Testautomatisierung für eine Standardsoftware: Praxisbericht eines Schweizer Versicherungsunternehmens

Ohne Zweifel hat sich die Testautomatisierung in Großprojekten, insbesondere wenn bei vielen Testzyklen in kurzer Zeit getestet werden soll, in der IT-Branche etabliert. Bei vielen Unternehmen erfolgt der Einsatz einer Standardsoftware – dabei kommt das Thema Testen aber häufig zu kurz. Es stellt sich auch die Frage, welche Testmethode bei einer Standardsoftware geeignet ist. Die Suva Schweiz hat sich für eine Keyword-Driven Testmethode als Testansatz zur Testautomatisierung einer Standardsoftware entschieden. Dieser Praxisbericht zeigt am Beispiel des schweizerischen Versicherungsunternehmens Suva, wie Testautomatisierung für eine Standardsoftware möglichst wartungsarm mit größtmöglicher Wiederverwendung erfolgreich in den Entwicklungsprozess eingebunden werden kann.

Die Suva ist ein selbstständiges Unternehmen des öffentlichen Rechts und versichert rund 121.000 Unternehmen bzw. 1,95 Mio. Berufstätige gegen die Folgen von Unfällen und Berufskrankheiten. Arbeitslose sind automatisch bei der Suva versichert. Zudem führt sie im Auftrag des Bundes seit 2005 auch die Militärversicherung. Die Dienstleistungen der Suva umfassen Prävention, Versicherung und Rehabilitation (vgl. [Suv]). Die Suva stattet einen wesentlichen Teil ihrer Geschäftsprozesse mit einer neuen Software aus. Hierzu wird eine parametrierbare Standardsoftware verwendet und im Rahmen eines Projekts implementiert und an die Bedürfnisse der Suva angepasst.

Architektur

Abbildung 1 zeigt grob die Testautomatisierungs-Architektur der Suva. Grundsätzlich wird eine strikte Trennung zwischen den Spezifikationen der Testabläufe mit den zugehörigen Testfällen und der Ausführung der Testabläufe mit den zugehörigen Testfällen verfolgt. Organisatorisch gibt es ein Scrum-Team (vgl. [Scr]), bestehend aus mehreren Personen mit unterschiedlichen Rollen, die die Testfälle spezifizieren und implementieren (dazu später mehr).

Wesentliche Erfolgsfaktoren für eine gute Testautomatisierungs-Architektur sind eine möglichst einfache Wartung, eine größtmögliche Wiederverwendung und eine adäquate Performance bei der Ausführung der automatisierten Testfälle. Diese Erfolgsfaktoren gelten nicht nur für die Ausführungsebene mit den implementierten automatisierten Testfällen, sondern auch auf Testspezifikationsebene. Zudem sollte eine Testautomatisierungs-Architektur in der Lage sein, Tools für die Testspezifikation sowie Tools zur Automatisierung der Testfälle beliebig und mit möglichst wenig Aufwand auszutauschen.

Auf der Spezifikationsebene werden als Test-Case-Repository zur Erfassung, Durchführung und Management der Testabläufe mit den zugehörigen Testfällen derzeit primär die Tools „JIRA“ (vgl. [Atl]) und „HP QC Quality Center“ (vgl. [Hew]) verwendet.

Auf der Ausführungsebene nimmt die Software „STEP“ (Scalable Test Execution Environment) (vgl. [step15]) einen Testfall-Request in Form einer XML-Struktur von der Spezifikationsebene entgegen. STEP ist die strategische Lösung für automatisierte Softwaretests und eine Eigenentwicklung von Suva. Bevor der Request an einen Testautomaten für eine entsprechende Ausführungsumgebung (Produktion, Integra-

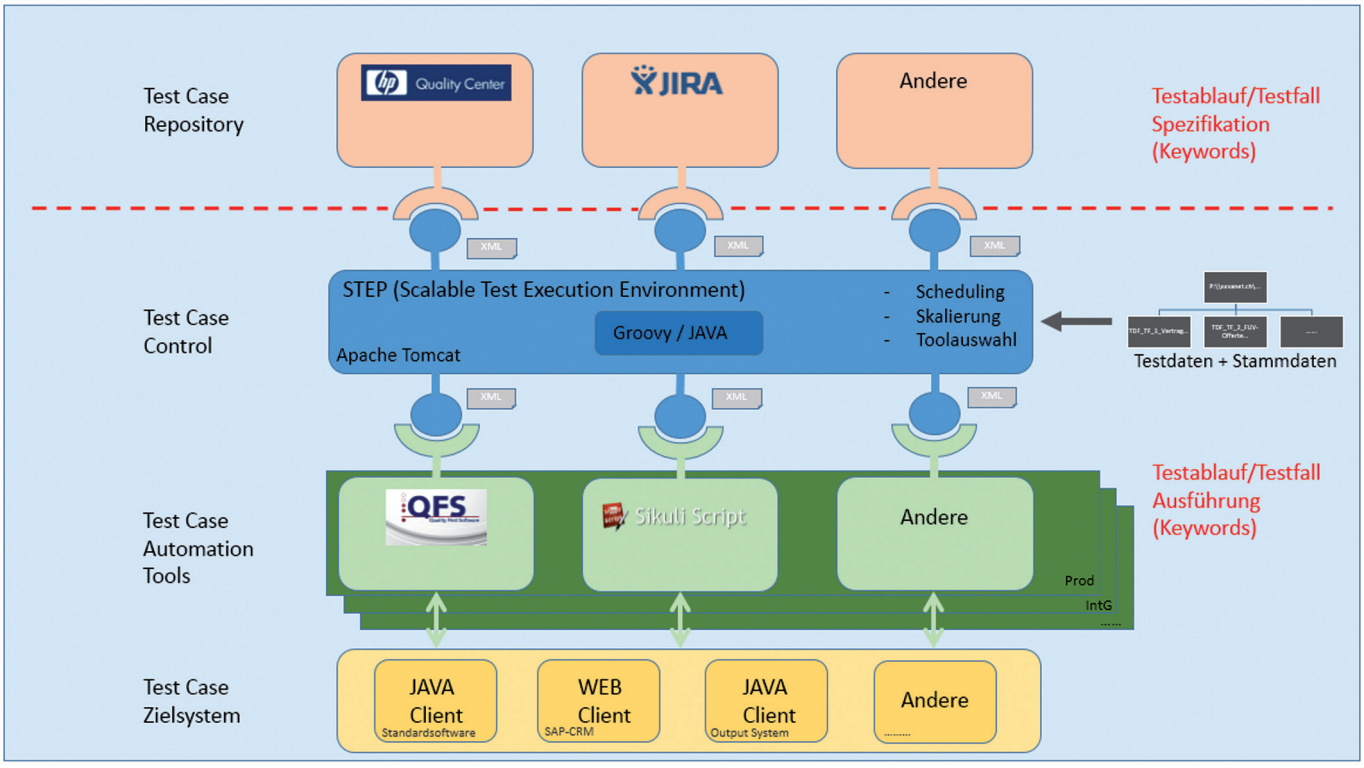

Abb. 1: Die Testautomatisierungs-Architektur.

```
<SAP_JurPartnerErfassen
  Name1="Name"
  GueltigAb="{Today('dd.mm.yyyy')}"
    ...
  Zuständigkeit="WEG-Bereich A03"
  AutoUnfallnummer="00"
/>
<Expected
  check1="Meldung == 'erfolgreich'"
  zuweisung1="{MyPartnerNr}:PartnerNr"
  onFailure="exit"
/>
```

Abb. 2: Keyword-XML-Struktur.

tion usw.) zur Ausführung weitergeleitet wird, überprüft die STEP-Software, ob die XML-Struktur einen syntaktisch korrekten Testfall enthält. Der Automat – die Suva verwendet derzeit hier im Wesentlichen das Test-Case-Automatisierungs-Tool „QF-Test“ (vgl. [QFS]) – erhält von STEP diese XML-Struktur und führt den Testfall gegen das entsprechende Zielsystem aus. Nach der Ausführung wird über STEP wiederum eine XML-Struktur als Antwort an die Spezifikationsebene hochgereicht, die Rückgabewerte bzw. Fehlermeldungen enthält. Über Adaptoren können das Test-Case-Repository und/oder das Test-Case-Automatisierungs-Tool beliebig ausgetauscht werden.

Hauptsächlich werden über die Architektur funktionale Regressionstests zur Entlastung der Fachtester durchgeführt – diese decken circa 50 Prozent aller Tests ab und geben bei einem neuen Release-Zyklus der Standardsoftware rasch Auskunft über die Qualität des neuen Releases. Der Schwerpunkt in diesem Praxisbericht liegt auf der Ausführung der Regressionstests.

Horizontal dazu werden aber auch automatisierte Regelwerktests, täglich Smoke-Tests, Rollen/Berechtigungen-Tests, Output (Dokumenten)-Tests, Batch-Tests und Monitoring-Tests zur Überprüfung der Funktionstüchtigkeit der Schnittstellen zwischen den Umsystemen (SAP CRM usw.) durchgeführt. Auch werden über die Architektur Last- und Performancetests gefahren.

Zudem werden synthetische Testdaten sowie produktive Stammdaten generiert. Hierzu können über die STEP-Software Excel-Sheets eingelesen werden. Jede Zeile des Excel-Sheets enthält die Daten eines Testfall-Requests, die Zeilen werden dann von STEP iterativ abgearbeitet. Testfälle können einerseits sequenziell ausgeführt werden oder aber auch derzeit bis zu 120 Testfälle parallel.

Wahl der Testmethode

Die verwendete Standardsoftware besteht im Wesentlichen aus einer Vielzahl in sich abgeschlossener Tasks, die wiederum teilweise voneinander abhängig sind. Einen Task kann man auch als Use-Case (Anwendungsfall) interpretieren, d. h. als einen wiederkehrenden, abgeschlossenen Ablauf, beispielsweise „Partner erzeugen“ oder „Fallmeldung erzeugen“.

Hier bietet sich die *Keyword-Driven Testmethode* (vgl. [Wik]) an, bei der wiederkehrende Aktionen und Abläufe – im Fall der Standardsoftware auf einem granularen Level in Form von abgeschlossenen Tasks – in so genannte Keywords gekapselt werden. Ein Testablauf besteht dann in der Regel aus einem oder einer Aneinanderreihung einzelner Keywords. Ein Testfall ist dann eine konkrete Instanz eines Testablaufs, bestehend aus *1* bis *n* Keywords mit den zugehörigen Daten innerhalb der einzelnen Keywords.

Ein Keyword (**siehe Abbildung 2**) wird in einer XML-konformen Sprache in Prosa formuliert, hat einen sprechenden Namen, verfügt über *1* bis *n* Ein- und Ausgabeparameter sowie konkrete Testdaten für die Parameter. Um die Wartbarkeit zu erhöhen, werden Keywords auf der Spezifikationsebene als Vorlagen im JIRA bzw. im HP QC Quality Center abgespeichert, die Keywords können dann von beliebigen Testfällen einfach angezogen werden.

Fachexperten erstellen die einzelnen Keywords in Form von XML-Strukturen und implementieren dann die Testfälle auf der Spezifikationsebene, die aus einer Aneinanderreihung von *1* bis *n* Keywords (so genannten *Steps*) bestehen. Die Fachexperten gewährleisten somit, dass fachlich das Richtige automatisiert wird. Da die STEP-Software zur Evaluation von Ausdrücken „Groovy“ (vgl. [Gro]) einsetzt, kann der Fachexperte in den Keywords auch Variablen, Funktionen und Operatoren in Groovy-Syntax beim Aufruf (**oberer Abschnitt in Abbildung 2**) und bei der Verifikation des Ergebnisses (**unterer Abschnitt in Abbildung 2**) verwenden. Dies führt zu einer besseren Wartbarkeit und Flexibilität der Testfälle auf Spezifikationsebene. Implementierungsexperten programmieren auf der Ausführungsebene dann die einzelnen Keywords, das heißt im Testautomaten werden dann nur noch die Keywords implementiert.

Erstellung der Testfälle

Die Hauptgründe für die Testautomatisierung bei der Suva sind die Entlastung der manuellen Fachtester sowie die Sicherstellung der funktionalen Regressionstests über den gesamten Lebenszyklus der Anwendung über Jahre hinweg, was auch das Investment in eine Testautomatisierungs-Architektur mit einer entsprechenden Testorganisation rechtfertigt, da der ROI eher bei mehreren Jahren liegt.

Der Testentwicklungsprozess ist komplett getrennt vom Entwicklungsprozess der Standardsoftware. Wie bereits erwähnt, implementieren die Fachexperten die Testfälle auf Spezifikationsebene im JIRA bzw. im HP QC Quality Center, wo die Testfälle verwaltet werden. Je nach Fachbereich sind verschiedene Fachexperten für die Testfälle zuständig. Die Implementierungsexperten wiederum verwenden für die Implementierung der Keywords im Testautomaten ein eigenes Versionsmanagement-Werkzeug.

Als Vorgehensrahmen für den Testentwicklungsprozess wird Scrum eingesetzt. Das komplette Testteam trifft sich täglich zu einem „Daily Scrum Meeting“, um sich einen

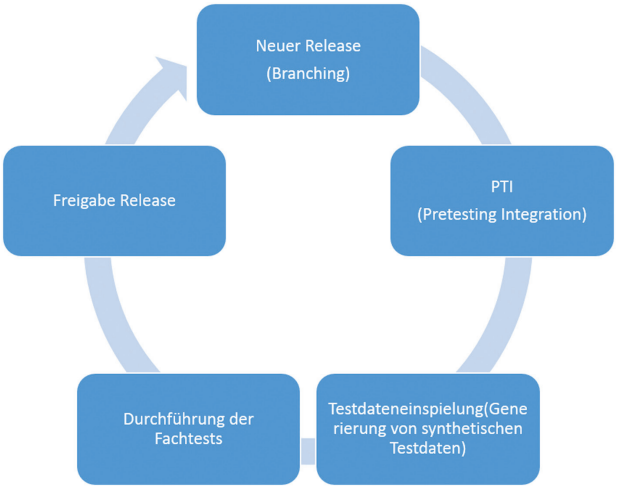

Abb. 3: Ablauf eines Release-Zyklus.

Überblick über den aktuellen Stand der Arbeit zu verschaffen. Im Daily Scrum Meeting werden von den Fachexperten auch automatisierte Testfälle angesprochen, die nicht mehr laufen bzw. einen Fehler ins HP QC Quality Center zurückgeben. Die Implementierungsexperten analysieren die Fehler mit Hilfe der Fachexperten, passen dann die zugehörigen Keywords im Testautomat an und informieren im Anschluss die Fachexperten, die dann die fehlerhaften automatisierten Tests neu starten.

Ein Sprint dauert zwei Wochen. Am Ende eines Sprint finden ein Sprint-Planning und eine Sprint-Retrospektive statt. Im Sprint-Planning wird festgelegt, welche Anforderungen für das automatisierte Testen aus dem Product Backlog für den nächsten Sprint umgesetzt werden können. Das Testteam legt dann zusammen fest, welche automatisierten Tests durchgeführt werden müssen, welche Testdaten generiert werden müssen, welche Testfälle auf Spezifikationsebene und welche Keywords im Testautomat implementiert bzw. angepasst werden müssen. Dann werden einzelne Tasks definiert, die in einem Taskboard festgehalten werden. Zum Schluss findet noch eine Sprint-Retrospektive statt, wo im Team offen diskutiert wird, was generell bei der Testautomatisierung gut gelaufen ist und was nicht. Falls es im letzten Sprint Probleme gab, werden konkrete Verbesserungsschritte definiert, die dann für den nächsten Sprint schriftlich festgehalten werden.

Es gibt zwei- bis vierwöchige Testzyklen. Automatisiert werden die Geschäftsprozesse der Standardsoftware inklusive der wichtigsten Umsysteme, wie SAP CRM (Partnerverwaltung) und das Output-System (Drucken/Speichern von Dokumenten). Ein Zyklus läuft grundsätzlich nach folgendem Schema ab (**siehe auch Abbildung 3**):

- Ein neuer Release der Standardsoftware wird deployed und es findet ein *Branching* der Testfall-Entwicklungsstränge statt. Das Branching ist notwendig, da unterschiedliche Releases der Anwendung auf unterschiedlichen Umgebungen getestet und die Keywords entsprechend angepasst werden müssen.
- Anschließend findet eine so genannte *Pretest Integration (PTI)* statt, wo alle automatisierten funktionalen Regressionstests durchgeführt werden.
- Nach einem erfolgreichen PTI werden die synthetischen Testdaten erzeugt, indem die vordefinierten Excel-Sheets von der STEP-Software abgearbeitet werden.
- Anschließend finden die Fachtests statt und schließlich wird das Release der Standardsoftware freigegeben.

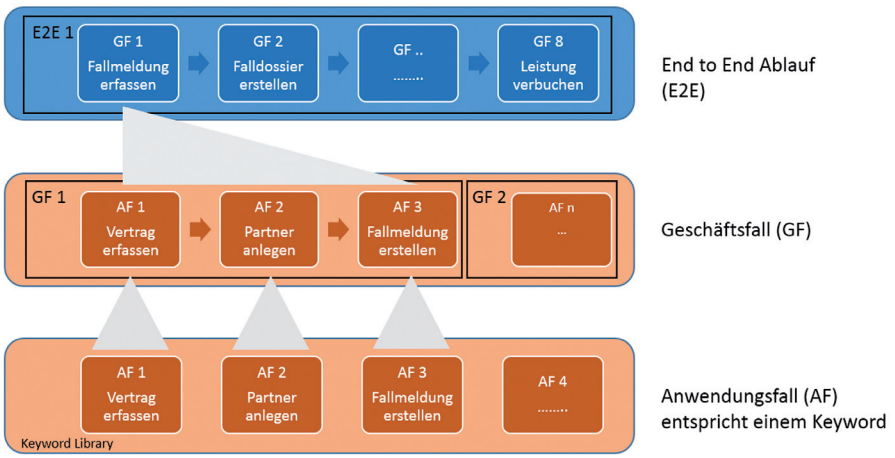

Abb. 4: Zusammenhang zwischen E2E-Abläufen, Geschäftsfällen und Anwendungsfällen.

Bei den funktionalen Regressionstests werden im Wesentlichen drei Testfallebenen unterschieden (**siehe Abbildung 4**):

- *E2E-Testfalle: End-to-End-Abläufe (E2E)* beginnen und enden beim Kunden, setzen sich aus mehreren *Ge-*

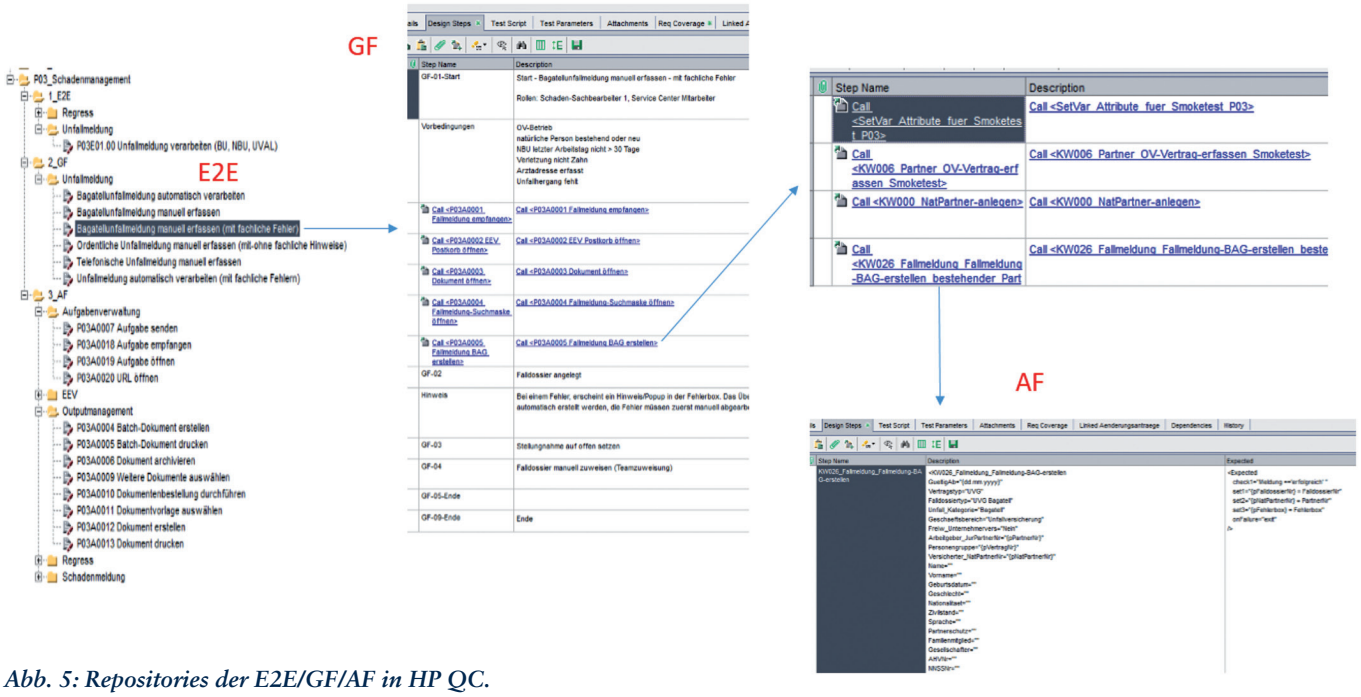

Abb. 5: Repositories der E2E/GF/AF in HP QC.

schäftsfällen (GF) zusammen, können system- und fachübergreifend sein und beziehen die Umsysteme mit ein.
- *GF-Testfälle:* GF setzen sich aus mehreren *Anwendungsfällen (AF)* zusammen, sind nicht fachübergreifend und können systemübergreifend sein.
- *AF-Testfälle:* AF in der untersten Ebene als wiederverwendbare, in sich abgeschlossene Testfälle sind in Form von Keywords implementiert, sind stark systembezogen und entsprechen in der Regel einem abgeschlossenen Task im System.

Im HP QC auf Testfallspezifikationsebene werden somit die einzelnen Testfallebenen durch eine Aneinanderreihung von Anwendungsfällen in Form von implementierten Keywords zusammengebaut. **Abbildung 5** zeigt beispielhaft die Repositories (Ordnerstrukturen) der Testfälle für einen bestimmten Fachbereich im HP QC Quality Center von einem E2E über einen GF bis zum AF, also dem implementierten Keyword.

Implementierung der Testfälle mit QF-Test

Für Java- und Web-Applikationen hat sich die Suva für das Tool „QFTest“ (vgl. [QFS]) der Softwarefirma QFS entschieden. QFTest hat sich als stabiler Testautomat zur Automatisierung von Java- und Web-Applikationen erwiesen. Testfälle können durch eine Vielzahl vorgegebener Knoten zusammengebaut werden oder – falls diese nicht genügen – durch Skripte mit den Programmiersprachen Jython (vgl. [Jyt]) und/oder Groovy (vgl. [Gro]) erweitert werden, wodurch selbst die Implementierung sehr komplexer Testszenarien möglich ist.

Wie bereits erwähnt, werden im Testautomaten nur noch die einzelnen Keywords implementiert. Somit kann man den Testautomaten von fachlichen Informationen weitestgehend freihalten – der Testautomat konzentriert sich dann nur noch auf seine Hauptaufgabe: die eigentliche Testausführung.

Auch bei der Implementierung der Keywords mit QFTest sind die wesentlichen Erfolgsfaktoren die Wiederverwendbarkeit und die Wartungsfreundlichkeit der implementierten automatisierten Tests. Die im Folgenden beschriebenen Techniken sind unabdinglich für einen stabilen Testablauf.

Generische Komponenten

Das Wichtigste bei der Implementierung von automatisierten Testfällen ist die stabile und zuverlässige Erkennung der grafischen Komponenten. Werden diese Komponenten für jeden Testfall neu aufgezeichnet, so ist die Chance groß, dass bei einem neuen Entwicklungszyklus der Testfall nicht mehr läuft, da für die Komponenten beim Aufzeichnen flüchtige IDs generiert werden. Wenn der Entwickler dann an der Komponente beispielsweise Namensänderungen vornimmt oder sich die Komponentenhierarchie stark ändert, wird diese Komponente nicht mehr erkannt. Die Lösung hierzu sind *generische Komponenten*. Dabei wird von einer aufgezeichneten Komponente auf eine in der Klassenhierarchie möglichst höher stehende Komponente verwiesen. Die aufgezeichnete Komponente wird dann zu einer generischen Komponente, die dann

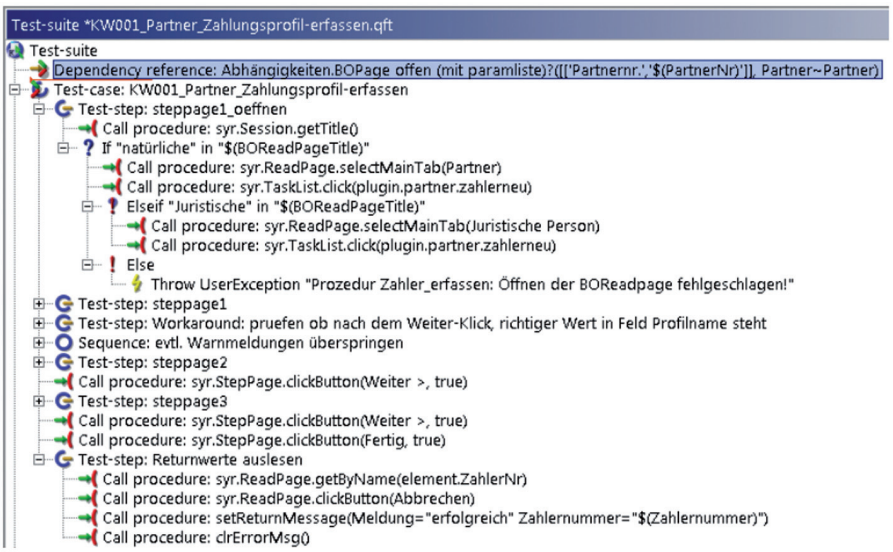

Abb. 6: Testfall-Implementierung im QF-Test.

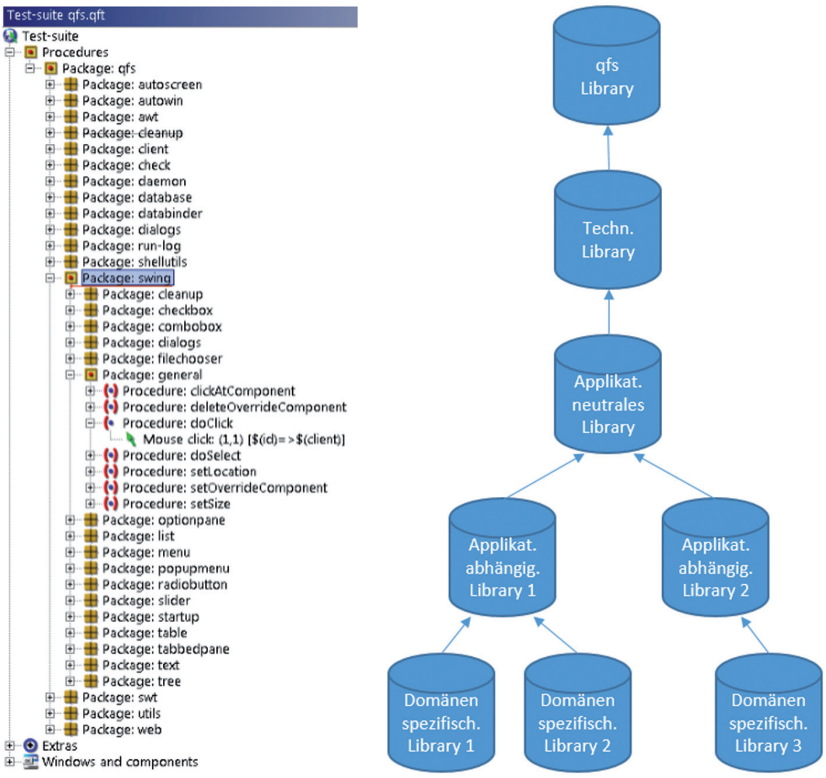

Abb. 7: Library-Hierarchie im QF-Test.

über Merkmale, die das QFTest Tool für eine Familie von grafischen Elementen (beispielsweise Labels, Buttons oder Eingabefelder) bereitstellt, und/oder *reguläre Ausdrücke* (vgl. [Goy13]) flexibilisiert wird. Das Ganze wird dann in eine Prozedur gepackt und in Form von Prozedurübergabe-Parametern parametrisiert. Der Testautomatisierer zeichnet dann für einen Testfall keine Komponenten mehr auf, sondern ruft nur noch vorgefertigte, wiederverwendbare Prozeduren auf (siehe Abbildung 6), die er je nach Bedarf beim Aufruf parametrisiert.

Namensauflöser (Resolver)

QFTest verwendet stabile Algorithmen zur Erkennung der grafischen Komponenten. Es kommt aber vor, dass beispielsweise ein Entwickler den Namen bzw. die ID einer grafischen Komponente ändert oder auch gar kein Name vorhanden ist. In solchen Fällen können vom QFTest-Tool bereitgestellte *Naming Resolver Interfaces* verwendet werden. Mit diesen können die Namen der Komponenten geändert oder gelöscht oder auch ein eigener Algorithmus zur Komponentenerkennung implementiert werden. Die Naming Resolver Interfaces enthalten Methoden, die mit einem eigenen Algorithmus implementiert werden können. Der Testautomatisierer programmiert dann eigene Resolver-Klassen, die diese Naming-Resolver-Interface-Methoden implementieren. Diese implementierten Methoden werden dann ausgeführt, wenn QFTest eine Komponente ermittelt hat.

Abhängigkeiten (Dependencies)

Ein weiteres gutes Feature zur einfacheren Wartung und Minimierung der Testfallimplementierung sind so genannte *Dependencies* (siehe Abbildung 6), in denen man Vorbedingungen definieren kann, die vorab vor dem Start eines Testfalles ausgeführt werden, beispielsweise die Navigation zu einem bestimmten Fenster für domänengleiche Testfälle.

Auslagern in Prozeduren

Zwecks Wiederverwendbarkeit werden sämtliche Zugriffe auf grafische Komponenten – beispielsweise Lesen von oder Schreiben in eine Komponente oder einzelne Testsequenzen in Prozeduren – ausgelagert, die dann von jedem implementierten Testfall vom Testautomatisierer parametrisiert aufgerufen werden können.

Links

[Atl] Atlassian, JIRA, siehe: `www.atlassian.com/software/jira`

[Goy13] J. Goyvaerts, Regular Expressions, 2013, siehe: `www.regular-expressions.info`

[Gro] Groovy Programming Language, siehe: `http://groovy-lang.org/`, `http://groovy.codehaus.org/`

[Hew] Hewlett-Packard Development Company, L.P., Quality Center, Enterprise, siehe: `www8.hp.com/us/en/software-solutions/quality-center-quality-management/index.html`

[Jyt] Jython, The Jython Project, siehe: `www.jython.org`

[QFS] QFS GmbH, Quality First Software, siehe: `www.qfs.de`

[Scr] Scrum.org Homepage, siehe: `www.scrum.org/resources/what-is-scrum`

[step15] STEP Scalable Test Execution Environment, von den Suva-Mitarbeitern Boris Basic und Jérôme Comte entwickelt, siehe: `http://suraww.suva.ch/Public-STEP-ObjektSpektrum/2015-Public-STEP-Konzept-ObjektSpektrum-Artikel.pdf`

[Suv] Suva Homepage, siehe: `www.suva.ch`

[Wik] Wikipedia, Keyword-Driven Testing, siehe: `http://de.wikipedia.org/wiki/Keyword-Driven_Testing`

Wiederverwendbare Librarys

Da die Prozeduren aus den Testfällen heraus wiederverwendet werden, werden die Prozeduren wiederum in Librarys ausgelagert, die in den jeweiligen Testfällen dann einfach inkludiert werden. Die Librarys wiederum werden hierarchisch aufgeteilt nach Verantwortlichkeiten in applikationsspezifische, applikationsneutrale und technische Librarys (siehe Abbildung 7). An oberster Stelle steht die von QFTest mitgelieferte „qfs-Library“, die hilfreiche, wiederverwendbare Prozeduren enthält.

Fazit

Die Keyword-Driven Testmethode als Testansatz zur Testautomatisierung einer Standardsoftware hat sich bewährt und wurde erfolgreich eingesetzt.

Ein wesentlicher Erfolgsfaktor war auch die Struktur der Testorganisation, also ein Scrum-Team bestehend aus mehreren Personen mit unterschiedlichen Rollen. Es gibt Fachexperten für die Fachtest-Automatisierung, die die Testfälle mit den Keywords mit sprechenden Namen auf Spezifikationsebene erstellen und implementieren. Und es gibt Implementierungsexperten, die in der Lage sind, ohne entsprechendes Fachwissen die Keywords im Testautomaten zu implementieren. Derzeit gibt es zwölf Fachexperten, aufgeteilt in verschiedene Fachbereiche, die die Testfälle implementieren, und zwei Implementierungsexperten, die die Keywords implementieren und warten.

Zwischenzeitlich wurden von den Implementierungsexperten über 260 wiederverwendbare Keywords umgesetzt. Derzeit werden von der Testautomatisierungs-Architektur pro Stunde circa 500 Keywords, am Tag im Schnitt bis zu 80.000 Keywords mit insgesamt 120 QFTest-Daemons aufgerufen und verarbeitet.

Der Aufbau einer Testautomatisierungs-Architektur mit der zugehörigen Testorganisation ist zwar aufwändig und mit einem entsprechenden Investment verbunden, zahlt sich aber im Endeffekt durch eine höhere Qualität der Software aus, da innerhalb von kurzen Release-Zyklen ein manuelles Testen aller Testfälle zeitlich nicht möglich ist. Zudem lassen sich bequem Testdaten und produktive Stammdaten generieren. Wesentliche Erfolgsfaktoren hierzu sind aber eine stabile Ausführung der automatisierten Tests, ein vertretbarer Wartungsaufwand der Testfälle durch eine möglichst hohe Wiederverwendbarkeit und Flexibilität auf allen Ebenen der Testautomatisierungs-Architektur. ||

Der Autor

|| Michael Reiser
(michael.reiser@riotec.ch)
ist selbstständiger Diplom-Informatiker (FH) und arbeitet seit über 20 Jahren in der Softwareentwicklung in Großprojekten, in den letzten drei Projekten seit sechs Jahren als Testautomatisierer.