Delft University of Technology
Software Engineering Research Group
Technical Report Series

Online Testing of Service-Oriented Architectures to detect State-based Faults

Michaela Greiler, Hans-Gerhard Gross, Arie van Deursen

Report TUD-SERG-2009-029

TUD-SERG-2009-029

Published, produced and distributed by:

Software Engineering Research Group
Department of Software Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4
2628 CD Delft
The Netherlands

ISSN 1872-5392

Software Engineering Research Group Technical Reports:
http://www.se.ewi.tudelft.nl/techreports/

For more information about the Software Engineering Research Group:
http://www.se.ewi.tudelft.nl/

Online Testing of Service-Oriented Architectures to detect State-based Faults

Michaela Greiler
supervised by Hans-Gerhard Gross and Arie van Deursen

Delft University of Technology, Delft, The Netherlands,
{m.s.greiler|h.g.gross|arie.vanDeursen}@tudelft.nl

Abstract. Service-oriented architectures have found their way into industry to enable better business-to-business cooperations. With this software architecture new challenges for software development and testing appeared.
In this proposal we discuss the problem of testing these complex, and distributed systems in dedicated test environments. We argue that state and configuration of the production system can influence system behavior in an unexpected way, and that test environments do not reflect the final system adequately. Therefore, we propose the development of an online testing method to detect state-based faults, and discuss related research challenges and solutions.

1 Introduction

A service-oriented architecture (SOA) is a software architecture that aims at reusability and interoperability by exposing functionality as loosely coupled services, that can be composed to form business processes. Testing such systems is aggravated by stakeholder separation: third party services are black-boxes for integration testers, and influence in their evolution and maintenance is not permitted. Dynamic features, like ultra-late binding or dynamic composition of services, allow a flexible and dynamic way of composing a concrete system just at runtime, but restrict the ability to test a priori. The context in which a service will be used is often unknown at the service's development time, imposing problems for service testers to predict and foresee possible requirements and usage scenarios. The limited testability of service-oriented systems imposes the need of reconsidering and redesigning traditional, and inventing new testing methods and processes, as mentioned in [5, 8, 10, 12].

This proposal suggests online testing as a method to identify state-based faults in online reconfiguration of service-oriented systems. Online testing is testing carried out in the realm of the production system, and in parallel to its operational performance. In our context, state comprises information about previous executions, the concrete set of installed and active programs, as well as history of preceding configuration activities. Reconfiguration includes evolution of business processes, services, business logic and changes in the execution environments.

The proposal is structured as follows: first, related work is summarized. Section 3 states the problem and formulates the research hypothesis. The related research challenges and possible solutions are discussed subsequently. The research plan is outlined in Section 5, followed by the evaluation plan in Section 6. The expected outcomes are summarized in Section 7.

2 Related Work

Influenced by the need of new testing methods and techniques for SOA, many approaches on unit, integration, interoperability, and regression testing can be found in the literature [4, 6]. We are focusing especially on related work concerning integration and online testing, where in particular the lack of control of integrated services e.g., to access the service code or to put a service in test mode, and the lack of information of integrated services e.g., to generate stubs, aggravate testing.

Similar to online testing is in vivo testing [7], also dubbed as perpetual testing[1], used to assess live applications after deployment. However, while in vivo testing focuses on performing unit tests, our proposal focuses on integration testing of service compositions. The scope of the test naturally affects the test approach, the test cases required, and their number. Online testing, as we see it, does not replace offline integration testing, but offers an efficient approach to address reconfiguration faults that cannot be identified in the offline test environment.

Bertolino et al. [1, 2] have also looked at the field of online testing, with interoperability testing, approaching the testability problem by redirecting service invocations to stubs. They suggest that the originally passive registry should take over an active audition role, and check the correctness of a service against its specification before listing it. Services already active in the SOA do not take part in the testing process. In contrast, our method will check the service implementation against expectations (specification) of requesting services, and not against its own specification.

Heckel et al. [11] discuss an approach, in which a central third party, called the discovery agency, checks that a service conforms with its specification, i.e., a model, by performing full behavioral and functional tests, before it is registered. Later on, other services can compare their requirements against the model provided by a discovered service. Their assumption is that the service could be faulty, either unintentionally because of insufficient testing, or intentionally for introducing malicious services. This differs from our online testing in which we focus on integration faults that are not easily caught using just offline testing. Further, it is not obvious in their description how a service checks its requirements model against the service specification model.

To run a test in the production system, undesired side effects need to be avoided, thus the system must provide test isolation, which has been addressed by [3, 9, 13]. Our approach to achieve test isolation differs from these approaches where a test sensitive component is cloned, replaced with a proxy, either in test mode or operational mode, or where testing is completely aborted. In our online testing framework, a component is not cloned, but two versions of a service are instantiated, whereby inheritance is used to allow test isolation, which makes the solution less expensive. Since both services are available at the same time, testing does not block the operational service, and can be executed concurrently.

All testing approaches abstract away from the complex and heterogeneous environments in which the services are operated, and do not mention their influence on systems' behavior.

[1] http://www.ics.uci.edu/∼djr/edcs/PerpTest.html

3 Problem Statement and Research Hypothesis

Testing of SOA systems means testing of organization-spanning systems-of-systems. In most cases it is not feasible to set up an accurate test environment, that represents the final environments of all involved parties in which services are used. Many parts of third parties have to be stubbed or mocked, even if there is a lack in provided information. Different and heterogeneous execution environments and their configuration have to be recreated, whereas a clean initial system state still does not reflect the production environment. Beside resource limitations, also time restrictions hinder adequate set-ups, leading to the risk that even tested software fails during online integration in the production environment.

We want to determine how the state of a system can influence its behavior in an unexpected way, and develop an online testing method for identifying state-based faults during online reconfiguration of service-oriented architectures.

On account of this the research hypothesis is: "Online testing is an effective strategy for detecting state-based faults in service-oriented systems."

4 Research Challenges and Proposed Solutions

An efficient online testing method to detect reconfiguration faults has to address following challenges.

What are the typical reconfiguration faults in SOAs, and which state and configuration produce those, even if software and services have been offline tested?

The main challenge is to determine which state and configuration data influence program execution in a SOA environment. This has to be narrowed down by the question: "Which information is often left out in the set-up of test environments?".

To define possible faults it has to be clear how reconfigurations of service-centric systems are propagated to local and remote systems. Possible faults have to be partitioned in those that can easily be detected during offline testing, and those that cannot. Latter faults have to be subdivided again, based on the question: "Which errors are already handled by the middleware, and which have to be handled by services and systems themselves?"

Regarding state and configuration, security and authentication policies are often not known or accurately implemented in a test environment. Many errors, like service unavailability and deployment errors, will be detected and partly solved by the middleware. During composition, and service execution, typical integration faults, like interface, data format, and communication protocol mismatches are to be expected. Further, dependency errors caused by evolution of parts of the system can occur, especially if some of the services and systems involved are stubbed or mocked in the test environment. It is important to analyze the impact of software evolution on the existing systems, either local or remote, and which layers of a SOA are affected. This can reach from changes in the business process layer, to the business logic layer and also to changes in the execution environment. Also application tolerance, which is the tolerance of multiple applications for each other, has to be addressed if a new services is

integrated. Changes in the backend of a service, be it the business logic or the configuration of the execution environment, can have rippling effects on local or even remote systems. These changes can be for instance, policies, new installed software, changes of database schema or changes of the database management system. Such modifications can cause service level agreement violations, because the network load increases, the new database management system responses slower, or access is now denied.

How should an effective online testing framework for SOA integration and system testing be designed, and what are the requirements for such a framework?

Challenging during design and development of an online testing method, are test isolation, performance, and application enhancement effort. If a system's behavior is assessed online, it has to be guaranteed that testing is isolated from the production system, and no unintended side effects appear. Test execution can only take place if no performance decreases or even unavailability of service are experienced. Enhancement costs for applying online testability mechanisms have to correspond to the positive impact accompanied with them. This implies that the tradeoff of online testing is corporeal. "But how can the effectiveness of an online testing method be assessed and measured?" Evaluation could be based on the capability of online testing to reveal faults in contrast to offline testing.

Information regarding the state of execution environments and infrastructures (e.g. application server, messaging bus, etc.) should support online testing to reveal faults. "But what is an accessible and uniformed way to provide state information to testing entities?"

Monitoring capabilities can be used to notify about reconfiguration changes in the system. An observer service, installed on each server involved in the SOA environment, could provide access to this information to interested and authenticated testing entities.

How can test cases, oracles, and system models be derived or generated?

In many SOA environments, abstract business models, business process models or UML documents, as well as service descriptions are present. Those represent a good foundation to derive system models and test cases. Test cases should not represent full behavioral tests, but address aspects that could not been tested offline.

To solve the oracle problem, functional equivalence testing can be applied. The running system is considered to be correct and the test output for testing the new, adapted version of the system is compared with the output of the old, stable system. Capture and replay techniques can help to increase observability, and controllability of the testing process, because old input and output sequences are recorded and reused ad libitum.

5 Research Plan

We have implemented an online testing framework providing test isolation in a case study, based on OSGi (Open Service Gateway Initiative), to evaluate the fault finding capabilities of our online testing approach. We could gain a first insight in state-based faults, mainly caused by missing required packages, wrong class bindings, and inconsistencies in the overall system. Services have been consistent with their specification, but when integrated in the execution environment, inconsistencies between different services emerged.

In the coming year, we plan to set up a SOA laboratory in order to examine configuration and state information that is significant during system reconfiguration, and to understand better how these influence the testing effectiveness and accuracy. Based on the results we want to develop a fault taxonomy for faults caused by evolutions in distributed, heterogeneous runtime environments.

In the next year, our online testing method has to be enhanced to identify these faults, and to be applicable in industrial SOA environments (e.g., IBM product suite).

Subsequently, we will explore capabilities to reduce the application enhancement effort. This will include automatic generation of test cases, and of testability artifacts allowing runtime service assessment. With this outcome, the online testing method will be extended to an open source online testing framework, providing support for test generation and execution. Throughout the research, the outcomes will be evaluated based on case studies.

6 Evaluation Plan

The SOA laboratory is a network of distributed servers, on which services, installed in different execution environments, form actual applications, inspired by industry. Communication is handled by a centralized service registry and bus.

This laboratory will be used to conduct representative case studies, following Yin [14], to evaluate our online testing method. Update scenarios of different parts of the system will function to determine the accuracy of the test environment to prevent failures, and the fault finding capabilities of our method.

In collaboration with our industrial partner, we also want to assess the effort to prepare an application, with support of our framework, for online testing. For that, we will measure, for example, the required time and the number of necessary changes in the existing system. A comparison of effort and fault finding capabilities should indicate if online testing has a positive tradeoff.

7 Expected Outcomes

The resulting contributions of the doctoral studies include:

1. A new approach for online testing to detect state-based faults.
2. A series of fault models classifying faults that appear especially in the production environments.
3. Empirical evidence concerning the effectiveness of the proposed approach.
4. An open source tool for online test support, including generation and execution of test cases.

Our results should corroborate the hypothesis by indicating the effectiveness of online testing to detect state-based faults during system reconfiguration.

References

1. A. Bertolino, L. Frantzen, A. Polini, and J. Tretmans. Audition of web services for testing conformance to open specified protocols. In *Architecting Systems with Trustworthy Components*, pages 1–25. Springer, 2006.
2. Antonia Bertolino, Guglielmo Angelis, Lars Frantzen, and Andrea Polini. The plastic framework and tools for testing service-oriented applications. In *Software Engineering: International Summer Schools, ISSSE 2006-2008, Salerno, Italy, Revised Tutorial Lectures*, pages 106 – 139, Berlin, Heidelberg, 2009. Springer-Verlag.
3. Daniel Brenner, Colin Atkinson, Oliver Hummel, and Dietmar Stoll. Strategies for the runtime testing of third party web services. In *SOCA '07: Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications*, pages 114–121, Washington, DC, USA, 2007. IEEE Computer Society.
4. A. Bucchiarone, H. Melgratti, S. Gnesi, and R. Bruni. Testing service composition. In *Proceedings of the 8th Argentine Symposium on Software Engineering (ASSE'07) Mar del Plata, Argentina*, pages 29–31, August 2007.
5. Gerardo Canfora and Massimiliano Di Penta. Testing services and service-centric systems: Challenges and opportunities. *IT Professional*, 8(2):10–17, 2006.
6. Gerardo Canfora and Massimiliano Penta. Service-oriented architectures testing: A survey. In *Software Engineering: International Summer Schools, ISSSE 2006-2008, Salerno, Italy, Revised Tutorial Lectures*, pages 78–105, Berlin, Heidelberg, 2009. Springer-Verlag.
7. Matt Chu, Christian Murphy, and Gail Kaiser. Distributed in vivo testing of software applications. In *ICST '08: Proceedings of the International Conference on Software Testing, Verification, and Validation*, pages 509–512, Washington, DC, USA, 2008. IEEE Computer Society.
8. Schahram Dustdar and Stephan Haslinger. Testing of Service-Oriented Architectures - A practical approach. In *Object-Oriented and Internet-Based Technologies*, pages 97–109. Springer Berlin/ Heidelberg, 2004.
9. Alberto González, Éric Piel, and Hans-Gerhard Gross. Architecture support for runtime integration and verification of component-based systems of systems. In *1st International Workshop on Automated Engineering of Autonomous and run-time evolving Systems (ARAMIS 2008)*, pages 41–48, L'Aquila, Italy, September 2008. IEEE Computer Society.
10. Michaela Greiler, Hans-Gerhard Gross, and Khalid Adam Nasr. Runtime integration and testing for highly dynamic service oriented ICT solutions – an industry challenges report. In *TAIC-PART '09: Proceedings of the Testing: Academic & Industrial Conference on Practice And Research Techniques*, pages 51–55, Windsor, UK, 2009. IEEE.
11. Reiko Heckel and Leonardo Mariani. Automatic conformance testing of web services. In *Proc. Fundamental Approaches to Software Engineering (FASE)*, pages 34–48. Springer, 2005.
12. Michael P. Papazoglou, Paolo Traverso, Schahram Dustdar, and Frank Leymann. Service-oriented computing: State of the art and research challenges. *Computer*, 40(11):38–45, 2007.
13. Dima Suliman, Barbara Paech, Lars Borner, Colin Atkinson, Daniel Brenner, Matthias Merdes, and Rainer Malaka. The morabit approach to runtime component testing. In *COMPSAC '06: Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC'06)*, pages 171–176, Washington, DC, USA, 2006. IEEE Computer Society.
14. R. K. Yin. *Case Study Research, Design and Methods*. Sage Publications, Beverly Hills, CA, second edition, 1994.

TUD-SERG-2009-029
ISSN 1872-5392