Chapter 11
State-Based Evolution Management of Risk-Based System Tests for Service-Centric Systems

Michael Felderer
University of Innsbruck, Austria

Berthold Agreiter
University of Innsbruck, Austria

Ruth Breu
University of Innsbruck, Austria

ABSTRACT

For various reasons, service-centric systems are subject to continuous evolution. Therefore, regular adaptations to their tests are essential to keep, or even improve, their quality of service. This chapter presents a model-based approach to manage tests for evolving service-centric systems. We do so by attaching state machines to all model elements of our system model and test model to manage the consistent evolution of the system and its tests. In our approach, a modification to an arbitrary model element is propagated to related model elements. As a consequence, also these model elements may change their state. The process integrates continuous risk assessment which enables the prioritization of tests for optimized test selection and subsequent test execution. Our system and test evolution management process is demonstrated in a case study from the home-networking domain.

DOI: 10.4018/978-1-61350-438-3.ch011

INTRODUCTION

A software system must evolve, or it becomes progressively less satisfactory (Lehman, 1980). Although evolution has been investigated for model-driven system development (R. Breu, 2010), such aspects have been neglected for model-driven *system testing* so far. Nevertheless, testing is very important during evolution (Moonen, Deursen, Zaidman, & Bruntink, 2008), and maintenance of test models is a key factor to make model-driven testing applicable at all. The evolution of a system provides additional information that supports the selection of test models and therefore the generation of an optimal test suite by analogy to classical regression testing (Rothermel & Harrold, 1998).

In many application domains, testing has to be done under severe pressure due to limited resources with the consequence that only a subset of all relevant test cases can be executed. In this context, *risk-based testing* (RBT) (Amland, 2000) is more and more applied to prioritize test cases based on assigned risks, i.e., the chance of damage determined by its probability and impact. The results of a test run are then used to reassess risks for future test prioritization. Risk-based testing can be integrated into a system and test evolution management process in a natural way.

In this chapter, we present a state-based evolution management methodology for dynamically evolving service-centric systems that integrates risk-based testing techniques for the prioritization of test cases. We put the main focus on how the evolution influences tests so to be able to build optimal test suites, and on how risk-based testing can be considered in the evolution process to have a further criterion for test selection. We enhance our previous evolution management approach (M. Felderer, B. Agreiter, & R. Breu, 2011) by risk-based testing to provide additional information for optimized test selection. Our approach is based on a state-model based computation of test models and allows for regression testing of service-centric systems.

Arising application scenarios have demonstrated the power of service-centric systems. This ranges from the exchange of health related data among various stakeholders in healthcare, over the cross-linking of traffic participants, to home-network control systems. Taking the latter as an example, consider only a few device types are initially integrated in the scenario and will successively be extended by new actor instances, improved underlying infrastructure or new resp. modified functionality. The importance for systematically managing the evolution of service-centric systems is also reflected by a special volume on continual service improvement within the ITIL standard for service management (OGC, 2007).

In our methodology, each changeable artifact, i.e., a model element, has a state machine attached that describes its current state and possible future states of the artifact. The state machine triggers resp. receives events to compute the new state of its corresponding artifact. Consequently, we work with a model where any change may influence multiple model elements. The states of the model elements describe their condition, e.g., whether a service is executable, or whether a requirement is associated with a test.

Following the widely used classification in (Leung & White, 1989), the *type of a test* can be *evolution* for testing novelties of the system, *regression* for testing non- modified parts ensuring that evolution did not impact parts supposed not to be modified, *stagnation* for ensuring that evolution did actually take place and changed the behavior of the system, and *obsolete* for tests which are not relevant anymore. The type of a test is computed by the state machines mentioned before. Based on this type, and the risk value of a test and the test requirement, a test suite is determined. This supports safe regression testing (Rothermel & Harrold, 1998), which identifies all test cases in