Telling TestStories – Modellbasiertes Akzeptanz–Testen Serviceorientierter Systeme

Michael Felderer, Joanna Chimiak–Opoka, Ruth Breu
Universität Innsbruck
{michael.felderer,joanna.opoka,ruth.breu}@uibk.ac.at

Zusammenfassung

In diesem Artikel geben wir einen Überblick über *Telling TestStories*, ein modellbasiertes Testverfahren und Testframework, welches speziell für Akzeptanztests von Serviceorientierten Systemen geeignet ist und derzeit im Rahmen einer Forschungskooperation zwischen der Universität Innsbruck und der Firma softmethod[1] entwickelt wird.

1 Motivation

Moderne Serviceorientierte Systeme werden immer komplexer, was hohe Anforderung an ihre Qualitätssicherung stellt. Die Qualitätssicherung Serviceorientierter Systeme weist allerdings eine Reihe von Spezifika wie die Komplexität des Aufbaus der Testinfrastruktur auf, die durch die heute verfügbaren Testmethoden und Testframeworks nicht abgedeckt sind.

Telling TestStories, kurz *TTS*, versucht diese Probleme durch die Bereitstellung einer Testmethode und eines Testframeworks für die modellbasierte Definition, Durchführung und Auswertung von Akzeptanztests auf Basis von Anforderungsspezifikationen zu lösen.

Im Bereich des modellbasierten Testens von Anforderungsspezifikationen gibt es derzeit erst wenige Ansätze, die sich vor allem auf die automatische Generierung von Testfällen aus Modellen der Anforderungsspezifikation fokussieren wie [BL02, NF06] oder das EU-Projekt Agedis[2].

Unser Ansatz unterscheidet sich von diesen existierenden Ansätzen vor allem darin, dass die Testfälle nicht aus der Anforderungsspezifikiation abgeleitet werden, sondern auf Basis der Elemente der Anforderungsspezifikation von Testern oder Kunden modelliert und ausgeführt werden. Dies hat den Vorteil, dass keine vollständige Anforderungsspezifikation - wie sie in der Praxis selten vorliegt - benötigt wird, um zuverlässige Testpläne entwickeln zu können.

In den folgenden Abschnitten geben wir einen Überblick über die TTS–Methode bzw. das TTS–Framework, beschreiben unsere Implementierung und geben abschließend einen Ausblick.

[1] http://www.softmethod.com
[2] http://www.agedis.de

2 TTS Methode und Framework

In diesem Abschnitt erklären wir zunächst unsere Methode zur Definition, Durchführung und Auswertung von Tests und anschließend das Framework zu deren Umsetzung.

TTS Methode In TTS dienen die Modellelemente einer Anforderungsspezifikation der Erstellung sogenannter *Test Stories*, welche sequentielle oder verteilte Abläufe modellieren, in welchen Akteure untereinander und mit dem zu testenden System interagieren. Test Stories stellen also Testabläufe auf fachlicher Ebene dar und werden durch Aktivitätsdiagramme, Sequenzdiagramme oder textuell ähnlich wie in TTCN–3[3] beschrieben. In Abbildung 1 ist eine Test Story für einen einfachen Login–Vorgang dargestellt.

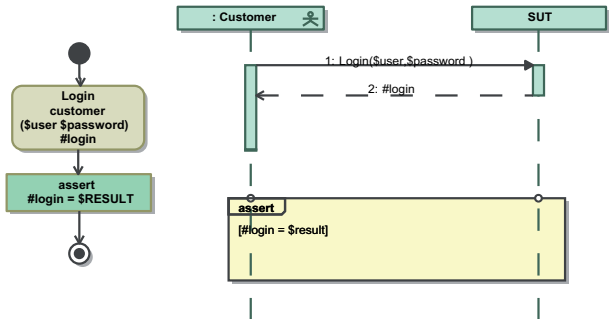

Abbildung 1: Test Story für einen Login–Vorgang als Aktivitäts- und Sequenzdiagramm

Test Stories basieren lediglich auf folgenden Modellelementen der Anforderungsspezifikation: *Akteure*, *Services* und *Klassen*. Die Klassen werden in einem fachlichen Klassendiagramm und die Akteure und Services in einem Use-Case-Diagramm der Anforderungsspezifikation bereitgestellt. Diese geringen formalen Anforderungen an die Spezifikation, machen unseren Ansatz für Kunden überschaubar und damit anwendbar. Im Beispiel aus Abbildung 1 ruft der Akteur Customer das Service Login mit den Parametern username und password auf. Die Parameter werden dabei mit einem '$' eingeleitet, was symbolisiert, dass die Werte aus einer Datentabelle – ähnlich wie im Fitnesse

[3] http://www.ttcn-3.org

Framwork[4] – stammen.

Im nächsten Schritt werden die Test Stories von einem Entwickler oder Testingenieur um ausführbare Artefakte, die wir *Fixtures* nennen und bei denen es sich etwa um die Zuordnung ausführbarer Komponenten zu den Services oder um Code für den Aufbau der Testinfrastruktur handelt, angereichert. Anschließend werden Test Stories, Datentabellen und Fixtures in Code der Zielsprache übersetzt, der dann in einer verteilten Testumgebung ausgeführt wird. Abschließend wird aufgrund der im assert–Element (siehe Abbildung 1) definierten Testbedingung für jeden Testfall ein Testverdikt berechnet und in den Datentabellen der Test Stories oder in der Anforderungsspezifikation annotiert, um den Test dann analysieren zu können.

Framework Um die oben beschriebene Methode umzusetzen, haben wir ein Framework entwickelt, das in Abbildung 2 dargestellt ist.

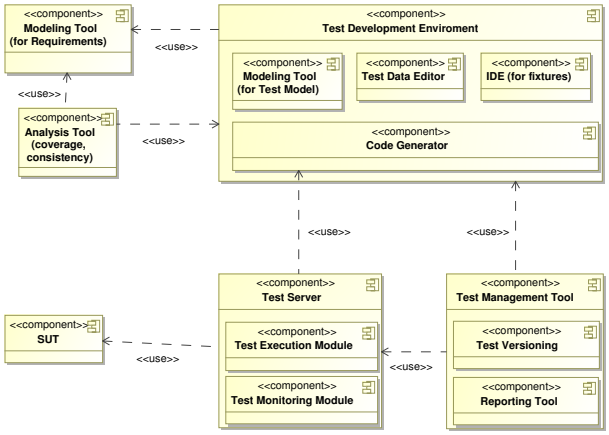

Abbildung 2: Telling TestStories Framework

Das *Modellierungswerkzeug* stellt die Anforderungsspezifikation im XMI-Format für die *Testentwicklungsumgebung* bereit. Diese besteht aus einem Modellierungswerkzeug zur Erstellung der Test Stories, sowie je einem Editor zur Verwaltung der Datentabellen und zum Erstellen der Fixtures. Über einen Codegenerator wird Testcode erzeugt, der auf einem *Test Server*, bestehend aus einer Ausführungskomponente und einer Monitoringkomponente, gegen das zu testende System ausgeführt wird. Die Testergebnisse werden in einem *Testverwaltungswerkzeug* versioniert und aufbereitet, um sie dann in der Testentwicklungsumgebung zu integrieren.

Im nächsten Abschnitt beschreiben wir kurz unsere prototypische Umsetzung dieses Frameworks in Java.

3 Implementierung

Wir haben das Telling TestStories Framework in Java implementiert. Die Testentwicklungsumgebung haben wir in Eclipse als Menge von Plug-ins umgesetzt. Ein Akzeptanztest wird als Projekt in Eclipse verwaltet, die Anforderungen werden im EMF XMI-Format importiert und können für die Erstellung von Test Stories verwendet werden, wofür wir einen eigenen Editor mit xtext aus dem openArchitectureWare Framework[5] implementiert haben. Die Datentabellen werden in HTML erstellt und Fixtures stellen Codefragmente in Java dar, welche in Eclipse automatisch benutzerfreundlich verarbeitet werden können. Die Generierung von Testfällen in der Zielsprache Java erfolgt ebenfalls mit openArchitectureWare. Der Testcode wird anschließend in einer einfachen Testumgebung, die mit Reflections arbeitet, ausgeführt. Das Testmanagement haben wir noch nicht umgesetzt. Die Überprüfung von Konsistenz- und Überdeckungskriterien des Testmodells gegen die Anforderungsspezifikation werden wir mit unserem SQUAM Framework [CO+06] durchführen.

4 Schlussfolgerung und Ausblick

Telling TestStories stellt einen neuen modellbasierten Ansatz für Akzeptanztests auf Basis der Anforderungsspezifikation dar. Testszenarien können damit bereits parallel zur Erstellung der Anforderungen definiert, analysiert und ausgeführt werden. Test Stories werden von Fachexperten verstanden oder können sogar von diesen erstellt werden, was entscheidend zur Erhöhung der Qualität der Anforderungsspezifikation beitragen kann.

Wir werden den Telling TestStories Ansatz sowohl anwendungsbezogen als auch wissenschaftlich erweitern. Anwendungsbezogen werden wir die aktuelle Implementierung vervollständigen und verbessern, um sie dann auf umfangreichere Serviceorientierte Systeme aus der Praxis anzuwenden. Wissenschaftlich werden wir Fragen der Konsistenz und Vollständigkeit sowie Kriterien für die automatische Generierung der Testfälle untersuchen.

Literatur

[BL02] Lionel C. Briand and Yvan Labiche. A UML-based approach to system testing. *Software and System Modeling*, 1(1), 2002.

[CO+06] Joanna Chimiak-Opoka et al. Tool–Supported Systematic Model Assessment. volume 82 of *Lecture Notes in Informatics (LNI)—Proceedings*. Gesellschaft fuer Informatik, 2006.

[NF06] Clementine Nebut and Franck Fleurey. Automatic Test Generation: A Use Case Driven Approach. *IEEE Trans. Softw. Eng.*, 32(3), 2006.

[4] http://www.fitnesse.org

[5] http://www.openarchitectureware.org