

Modellgetriebene Systemtests mit Telling TestStories

Michael Felderer, Frank Fiedler, Philipp Zech, Ruth Breu

Die stetig steigende Anzahl und Komplexität Serviceorientierter Systeme erfordert die Bereitstellung geeigneter Methoden und Werkzeuge für deren Systemtest. Erfolgversprechende Testansätze hierzu müssen die besonderen Eigenschaften solcher verteilten Systeme berücksichtigen wie die Integration unterschiedlicher Komponenten- und Kommunikationstechnologien, den fehlender Zugriff auf die Implementierung und Steuerung einzelner Services sowie die dynamische Änderung und Integration von Services.

Modellgetriebenes Testen, d.h. die Ableitung von ausführbarem Testcode aus Testmodellen analog zu MDA Ansätzen, ist deshalb für den Systemtest besonders geeignet, weil es eine technologie- und implementierungsunabhängige Sichtweise auf Tests ermöglicht und die Tests mit verhältnismäßig geringem Aufwand an geänderte Erfordernisse angepasst werden können.

Im Forschungskooperationsprojekt Telling TestStories (TTS) wurde eine neue Methode und ein prototypisches Tool für den modellgetriebenen Systemtest Serviceorientierter Systeme entwickelt. Im Gegensatz zu anderen modellbasierten Testansätzen stützt sich TTS auf voneinander unabhängige, aber über Modellelemente verbundene System- und Testmodelle. Dies ermöglicht es, im Sinne eines testgetriebenen Ansatzes, Testmodelle vor oder in Verbindung mit Systemmodellen zu erstellen.

Das Systemmodell beschreibt plattformunabhängig die Struktur und das Verhalten des Serviceorientierten Systems. Dazu werden die beteiligten Akteure mit den von ihnen angebotenen und benötigten Services modelliert. Jedes Service besteht aus Operationen, deren Semantik über Vor- und Nachbedingungen definiert werden kann. Das Verhalten des Gesamtsystems wird in globalen Workflows und jenes einzelner Komponenten in lokalen Workflows definiert.

Das Testmodell beschreibt die Testkonfiguration und das Testverhalten in Form sogenannter Teststories. Diese stellen durch Kontrollstrukturen gesteuerte Abfolgen von Operationsaufrufen dar, welche zusätzlich Zusicherungen enthalten, um das gewünschte Testverhalten festzulegen. Teststories können generisch definiert werden, indem die konkreten Testdaten – ähnlich wie im FIT Framework – in Tabellen abgelegt werden.

Teststories werden in ausführbaren Code der Testumgebung transformiert. Durch Adapter werden die einzelnen Operationsaufrufe in Systemaufrufe beliebiger Zieltechnologien wie Web Services oder RMI übersetzt.

Für das System- und das Testmodell gibt es jeweils Metamodelle, welche die Definition von Konsistenz- und Überdeckungsbedingungen ermöglichen und welche man für die Transformation der Teststories in ausführbaren Testcode benötigt. Die Metamodellelemente sind auf UML Metaklassen abbildbar und können somit mit gängigen UML Werkzeugen erstellt und bearbeitet werden. Weiters kann das Systemmodell auf SoaML und das Testmodell auf das UML Testing Profile abgebildet werden, was die Kompatibilität zu gängigen Standards sicherstellt.

Falls das Systemmodell bzw. das Testmodell vollständig ist, können auch Verhaltensartefakte des einen aus dem anderen generiert werden.

Eine wichtige Eigenschaft des Ansatzes ist die durchgängige Nachverfolgbarkeit zwischen den Systemanforderungen, den einzelnen Service-Operationen, den Teststories und der Implementierung. Dadurch wird es möglich, unerwartetes Systemverhalten den Anforderungen bzw. den System- und Testmodellelementen zuzuordnen. Dies ermöglicht die iterative Erstellung, Transformation, Ausführung und Analyse von System- und Testmodell vor dem Hintergrund sich ändernder Anforderungen und Services.

TTS wird derzeit in einer Fallstudie aus dem Telekommunikationsbereich evaluiert. Weitere Testprojekte für das funktionale Testen einer klassischen SOA Anwendung und das Security-Testen einer Zugriffssteuerung sind in Vorbereitung.

Nähere Informationen zu TTS können unter <http://teststories.info> abgerufen werden.