Generation of Conformance Test Suites for Compositions of Web Services Using Model Checking

José García-Fanjul, Claudio de la Riva, Javier Tuya
Computer Science Department, University of Oviedo
Campus de Viesques s/n, Gijón, SPAIN
jgfanjul@uniovi.es, claudio@uniovi.es, tuya@uniovi.es

Abstract

Testing compositions of web services is complex, due to their distributed nature and asynchronous behaviour. However, research in this field is scarce. We propose a new testing method for compositions of web services. A formal verification tool (the SPIN model checker) will be used to automatically generate test suites for compositions specified in an industry standard language: BPEL. Adequacy criteria will be employed to define a systematic procedure to select the test cases. Preliminary results have been obtained using a transition coverage criterion.

1. Introduction

Web services are becoming the default choice when implementing distributed software. They are asynchronous, low-coupled and platform-independent. The composition of web services (specified with languages such as BPEL [18]) enables the implementation of interoperable business processes. Furthermore, it has encouraged an increasing investment in this kind of software worldwide, which doubled from 2003 to 2004, reaching $2.3 billion. That figure is expected to continue to grow and become $15 billion by 2009, according to IDC research studies [19]. This high acceptance by industry has led to concerns regarding the testing processes of web services software. Canfora and Di Penta [5] and Zhang and Zhang [24] have identified a number of unresolved challenges in the application of traditional software testing technologies to web services such as:

1. The need to remotely test web services, with its associated cost.
2. The impact that the limited information exposed about a web service has on the design of test cases.
3. The ability to dynamically search and invoke web services.

Bearing in mind the above mentioned challenges, in this research abstract we will propose a new testing method for compositions of web services. Related work will be reviewed in Section 2. Then, in Section 3, the proposal is specified. The abstract ends with the expected contributions of this research, in Section 4.

2. Related work

Research in verification and validation applied to compositions of web services may be basically classified in two categories: papers describing formal verification approaches and others that use testing techniques.

Most of the research in this field has been directed towards formal verification. However, recent results show the limited feasibility of automated verification applied to compositions of web services [4] [11]. The goal of formal verification approaches is to decide whether it may be said that certain properties hold in the composition under study. Fu et al [12] use the SPIN model checker to formally verify compositions of web services specified in BPEL. Their approach thus shares with ours the use of SPIN and the need to build a model for the business process, as it will be explained in Section 3. They do not generate test cases, as they use the model checker to verify certain (selected by hand) properties. In the same line of work, Foster et al [10] use Finite State Processes (FSP) to model compositions of web services and describe the use of the LTSA tool [9] to formally verify BPEL specifications. They propose specifying the desired properties in terms of Message Sequence Charts, a technique included in the Unified Modelling Language (UML). Using a different model and verification paradigm, Narayanan and McIlraith [20] propose annotating web services with semantic descriptions (DAML-S) of their capabilities, to subsequently encode these in a Petri Net.

Regarding testing approaches, Chun and Offutt [6] and Offutt and Xu [21] describe the application of mutation analysis and data perturbation in the testing of web services. Their processes are defined at the unit level, so the targets are the individual web services and not their composition. Bertolino and Polini [3] propose a framework for dynamic testing of web services interoperability. They introduce a testing stage called "audition" before the services are published on a UDDI registry. In combination with verification techniques, Huang et al [17] describe a method to test composite web services. They explicitly specify the web services behavior (using OWL-S) and define the desired properties by hand. Then, they use model checking to ascertain whether the properties hold.

In summary, many of the above mentioned works (such as [12], [17] or [20]) rely on the explicit annotation of web services behaviour. Thus, further research is needed on testing compositions of web services with no added knowledge but the specification of the composition itself. Furthermore, the selection of test cases is done manually in most of the papers we have found on testing compositions of web services. New research should spot the adoption of automatic algorithms for test case selection on this field.

3. Conformance testing of web services compositions

Our research hypothesis is that a new method for generating test suites for compositions of web services is needed, with the following characteristics:

- It will be static, so there will be no need to execute the software for obtaining the test cases. Thus, we avoid the cost of remotely executing the web services and undertake the first of the challenges listed in the introduction.
- The only required input will be a specification of the composition in BPEL. The obtained test cases will be independent from the particular implementation and we adhere to industrial standards. This decision is meant to tackle the third challenge listed in the introduction.

To build such a new method, we will rely on an existing technique called model checking [7]. It is a formal verification technique that enables the automatic detection of whether certain properties hold in a model. It has a number of well documented applications, ranging from the verification of protocols [23] to fault detecting in software systems [15]. SPIN is one of the most commonly used model-checking tools [16]. Using SPIN, properties can be specified by assertions in the model or shaped as Linear Temporal Logic (LTL) formulae. The tool searches all the possible states within the model and checks whether the properties hold. If not, it gives a trace of the steps illustrating the violation of the property, which is called a counterexample.

Model checking is commonly used for systems verification, but it can be applied to generate test cases [1] [14]. In order to obtain a test case for a certain requirement C, the model checker is fed with a model for the software and a LTL formula stating that C never holds. The output obtained from the tool is hence a counterexample in which the software fulfils C. That counterexample can be transformed into a test case, as it describes an execution of the software in which the desired test requirement holds.

The above technique can be adapted to generate test case specifications for conformance testing of BPEL compositions (here we use the term "conformance testing" as defined in [2]). Our method, which is depicted in Figure 1, comprises four steps: Step 1: Transforming BPEL to PROMELA (the input language of SPIN), Step 2: Applying an adequacy criterion, Step 3: executing the model checker (and obtaining a counterexample) and finally Step 4: Test case specification.

First of all, the business process must be transformed into PROMELA. We will also need to model the external behaviour of the different web services (called partners in BPEL) that participate in the business process. The BPEL specification does not directly include information about their behaviour. Thus, a mock model will be constructed for each partner based upon its interface with the business process.

Secondly, in order to produce test cases, test requirements must be identified. As it has been said before, this is commonly done by hand. Yet, we will describe systematic procedures to obtain test requirements from different adequacy criteria. These criteria will guide the instrumentation of the PROMELA code, in order to discern if an execution of the model meets the test requirements. In addition, LTL properties will be properly constructed expressing the negation of the identified requirements.

The third step is the execution of the model checker. The counterexample obtained from a SPIN run is a sample execution of the BPEL process in which the test requirements included in the LTL are exercised.

Lastly, to specify the test case, we will analyse how to get relevant information from the counterexample generated with SPIN. The test case specification will include the inputs and the desired output, both of them expressed in terms of the information exchanged between the business process and the partners.

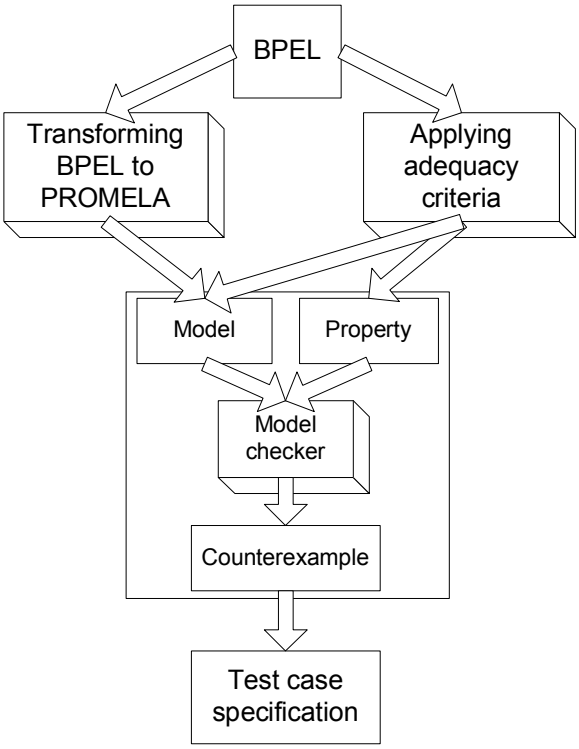

Figure 1. Overview of the proposed method.

Regarding the evaluation of the method, its application to real-life compositions of web services must be taken into account. However, as there are not many publicly available compositions [8], synthetic ones may need to be constructed. To validate our results, we will use fault-injection techniques with different implementations of the same composition. If our method is sound, the execution of the obtained test suite will enable us to discriminate the correct composition from the faulty ones. We also plan to contrast the faults detected using our method to the ones obtained applying other techniques. Controlled experiments will also be tried to further validate our approach.

In our preliminary work [13], we use a transition coverage criterion (taken from [22]) to select the test cases. Specifically, the criterion states that the resulting test suite must include test cases that cause every transition in the BPEL specification to be taken. To do so, on the second step of our method, transitions are identified in the BPEL specification and mapped to PROMELA. Also, a LTL property is constructed for each transition to find a counterexample for that given transition (a run in which the transition is exercised). To build a test suite that meets the above defined transition coverage criterion, the model checker should be executed as many times as transitions are identified in the BPEL. To reduce the number of test cases, all the transitions covered with each counterexample are taken into account. In our first case study, using the well-known "loan approval" sample composition, the number of test cases obtained is the minimum required to give transition coverage for the specification.

4. Expected contributions

The main contribution of our research will be the definition of a new method to obtain conformance test suites for compositions of web services. The method will rely on a model checking tool (SPIN) for obtaining test cases specifications from a model of the business process.

Our research will address how to transform a BPEL specification to a PROMELA model. Test cases will be automatically selected to fulfil certain adequacy criteria. Thus, we will describe procedures to:

- instrument PROMELA code, considering those criteria;
- construct LTL properties for the counterexamples to show sample executions of the model that meet the criteria;
- automatically obtain a test suite specification from the counterexamples that SPIN provides.

After the preliminary case studies, immediate lines of work are the application of different adequacy criteria, such as those described by Offutt et al in [22] and its automation. Research will also be directed to fully determine the scalability of the method.

5. Acknowledgements

This work is supported by the Ministry of Science and Education (Spain) under the National Program for Research, Development and Innovation, projects IN2TEST (TIN2004-06689-C03-02) and REPRIS (TIN2005-24792-E).

6. References

[1] P. Ammann, P.E. Black and W. Majurski, Using Model Checking to Generate Tests from Specifications, Second IEEE International Conference on Formal Engineering Methods, Brisbane (Australia), 1998, pp 46-.

[2] A. Bertolino and E. Marchetti, A Brief Essay on Software Testing, Chapter of *Software Engineering. Volume 1: Development process*, Third Edition, IEEE Computer Society/Wiley Interscience, 2005, pp. 393-411.

[3] A. Bertolino and A. Polini, The Audition Framework for Testing Web Services Interoperability, 31st EUROMICRO

Conference on Software Engineering and Advanced Applications, Porto (Portugal), 2005, pp. 134-142.

[4] T. Bultan, X. Fu and J. Su, Analyzing Conversations of Web Services, *IEEE Internet Computing*, 10(1), IEEE, 2006, pp. 18-25.

[5] G. Canfora and M. Di Penta, Testing services and service-centric systems: Challenges and opportunities, *IT Professional*, 8(2), IEEE, 2006, pp.10–17.

[6] S. Chun, and J. Offutt, Generating Test Cases for XML-based Web Component Interactions Using Mutation Analysis, 12th IEEE International Symposium on Software Reliability Engineering, Hong Kong (PRC), 2001, pp. 200-209.

[7] E.M. Clarke, O. Grumberg and D.A. Peled, *Model Checking*, The MIT Press, 2000.

[8] J. Fan and S. Kambhampati, A Snapshot of Public Web Services, *SIGMOD Record*, 34 (1), ACM, 2005, pp. 24-32.

[9] H. Foster, S. Uchitel, J. Magee and J. Kramer, Tool Support for Model-Based Engineering of Web Service Compositions, IEEE International Conference on Web Services, Orlando (USA), 2005, pp. 95-102.

[10] H. Foster, S. Uchitel, J. Magee and J. Kramer, Model-based Verification of Web Service Compositions, 18th IEEE International Conference on Automated Software Engineering, Montreal (Canada), 2003, pp 152-163.

[11] X. Fu, T. Bultan and J. Su, Synchronizability of Conversations among Web Services, *IEEE Transactions on Software Engineering*, 31(12), IEEE, 2005, pp. 1042-1055.

[12] X. Fu, T. Bultan and J. Su, Analysis of Interacting BPEL Web Services, Thirteenth International World Wide Web Conference (WWW 2004), New York (USA), 2004, pp. 621-630.

[13] J. García-Fanjul, J. Tuya and C. de la Riva, Generating test cases specifications for BPEL compositions of web services using SPIN, International Workshop on Web Services - Modeling and Testing, Palermo (Italy), 2006, pp. 83-94.

[14] E.L. Gunter and D. Peled, Model checking, testing and verification working together, *Formal Aspects of Computing*, 17(2), Springer, 2005, pp. 201-221.

[15] K. Havelund, M.R. Lowry and J. Penix, Formal Analysis of a Space-Craft Controller Using SPIN, *IEEE Transactions on Software Engineering*, 27(8), IEEE, 2001, pp. 1000-9999.

[16] G.J. Holzmann, *The SPIN Model Checker: Primer and Reference Manual*, Addison-Wesley Professional, 2003.

[17] H. Huang, W. Tsai, R. Paul and Y. Chen, Automated Model Checking and Testing for Composite Web Services, Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, Seattle (USA), 2005, pp. 300-307.

[18] IBM, Business Process Execution Language for Web Services version 1.1, URL: http://www-128.ibm.com/developerworks/library/specification/ws-bpel/.

[19] IDC, Research Reports, URL: http://www.idc.com/.

[20] S. Narayanan and S.A. McIlraith, Analysis and simulation of Web services, *Computer Networks*, 42(5), Elsevier, 2003, pp. 675-693.

[21] J. Offutt, and W. Xu, Generating Test Cases for Web Services Using Data Perturbation, *ACM SIGSOFT Software Engineering Notes*, 29(5), ACM, 2004, pp. 1-10.

[22] J. Offutt, S. Liu, A. Abdurazik and P. Ammann, Generating Test Data From State-based Specifications, *The Journal of Software Testing, Verification and Reliability*, 13(1), Wiley, 2003, pp. 25-53.

[23] A.W. Roscoe and P.J. Broadfoot, Proving Security Protocols with Model Checkers by Data Independence Techniques, *Journal of Computer Security* 7(2-3), IOS Press, 1999, pp. 147-190.

[24] J. Zhang and L.J. Zhang, "Web Services Quality Testing", *International Journal of Web Services Research*, 2(2), Idea Group, 2005, pp. 1-4.