Johann Gietl

(johann.gietl@sogeti.de)
verfügt als iSQI Quality Assurance Management Professional über tiefe Kenntnisse in der Testautomatisierung und dem Testmanagement. Seit 2011 ist er für Sogeti Deutschland als Managing Consultant tätig.

Testautomatisierung heute

Erfolgsfaktoren bei der automatischen Ausführung von GUI-Tests

Die meisten Firmen stellen im System- und Abnahmetest für das GUI-Testen individuelle Testautomatisierungslösungen zusammen, und verbrauchen damit sehr viel Kraft und Ressourcen. Sucht man nach Prozessen und Werkzeugen, so wird man zwar fündig, aber beim Einsatz ergeben sich vielfältige Probleme, die es zu umschiffen gilt. Dieser Artikel beschreibt den aktuellen Stand der Technik zur Lösung dieser Probleme und liefert eine praktische Checkliste (siehe Abbildung 1), um die automatische Ausführung von Testfällen erfolgreich planen und durchführen zu können.

Es ist nicht optimal, wenn die Fachbereichsexperten manuelle Testfälle erstellen und den Programmierern zur Automatisierung übergeben. Deshalb ist es inzwischen üblich, dass der Fachbereich in den Testautomatisierungsprozess einbezogen wird. Die Testautomatisierung wird unter den Gesichtspunkten Motivation, Anforderungen, Konzepte und Prozesse beleuchtet. In den letzten zehn Jahren entstand die hier vorgestellte Checkliste aus der Dokumentation von Strukturen und Konzepten von zahlreichen Testautomatisierungsprojekten, unter anderem bei Allianz, Audi, BMW, KDG, Kion, Siemens und Zurich.

Der Begriff Testautomatisierung im Softwaretest wird sehr oft mit unterschiedlichen Bedeutungen verwendet. Dieser Artikel beschränkt sich bei der Definition des Begriffs auf die automatisierte Ausführung des System- und Abnahmetests. Es wird im Folgenden auf die zehn wichtigsten Themen eingegangen, die wirklich essenziell für den Erfolg im praktischen Einsatz sind. Die Reihenfolge der Themen ergibt sich aus der zeitlichen Abfolge der Themen und stellt keine Priorisierung dar.

Motivation

Manager bemängeln bei der Testautomatisierung die hohen Kosten und die Haltbarkeit von Testsuiten und suchen nach Wegen zur Erhöhung der Produktivität und zur Vermeidung der Wartungsfalle.

Erhöhung der Produktivität

Ab einer gewissen Anzahl wird die automatisierte Ausführung eines Testfalles durch eine Maschine kostengünstiger als dessen manuelle Ausführung durch einen Mitarbeiter. Ist die genügend häufige Ausführung eines manuellen Testfalles geplant, rentiert sich die Investition in die Erstellung und Wartung eines automatischen Testfalles, der ohne Tester „vollautomatisch" ausgeführt werden kann. Durch die Aufhebung der Trennung von manuellem und automatisiertem Testfall (siehe Abbildung 2) in einem integrierten Testfall, der sowohl vom Fachbereichsexperten als auch vom Automatisierungsingenieur ausgeführt und angepasst werden kann, lässt sich die Produktivität bis zum Faktor 5 erhöhen. Wenn der Fachbereich Geschäftsprozesse dokumentiert, werden gleichzeitig automatisierte Testfälle erstellt. Automatisiert der Automatisierungsingenieur gut verstandene Geschäftsprozesse, werden dabei auch komplexe Testszenarien erstellt.

Vermeidung der Wartungsfalle

Um die Wartung automatisierter Testfälle zu gewährleisten, ist zudem eine tragfähige und gleichzeitig flexible Testware-Architektur mit verschieden Abstraktionsebenen nötig (vgl. [Gra12]). In der oberen Ebene der Testware-Architektur (siehe Abbildung 3) können die Fachexperten durch verständliche Schlüsselwörter Tests anpassen, ausführen oder sogar selbst erstellen. Dieser Ansatz ist inzwischen weit verbreitet. Im Mittelteil befindet sich ein Framework, in dem die Testware organisiert ist. In der unteren Ebene befinden sich strukturierte Skripte zur vereinfachten Wartung. Dies

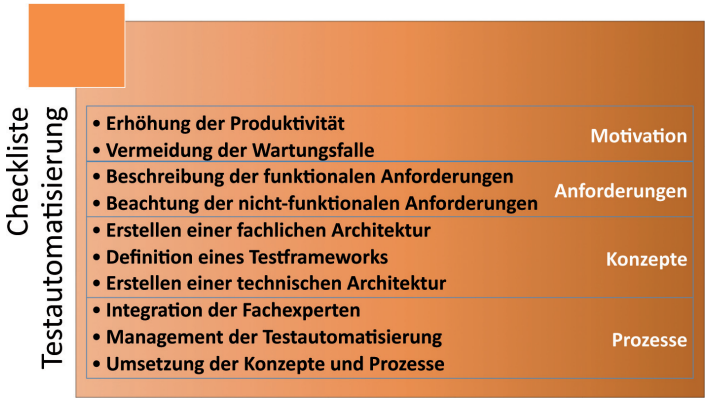

Abb. 1: Checkliste Testautomatisierung.

Abb. 2: Integration von manuellem und automatisiertem Test.

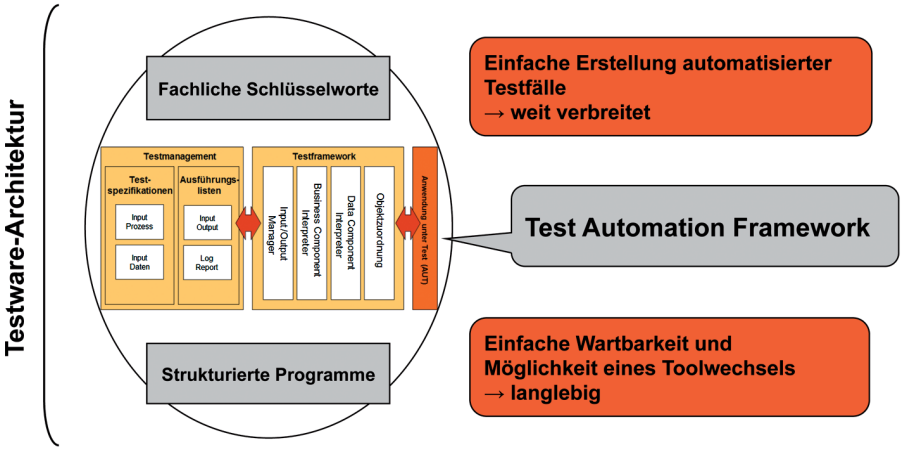

Abb. 3: Testware-Architektur.

sichert die Langlebigkeit der automatisierten Tests.

Anforderungen

Nach den Motivationsthemen Produktivität und Wartungsfalle müssen nun die Anforderungen festgelegt werden. Die wesentlichen Elemente der funktionalen Anforderungen werden als Use Cases definiert, während die nichtfunktionalen Anforderungen als einfache Liste realisiert werden.

Beschreibung der funktionalen Anforderungen.

Im Folgenden sind beispielhaft die Use Cases zur automatischen Ausführung eines Tests dargestellt (siehe Abbildung 4).

Mit *Aktiviere Testausführung* initiiert der Fachexperte die Testausführung durch den Start einer Ausführungsliste. Die Tests werden vom *Test Management Tool* abgeholt. Über *<<include>>* wird der Use Case *Interpretiere Test* eingebunden, der die Tests zeilenweise abarbeitet und ausführbaren Code erstellt. Wenn Fehler auftauchen, wird *Sende Fehlerbenachrichtigung* aufgerufen. Des Weiteren initiiert das Testframework den Use Case *Führe Test aus* mit dem Actor *Test Automation Tool*, der den Code ausführt. Schließlich wird im Use Case *Erstelle Testlog* für jede Testzeile ein Logeintrag erstellt. In ähnlicher Weise werden die Use Cases für die Testfallspezifikation und für den (Fehler-)Report erstellt.

Beachtung der nichtfunktionalen Anforderungen

Einer ausführlichen Betrachtung bedürfen neben den funktionalen auch die nichtfunktionalen Anforderungen. Dabei gilt es, die folgenden Qualitätsmerkmale als Anforderungen aufzulisten, entsprechend zu gewichten und mit Abnahmekriterien zu versehen. Je wichtiger eine Anforderung bewertet wird, umso genauer müssen dazu Messwerte definiert und überprüft werden.

- **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, Komponenten und Funktionen mehrmals einsetzbar sein.
- **Einfachheit:** Die Komplexität sollte für die verschiedensten Fachbereiche leicht überschaubar und intuitiv erfassbar sein.
- **Schnelligkeit:** Daneben sollte es schnell möglich sein, neue Tests, Komponenten und Funktionen zu erstellen sowie Anpassungen daran vorzunehmen.
- **Effektivität/Effizienz:** Selbstverständlich ist es notwendig, effektiv und effizient Testfälle erstellen zu können.
- **Robustheit:** Des Weiteren muss die stabile automatische Ausführung der Testfälle in den heißen Projektphasen gewährleistet werden.
- **Wirtschaftlichkeit:** Das gesamte Paket muss rentabel sein.

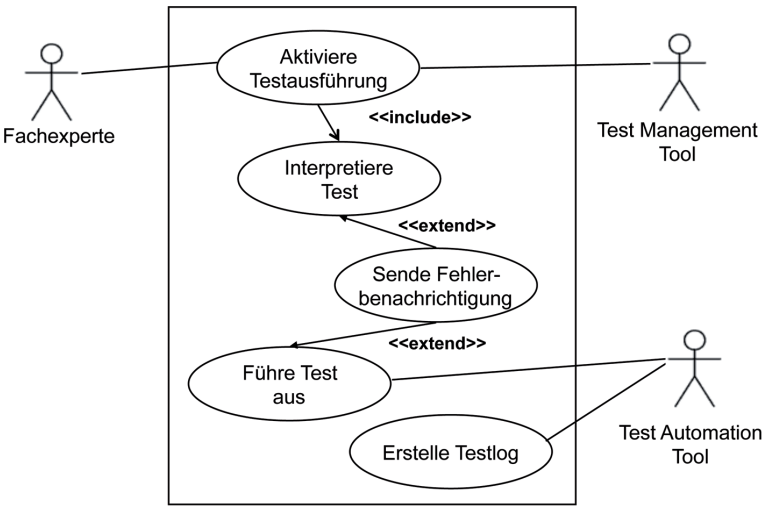

Abb. 4: Use Cases Automatische Testfallausführung.

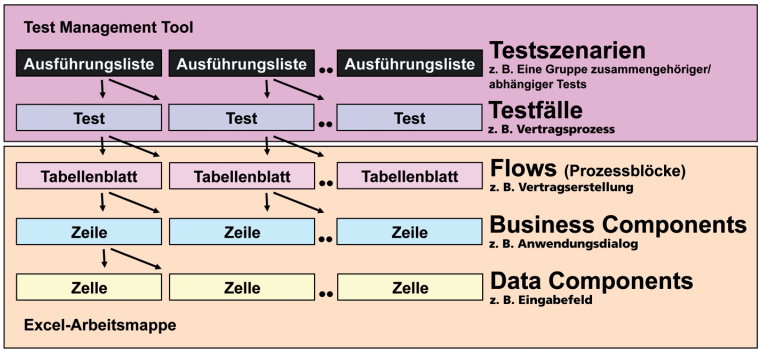

Abb. 5: Fachliche Architektur.

- **Standardisierung:** Schließlich sollten standardisierte Werkzeuge, Methoden und Prozesse für eine nachhaltige Verwendung der Ergebnisse sorgen.

Konzepte

Nach Dokumentation und Priorisierung der Anforderungen müssen für die drei Abstraktionsebenen *Fachliche Architektur*, *Testframework* und *Technische Architektur* tragfähige Konzepte definiert werden.

Definition der fachlichen Architektur

Die fachspezifische Schnittstelle bildet die Kommunikationsbasis zwischen den fachlichen Testdesignern und den Testingenieuren. Die fachspezifische Testsprache wird in der Literatur oft *Domain Specific Test Language (DSTL)* genannt. Es ist zum einen darauf zu achten, dass die Sprache einfach und intuitiv gehalten wird und zum anderen, dass der Formalismus auf das Notwendigste beschränkt wird, um Geschäftsprozesstests im System- und Abnahmetest automatisch auszuführen. Das etablierte Grundprinzip ist eine Folge von Zeilen der Form *Schlüsselwort;Datum;Datum;…* (Beispiel: *Login;User:=Mustermann;Password:=1234*) ohne Schleifen und Sprünge. Als optimale Darstellung zur einfachen Handhabung der Tests haben sich Tabellenblätter bewährt, die von fast allen Beteiligten ohne großen Schulungsaufwand bereits beherrscht werden.

In Abbildung 5 sind fünf Ebenen einer fachlichen Architektur dargestellt. Ganz oben im Test Management Tool befinden sich die Ausführungslisten, darunter die Testfälle. Die weiteren Ebenen sind hier als Excel-Arbeitsmappen integriert. Jedes Tabellenblatt entspricht einem Ablauf (oder *Flow*), jede Zeile entspricht einem Prozessschritt, auch Geschäftskomponente oder *Business Component (BC)* genannt. Bei besonders komplexen Anwendungen können für die BCs weitere Ebenen definiert werden. Zum Beispiel die Ebenen *Niedrig* für einfache Steuerungselemente, *Mittel* für Dialoge und *Hoch* für Teilprozesse. BCs bestehen aus einer Zelle, die das Schlüsselwort enthält und den dazu zugehörigen Datenzellen, die hier *Data Component (DC)* genannt werden. Die physische Trennung von Testschritten und Testdaten machen das Testdesign bzw. die manuelle Testausführung unnötig kompliziert. Um diese Trennung zu vermeiden, werden hier DCs definiert. Eine DC besteht aus einem Parameternamen und einem Wert. Der Parametername ergibt sich zumeist direkt aus den Feldbezeichnern der zu testenden Anwendung. Bei den Werten werden die drei Datenarten *statisch*, *dynamisch* und *unmittelbar* unterschieden:

- Die statischen Daten sind bei allen Testfallausführungen konstant und werden als konkrete Parameterwerte spezifiziert.
- Dynamische Daten werden zu Beginn der Testfallausführung interpretiert und in konkrete Parameterwerte umgewandelt. Ein Beispiel hierfür ist die DC *<Today>*, die zu Beginn der Testausführung in das aktuelle Tagesdatum umgewandelt wird.
- Unmittelbare Daten sind erst an einem bestimmten Punkt während der Testfallausführung verfügbar. Dabei handelt es sich meist um Prüfungen von Ergebniswerten oder um die Ausgabe von z. B. Vertragsnummern.

Alle DCs werden direkt bei den Geschäftsprozessschritten (oder BCs) spezifiziert. Bei der Erstellung und Strukturierung von Testfällen, Flows, BCs und DCs ist auf die Lesbarkeit und Wartbarkeit zu achten.

Definition des Testframeworks

Nach der fachlichen Architektur geht es jetzt an das Framework zur Verbindung der fachlichen und technischen Ebenen.

In Abbildung 6 ist die dreiteilige Struktur eines Testframeworks abgebildet. Auf der linken Seite befinden sich die Testfallspezifikationen mit den Tabellenblättern *Input-Prozess*, *Input-Daten* und *Input/Output-Daten* sowie Log- und Report-Dateien. In der Mitte befindet sich der Kern des Testframeworks. Der Input/Output Manager liest die Testfälle zur Laufzeit ein, bearbeitet diese und speichert die Ergebnisse ab. Der Business Component Interpreter interpretiert die Schlüsselwörter und erzeugt lauffähige Befehle. Der Data Component Interpreter löst zur Laufzeit die dynamischen und unmittelbaren Testdatenwerte auf. Schließlich erfolgt noch die Zuordnung der Testobjekte auf die Zielanwendung.

Definition der technischen Architektur

In der untersten Abstraktionseben werden nun die technischen Konzepte definiert. Die Erstellung der Skripte erfordert strukturierte Techniken und Richtlinien. Für die Konzeption eines technischen Architekturelementes sind hier exemplarisch die Beziehungen der Sheets dargestellt, welche die Benutzerschnittstelle definieren (siehe Abbildung 7).

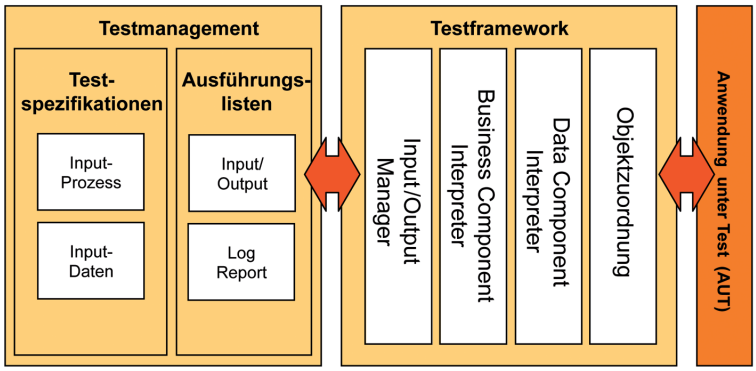

Abb. 6: Beispiel eines Testframeworks.

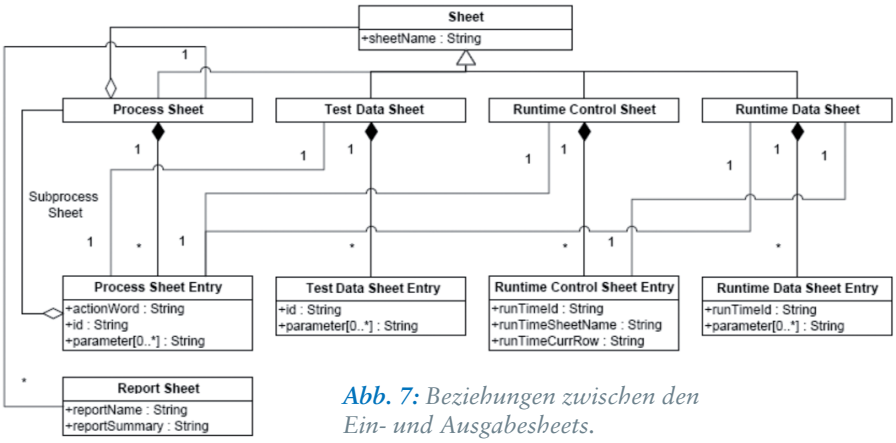

Abb. 7: Beziehungen zwischen den Ein- und Ausgabesheets.

Das *Process Sheet* (Tabellenblatt) enthält *Process Sheet Entries* (BCs). *+actionWord"* ist die technische Realisierung der Schlüsselwörter und die *+parameter* implementieren die Data Components (DCs). Daneben existieren noch einige weitere Datensheets, um die verschiedensten Testanforderungen zu realisieren. Das *Test Data Sheet* enthält *Test Data Sheet Entries* (Testdatensätze). Das *Runtime Control Sheet* enthält technische Informationen zur Verwaltung der Testabläufe, während die *Runtime Data Sheets* die Ausgabewerte zur Laufzeit aufnehmen. Im *Report Sheet* werden die Protokolleinträge und Testergebnisse geführt. Für die Programmierung der BCs und DCs empfiehlt sich eine strikt formalisierte „Inline"-Dokumentation. Aus den standardisierten Kommentaren lassen sich Wertehilfen und Füllfunktionen generieren, die den Testdesigner bei der Erstellung und Wartung von Tests mit BCs und DCs optimal unterstützen. Für die Programmierung sind natürlich alle sinnvollen und standardisierten Architekturelemente denkbar (vgl. [Few99]). Nicht vergessen werden sollten auch (Code-)Reviews und Entwicklertests.

Prozesse

Sind die Konzepte geklärt, ist für die Umsetzung ein gewisses Maß an Change Management nötig. Hier müssen die Prozesse zur Integration der Fachexperten und im Testmanagement angepasst werden.

Integration der Fachexperten

Zu Beginn eines Testautomatisierungsprojektes müssen sich die Fachexperten und Testingenieure auf gemeinsame Schlüsselwörter einigen. Der Fachbereich muss dabei einige Formalismen in Kauf nehmen, während die IT alle technischen Merkmale von der obersten Abstraktionsebene fernhalten sollte. Sind die ersten automatisierten Testfälle am Laufen, kann der Fachbereich bereits zügig neue Varianten dieser Geschäftsobjekte durch Kopieren und Editieren erstellen (siehe Abbildung 8). 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.

Testmanagement

Die rollenbasierte Definition der Arbeitspakete ist im Testautomatisierungsprozess unerlässlich. Der Fachexperte ist zuständig für das Testdesign und die Auswahl der Testfälle. Der Testautomatisierer programmiert die BCs, wartet sie und integriert die Testszenarien in den Regressionstest. Die Freigabe erfolgt durch den Testmanager, während die regelmäßige Ausführung im Zuständigkeitsbereich der Fachexperten liegt. Der Testmanager trägt die Verantwortung für den Gesamtprozess vom Testdesign bis zur regelmäßigen Ausführung.

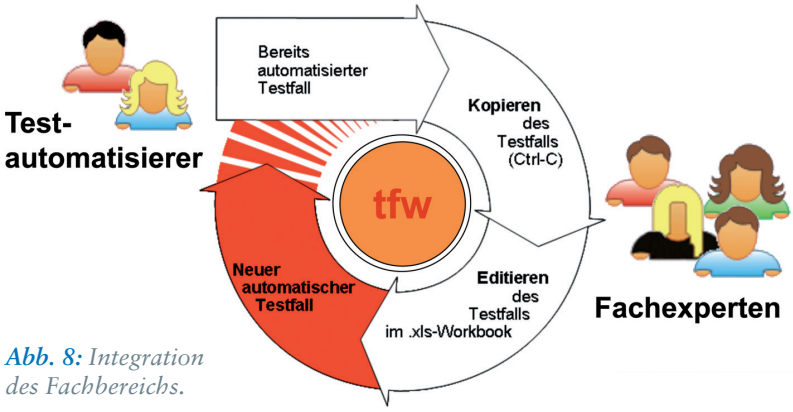

Abb. 8: Integration des Fachbereichs.

Umsetzung der Konzepte und Prozesse

Wichtiger noch als die vorangegangenen Punkte der Checkliste ist die Realisierung einer Testautomatisierung: „Just do it!". Um ein besseres Gefühl für die verschiedenen Aufgaben zu bekommen, sollte man klein beginnen, um dann die großen Projekte ebenso zu beherrschen. Ein großer Vorteil des Frameworks ist, dass es sich innerhalb kürzester Zeit sehr schnell an ein neues Projekt anpassen lässt. So dauert es meist nicht mal einen Tag, bis der erste Testfall fertiggestellt ist. Eine flächendeckende Implementierung in Großunternehmen mit zahlreichen Projekten erfordert erheblichen Abstimmungsaufwand und dauert natürlich wesentlich länger. Ein bisher noch nicht genannter Faktor ist die Besetzung der Testdesigner und der Testingenieure. Der Testdesigner benötigt fachliche Expertise, der Testingenieur Proammierkenntnisse. Um optimale Ergebnisse im Regressionstest erzielen zu können, müssen beide Rollen auch fundierte Kenntnisse des Testprozesses und der Testwerkzeuge aufbauen.

Fazit

Jedes vorgestellte Thema für sich ist ein wesentlicher Faktor, der sorgfältig berücksichtigt werden sollte, da ansonsten das Testautomatisierungsprojekt zur Kostenfalle werden kann. Die Erfolgsfaktoren wurden anhand einer Checkliste für GUI-Testautomatisierung dargestellt. Nachhaltige Ergebnisse zeigen sich insbesondere bei Unternehmen, die in der Testautomatisierung einen langfristigen Horizont haben und erstellte Testartefakte in mehreren Projekten wiederverwenden können. ■

Referenzen

[Dor] Website Dorothy Graham, siehe http://www.dorothygraham.co.uk.

[Few99] M. Fewster, D. Graham, Software Test Automation, Addison Wesley, 1999.

[Goo] Google Automation, siehe http://www.googleautomation.com.

[Gra12] D. Graham, M. Fewster, Experiences in Test Automation, Addison Wesley, 2012.