Contract Based, Non-invasive, Black-Box Testing of Web Services

Michael Averstegge
(Advisor: Bernd J. Kraemer)

FernUniversitt Hagen, Germany
Department of Electrical Engineering
Michael.Averstegge@web.de

Abstract. The Web service standard represents a prominent conversion of the SOA paradigm increasingly used in practice. The (not so knew) technical aspects in combination with the practices introducded by the WEB, lead new challanges in testing Web services often stated in literature. Introduced in this paper is a not invasive functional testing approach for Web services based on the Design by Contract (DbC) paradigm. By using formal semantic specification in a consequent manner we can present a generic testing approach which enables us to introduce quality metric measurements not before viable in traditional testing in a practicable way. We present results of our first basic study at the Schweizer Bundesbahn (SBB) Informatik in Bern.

Keywords: Web service, design by contract, generic testdriver, generic oracle, semantic specification, conformity, black-box test, monitoring, contract checker, test data generation, non-invasive.

1 Introduction

Web services are the main realization of service oriented architecture (SOA) and are widely considered to be the strategic model for loose coupled, distributed computing ([9], [7], [4], [13]).

Its often stated that unforseen dynamically use of Web services leads to new challanges in the field of testing ([5], [1]). The WSDL specification ([12]) is a syntactical contract of a Web service and ist therefor on the first level of contracts ([3]). New conceptual enhancements of the traditional test techniques and methods is postulated ([8], [6]).

Immens growing dependencies and fast changing interfaces make it impossible to test bug fixes as well as newly added interfaces in manifold usage environments. Our answer to the challanges is a new fully generic, non-invasive functional test-technique method of contract based Web service functions. Test oracle and test driver are generally derived from the contract and the WSDL specification. By this newly introduced technique we can use the structural coverage proof of the formal semantic specification as a quality metric ([2]).

One basic side effect is the economic efficiency: writing testdrivers manually is obsolete.

P.P. Maglio et al. (Eds.): ICSOC 2010, LNCS 6470, pp. 695–698, 2010.

2 Contract-Based Testing

We connote a contract to a WSDL and its correspondig Web service. The contract is given in XML-syntax specified by an XSD definition. The expression of contract conditions are given in OCL. This language is used to be adoptable to model based approaches.

The technical architecture for the interaction of any client, VWS, monitor agents and WSUT is depicted in figure 1.

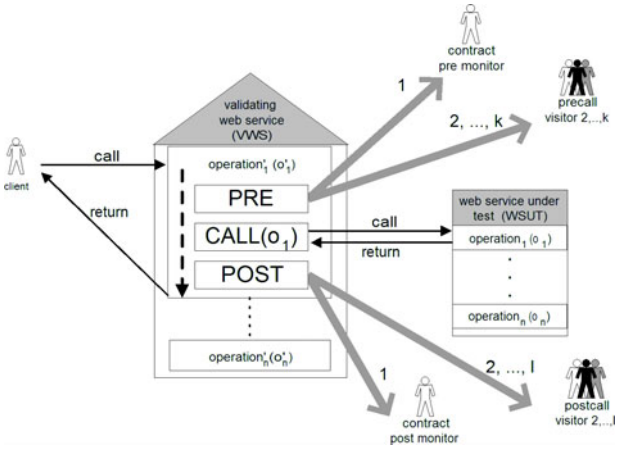

Fig. 1. VWS-Proxy and Monitor

The framework includes following functional kernels:

1. Proxy and monitor-agents generation,
2. Type-system analysis and persisting,
3. Runtime monitoring of the Web service under test (WSUT),
4. Testdriver generation,
5. Oracle generation,
6. Testdata import,
7. Test executing. i.e.:
   (a) running generic testdriver,
   (b) agent monitors and persists runtime values,
   (c) generic oracle,
      i. verifies pre-/post-conditions and invariants,
      ii. determines coverage metrics qualities of expression and atomic expression and
      iii. determines functional correctness of the particular WSUT function.

The observer pattern allows to plug in in further agents, especially ones that break the proxy conformity by, e.g., refusing calls with actual parameter values which don't achieve the preconditions and can be used to implement Meyer's *Modular Protection* ([10], pp. 345). We tested this first and necessary condition in a study at real business WSUT applications (ticketing system) at the Schweizer Bundesbahn (SBB).

2.1 An Example: Bookflight

We give a simple example to demonstrate the functionality and the benefit of using business rules in form of contracts right from the start.

Let's take the often used example of booking a flight.

precondition	`flight_nof > children_nof and children_nof >= 0 and location_from <> location_to`
postcondition	`result.akzeptedOrder.flight_nof > 0`

Fig. 2. Simple contract (3)

From the VWS-System GUI we start the test run by clicking a button to start to formentioned mechanism.

The result is depicted in figure 3. We focus on branche coverage (Branch), simple condition coverage (SCC), modified decision condition coverage (MCDC) and multiple condition coverage (MCC).

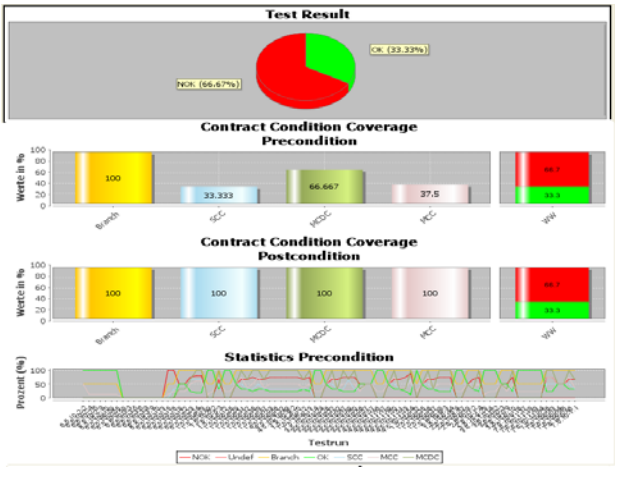

Fig. 3. Oracle Report (3)

In a study we implemented a testdata generator which uses the contract, transforms it to an equation with n unknowns (n = number of distinct atomar conditions) and calling the solver GlassTT ([11]). This works for about 80% of conditions used in the WSDL interfaces. This technique is also very effective in evaluating boundary values.

In further studies we will collect data in professional projects to measure and prove the aformentioned benefits in terms of numerary.

3 Summary

We presented the non-invasive validating system VWS framework, with which Web services can be supervised and tested based on contract in a contract-based way.

The non-invasive approach is realized by inserting an intermediate, transparent layer between caller and supplier that serves for validating the WSUT.

We introduce with the VWS testing framework a new technical and methodic way of testing Web services, based on behavioral contractual level. Conceptually this is not limited to a specific component technology, but for Web service testing it generates the most dramatic earnings. This method enables us to adopt structural quality metrics in thorough generic functional testing. The aforementioned metrics could not be measured practicable without formal semantic specification before.

References

1. Andrikopoulos, V., Benbernou, S., Papazoglou, M.P.: Evolving services from a contractual perspective. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009. LNCS, vol. 5565, pp. 290–304. Springer, Heidelberg (2009)
2. Averstegge, M.: Generisches testen von webservices. OBJEKTspektrum 4(4) (2010)
3. Beugnard, A., Jézéquel, J.-M., Plouzeau, N., Watkins, D.: Making components contract aware. Computer 32(7), 38–45 (1999)
4. Dustdar, S., Haslinger, S.: Testing of service-oriented architectures - a practical approach. In: Weske, M., Liggesmeyer, P. (eds.) NODe 2004. LNCS, vol. 3263, pp. 97–109. Springer, Heidelberg (2004)
5. Farooq, A., Dumke Reiner, R., Georgieva, K.: Challenges in evaluating soa test processes. In: Dumke, R.R., Braungarten, R., Büren, G., Abran, A., Cuadrado-Gallego, J.J. (eds.) IWSM 2008. LNCS, vol. 5338, pp. 107–113. Springer, Heidelberg (2008)
6. Heckel, R., Lohmann, M.: Towards contract-based testing of web services (2004)
7. Huang, H., Tsai, W.-T., Paul, R., Chen, Y.: Automated model checking and testing for composite web services. In: ISORC , pp. 300–307 (2005)
8. Martin, E., Basu, T.X.S.: Automated testing and response analysis of web services. In: Proc. the IEEE International Conference on Web Services (ICWS 2007), Application Services and Industry Track, July 2007, pp. 647–654 (2007)
9. Martínez, A., Martínez, M.P., Jiménez-Peris, R., Pérez-Sorrosal, F.: Zenflow: A visualweb service composition tool for bpel4ws. In: Proceedings of the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2005 (2005)
10. Meyer, B.: Object-Oriented Software Construction, 2nd edn. Prentice-Hall, Englewood Cliffs (1997)
11. Kuchen, H., Müller, R., Lembeck, C.: GlassTT – a Symbolic Java Virtual Machine Using Constraint Solving Techniques for Glass-Box Test Case Generation. Technical Report 102, Universität Münster, Institut für Informatik (2003)
12. W3C. Web Services Description Language (WSDL) 1.1 (2001), `http://www.w3.org/TR/wsdl` (last visited (2010.06.03))
13. Yang, J.: Web service componentization. Commun. ACM 46(10), 35–40 (2003)