Robert Huber
(huber.robert2@muenchener-verein.de)
ist Testmanager beim Münchener Verein. Er verfügt über mehr als 20 Jahre Erfahrung in Testautomatisierungslösungen für GUIs für Java und Weboberflächen und verantwortet aktuell als Testmanager die Einführung von PSLife von adesso bei der Versicherungsgruppe Münchener Verein.

Hybrid Keyword-Driven Testing beim Münchener Verein

Hybrid Keyword-Driven Testing ist eine Ergänzung und Weiterführung des klassischen Keyword-Driven Testings, wobei generische Objekte verwendet werden. Diese sind losgelöst von der verwendeten Programmiersprache der Oberfläche und damit auch losgelöst von jeder Erkennungslogik, die einer Automatisierung normalerweise zugrunde gelegt wird. Das bedeutet mit anderen Worten, dass alle Objekte, die zur Ablaufsteuerung eines Testautomaten benötigt werden, zur Laufzeit dynamisch erstellt werden. Typische wartungsintensive und langwierige Aufzeichnungs- und Erfassungsschritte sind hierbei nicht notwendig. Während der Ausführung der Testskripte werden alle notwendigen Variablen aus dem Keyword-Driven Ausführungsplan gelesen und in ein ablauffähiges Skript umgewandelt. Dieses Vorgehen ist ideal bei agilen Projekten und ermöglicht bereits zu einem sehr frühen Zeitpunkt über den gesamten Releasezyklus Testautomatisierungen einzusetzen, bei denen die Wartungskosten nicht zur „Wartungsfalle" werden.

Motivation

Neue Produkte beim Fachbereich einzuführen, ist mit viel Testaufwand verbunden. In diesem Artikel soll dargestellt werden, wie die End-to-End-Tests der funktionalen Anforderungen so früh wie möglich beschrieben und automatisiert werden können, ohne in die Wartungsfalle zu geraten. Java- und HTML-Anwendungen sollen in kürzester Zeit von Beginn an automatisiert getestet werden können.

Ziel war es beim Münchener Verein (kurz MV), ein neues Bestandssystem, „PSLife" der adesso-Gruppe, einzuführen mit folgenden Anforderungen an die Automatisierung:

- Automatisierung beinhaltet ca. 150 Standardprozesse des Vertragslebenszyklus in der Lebensversicherungsbrache: vom Antrag über die Vertragsänderung bis in die Leistung,
- beinhaltet anpassbare Testfälle für alle marktgängigen Lebensversicherungsprodukte,
- beinhaltet Szenarien von unterschiedlichen Testfallkombinationen,
- Auswahl von Testfällen für monatliche Releasetests und Performanztests,
- enger Zeitplan ist vorgegeben, von der ersten Anforderung bis zur Umsetzung,
- begrenzte Ressourcen beim Fachbereich, um die Testfälle zu erstellen und dann durchzuführen,
- stabile und zukunftsweisende Automatisierung, die für künftige Projekte eingesetzt werden kann.

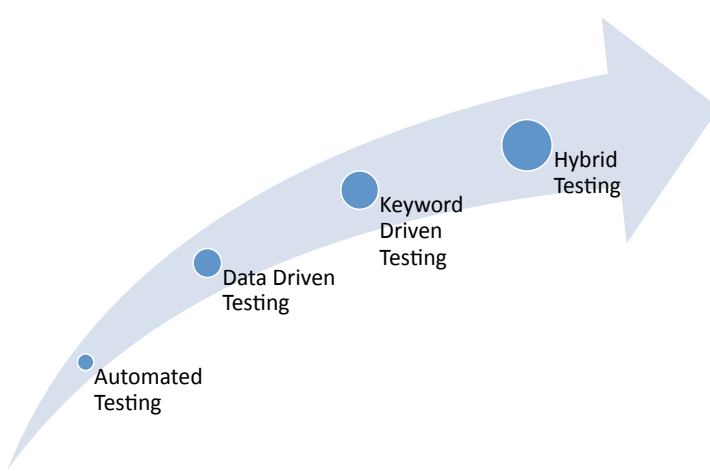


Abb. 1: Testautomatisierungsarten.

Deshalb wurde beim MV beschlossen, bei der Automatisierung die Strategie des Hybrid Keyword-Driven Testings [Wri10] einzusetzen.

Begriffserklärung

Versucht man die folgenden Punkte bei einer Automatisierung zu erfüllen und zu optimieren, spricht man von Hybrid Testing (vgl. auch Abbildung 1):

- *Flexibilität*: Ein Testframework muss anpassbar sein. Bei Änderungen in der Zielanwendung darf kein übermäßiger Aufwand bei der Anpassung des Frameworks entstehen.
- *Wiederverwendbarkeit:* Ebenso sollten die Tests, die Funktionen und die Verweise auf die grafischen Objekte mehrmals und daher wiederverwendbar sein.
- *Einfachheit:* Die Komplexität sollte für die verschiedensten Fachbereiche leicht überschaubar und intuitiv erfassbar sein.
- *Schnelligkeit/Effektivität/Effizienz:* Des Weiteren sollte es schnell möglich sein, neue Tests und Funktionen zu erstellen sowie Anpassungen daran vorzunehmen.
- *Robustheit:* Des Weiteren muss die stabile automatische Ausführung der Testfälle in den heißen Projektphasen gewährleistet werden.
- *Wirtschaftlichkeit:* Die gesamte Testautomatisierung muss so früh wie möglich rentabel sein.

Um die Testfälle zu strukturieren, wird zusätzlich Keyword-Driven Testing verwendet.

Zunächst werden die zu testenden Aktionen oder Operationen in der Anwendung (oder in den Anforderungen für die Anwendung) analysiert. Eingabeaktionen und -abläufe werden in Keywords (Schlüsselwörtern) gekapselt und stehen klassischerweise in Tabellen zur Verfügung, die dann für den Testautomaten verwendet werden.

Beim Keyword-Driven Testing erscheint der Aufwand zu Beginn höher als bei einfach aufgenommenen Skripten. Jedoch macht sich sorgfältige Planung bei der folgenden Testerstellung und -wartung bezahlt. So fördert Keyword-Driven Testing eine stabile und übersichtliche Teststruktur, die für alle Testfälle wiederverwendet werden kann. Auch kann die gleiche Struktur projektübergreifend verwendet werden, sodass der Fachbereich nur ein einziges Mal in die Struktur eingewiesen werden muss. Die Testfälle können wahlweise manuell oder bevorzugt automatisiert durchgeführt werden.

Beim Hybrid Testing werden die benötigten Objekte bereits zu Beginn generisch beschrieben und können dann auch später bei den End-to-End-Tests im Testautomaten für die Automatisierung verwendet werden.

Diesem Ansatz folgend, ist es möglich, bereits frühzeitig mit Testfallbeschreibungen für den Testautomaten zu beginnen. Idealerweise bereits zusammen mit den User Requirements als Anwendungsfälle.

Vergleich zwischen traditionellem und hybrid automatisiertem Testen

Je früher man die Testbeschreibungen für Testfälle erfasst und für die Automatisierung wiederverwendet, desto größer ist der Benefit. Testfälle können schon in einem sehr frühen Stadium Lücken bei der Spezifikation aufzeigen und dann Schritt für Schritt vervollständigt werden, um schließlich bei der Testdurchführung zum Einsatz zu kommen. Der Ansatz ist, die gleiche Testfallbeschreibung für alle Stufen der Entwicklung (User Requirements, System Requirements, Design und Implementierung, siehe Abbildung 2) zu verwenden.

Frühzeitige Integration schafft hier den Vorteil, viele Vorgänge nur einmal klar strukturiert beschreiben zu müssen, um sie später manuell und automatisiert ablaufen lassen zu können. Beim Hybrid Testing versucht man so frühzeitig wie möglich, Testfälle zu erstellen und diese über den gesamten Entwicklungszyklus zu vervollständigen und zu pflegen, ähnlich dem Ansatz von Acceptance Test-Driven Development (ATDD). Zusätzlich werden jedoch alle benötigten Objekte zur Testautomatensteuerung generisch vorab beschrieben und während des Testablaufs dynamisch erstellt. Gerade auch in agilen Projekten mit häufigen Änderungen im Ablauf und an der Oberfläche ist dies von großem Vorteil.

Beim traditionellen Testen erfolgt erst kurz vor Auslieferung die Erstellung der Testfallbeschreibungen mit den dann möglichen Testautomatisierungen und benötigten Objekten zur Steuerung. Häufig sind hier auch Testautomatisierungen erst nach dem ersten Release sinnvoll, da der Aufwand zu groß ist, die Automatisierungen an jede GUI-Änderung anzupassen. Ein klassischer Fall ist hier die Verwendung von „Capture-Replay"-Tools für die Automatisierung, die sehr pflegeintensiv sind. Siehe auch den Abschnitt „Wartungsfalle" weiter unten.

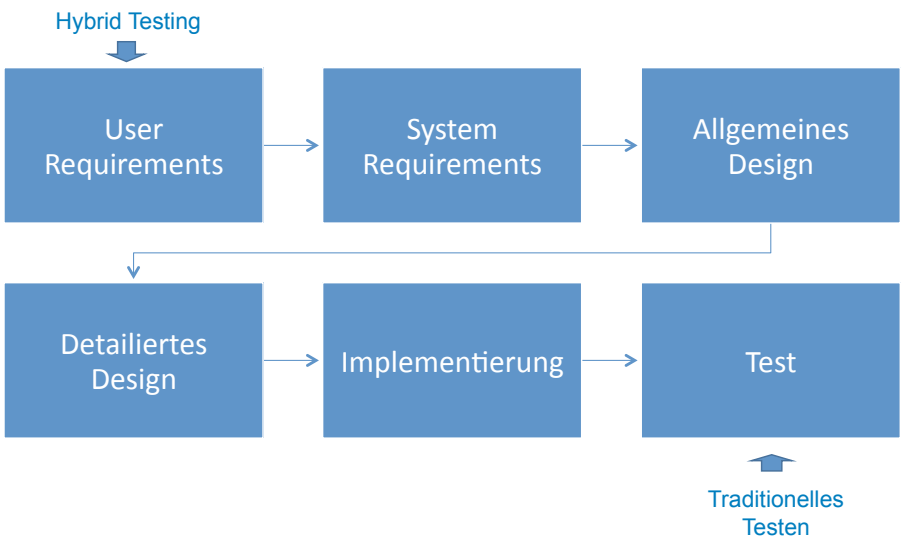


Abb. 2: Test: Traditionell versus hybrid.

Anforderungen

Um Hybrid Testing umsetzen zu können, muss eine schlanke Werkzeugkette eingesetzt werden für folgende Tätigkeiten:

- Testfälle beschreiben,
- Testfälle ausführen,
- Testfälle auswerten,
- Testfälle planen/gruppieren/strukturieren.

Der Fachbereich sollte alle Testfälle komplett selbstständig erstellen, ausführen und warten können.

Wichtig ist, dass so wenig zusätzliche Tools wie möglich verwendet werden, da sonst die Pflege der eingesetzten Produkte mit den verbundenen Kosten ebenfalls schnell zur Wartungsfalle wird. Die verwendeten Tools müssen einmal initial auf-

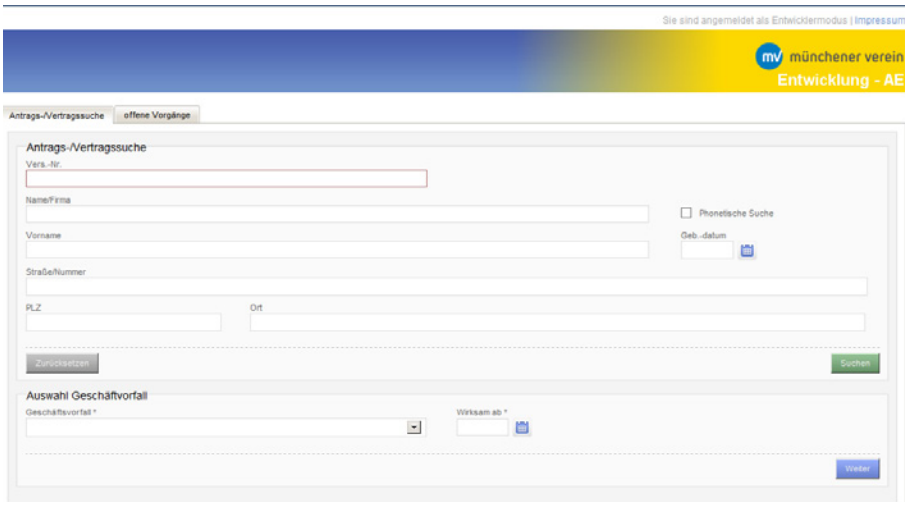


Abb. 3: Eingabemaske von PSLife.

gesetzt und konfiguriert werden. Für alle zukünftigen Projekte können diese dann uneingeschränkt wiederverwendet werden. Beim Münchener Verein konnte die Anzahl der Tools auf drei beschränkt werden:

- *Basistesttool:* Wir haben MS-Excel eingesetzt und damit die Testfälle erstellt, beschrieben, geplant und strukturiert. Darüber hinaus wurde Excel auch für die Objekt-Repository-Pflege und das Auslesen und Rückspielen der Ergebnisse verwendet.
- *Steuerung und Auswertung:* Hier wurde Jenkins eingesetzt, um den Testautomaten von jedem beliebigen Ort und zu jeder beliebigen Zeit über Crontab-Einträge auf virtuellen Maschinen starten zu können.
- *Testautomat:* Verwendung von QF-Test als Testautomat für alle GUI-Aktionen. Dieser wird angesteuert über Jenkins und verarbeitet dann die ihm zur Verfügung stehenden Excel-Dateien.

Anforderung an die Teststeuerung

Es ist wichtig, die Tools über einen zentralen Einstieg steuern zu können. Wir haben uns hierbei für Jenkins entschieden. Jenkins bietet eine simple Weboberfläche an, die der Fachbereich von jedem Arbeitsplatz starten kann. Der von Jenkins gesteuerte und überwachte Testautomat läuft im Hintergrund auf einer beliebigen virtuellen Maschine. Während des Testlaufs werden automatisch Reporte angelegt und als HTML-Dokumente direkt zur Auswertung an Jenkins übergeben.

Anforderung an das Basistesttool

Mit MS-Excel erhält man ein sehr mächtiges Werkzeug, mit dem man Testfälle beschreiben, ausführen, auswerten und auch die gesamte Planung/Steuerung durchführen kann. Der Fachbereich ist hier in kürzester Zeit in der Lage, selbstständig all diese Tätigkeiten durchzuführen, da Wissen und Know-how zu Excel meist vorhanden ist beziehungsweise schnell aufgebaut werden kann. Auch ist dieses Werkzeug in nahezu allen Firmen ohne Einschränkungen und zusätzlichen Kosten verfügbar.

Obwohl Excel kein klassisches Testtool ist, kann es hierfür sehr gut verwendet werden, weil es die meisten Testautomaten in der Zwischenzeit auswerten können.

Anforderung an den Testautomaten

Zuerst muss also ein Testautomat gefunden werden, der es uns ermöglicht, die Erkennung dieser Objekte auf diese Anwendersicht zu reduzieren. Des Weiteren muss es möglich sein, den Automaten im Falle einer unsauberen Erkennung zu steuern beziehungsweise ihm das richtige Verhalten beizubringen. Hierfür haben wir uns beim Münchener Verein für den Testautomaten QF-Test entschieden.

QF-Test bietet eine flexible Objekterkennung an, welche wir mit sogenannten Resolverskripten des Tools an unsere Bedürfnisse anpassen konnten. Hier ist es möglich, durch unterschiedliche Resolverskripte für Java-Anwendungen und durch den CustomWebResolver-Ansatz für Webanwendungen die Objekterkennung gezielt zu steuern, ohne alle Objekte vorher aufgezeichnet zu haben. Mit Excel können diese Objekte dann dynamisch zur Laufzeit komplett erkannt und innerhalb von Sekunden ausgeführt werden.

Der initiale Aufwand zur Erfassung aller Objekte für unsere Anwendung lag hier im Bereich weniger Tage. Nötigenfalls kann man dazu auch externe Experten hinzuziehen, sodass die Dauer noch kürzer ist.

Man erhält bei diesem Vorgehen eine stabile Testautomatisierungsumgebung, die nicht anfällig ist für Änderungen der Eigenschaften der Objekte, die einen Fachbereich nicht interessieren. Den Fachbereich interessiert zum Beispiel nur der Text des Labels eines Eingabefeldes auf einer bestimmten Maske und nicht eine von einem Framework generierte technische ID oder ein Pfad von Objekten durch einen Objektbaum.

Erkennung und Steuerung der Objekte

Um eine Automatisierung frühzeitig erstellen zu können, die eine GUI zielsicher bedient, sind eindeutige Labels und eine Handvoll Objekte zur Steuerung notwendig.

Checkliste für eindeutige Bezeichner der GUI-Elemente:

- Verwendung keiner technischen IDs, die sich durch die Entwicklung ändern können! Verwendung *nur* von „Klarnamen“, wie der Fachbereich diese auch an der Oberfläche sieht.
- „Labels“ von Textfeldern sind gleichzeitig die eindeutigen Identifier, um die Automatisierung zu steuern.
- Keine weiteren versteckten Identifier werden verwendet!
- Für ein mehrfaches Auftauchen desselben Labels muss der Testautomat eine Lösung finden.

Steuerobjekte:

- Man benötigt verschiedene Eingabetypen, die mit einer Handvoll Steuerbegriffen verbunden werden: „Klicke“, „Setze“, „Vergleiche Wert“ genügen, um die meisten Anwendung komplett zu steuern.

Abbildung 3 zeigt das Beispiel einer Eingabemaske. Wandelt man diese Informationen einer jeden Maske in eine lesbare minimalisierte Form um, erhält man ein komplettes Objekt-Repository (siehe Abbildung 4) für jedes GUI-Element auf jeder Maske. Jedes Objekt kann damit zur Laufzeit eindeutig identifiziert werden (siehe Abbildung 5).

Verwendet man dieses Objekt-Repository als Basis, können nun beliebige Testfälle erstellt werden. Jedes Objekt erhält eine eindeutige Objektbeschreibung auf jeder Seite, die grundsätzlich dem entspricht, was auch der Anwender sieht und auch die

qft_id Main	sort1	sort2	sort3	label	matches	typ
5	Antrags/Vertragssuche					
6				Vers.-Nr.		Text
7				Name/Firma		Text
8				Phonetische Suche		Checkbox
9				Vorname		Text
10				Geb.-datum		Datum
11				Straße/Nummer		Text
12				PLZ		Text
13				Ort		Text
14				Zurücksetzen		Button
15				Suchen		Button
16				Rolle		Tabelle_Link
17			Auswahl Geschäftvorfall			
18				Geschäftsvorfall		Combobox
19				Wirksam ab		Datum
20				Weiter		Button
21	Produktauswahl					

Abb. 4: Objekt-Repository.

Minimalanforderung für die Steuerung des Testautomaten ist.

Die Testallbeschreibung muss nun ebenfalls in einer normierten Form in Excel erfolgen. Der Automat kann aus diesen Excel-Tabellen ausführbare Objekte zur Laufzeit erstellen.

In Abbildung 6 werden Testfälle und deren Beziehung zur Oberfläche angezeigt. In dem Beispiel werden folgende zwei Testfälle (Spalten) durchgeführt:

- Testfall 1 (Spalte I): Namen, Vorname, Ort eingeben und dann „Suchen“ klicken.
- Testfall 2 (Spalte J): Geschäftsvorfall auswählen, Datum eintragen und dann „Weiter“ anklicken.

Die Informationen aus dem Objekt-Repository werden aber im Testfall nur angezeigt! Sie sind mit dem Objekt-Repository dynamisch verlinkt (als Excel-Referenz). Somit gibt es ein zentrales Objekt-Repository, welches für alle Testfälle verwendet wird. Dieses Objekt-Repository wird nun zur Laufzeit ausgelesen und für die Objekterkennung an der Oberfläche verwendet.

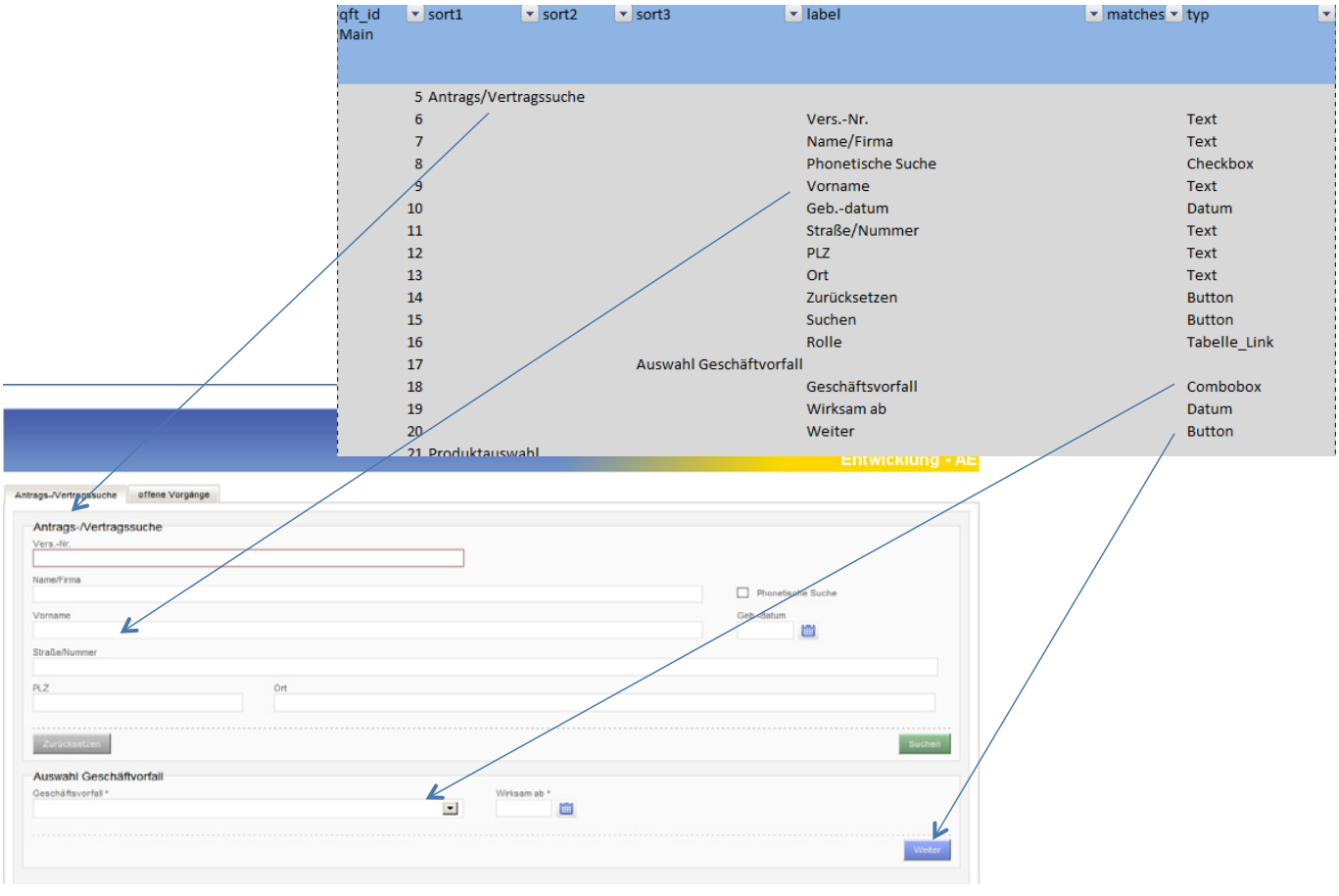
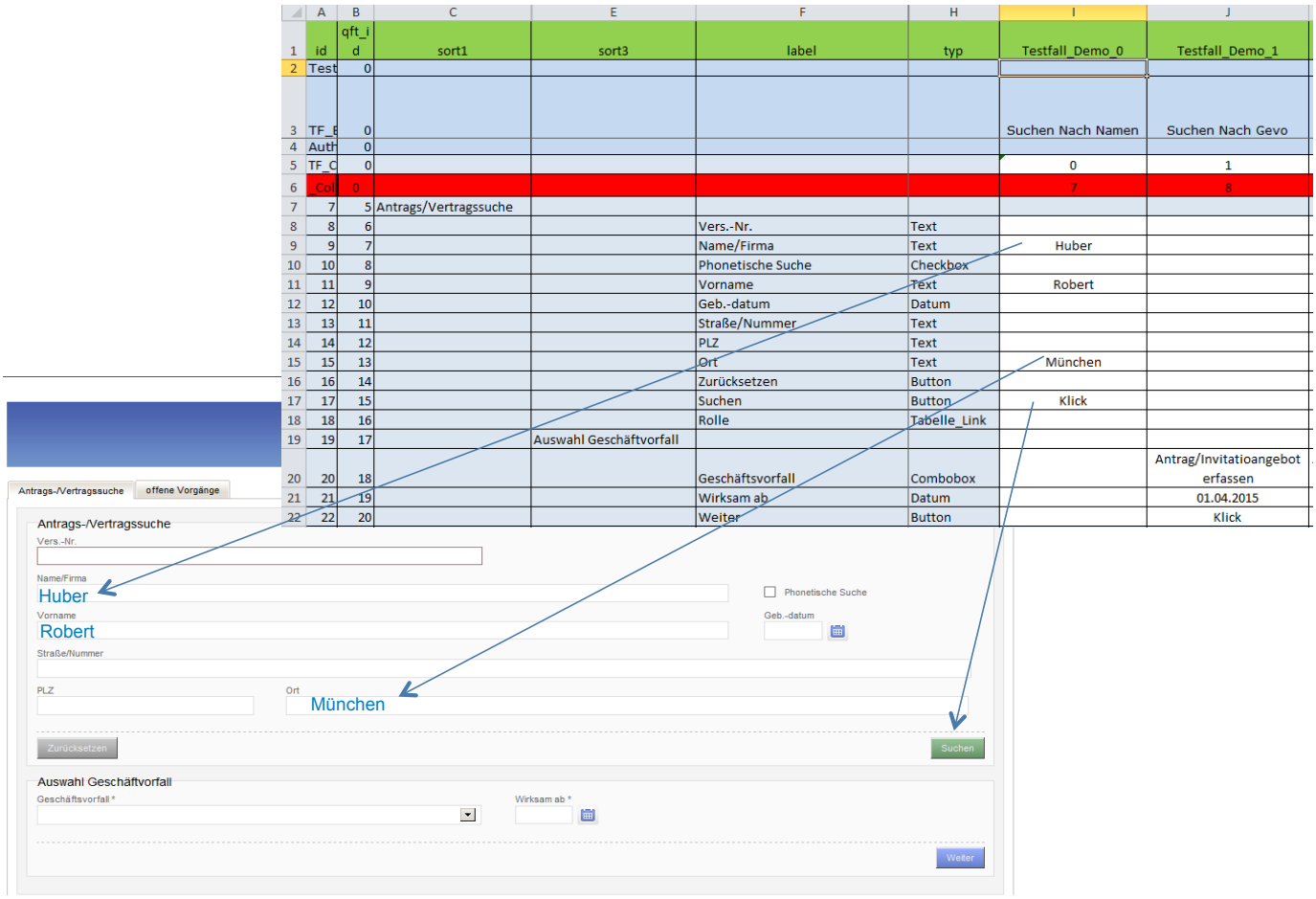


Abb. 5: Objekt-Repository-Beziehungen.



	A	B	C	E	F	H	I	J
1	id	qft_id	sort1	sort3	label	typ	Testfall_Demo_0	Testfall_Demo_1
2	Test	0						
3	TF_E	0					Suchen Nach Namen	Suchen Nach Gevo
4	Auth	0						
5	TF_C	0					0	1
6	Col	0					7	8
7	7	5	Antrags/Vertragssuche					
8	8	6			Vers.-Nr.	Text		
9	9	7			Name/Firma	Text	Huber	
10	10	8			Phonetische Suche	Checkbox		
11	11	9			Vorname	Text	Robert	
12	12	10			Geb.-datum	Datum		
13	13	11			Straße/Nummer	Text		
14	14	12			PLZ	Text		
15	15	13			Ort	Text	München	
16	16	14			Zurücksetzen	Button		
17	17	15			Suchen	Button	Klick	
18	18	16			Rolle	Tabelle_Link		
19	19	17		Auswahl Geschäftvorfall				
20	20	18			Geschäftsvorfall	Combobox		Antrag/Invitatioangebot erfassen
21	21	19			Wirksam ab	Datum		01.04.2015
22	22	20			Weiter	Button		Klick

Abb. 6: Gliederung der Testfälle.

Idealerweise wird das Objekt-Repository automatisch von der Entwicklung zur Verfügung gestellt, sodass die erste Erstellung und Pflege schneller durchgeführt werden kann. Ist dies nicht der Fall, kann der Fachbereich dieses Objekt-Repository aber auch selbstständig erstellen und pflegen.

Wartungsfalle

Vermeidung der Wartungsfalle sollte das höhere Ziel der Automatisierung sein! Um die Wartbarkeit automatisierter Testfälle zu gewährleisten, ist zudem eine tragfähige und gleichzeitig flexible Testware-Architektur mit verschieden Abstraktionsebenen nötig:

- Auf der oberen Ebene können die Fachexperten durch verständliche Schlüsselwörter ein Objekt-Repository erstellen.
- Im Mittelteil befindet sich das Testautomaten-Framework, welches zentral über Jenkins gesteuert wird.
- Auf der unteren Ebene befinden sich strukturierte MS-Excel-Tabellen zur vereinfachten Wartung. Durch diese Trennung erhält man eine einfach zu wartende Testumgebung mit wenigen Tools.

Nachhaltige Ergebnisse zeigen sich hier insbesondere bei Unternehmen, die in der Testautomatisierung eine langfristige Investition sehen und erstellte Testartefakte in mehreren Projekten wiederverwenden können

Fazit

Der Fachbereich muss bei dem oben beschriebenen Vorgehen einige Formalismen in Kauf nehmen, während die IT alle technischen Details vom Fachbereich fern halten sollte. Laufen die ersten automatisierten Testfälle, kann der Fachbereich bereits zügig neue Varianten dieser Geschäftsobjekte durch Kopieren und Editieren erstellen. Dies ist der Punkt, an dem sich der Hebel für die Maximierung der Produktivität befindet. Die Fachexperten werden mit geringem Lernaufwand in die Lage versetzt, komplizierte Sachverhalte in automatisierten Tests ohne Programmierkenntnisse selbst zu realisieren, auch wenn die Oberflächen noch nicht oder nur in Teilen zur Verfügung stehen. ■

Links

[Adesso] Homepage adesso insurance solutions GmbH, siehe http://www.adesso-insure.de
[Jenkins] Homepage, siehe: https://jenkins.io/
[Micro] MS-Excel, Homepage von Microsoft, siehe: https://www.microsoft.com/de-de/
[PSLife] Homepage PSLife, siehe: http://www.pslife.eu
[QFS] QF-Test: Das GUI Testtool für Java & Web, Quality First Software, siehe: https://www.qfs.de
[WikiATDD] https://en.wikipedia.org/wiki/Acceptance_test-driven_development
[WikiKDT] https://de.wikipedia.org/wiki/Keyword-Driven_Testing
[Wri10] J. L. Wright, Hybrid Keyword Data Driven Automation Frameworks, ANZTB (The Australia and New Zealand Testing Board) Conference, 2.3.2010, siehe: http://www.automation.org.uk/downloads/HybridKeywordDataDrivenAutomationFrameworks.pdf