A Runtime Testing Framework for Web Services

Senol Arikan, Markus Hillenbrand, Paul Müller
University of Kaiserslautern
D-67653 Kaiserslautern, Germany
{arikan, hillenbr, pmueller}@informatik.uni-kl.de

1 Motivation

As a most famous implementation technology for service-oriented architectures (SOA), Web services are becoming even more popular in the world of distributed software systems. More and more distributed systems are being implemented using this technology. The Venice Service Grid [5] is a SOA based framework for building secure and dependable distributed applications; it supports service developers during creation, deployment, maintenance, and usage of Web services. Venice provides a solid authentication and role-based authorisation environment, which is based on several basic concepts like single sign-on and federations of trust [4].

Venice offers a secure service-based communication infrastructure for widely deployed Web services. As those Web services are deployed on the Internet by different providers, the functional adherence of the Web services implementation to the underlying specification cannot be guaranteed, e.g. providers can change the implementation of Web services at any time without notifying their customers. To be able to test the compliance of these services with their specification, Venice provides a sophisticated runtime testing framework.

The proposed testing framework enables Web service developers to test services fully automatically at runtime. This way, service developers and users can be sure that a service complies with its specification. If an error occurs during the actual service execution, e.g. in the case that a service cannot be reached or its output does not correspond the expected values, service developers or service providers will be notified about this error. The framework can also analyse test results and store them in a database for future observations.

2 The Venice Testing Framework

The architecture of the framework is shown in Figure 1. The framework is composed of several components like TestManager, UnitTestGenerator, InputGenerator, and a database which in combination execute the framework functionality. It uses other Venice services such as the Single Sign-on Service (SSO) or the Notification Service (NS). The Venice testing framework enables a user to test a new Web service (WS) after registration at the SSO service. As soon as the SSO service gets a request for a new service registration, the testing framework begins with automatic tests of the new Web service. The SSO service notifies the Test Service through the Notification Service. The Test Service marks the new service as “to test” and saves it in the Testing Registrations database. The Test Service uses a first in first out (FIFO) method for processing the Web services from the Testing Registrations database. It delivers the WSDL specification and test cases given by the service developer to the components UnitTestGenerator and InputGenerator.

These components of the framework are responsible for generating test suites and test inputs which are required for test execution. The UnitTestGenerator uses JCrasher [3], a test generation tool that can automatically generate JUnit [6, 2] tests for a given Java class. The UnitTestGenerator generates a test class from a WSDL file by using the Axis utility WSDL2Java [1]. The generated JUnit tests are gathered in a test suite and delivered to the Test Service. The test cases have to be described in XML format by the service developer. This XML file includes input data, expected results for this input data, the operation name as well as the description of the test case. The InputGenerator generates input data from this test case and returns it to the Test Service, which will be used for further test processes. The Test Service executes these test suites with given inputs and gets the test results. These results will also be analysed by the Test Service. It compares the test results with the expected outputs, defined in the given test case. If test outputs do not comply with the expected results or if the test process returns no results, the Test Service notifies the affected service users and service providers through the Venice Notification Service. When all the tests have been successfully completed, the tested service will be registered and stored into the database “Registered Services”.

Furthermore, the Test Service provides an external inter-
face. Thus, it can be called to test any other Web service. The service to be tested will be marked “to test” in the same way as above and the whole test procedure will be processed for it. Consequently, a user can call the Test Service periodically for any Web service, and check its conformity to the corresponding service specification at runtime.

3 Conclusion and Future Work

The Venice testing framework allows every Venice entity (i.e. user or service) to check the conformity of a Web service at both service registration time and runtime. This way an entity can test the functionality and the availability of a Web service automatically at any time. Additionally it is possible to frequently test the Web services after deploying them on a larger scale and present the testing results to interested users.

In this paper the basic ideas and fundamental components of the framework have been discussed along with a brief overview of the Venice testing framework architecture. As future work, the implementation will be finalised and evaluated. The automatic generation of test cases without given additional information by the service developer is a future goal. Finally, the coherent integration of the testing framework into the Venice Service Grid will expand the functionality of Venice and will support service developers in each step of a SOA-based software development process.

4 Acknowledgements

This work was funded by the German Ministry for Education and Research (BMBF project “iGreen” 01IA08005G). The authors are responsible for the content.

References


