Chapter 3
A Metamorphic Testing Methodology for Online SOA Application Testing

W. K. Chan
City University of Hong Kong, Hong Kong

S. C. Cheung
Hong Kong University of Science and Technology, Hong Kong

Karl R. P. H. Leung
Hong Kong Institute of Vocational Education, Hong Kong

ABSTRACT

Testing the correctness of service integration is a step toward assurance of the quality of applications. These applications however may bind dynamically to their supportive services using the SOA pattern that share the same service interface, yet the latter services may behave differently. In addition, a service may implement a business strategy, such as best pricing, relative to the behaviors of its competitors and the dynamic market conditions. As such, defining a test oracle that specifies the absolute expected outcomes for individual test cases is hard. Many existing works ignore these issues to address the problem of identifying failures from test results. This chapter studies an approach to online testing. Service testing is divided into two steps. In the spirit of metamorphic testing, the offline step determines a set of successful test cases to construct their corresponding follow-up test cases for the online step. These test cases will be executed by metamorphic services that encapsulate the services as well as applicable metamorphic relations. Thus, any failure revealed by the approach will be a failure of the service under test.

INTRODUCTION

Service-oriented architecture (SOA) is a kind of architectural reference model (Bass et al., 2003) to support distributed computing. A notable example is Web services (W3C, 2002). It promises to alleviate the problems related to the integration of applications of heterogeneous technologies (Mukhi et al., 2004; Kreger et al., 2003). In this reference model, a SOA application consists of a set of self-contained, communicating components, known as services, in which each service should make little
or no assumption about its collaborating services. This setting advocates the dynamic composition of services by using different configurations of supportive services.

Typical end-users of a decent SOA application, such as bank customers using an online foreign exchange trading service, may however expect consistent outcomes each time they use the service, probably except some upgrade of the services. For instance, the customers may compare the online foreign exchange service of a bank to similar services of other banks to judge whether the service is good in product quality. If a Business-to-Business (B2B) service provider is driven by a predefined business strategy, for example, to maintain its market share, then the criteria to define functional correctness of the service may vary according to the dynamic banking environment. As such, testers may not be provided with predefined expected test outcomes; and by the business nature, testers (technical professionals) may not know the decisions of business managers, and vice versa. Such a barrier in the application domain of the service under test makes the manual judgment on the test results by the testers ineffective and inefficient. This may lengthen the testing of services.

Services may be subject to both offline testing and online testing. Unlike the testing of conventional programs, services bind dynamically to other peer services when it is tested online. In some cases, the behaviors of these peer services are not known precisely in advance, and may evolve during testing. While the offline testing of services is analogous to the testing of conventional programs, the online testing of services needs to address new issues and difficulties.

In testing, testers should address both the test case selection problem and the test oracle problem (Beizer, 1990). As we have explained above, identifying failures from the test results is challenging in service testing. Therefore, we restrict our attention to the latter problem in this chapter.

A test oracle is a mechanism that reliably decides whether a test succeeds. In services computing, formal test oracle may be unavailable. Instead, the expected behavior of a service may evolve dynamically according to the environment. Such an expected behavior may be relative to the behaviors of competing services or other services. Tsai et al. (2004), for example, suggest using a progressive ranking of similar implementations of the same service interface to alleviate the test oracle problem. Their proposal is useful when all implementations should provide the same results on the same input. However, sometimes, the behaviors of different implementations of the same service (e.g., search engines) may vary. The test result of a particular group of implementations cannot be served reliably as the expected result of another group of implementations of the same service on the same test case. In addition, a typical SOA application may comprise collaborative services of multiple organizations. Knowing all the implementations of different organizations is hard (Ye et al., 2006). For example, a travel package consultant may bundle services of hotels, airlines and entertainment centers to personalize tour packages for a particular client. Without the implementation details, static analysis appears hard to be applied to assure the quality of the application. The black-box approach to checking the test results remains popular to assure the correctness of these applications.

Traditionally, the expected result of a program on a test case is defined in the absolute sense (Beizer, 1990). Having the program output for the test case, a tester compares the program output with the expected result. If they match, then no failure has been revealed. The test case is termed as a successful test case. On the other hand, if there is any mismatch, tester deems that the test case is failure-causing.

Metamorphic testing (MT) (Chan et al., 1998; Chen et al., 1998, 2002) is an approach to alleviating the test oracle problem in the relative