AUTOMATISIERUNG IST KEINE KÜR SONDERN PFLICHT

EIN PRAXISBERICHT

Es gibt eine große Anzahl von Tools, die die Automatisierung von Tests unterstützen. Die verschiedenen Werkzeuge haben viele Vorteile und erleichtern und beschleunigen die Testabläufe. Doch kein Tool bringt alles mit, was ein Tester braucht. Johann Gietl, Berater für Test Automation Engineering bei der Sogeti Deutschland GmbH, gibt einen Überblick über seine Arbeit, wie er Tests automatisiert und wie er sie verwaltet.

DIE STRATEGIE

Vor allem in agilen Projekten werden heute automatische Testfälle eingesetzt. Ein Test, der ohne Tester „vollautomatisch“ und beliebig oft ausgeführt werden kann, ist eine deutliche Arbeitserleichterung. Testautomatisierung ist kein Luxusgut mehr. Im Gegenteil. Sie ist heute eine essenzielle Notwendigkeit, ohne die Continuous Integration gar nicht mehr denkbar ist. Der Erfolg der Testautomatisierung ist für Firmen überlebenswichtig geworden.
Die Entwicklung einer erfolgreich automatisierten Testsuite erfolgt ähnlich wie der Prozess der normalen Anwendungsentwicklung. Sie wird als vollwertiger Bestandteil in eine übergreifende Teststrategie integriert. Ziel im ersten Sprint ist, die Anwendung im Kernprozess zu implementieren. Ein paar Sprints später sollte der automatisierte Test komplett im Design und Entwicklungsprozess umgesetzt sein. Im Idealfall ist die Strategie sehr einfach, klar und standardisiert. Ich nutze beispielsweise das Modell von Dorothy Graham [1] (Abb. 1).

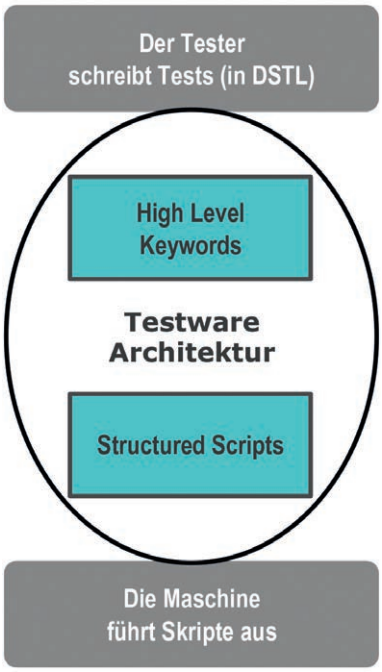


Abb. 1: Die zwei Abstraktionsebenen der Testware-Architektur

Bereits vor 20 Jahren begann man mit einem integrierten geschäftsprozessbasierten Testfall die Trennung zwischen manuellem und automatisiertem Testfall aufzuheben. Dieser Testfall kann sowohl von dem Experten des Fachbereichs als auch von dem Ingenieur für Testautomatisierung bzw. der Maschine ausgeführt und angepasst werden. Mit dieser Technik lässt sich die Produktivität bis zum Faktor 5 erhöhen.

Um die Wartung automatisierter Testfälle zu gewährleisten, reichen zwei Abstraktionsebenen. Auf der einen

Seite werden für den Tester verständliche Schlüsselwörter (High Level Keywords) innerhalb einer Domain Specific Testing Language (DSTL) definiert, um Tests anpassen und ausführen zu können. Auf der anderen Seite befinden sich die strukturierten Skripte in einer geläufigen Programmiersprache zur vereinfachten Wartung.

DIESE STRATEGIE ZU AUTOMATISIERTEN TESTS UMFASST DREI TEILE:

1) **der Tester mit den High Level Keywords,**
2) **die Testware-Architektur und**
3) **die Maschine mit den strukturierten Skripts.**

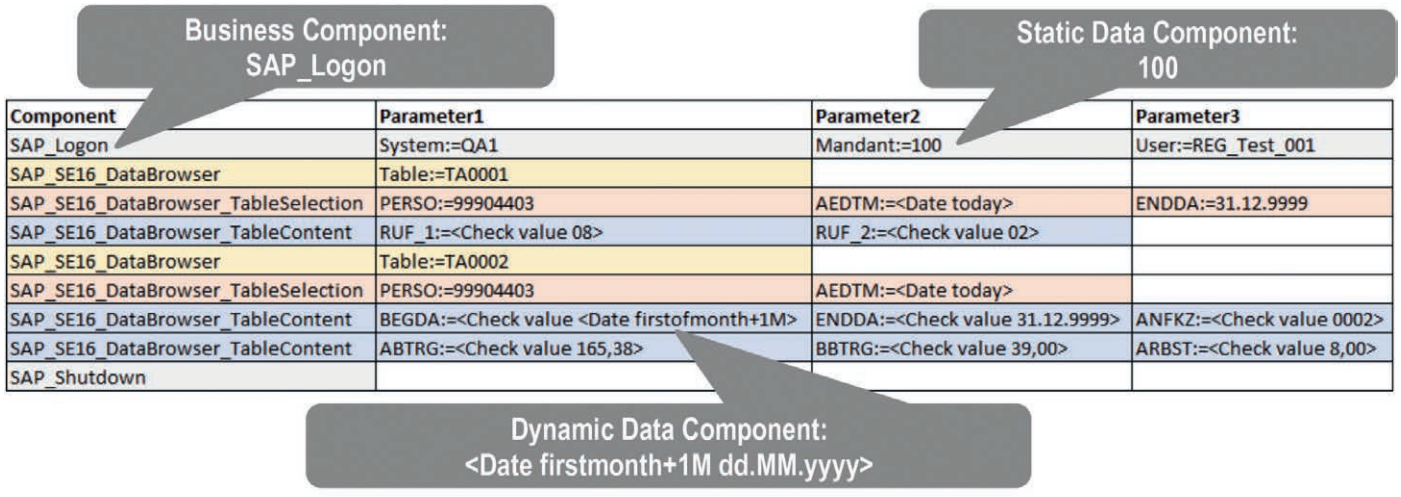
1) Der Tester

Der Tester soll den Testfall in einer domänenspezifischen Testsprache beschreiben. Bisher konnte mich noch kein Test-Tool-Hersteller mit einer proprietären Sprache begeistern. Außerdem wollte ich immer unabhängig bleiben. Microsoft Excel wird von allen Tool-Herstellern als kleinster gemeinsamer Nenner immer unterstützt und daher fiel meine Entscheidung bereits sehr früh (vor circa 20 Jahren) auf diese Software als Werkzeug für die Beschreibung von Testfällen. Ein weiterer Vorteil ist, dass es auch von den Testern in allen Branchen meist ohne weitere Installation und Schulung benutzt werden kann.
Die Form war ebenfalls ziemlich schnell klar. Wir definierten die „Business Component" als eine Excel-Zeile mit einem Schlüsselwort als Namen und mehreren „Data Components" als Parametername-Parameterwerte-Paar in den Zellen dieser Zeile. Der Parametername sollte intuitiv sein und ergibt sich zumeist direkt aus den Feldbezeichnern der zu testenden Anwendung.

In Abbildung 2 sehen wir einen grundlegenden Beispiel-Testfall, der in dieser Form seit 15 Jahren über unzählige Major Releases von Excel und SAP mit minimalem Wartungsaufwand läuft. Er enthält alle Basisfunktionen zum Ausfüllen verschiedener Dialoge. Mit fünf einfachen Business Components (SAP_...) und zwei Data Components (<Check> und <Date>), die durch wenige Zeilen Code implementiert wurden, lassen sich bereits zu einem beliebigen Zeitpunkt auf beliebig vielen SAP-Systemen sämtliche Datenbankfelder prüfen, zu denen der Tester berechtigt ist. Der Testfall ist so beschrieben, dass jeder, der jemals ein SAP R/3 System initial bedient hat, sofort mit der manuellen Ausführung starten und bei Änderungen den Testfall auch anpassen könnte.
Prinzipiell sehen alle automatisierten Testfälle so ähnlich aus. Sie können nahezu beliebig komplex ausgestaltet und erweitert werden. Zum Beispiel testet eine Fachtesterin den gesamten Produktlebenszyklus eines Rückversicherungsprogramms in einem Excel Workbook. Und ein Aktuar testet unter zusätzlicher Nutzung der Excel-Zellreferenzen regelmäßig einen monatlichen Abschluss. Nicht, dass ich solche Testfall-Monster empfehlen würde, aber es zeigt, dass mit dieser einfachen Technik mächtige Geschäftsprozesse automatisiert getestet werden können.

2) Die Testware-Architektur

Die tägliche automatisierte Bereit- und Sicherstellung der Testsysteme und des Test-Netzwerkes nahmen zu Großrechner-Zeiten einen beträchtlichen Teil der Arbeitszeit ein. Die Testautomatisierung startete erst auf der Client-Server-Architektur, auf der auch die zu testende Anwendung lief. Dies war die Zeit, in der auch der internationale Standard zur Testdoku-



Component	Parameter1	Parameter2	Parameter3
SAP_Logon	System:=QA1	Mandant:=100	User:=REG_Test_001
SAP_SE16_DataBrowser	Table:=TA0001		
SAP_SE16_DataBrowser_TableSelection	PERSO:=99904403	AEDTM:=<Date today>	ENDDA:=31.12.9999
SAP_SE16_DataBrowser_TableContent	RUF_1:=<Check value 08>	RUF_2:=<Check value 02>	
SAP_SE16_DataBrowser	Table:=TA0002		
SAP_SE16_DataBrowser_TableSelection	PERSO:=99904403	AEDTM:=<Date today>	
SAP_SE16_DataBrowser_TableContent	BEGDA:=<Check value <Date firstofmonth+1M>	ENDDA:=<Check value 31.12.9999>	ANFKZ:=<Check value 0002>
SAP_SE16_DataBrowser_TableContent	ABTRG:=<Check value 165,38>	BBTRG:=<Check value 39,00>	ARBST:=<Check value 8,00>
SAP_Shutdown			

Abb. 2: Ein Testfall mit High Level Keywords als Structured Script Components realisiert

Johann Gietl verfügt als iSQI Quality Assurance Management Professional über tiefe Kenntnisse in der Testautomatisierung und im Testmanagement. Seit 2011 ist er für Sogeti Deutschland als Berater für das Test Automation Engineering tätig.

mentation (IEEE 829) entstand, der bis heute die Grundlage für das Testen und damit auch für das automatisierte Testen bildet. Bei den Standards zur Testautomatisierung hat sich in den letzten Jahren eine Bewegung aus dem ISTQB herauskristallisiert, die sich wohl in den nächsten Jahren global durchsetzen wird. Im Kern des Syllabus für den Test Automation Engineer steht die „general Test Automation Architecture (gTAA)". (Abb. 3) [2][3]

Die gTAA bildet eine sehr gut strukturierte Basis für den Aufbau eines konkreten Systems zur Testautomatisierung. Das Framework beinhaltet immer implizit oder explizit die vier Ebenen Testdefinition, Testausführung, Testadaption und Testgenerierung.
Nach Record/Playback und Data-Driven Tests, haben wir vor ca. 15 Jahren sehr schnell begonnen, die Testgenerierung und die Testdefinition an MS Excel zu übergeben, das heute gar nicht mehr als Testautomatisierungstool wahrgenommen wird. Die Entscheidung für Excel im Test Definition Layer gab uns die Möglichkeit, das Testtool von Microfocus UFT einfach nach LeanFT bzw. Selenium zu migrieren. Die konkrete Architektur gab uns dabei die Übersicht, welche Komponenten ersetzt oder angepasst werden mussten.

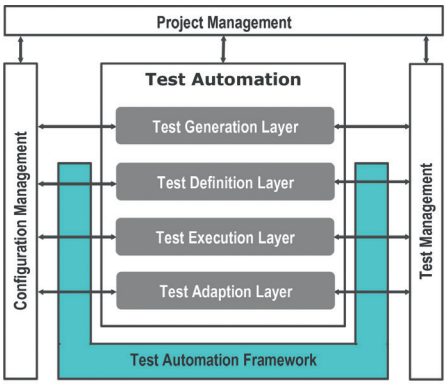


Abb. 3: Test Automation Framework innerhalb der „general Test Automation Architecture" nach ISTQB

3) Die Maschine

Strukturierte Skripte sind die Grundlage für eine stabile und gleichzeitig flexible Maschine. Da wir unseren Test Definition Layer bereits an Excel übergeben haben und den Test Adaption Layer unserem Testautomatisierungstool überlassen, müssen wir nur noch die Business- und Data-Components als einzelne einfache Funktionen im Test Execution Layer realisieren. Im günstigsten Fall können wir damit den Test der Standardanwendungen praktisch „scriptless" automatisieren.

Die dargestellten Komponenten in Abbildung 4 sind hier nur ein Vorschlag und können separat nach Bedarf im Automatisierungstool implementiert oder an andere Werkzeuge übergeben werden.

Testtool-Hersteller können einem all die Arbeit abnehmen, aber dennoch sollten alle wichtigen Aspekte und deren Umsetzung sorgfältig geprüft werden. Sind die Lizenz- und Service-Verträge einmal unterzeichnet, kann es bei Eigenentwicklungen manchmal sehr lange dauern, bis eine Erweiterung oder Anpassung geliefert wird. Es hat sich in unseren Projekten bewährt, Werkzeuge wie Jenkins oder GitHub, die bereits im Projekt von den Entwicklern genutzt werden, mit zu nutzen. Zumeist handelt es sich zudem bei der Testautomatisierung um einige minimal-invasive Eingriffe in die bereits vorhandene Projekt-Infrastruktur.

FAZIT

Eine einfache Strategie mit wartbaren Testfällen, einer klaren Testware-Architektur mit ultimativ standardisierten Werkzeugen wie z. B. Excel und sauber strukturierte Skripts sind die wesentlichen Faktoren für eine effiziente Testautomatisierung. Der ISTQB Syllabus und die Test Automation Patterns geben die Richtung vor und der

Test Automation Engineer befindet sich auf der sicheren Seite, um schnell und erfolgreich die Anwendungsentwicklung unterstützen zu können. ■

REFERENZEN:

[1] Experiences of Test Automation, Graham / Fewster, Addison-Wesley Professional, 2012

[2] Certified Tester Advanced Level Syllabus Test Automation Engineer, Working Group, 2016

[3] Test Automation Engineer, Pollner / Fewster / Schieferdecker, rockynook 2019

[4] A Journey through Test Automation Patterns, Gamba / Graham, Amazon Fulfilment, 2018

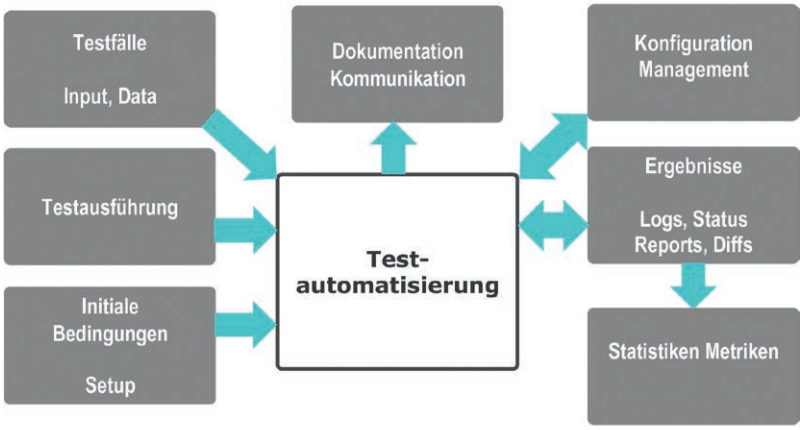


Abb. 4: Komponenten im Test Automation Framework frei nach Seretta Gamba [4]

Die Zertifizierung zum ISTQB® Test Automation Engineer können Sie auch bei iSQI erlangen.
MEHR INFORMATIONEN FINDEN SIE HIER: *isqi.org/de/istqb-certified-tester-test-automation-engineer#*