Defining an SLA-Aware Method to Test Service-Oriented Systems

Marcos Palacios
Supervised by: José García-Fanjul and Javier Tuya

University of Oviedo
palacios@lsi.uniovi.es

Abstract. In the scope of Service Oriented Architectures (SOA), Service Level Agreements (SLAs) are used to specify the conditions that have to be fulfilled by both service provider and consumer. These stakeholders need checking whether the executions of the services fulfill the conditions or not, so the evaluation is an important and not trivial task within the testing process. This paper describes the current state of an ongoing PhD research that aims to define an SLA-aware testing method to test service-oriented systems. A model will represent relevant information associated to the terms of the SLA and, from this model, interesting situations will be identified and prioritized. The exercitation of these situations will be performed through a suited design and execution of test cases. In addition to this, the approach could be complemented with monitoring techniques, determining which situations have already been executed in an operation environment in order to select a set of situations that remain unexercised and need to be tested.

Keywords: Service Level Agreements, Software Testing, Monitoring, Service Oriented Architectures.

1 Scope and Motivation

During last decade Service Oriented Architecture (SOA) has emerged as a promising solution to develop interoperable and highly dynamic applications by integrating available services over the web. In some scenarios it is necessary that the service provider and the consumer negotiate and agree a set of conditions related to the use of the services. Such conditions are often specified in a contract or technical document named Service Level Agreement (SLA).

Within the tasks included in the management of the SLAs, testing has been identified as a challenge. Monitoring is often proposed as a suited technique to observe the behaviour of the software and detect whether the services are violating the conditions of the SLA but these approaches have limitations. First of all, the problems in the software are detected after they have occurred (reactive approaches) so consequences can not be avoided. Furthermore, there can be situations that are not exercised during the period of time the system is being monitored but they may occur in the future. In critical scenarios where an SLA violation may lead to severe consequences for stakeholders, monitoring could be complemented with testing methods that anticipate the detection of problems identifying and exercising interesting situations through a suited test design (proactive

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 164–170, 2012.

approaches). This research will mainly focus on defining an SLA-aware method that allows testing systems in the aforementioned scenarios.

The content of the paper is organized as follows. Section 2 outlines the research focus and the objectives of this PhD research. Section 3 and 4 present the ongoing work with results achieved so far and our proposal to evaluate and test SLAs. In Section 5 we briefly describe the state of the research in the addressed topic. Finally, conclusions and expected results are outlined in Section 6.

2 Research Focus

The final objective of this research is to detect faults in the software that can lead to SLA violations so their correspondent consequences can be avoided or mitigated. Hence, the main question can be defined as follows:

- ***How can we test service-oriented systems with a defined SLA before the executions cause non-expected consequences for the stakeholders?***

This goal can be achieved addressing the following set of research questions:

1. How can an SLA be evaluated for purposes of detecting faults in the software?
2. How can we represent the content of the SLA and other relevant objective information in a model?
3. What potential error-prone situations can be identified from such model?
4. How can these situations be exercised through a test design and execution?

This ongoing PhD research aims at contributing to solve the problem of testing service-oriented systems with the elaboration of a test method that addresses the aforementioned questions. Furthermore, we will also consider complementarity between testing and monitoring:

5. How can we exploit the benefits of monitoring to collect information when the software is deployed in an operation environment?

3 SLAs and Their Evaluation

During the first steps of this ongoing research, we studied the state of the art in the field of testing service-oriented systems that allow a specific capability such as dynamic binding [8]. In this work, we identified that the testing of SLAs is a promising research line that has not probably received enough effort yet. Furthermore, although different languages have been proposed with the aim at standardizing the specification of SLAs, it has been WS-Agreement (WSAG) [1] which has received more attention so far. The specification of a WSAG is composed of three parts: name, context and terms. The latter describe the obligations that have to be fulfilled and are the most important elements of the SLA. A guarantee term includes the *Scope* which is the list of services this guarantee applies to, a *Qualifying Condition* which is an assertion under which the term is applied, and a *Service Level Objective* (SLO) that has to be guaranteed. These terms can be combined using three compositor elements: *All*, *OneOrMore* and *ExactlyOne* (logical AND, OR and XOR respectively).

The evaluation of an SLA requires making a decision about each of the elements specified in such SLA. Typically, the evaluation of an SLA may be depicted with a two-way traffic light indicator (green / red) which represents whether the agreement has been fulfilled or violated respectively. In order to make this decision, it is necessary to check the evaluation of all the terms of the SLA, also taking into account the compositor elements. Hence, the term can be considered as the most indivisible element of the SLA. At runtime and after having analyzed the collected information:

- A term is considered ***Fulfilled / Violated*** if and only if the services specified in the *Scope* have been executed, the *Qualifying Condition* is honored so the term is valid and the conditions specified in the *SLO* are satisfied /are not satisfied.

In addition to this two values *Fulfilled* and *Violated*, it is possible to identify and, in fact, WSAG does it, a third potential value that a term can take after its evaluation.

- A term is considered **Not Determined** if and only if the services specified in the *Scope* have not been yet executed.

The interpretation of this value according to WS-Agreement is that, at the moment of the evaluation, no activity regarding the term has happened yet or no activity is currently happening that allows evaluating whether the term is fulfilled or violated.

After studying the internal elements of a term and its interpretation according to the standard, we have identified a new situation where the term can be found after its evaluation and which has not been explicitly identified in WS-Agreement. This situation arises when the services specified in the Scope of the term have been executed but those executions do not satisfy the Qualifying Condition so such term is not valid and should not be applied for the purpose of the evaluation of the global SLA.

- A term is considered ***Inapplicable*** if and only if the services specified in the *Scope* have been executed but the *Qualifying Condition* is not satisfied.

As an SLA specifies a hierarchy of terms combined through the compositor elements, we will define a logic that allows unequivocally representing all the potential situations that may occur during the SLA evaluation process. It is necessary that the logic includes these two additional evaluations, leading to a four-valued logic where *Not Determined* and *Inapplicable* are two similar interpretations of the treatment of the null value in the three-valued logic (broadly studied in the context of DBMS and applied in the scope of software testing). The definition of this logic allows addressing the first research question of Section 2.

In addition to the potential values that terms can take during their evaluation, there are different factors that affect the evaluation and we will have to take into account in this PhD research, for example, the scope (we may have to evaluate only an specific term, a compositor element, the whole SLA, etc.) or the point in time of the evaluation (after each service execution, a specific number of executions, a temporal window, etc.).

4 Testing SLAs

Considering the logic described in Section 3 as a foundation to evaluate the SLAs, we have planned to define an SLA-aware method that contributes to improve the process of testing service based systems. This method is depicted in Fig. 1 where the SLA is always associated to a Software Under Test (SUT).

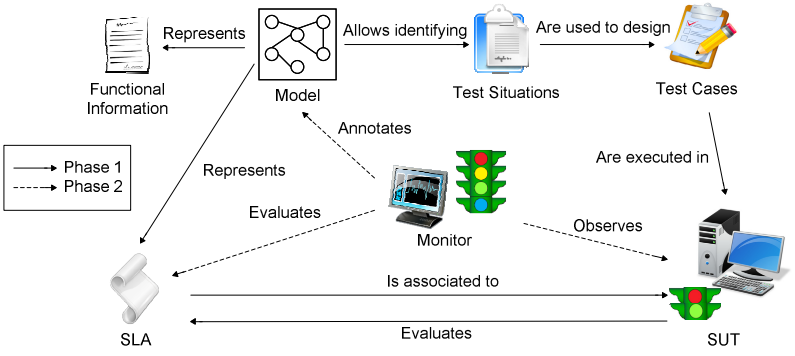

Fig. 1. SLA Test Method

One of the inherent characteristics of the management of SLAs is the capability to determine whether the conditions specified in the agreement are being fulfilled or not. Regarding this issue, the SUT must have a mechanism that observes the executions, collects data and, based on this information, performs the evaluation of the SLA. In fact, from our point of view this mechanism is part of the SUT and, hence, we assume that it also needs to be tested.

The method that will be developed during the first phase of this research is composed of the following processes, which will allow achieving our expected results:

1. Model the content of the SLA and other relevant information.
2. Define a set of criteria that allow the identification and prioritization of interesting test situations from the model.
3. Define procedures to design a set of test cases that exercise the selected situations.

Previous studies have represented the content of the SLA in a model [6][4]. In our approach, the development of this model should be useful in the process of testing service based systems and it is aligned with the second research question of Section 2. This model ought to contain, at least, all the relevant information specified in the SLA. Furthermore, we may consider representing other kind of information that could help to improve the testing process. This information can be extracted from different sources (requirements, behaviour models, etc.) and, although the SLAs typically contain non-functional characteristics, we consider that it could be interesting to represent some functional aspects of the SUT in the agreement. For instance, we are considering representing constraints about the evaluation of terms depending on the execution order of services, which could be specified using UML sequence diagrams.

From the previously constructed model and taking the proposed logic into account, different sets of test situations can be initially identified (aligned with the third research question of Section 2). The evaluation of terms, compositor elements or the global SLA with the values *Violated* (representing existing problems in the SUT), *Not Determined* and *Inapplicable* (both representing situations that have not been yet exercised) identifies potential error-prone situations that need to be tested before they lead to undesirable consequences for the stakeholders. Regarding this, a new problem may arise. In a previous work [7] we addressed the design of test specifications from

the conditions specified in the SLA. In such study, we realized that the number of situations identified from a low complexity agreement could be unmanageable. Hence, it is necessary to define criteria that allow reducing and prioritizing such situations in order to obtain a reasonable cost-effective set of tests. These criteria may be based on different factors such as business risks, economic penalty fees, etc.

The identified situations will be exercised through a suited design and execution of test cases (aligned with the fourth research question of Section 2). Typically, a test case should cover as many situations as possible in order to obtain a final reduced but effective set of test cases. This is even more important in SOA systems because we do not probably have unlimited and full access to the services but executions usually imply an economic cost. Thus, we will have to define criteria that allow us to design good test cases and decide their priority to be executed, overall if there are limitations that hinder the execution of the complete set of tests.

Finally and as we have previously outlined, we have planned to exploit the benefits of monitoring techniques to complement the approach (aligned with the fifth research question of Section 2). The observation of the software deployed in an operation environment could be a very useful task to recollect broad information and provide an interesting feedback. The monitor will show, for example, which situations remain unexercised and which situations are currently causing problems in the system. With this information, new decisions may be made in order to identify new situations or to prioritize existing ones in future regression tests.

5 Related Work

In the scope of service oriented computing, few approaches have addressed the identification of tests from the SLA specification. Di Penta et al. [3] perform black-box and white-box testing using Genetic Algorithms to detect SLA violations in service compositions where multiple services can be potentially invoked. In a previous work [7] we propose to test the conditions of the SLA specified in WSAG using a well-known testing technique, the Category Partition Method. Furthermore, there are approaches that do not specifically test SLAs but contribute to enable the necessary infrastructure. For example, Bertolino et al. [2] propose the PUPPET framework, which allows generating stubs from the WSAG, WSDL and BPEL specification of the services to test SLA-aware service compositions.

On the other hand, several works have addressed the testing of SLAs using monitoring approaches. Mahbub and Spanoudakis [6] propose to model and monitor the conditions specified in WS-Agreement using an Event Calculus (EC) based approach. Leitner et al. [5] propose a framework that allows monitoring and predicting SLA violations before they have occurred using machine learning techniques. Oriol et al. [9] aims at monitoring and detecting SLA violation at runtime using SALMon. This system allows triggering adaptive actions to correct the SLA violation after been detected. Raimondi et al. [10] proposed a system that automatically monitors SLAs, translating timeliness constraints into timed automata, which is used to verify traces of services executions.

6 Conclusions and Expected Results

This paper presents an overview of an ongoing PhD research in the field of testing service-based systems with Service Level Agreements. After having reviewed the state of the art in such research line [8], we have considered proposing an SLA test method, describing the issues we have planned to deal with, proposing potential solutions to solve them and stressing particular difficulties that arise from each of them.

In this research we expect to achieve a detailed description of how these systems can be tested according to the conditions that are specified in the SLA. This method is a proactive approach to identify, prioritize and exercise interesting test situations using a model that represents relevant information of the SLA. The second direction of the research involves exploiting the advantages of reactive approaches to improve and complete the potential information gathered from the tests.

Finally, a complementary and transversal expected result in this research will be the attempt to integrate existing solutions when possible. For instance, the execution of the tests could be performed using a framework (e.g. [2]) that makes easier the generation of a suited test infrastructure of the SUT. Furthermore, we will study how an existing monitoring system with the capability to check SLAs (e.g. [6]) could be adapted and integrated.

Acknowledgments. This PhD research is being partially funded by the Department of Science and Innovation (Spain) and ERDF funds within the National Program for Research, Development and Innovation, project Test4DBS (TIN2010-20057-C03-01) and FICYT (Government of the Principality of Asturias) Grant BP09-075.

References

1. Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web Services Agreement Specification (2007)
2. Bertolino, A., De Angelis, G., Frantzen, L., Polini, A.: Model-Based Generation of Testbeds for Web Services. In: Suzuki, K., Higashino, T., Ulrich, A., Hasegawa, T. (eds.) TestCom/FATES 2008. LNCS, vol. 5047, pp. 266–282. Springer, Heidelberg (2008)
3. Di Penta, M., Canfora, G., Esposito, G., Mazza, V., Bruno, M.: Search-based Testing of Service Level Agreements. In: 9th Annual Conference on Genetic and Evolutionary Computation (GECCO 2007), London, pp. 1090–1097. ACM, New York (2007)
4. Kotsokalis, C., Yahyapour, R., Rojas Gonzalez, M.A.: Modeling Service Level Agreements with Binary Decision Diagrams. In: Baresi, L., Chi, C.-H., Suzuki, J. (eds.) ICSOC-ServiceWave 2009. LNCS, vol. 5900, pp. 190–204. Springer, Heidelberg (2009)
5. Leitner, P., Michlmayr, A., Rosenberg, F., Dustdar, S.: Monitoring, Prediction and Prevention of SLA Violations in Composite Services. In: IEEE International Conference on Web Services (ICWS) Industry and Applications Track (2010)
6. Mahbub, K., Spanoudakis, G.: Monitoring WS-Agreements: an Event Calculus based Approach. Test and Analysis of Service Oriented Systems, Springer V, pp. 265–306 (2007)

7. Palacios, M., García-Fanjul, J., Tuya, J., de la Riva, C.: A Proactive Approach to Test Service Level Agreements. In: Fifth International Conference on Software Engineering Advances (ICSEA), pp. 453–458 (2010)
8. Palacios, M., García-Fanjul, J., Tuya, J.: Testing in Service Oriented Architectures with Dynamic Binding: A Mapping Study. Information and Software Technology 53(3), 171–189 (2011)
9. Oriol, M., Marco, J., Franch, X., Ameller, D.: Monitoring Adaptable SOA System using SALMon. In: Workshop of Service Monitoring, Adaptation and Beyond (MONA+), ServiceWave Conference (2008)
10. Raimondi, F., Skene, J., Emmerich, W.: Efficient Online Monitoring of Web-Service SLAs. In: Proceedings of the 16th ACM SIGSOFT Int. Symposium on Foundations of Software Engineering (SIGSOFT 2008/FSE-16) (2008)