented. In Section III some testing tools, using WSDL documents as a basis for testing, are discussed. Section IV focuses on the architecture of our testing tool (WSDLTest). Section V presents some results of testing a simple web service with WSDLTest. Finally, Section VI concludes the paper, highlighting some issues and future research directions.

II. Web services Testing

Testing services and service-centric systems poses new challenges to traditional testing approaches. Testing challenges derive primarily from the dynamic nature of web services and the clear separation of roles between the users, the providers, the owners, and the developers of a service. Automated service discovery and binding mean that the

This work was not supported by any organization

complete configuration of a system is known only at execution time, and this cause integration testing to be difficult. Canfora and Di Penta in [8] indicated unique features of services that add complexity to the testing.

Service testing can be performed at different levels namely:

- unit testing of atomic services and service compositions,
- integration/interoperability testing,
- regression testing,
- testing of non-functional properties.

Services testing is a recent area of investigation, numerous contributions have been presented in the literature, primarily in the areas of unit testing of services and orchestrations, integration testing, regression testing, and testing of non-functional properties. The literature also reports several investigations to improve the testability of services and service-centric systems. Canfora and Di Penta made SOA testing survey [8]. They also pointed out several problems which remain open and need additional research work e.g.: improving testability, combining testing and run-time verification, validating fully decentralized systems.

Comprehensive and excellent surveys on Web services testing can also be found in [9-12]. Several Web Services testing approaches were developed to address these new challenges, the survey can be found in [9]. Based on the surveyed Web Services testing approaches found in the literature, the existing approaches were divided by Ladam [9] into four classes:

1. WSDL-based test case generation,
2. mutation-based test case generation (e.g. [13, 14]),
3. test modeling (e.g. [15]),
4. XML-based approaches (e.g. [16]).

Hanna and Munro [17] proposed a framework that can be used to test the robustness quality attribute of a Web Service. This framework is based on analyzing the Web Service Description Language (WSDL) document of Web Services to identify what faults could affect the robustness attribute and then test cases were designed to detect those faults.

Bai and Dong [18] also are using the WSDL document to generate tests in a testing framework that includes test case generation, test controller, test agents and test evaluator.

Often the service is not able to satisfy the needs of a user but it is possible to combine existing services together in order to fulfill this need. The process of combining these services is called Web Service Composition (WSC). However much researches have been focused on the discovery, selection and composition of Web services, the testing of Web service composition is still immature [19]. Bucchiarone et al. [20] and Zakaria et al. [21] provide surveys focusing in on Web service composition. Rusli et al. provided [12] an interesting overview and evaluation of current approaches to WSC testing.

We concentrated on the WSDL-based approach to unit testing because it enables the automatic generation of tests.

A. WSDL

WSDL [4] - Web Services Description Language, is an abstract, XML-based language, which specifies location and functionalities offered by a web service. It contains specific information about the web service, for example needed parameters, returned data. XML schema is used for the presentation of WSDL description containing the messages send and received from the service. To communicate with the web service SOAP messages are exchanged and they are described by WSDL as operations. WSDL describes the format for interfaces, it can also describe interactivity of given service. WSDL description only shows the possible but not required interactions. The current, recommended by W3C [3], version of WSDL is 2.0. The older version - WSDL 1.1 is still quite common and in some software only this version is supported. Sections `<messages>` and `<portType>` were combined to create new section `<interface>` (in version 2.0)

WSDL 2.0 contains following sections:

- "*description*" - a container, inside which the remaining sections are located,
- "*types*" describes the data types send and received by a web service,
- "*interface*" describes the abstract functionality the web service provides (what messages it sends and receives, possible fault messages),
- "*binding*" provides information how to access the service,
- "*service*" provides information where to access the service.

There are also two optional sections in WSDL document:

- "*documentation*" provides textual description of a web service,
- "*import*" is used to import other XML schemas.

B. Usage of WSDL in testing

WSDL document is a description of functionalities provided by a web service. Inside this document, all "methods" and data types can be found. It is divided into several sections directly connected with each other. The section "types" contain data types, which are input and output parameters for operations. Operations are described in "interface" section. For each interface, there is a binding with the message protocol. The "binding" section of WSDL document provides knowledge about protocols bonded to given interface. "Service" sections provide information where to find endpoint for each binding. All those sections allow to determine the input and the output, as well as how and where to access the web service. Nonetheless, there is no information about what exactly the web service produces as output from input. To use effectively WSDL in testing software, following elements are needed:

- WSDL parser tool, which reads the WSDL document and extracts information.
- Oracle mechanism determining if the results of the test are correct.
- SOAP messages generator.

The process of testing based on WSDL document includes several steps. Firstly the WSDL document is parsed into

addressable memory objects. Information from parsing can be used to generate SOAP requests. Such message may contain either randomly generated parameters based on data type taken from WSDL document, or can be filled by a tester. Such approach to web service testing is applicable only when communication is bidirectional – request and response from the web service. When SOAP request is send, after a while the web service should send back response with results of its operation. These results may be later compared with oracle to determine if the web service is working properly.

III. TESTING TOOLS

There are several testing tools available on the market, which can be used for testing web services. The most recognizable tools are:

1. HP QuickTest Professional,
2. Parasoft SOAtest,
3. SOAPSonar,
4. SoapUI.

Some of these tools offer the validation of WSDL document provided by WS-I [22].

HP QuickTest Professional [23] is an automatic testing suite. It accepts both 1.1 and 2.0 versions of WSDL. The application performs automated tests on a variety of software and in many environments. It has graphical user interface, which allows performing regression and functional testing. The main testing algorithm identifies objects to be tested and performs testing of interface operations. Important function of HP QuickTest Professional is the ability to perform the validation of WSDL using the WS-I [22] tool. In this application the modification, insertion, and removal of test parts are easy. HP QuickTest Professional has a Web Service Testing Wizard Click that can be used for the definition of different options e.g. a checkpoint can be created or some actions may be selected. Everything defined in the wizard is later converted and inserted into the test. This tool has functions allowing the customization of reports with user-defined images and screenshots. Client performance-related errors (e.g. memory leakage) can be included in the standard reports with a direct link from the report to the test script. HP QuickTest Professional automates the design of tests and test cases.

Parasoft SOAtest [24] is used for testing applications build in Service-oriented Architecture. It can automate application testing, message/protocol testing, cloud testing and security testing. Its graphical user interface is Eclipse based (uses the same GUI library). It allows creating tests, defining the behavior of the tests and configuring specific tests (e.g. queue managers or database connections). SOAtest primary function is to test Web services. It automatically generates tests from key market platforms such as WSDL [4], UDDI [6], WSIL [7], XML [2], BPEL [25], HTTP traffic and other. SOAtest can validate WSDL documents and emulate the client or the server. Tests can be grouped to be executed in a sequence. Failed tests are highlighted. Parameters for requests can be entered by the user or can be read from a file. SOAtest can trace and visualize how messages and events flow through ESBs (Enterprise service bus) [26], message brokers, applications and databases, while tests are executed. Such tracing allows the interpretation of problems.

SOAPSonar [27] is testing and analysis tool specifically designed for Web Services. It can perform functional regression tests, performance tests; generate compliance reports, vulnerability checks and identity tests. It has clear graphical user interface. Tests are created via drag and drop selection. Application has also WSDL Region WS-I validation [22]. It accepts WSDL 1.1 and WSDL 2.0 documents. The professional edition of SOAPSonar has test flow management options and can create WSDL requests - response chain or data driven test for the exchange of messages. Functional tests use load testing for the performance monitoring. Security penetration tests are performed at the message layer. All versions of SOAPSonar use as test inputs and response analysis data from e.g. database tables or Microsoft®Excel files. Application can automatically change variables in message headers, message body, tasks and can change variables. SOAPSonar has also vulnerability mode, which allows associating each test request with a set of attack. It can parse the WSDL documents and generate a list of the operations available on interface described by it. It can be used to send SOAP request messages to the target Web service and capture the response.

SoapUI [28] is an open source web service testing application for service-oriented architectures. SoapUI has clear and easy to use graphical user interface, which allows for quick creation of any test scenarios. It requires WSDL files to generate tests, messages, validations and MockServices. SoapUI supports only 1.1 version of WSDL and it can easily create functional, regression, compliance, and load tests. SoapUI provides code free test environment, all tests are created by drag and drop actions. SoapUI Pro Test Debugging is an option, which allows following the flow of test, variables, properties requests and context. Data for data-driven tests can be provided from external editors such as Microsoft ® Excel. SoapUI MockServices is a feature, which allows mimicking Web Services before they are implemented. SoapUI uses WS-I profile [22] for 1.1 version of WSDL.

IV. WSDLTEST

WSDLTest is a simple application supporting unit testing of web services which are described in WSDL. In the Fig. 1 the high-level structure of WSDLTest is presented. External connections and internal interfaces are also shown. The frame at the top of figure, which consists of WSDL document and web service itself, represents remote location

that is accessed in different manner by WSDLTest application.

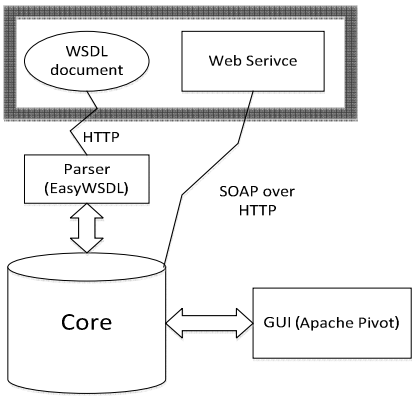

Fig. 1 Structure of WSDLTest

WSDL document is accessed via parser using HTTP connection, whereas web service is communicated thru SOAP messages transported over the HTTP connection. Core module consists of classes that support main functionality of application. WSDLTest is divided logically into Core, Parser and GUI. As a parser we used EasyWSDL Toolbox [29]. It can be used to parse WSDL 1.1 and WSDL 2.0 descriptions and transforms them in a unified object model (based on the WSDL 2.0 entities). Library is written in Java language and therefore can be used within java compatible application only. Moreover, WSDL parser can additionally export information from application object to an editable WSDL document. Other implementation details can be found in [30].

In the Fig. 2 main functions of core module are presented. Based on information from the parsed WSDL document interface bonded to SOAP [1] connection is found. From this interface all operations are extracted to determine what kind of parameters operations require and how many of them. GUI creates panel for input values of parameters or input of regular expression. WSDLTest generates all required parameters and performs tests.

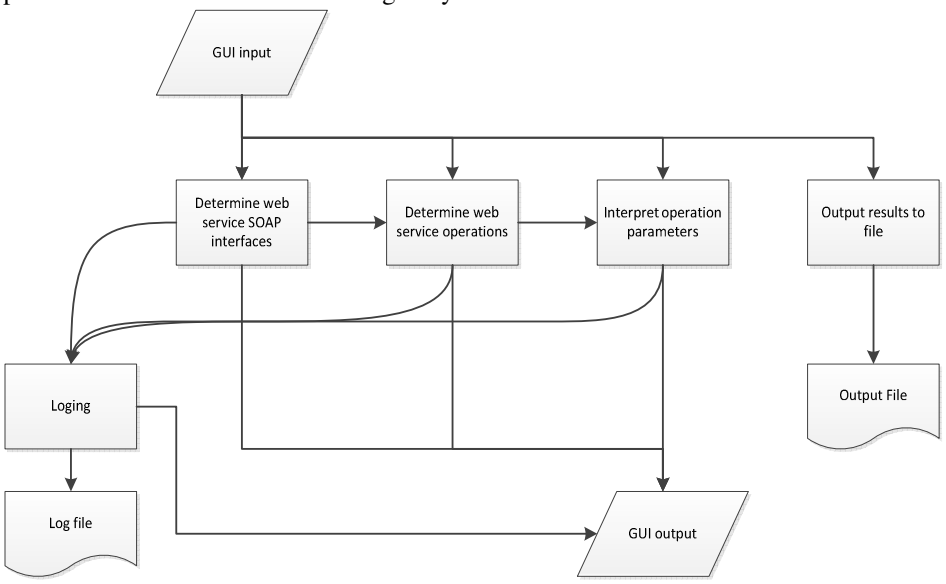

Fig. 2 Activities in Core module of WSDLTest

```
<s:element name="ValidateCardNumber">
        <s:complexType>
               <s:sequence>
                      <s:element minOccurs="0" maxOccurs="1"
                             name="cardType"
                             type="s:string"/>
                      <s:element minOccurs="0" maxOccurs="1"
                             name="cardNumber"
                             type="s:string"/>
               </s:sequence>
        </s:complexType>
</s:element>
<s:element name="ValidateCardNumberResponse">
        <s:complexType>
               <s:sequence>
```

```
14                          <s:element minOccurs="0" maxOccurs="1"
15                                 name="ValidateCardNumberResult"
16                                 type="s:string"/>
17                    </s:sequence>
18             </s:complexType>
19    </s:element>
20    ...
21    <wsdl:portType name="CCCheckerSoap">
22             <wsdl:operation name="ValidateCardNumber">
23                    <wsdl:documentation xmlns:wsdl=
24                           "http://schemas.xmlsoap.org/wsdl/">
25                           Please enter card type as VISA or
26                                 MASTERCARD or DINERS or AMEX
27                    </wsdl:documentation>
28                    <wsdl:input message=
29                           "tns:ValidateCardNumberSoapIn"/>
30             </wsdl:operation>
31    </wsdl:portType>
```

Fig. 4 WSDL document for web service ValidateCardNumbert

A. Testing modes

WSDLTest has three testing modes available for the user (shown in Fig. 3):

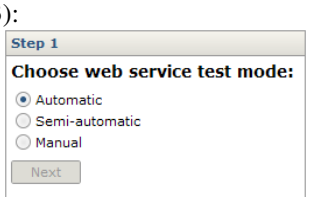

Fig. 3 Modes of operation of WSDLTest

1- *Automatic* – all tests are performed automatically. User only specifies the number of tests to be performed. All possible operations (functions of web service) defined in WSDL document are tested. For each operation random parameters are generated based on theirs defined in XML types.

2- *Semi-automatic* – parameters of operations are provided as regular expressions. For these expressions values of parameters, used in tests, are generated. User can specify number of tests and operations to be tested.

3- *Manual* – the simplest of modes, parameters of operations are given explicitly by the user.

In semi-automatic and automatic mode the oracle, containing expected test results, can be defined.

V. Example

Web service Validate Card Number [31] is a simple web service with only one method named `ValidateCardNumber`, used for credit card number validation. In Fig 4 the definition of operation `ValidateCardNumber` [lines 22-30] is given, input parameter of operation `ValidateCardNumber` [lines 1-10] and output of operation parameter `ValidateCardNumberResponse` [lines 11-19].

This service was tested using semiautomatic mode. In first test the parameter `cardNumber` was inputted as regular expression which matches any random sequence of digits of the length equal 16 (typical length of credit card number). Input window is presented in Fig.5.

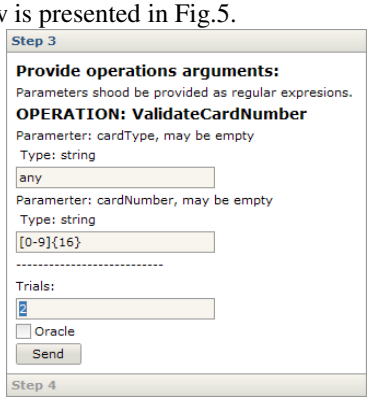

Fig. 5 Input window of WSDLTest – setting parameter values

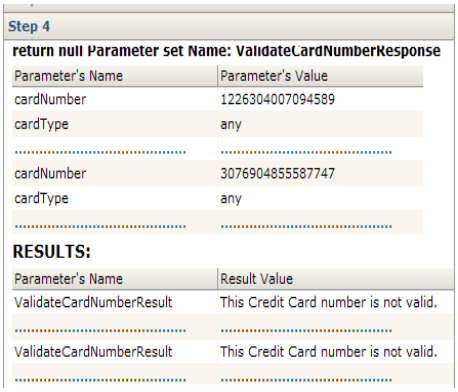

Fig. 6 Result of tests for parameters given as digits

Results, presented in Fig. 6, indicate whether the card number generated by WSDLTest is valid or not. The parameter `cardNumber` is defined as string (line 7 in Fig. 4.), so random string consisting of letters was used in next test. Credit card number can consist only of numbers, so error should be reported. WSDLTest was able to find the errort inside returned SOAP message. Results are shown in Fig.7.

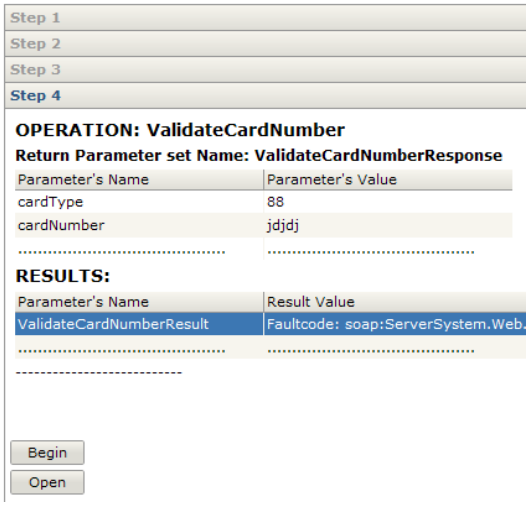

Fig. 7 Result of tests for parameters given as random string

VI. Conclusion

Web service can be specified in a WSDL [4] document. This document may be used as a basis in the automatic testing of the web service. We developed the application, called WSDLTest, which uses the WSDL document to prepare unit tests of web service. WSDLTest allows user, with basic technical knowledge, to test a web service. Testing can be performed in fully automatic mode for theoretically infinite number of trials with different parameters. The automated process of testing includes the generation of parameter values, the comparison of results with oracle and a basic validity check of the WSDL document. Tester can also provide parameter values and oracle in form of regular expression. Such form is more general than the explicit one. Regular expressions used in oracle are used to generate expected values of results, which are compared with actual results of tests.

WSDLTest can test a variety of web services described by WSDL. In this paper we presented simple example (section V). Even on this simple example it can be easily noticed that WSDLTest can be very useful in testing web services. Our tool does not have all functionalities available in other tools presented in section III, e.g. is not able to do load testing. In current version, WSDLTest is not able to read parameters from external files or mimicking web service, such functions are usually available in commercial tools. It might have some problems in testing web services with very complex data. These drawbacks can be eliminated in future versions.

WSDLTest has also an advantage over other similar tools. It can generate values of parameters from regular expressions; such functionality is not common in other tools.

References

All URL in references were valid in May 2014.

[1] SOAP: Simple Object Access Protocol, http://www.w3.org/TR/soap/.
[2] XML – Extensible Markup Language, http://www.w3.org/XML/.
[3] W3C Official Website:, http://www.w3.org/.
[4] WSDL: http://www.w3.org/TR/wsdl.
[5] UDDI: http://uddi.xml.org/.
[6] WSIL www.ibm.com/developerworks/webservices/library/ws-wsilspec.html
[7] F. Elberzhager, A. Rosbach, R. Eschbach, J. Münch, "Reducing Test Effort: A Systematic Mapping Study on Existing Approaches", *Information and Software Technology*, vol. 54, no. 10, October 2012, pp. 1092-1106, DOI=10.1016/j.infsof.2012.04.007.
[8] G. Canfora, M. Di Penta, M. 2009." Service-Oriented Architectures Testing: A Survey". In *Software Engineering*, De Lucia, A. and Ferrucci, F., Ed. Springer, 78-105. DOI= 10.1007/978-3-540-95888-8_4.
[9] M. I. Ladan, "Web Services Testing Approaches: A Survey and a Classification", in F. Zavoral et al. (Eds.): NDT 2010, Part II, CCIS 88, pp. 70–79, 2010. Springer-Verlag Berlin Heidelberg 2010
[10] M. Bozkurt, M. Harman, Y.Hassoun, Y. *Testing Web Services: A Survey*. Technical Report. King's College London, 2010.
[11] A. Metzger, S. Benbernou, M. Carro, M. Driss, G. Kecskemeti, R. Kazhamiakin, K. Krytikos, A. Mocci, A., E. Di Nitto, B. Wetzstein, F. Silvestri, 2010. "Analytical Quality Assurance". In *Service Research Challenges and Solutions for the Future Internet*, Papazoglou, M., Pohl, K., Parkin, M. and Metzger, A., Ed. Springer, 209-270. DOI= 10.1007/978-3-642-17599-2_7
[12] H. M. Rusli, M. Puteg, S. Ibrahim, and S. G. H. Tabatabaei, "A comparative evaluation of state-of-the-art web service composition testing approaches," in *International Workshop on Automation of Software Test*, 2011, pp. 29-35
[13] Reda, S., Nashat, M.: Testing Web Services. IEEE Software (2005)
[14] Andre, L., Silvia Regina, V.: Mutation Based Testing of Web Services. IEEE Software (2009)
[15] Feudjio, A.-G.V., Schieferdecker, I.: Availability Testing for Web Services, ISSN 0085- 7130 Telektronikk 1 (2009)
[16] Tsai, W.T., Paul, R., Song, W., Cao, Z.: An XML-Based Framework for Web Services Testing. IEEE Software (2002)
[17] S. Hanna, M. Munro: "An Approach for WSDL-Based Automated Robustness Testing of Web Services". *Information Systems Development Challenges in Practice, Theory, and Education* 2 (2008)
[18] X. Bai, W. Dong: 'WSDL-Based Automatic Test Case Generation for Web Services Testing". *IEEE Software* (2005)
[19] G. Canfora, M. Di Penta: "Testing services and service-centric systems: challenges and opportunities". *IT Professional*. 8, 2 (March-April 2006), 10-17. DOI= 10.1109/MITP.2006.51.
[20] A. Bucchiarone, H. Melgratti, F. Severoni, F. 2007. "Testing service composition". In *Proceedings of the 8th Argentine Symposium on Software Engineering* (Mar del Plata, Argentina, August 29-31, 2007). ASSE 2007.
[21] Zakaria, Z., Atan, R., Ghani, A. A. A., Sani, N.F.M. 2009. „Unit testing approaches for BPEL: A systematic review" In *Proceedings of the 2009 Asia-Pacific Software Engineering Conference* (Penang, Malaysia, December 1-3, 2009). APSEC 2009. IEEE, 316-322. DOI= 10.1109/APSEC.2009.7
[22] WSI: http://www.oasis-ws-i.org
[23] HP QuickTest Professional: http://www8.hp.com/us/en/software-solutions/unified-functional-testing-automation/.
[24] Parasoft: http://www.parasoft.com/soatest .
[25] BPEL: http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf
[26] ESB: http://en.wikipedia.org/wiki/Enterprise_service_bus.
[27] SOAPSonar: http://www.crosschecknet.com/products/soapsonar.php.
[28] SoapUI: http://www.soapui.org/.
[29] Easy WSDL Toolbox, http://easywsdl.ow2.org/.
[30] M. Kurek, M. Purwin: *Automatic Testing of Web Services Based on WSDL Document*, Bachelor Thesis, Institute of Computer Science, Warsaw University of Technology, 2014.
[31] WSDL, "Credit Card", http://www.webservicex.net/CreditCard.asmx?WSDL.