Chapter 12

An Agile and Tool-Supported Methodology for Model-Driven System Testing of Service-Centric Systems

Michael Felderer
University of Innsbruck, Austria

Philipp Zech
University of Innsbruck, Austria

Ruth Breu
University of Innsbruck, Austria

ABSTRACT

In this chapter, the authors present an agile and model-driven system testing methodology for service-centric systems called Telling TestStories. The methodology has a tool implementation and is based on separated system, requirements, and test models that can be validated in an integrated way. Test models contain test stories describing test behavior and test data in an integrated way. The underlying testing process is iterative, incremental, and supports a test-driven design on the model level. After a general overview of the artifacts and the testing process, the authors employ the methodology and the tool implementation on a case study from the healthcare domain.

INTRODUCTION

The number and complexity of service-centric systems for implementing flexible inter- and intra-organizational IT based business processes is steadily increasing. Arising application scenarios have demonstrated the power of service-centric systems that consist of peers providing and requiring services (Breu, 2010). These range from the cross-linking of traffic participants, over new business models like Software-as-a-Service, to the exchange of health related data among stakeholders in healthcare. Elaborated standards, technologies and frameworks for realizing service-centric systems have been developed, but system testing tools and methodologies have been neglected so far (Canfora & Di Penta, 2008).

DOI: 10.4018/978-1-4666-2503-7.ch012
The trend in industrial software development is going towards agile development processes that provide continuously growing executable systems. Agile processes (Fowler & Highsmith, 2001) are based on iterative and incremental development and testing, where requirements and solutions evolve through collaboration between cross-functional teams and support for (semi-) automatic system testing is highly important.

Agile system testing methodologies for service-centric systems, i.e., methods for evaluating the system’s compliance with its specified requirements, have to consider specific issues that limit their testability including the integration of various component and communication technologies, the dynamic adaptation and integration of services, the lack of service control, the lack of observability and structure of service code, the cost of testing, and the importance of Service Level Agreements (SLA).

A system testing methodology which considers model-driven testing and agile methods is particularly suitable for system testing of service-centric systems. Such a methodology supports the definition of requirements by modeling tests in a very early phase of the development process and on a high level of abstraction. The assignment of tests to requirements on the model-level makes requirements executable and enables the definition plus the modification of requirements and tests in a collaborative way by various stakeholders. Modeled tests can be adapted easily to changing requirements, they support the optimization of test suites without a running system, they provide an abstract technology and implementation independent view on tests, and they allow the modeling and testing of service level agreements. The latter especially allows for defining test models in a very early phase of system development even before or simultaneous with system modeling supporting test-driven development on the model level.

In this chapter we introduce an agile and model-driven system testing methodology called Telling TestStories (TTS) that is based on tightly integrated platform-independent requirements, system and test models. The approach is capable of iterative and incremental test-driven development on the model level, and guarantees high quality system and test models by checking consistency, completeness and coverage. Additionally, TTS provides full traceability between the requirements, the system and test models, and the executable services of the system. The collaboration between stakeholders for defining and adapting tests is supported by test models and the tabular representation of test data and test results as in the Framework for Integrated Testing (FIT). Our approach is therefore not only model-driven but also tabular. The tool environment of TTS supports these features in a stringent way and integrates domain experts, test experts, and system experts into a process of collaborative requirements, system and test design. The TTS framework adheres to a test-driven development approach, thus it allows the execution of test stories in early stages of system development and supports the evolution of the underlying system. Our methodology can also be employed for acceptance testing because we consider it as system testing performed by a customer.

There are several system testing methodologies and tools available. Unlike TTS, these frameworks do not combine tabular and model-driven system testing.

FIT/Fitnesse (Mugridge & Cunningham, 2005) is the most prominent framework which supports system test-driven development of applications allowing the tabular specification, observation and execution of test cases by system analysts. TTS is due to its tabular specification of test data based on the ideas of FIT/Fitnesse. But additionally, TTS is model-driven supporting validation, maintenance and traceability of test designs on the model level. Tests in TTS are also more expressive than FIT tests because TTS supports to model control flows in tests.

A model-driven approach to testing SOA with the UML 2.0 Testing Profile (OMG, 2005) is defined in (Baker et al., 2007). It focuses on web services technology and uses the whole set