A Framework for Modeling and Testing of Web Services Orchestration

Lina Bentakouk[1,2], Fayçal Bessayah[3], Mounir Lallali[1,2], Wissam Mallouli[4], Andrey Sadovykh[5]

[1]Univ Paris-Sud, LRI, UMR 8623, Orsay F-91405; [2]CNRS, Orsay, F-91405; {lina.bentakouk; mounir.lallali}@lri.fr
[3]TELECOM SudParis - CNRS SAMOVAR, F-91011 Evry, France; faycal.bessayah@it-sudparis.eu
[4]Montimage, 39 rue bobillot, 75013 Paris; wissam.mallouli@montimage.com
[5]SOFTEAM, 21avenue Victor Hugo, 75016 Paris; andrey.sadovykh@softeam.fr

***Abstract*—The Web Services and service-based systems gained extreme popularity in the recent years. While various testing frameworks exist for web services, they mostly cover unit testing and do not take into account the complete architecture built with service compositions. In our research we start from a high level system model expressed with SOA Logical Architecture UML2 Profile, from which we generate BPEL orchestration. Then formal testing techniques for a black box approach and several tools are used to check the conformance and the robustness of this orchestration.**

In this paper, we overview methods and tools being developed in the frame of the French national research project WebMov[1]. Finally, we present the case studies being executed and preliminary lessons learnt.

I. Introduction

SOA (Service Oriented Architecture) concept is becoming increasingly more important with the emerging popularity of Web services (WS) technology which offers a complete interoperability between systems regardless of their platform, implementation or underlying architecture. WS can be composed in a centralized way using WS-BPEL (Web Service Business Process Execution Language) which is able to manage a perfect orchestration between different service components. However, this orchestration induces an increase in size and complexity of services and consequently, leads to a major challenge for software testing.

The issue of testing Web service has been addressed differently depending on the available service description level and on the testing generation approach [1], [2], [3], [4], [5].

In this paper, we present a framework for testing Web services orchestration. The objectives of this platform are manifold. we define a methodology based on a high level abstraction view and an SOA based logical architecture for the design and composition of WS. Using the WS-BPEL model derived from this architecture, we derive formal models to generate test cases that take into account the specificities of a service orchestration. Our framework also addresses the issue of conformance testing, robustness testing, and the passive testing using monitoring techniques.

The rest of this paper is structured as follows: In section 2, we explain our approach. In section 3 we describe the tools chain developed. In section 4, we present the case studies on which we have applied our approach. Finally we conclude and present future work in Section 5.

II. Approach

Our approach contributes to the modeling and testing of Web services orchestration. It aims to cover the different steps of Web services life cycle. We define a methodology based on a high Level abstraction view and service oriented architecture based logical architecture for the design and composition of WS. This logical architecture permits to produce a BPEL model of Web service. Based on this model, we derive formal models of the composition from which functional tests are generated automatically to cover widely the Web services characteristics. These formal models allow us to apply model-based testing techniques and to reason with a non ambiguous model to automate the test case generation. To test effectively WS, a combination of techniques (active/passive testing, black/Gray box testing, conformance/robustness testing) were implemented. The robustness of the services is tested using fault injection techniques and a tool dedicated to this.

III. Description of The Tools Chain

Various tools have been developed under the project WebMov. These tools are intended to verify the conformance and robustness of a service orchestration at different levels and with different methods. Theses tools are integrated in a dedicated framework as shown in Fig. 1.

The *Modelio* CASE Tool[2] provides a UML2 Profile for the Logical Architecture language. This is a business architect dedicated language proposing a very limited set of concepts: Messages, Interfaces, Components, BPMN processes for

[1]This work is supported by the projects "WEB service MOdelling and Validation" (WEBMOV) of the French National Agency for Research (http://webmov.lri.fr/).

[2]Modelio CASE Tool by SOFTEAM, http://modeliosoft.com

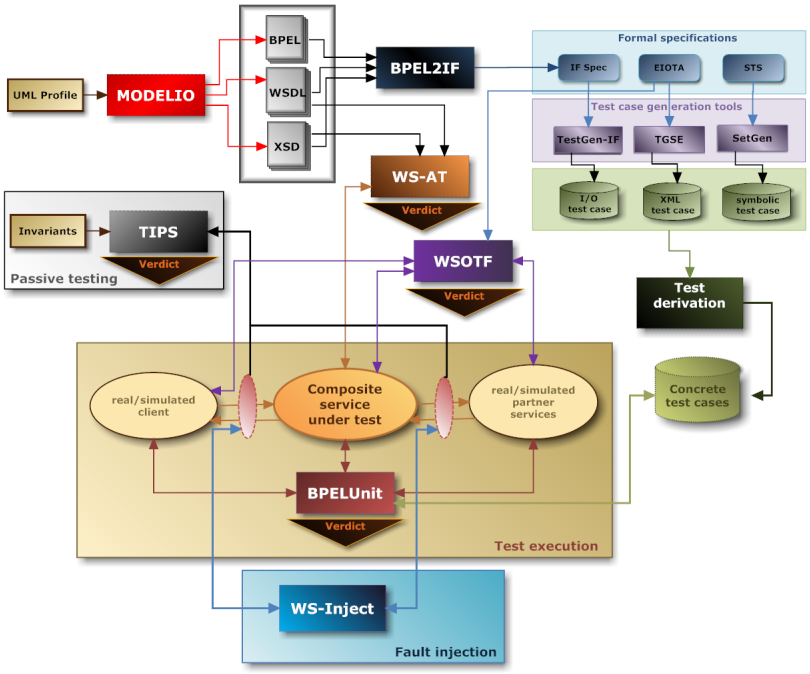

Figure 1. WebMov tools chain.

component internal behavioral specification. With a *Modelio* model-to-model transformation a platform specific model is created an initial model with UML2 Profiles for XSD, WSDL and BPEL. This model may be refined by a platform specialist with additional technical information. Then XSD, WSDL and BPEL files are created with dedicated model-to-code transformations integrated into *Modelio*. These files are inputs for the subsequent test activities described below.

The *WS-AT*[3] tool is used to perform a unit test of individual Web services. This tool gives a testing verdict on conformance of the Web service according to its WSDL interface description and allows to check the robustness of the service. *WS-AT* checks the operations existence, the exception management, the type of returned values and it observe whether each operation does not crash or hang by calling it using unusual values.

The specification files (WSDL and BPEL) are considered as a starting point for different methods and tools (e.g *BPEL2IF* tool). The BPEL file is considered as the specification model from which a formal model is generated to avoid any ambiguity. Based on the formal models a set of test generation tools is conceived and implemented (e.g. IF model[4] for TestGen-IF[5] tool, EIOTA[6] for TGSE [7] and STS [8] for SetGen[9][6]). Test cases generated by these tools are executed against the service orchestration and a verdict is given.

WSInject tool is a Web service fault injector able to inject both interface and communication faults. In the context of WebMov, this tool is used to study Web services robustness which is defined as the ability of a Web service to behave correctly when running in a stressful environment. *WSInject* is a script driven tool. Users can specify the type of faults and the injection conditions using a simple and powerful script grammar. During the experiments, the injector intercepts all SOAP messages exchanged between service partners. Then, according to the injection script, it will inject errors only on messages fitting the injection conditions.

The *TIPS*[10] tool allows to passively test a deployed orchestration service. The passive testing means that the service is tested without any interference by the tester. *TIPS* collects the traces of the service and checks if a set of properties called invariants are respected or not. The invariants are a set of behavioral constraints with time constraints that must be respected by the orchestration under test. The tool allows also to check the sequence order of exchanged messages, the communicated data, etc. The passive testing can be combined with an active testing and fault injection techniques in order to emit a verdict about the correctness of the deployed Web services.

[3]Web Service - Automatic Testing
[4]Intermediate Formalism model
[5]Test Generation with IF simulator
[6]Extended Timed Input Output Automaton
[7]Test, Generation, Simulation and Emulation

[8]Symbolic Execution System
[9]Symbolic Execution Tree Generation
[10]Testing Invariant for Protocols and Services

IV. Case studies

We have evaluated the WebMov tool chain on two Web services: the Travel Reservation System (TRS) and the Loan Approval Web Services. This tool chain will also be evaluated on a third Web service call Product Retriever. In this section we briefly present some results of experimentations performed on TRS.

A formal specification of the TRS Web service has been derived from its BPEL and WSDL files using BPEL2IF tool. This IF specification is used as a first input to TestGen-IF tool in order to automatically generate a set of test cases. Other tools like WSOTF and WS-AT were also used for tests generation. The number of generated test cases changed from a tool to another depending on the length of each test case and on the used data. Only a fail verdict was obtained for TestGen-IF tool and this was due to the application of the test cases to a mutant of TRS Web service where the messages correlation was removed.

The passive testing technique has been also applied on the TRS Web service using TIPS tool. Twelve functional properties that the TRS has to respect and called invariants have been designed based on the TRS test objectives. The obtained verdicts were Pass for some invariants and inconclusive for the others since they were never tested in the collected traces.

We also applied fault injection to the TRS application. Faults were injected in integer fields of messages between the BPEL client and its partner services. In all cases the BPEL process had a normal end. This indicates a lack of robustness, since the application accepts any values as valid.

V. Conclusion and Perspectives

In this paper we presented the WebMov approach and framework for testing Web Service orchestrations. The approach is based on modern SOA modelling languages, model transformations, code generation MDE methods and strong involvement of formal methods for testing of various aspects. The experiments with case studies showed the applicability and benefits of this approach with regards to industrial applications. The Future Internet and Cloud Computing are rapidly evolving research areas, that require thorough testing of all aspects including security and privacy. We believe that our approach can be extended to SOA compositions and cloud computing and help in addressing of the new coming challenges.

References

[1] P. Mayer, "Design and implementation of a framework for testing bpel compositions," Ph.D. dissertation, Leibnitz University, Germany, 2006.

[2] C. Bartolini, A. Bertolino, E. Marchetti, and I. Parissis, *Architecting Dependable Systems*, ser. LNCS, 2008, vol. 5135, ch. Data Flow-Based Validation of Web Services Compositions: Perspective and Examples.

[3] C. Bartolini, A. Bertolino, E. Marchetti, and A. Polini, "Towards Automated WSDL-Based Testing of Web Services," in *Proc. of ICSOC*, 2008.

[4] L. Frantzen, J. Tretmans, and R. de Vries, "Towards model-based testing of web services." in *Proc. of WS-Mate*, 2006.

[5] L. Frantzen, M. Huerta, Z. Kiss, and T. Wallet, "On-The-Fly Model-Based Testing of Web Services with Jambition," in *Proc. of WS-FM*, 2009.

[6] L. Bentakouk, P. Poizat, and F. Zaïdi, "A Formal Framework for Service Orchestration Testing based on Symbolic Transition Systems," in *Proc. of TESTCOM*, 2009.