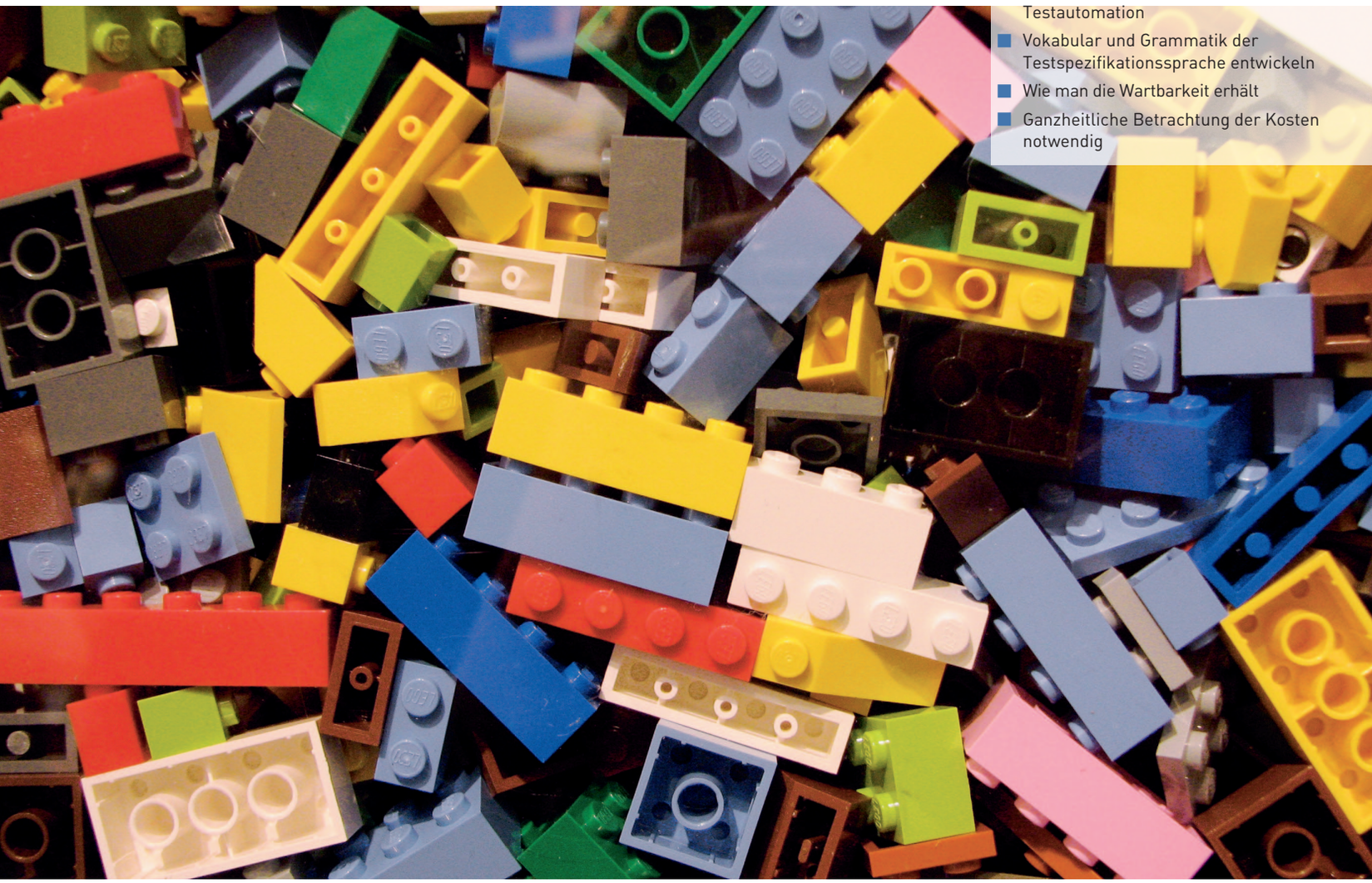
In diesem Beitrag:
- Das Warum und das Wie der Testautomation
- Vokabular und Grammatik der Testspezifikationssprache entwickeln
- Wie man die Wartbarkeit erhält
- Ganzheitliche Betrachtung der Kosten notwendig



Testautomation für eine hochparametrierbare, geschäftskritische Standardsoftware

Adcubum hat für seine Standardsoftware Syrius eine Testautomation aufgebaut. Die Anforderungen waren, einerseits über die zunehmende Anzahl Releases und Kunden zu skalieren und andererseits sowohl der hohen Parametrierbarkeit als auch der grossen Entwicklungsdynamik Rechnung zu tragen.

Infos zum Autor

Andreas Kellerstrass Dipl. Math. (TU Darmstadt),

17 Jahre Erfahrung in Softwareentwicklung, Projektmanagement und IT-Beratung. Seit 8 Jahren intensive Beschäftigung mit Testing, insbesondere mit der Umsetzung in Grossprojekten. Seit 2010 bei der Adcubum AG für QA und Testing verantwortlich

Mit der Etablierung der Standardsoftware Adcubum Syrius in der Versicherungswirtschaft hat die Zahl der Kunden und damit auch der Releases zugenommen. Um die Qualität der Releases sicherstellen zu können, hat die Adcubum eine Testautomation aufgebaut.

Das übliche Vorgehen beim Start einer Testautomation zielt zumeist primär darauf, möglichst schnell einen Anfangserfolg zu erzielen und viele, wirksame Tests zu implementieren. Damit begegnet man der Kritik an den hohen Investitionskosten wirkungsvoll und sichert sich die Unterstützung des Managements. Die Ernüchterung folgt dann meist ebenso schnell, wenn die Grenze der Wartbarkeit erreicht ist und die laufenden Kosten das Investment bei Weitem übersteigen. Zentraler Anspruch bei Adcubum war neben der zeitnahen Wirksamkeit, die Wartbarkeit in Zukunft zu sichern und das hohe Investment langfristig zu schützen.

Der Keyword- und Datadriven-Ansatz ist ideal für den Test von Standardsoftware

Methodisch stützt sich die Testautomation bei Adcubum auf einen Keyword- und Datadriven-Ansatz ab. Dieser erweist sich optimal für den Test von Standardsoftware. Gemäss diesem Ansatz zerfällt die Testspezifikation in eine Testablaufdefinition und die Definition des eigentlichen Testfalles. Die Testablaufdefinition ist eine Abfolge von fachlichen Aktionen (Keywords), die frei von konkreten Daten sind. Der Testfall selbst besteht aus einer Menge an Daten, die der Testablauf zur Durchführung benötigt (Datadriven). Testfälle sind entsprechend konkrete Ausprägungen eines generischen Testablaufs. Wesentlich für eine maximale Flexibilität und Wiederverwendbarkeit ist: In der eigentlichen Automation werden nicht mehr Testfälle oder Testschritte implementiert, sondern nur noch die Keywords sowie der allgemeine Prozess der Ablaufsteuerung.

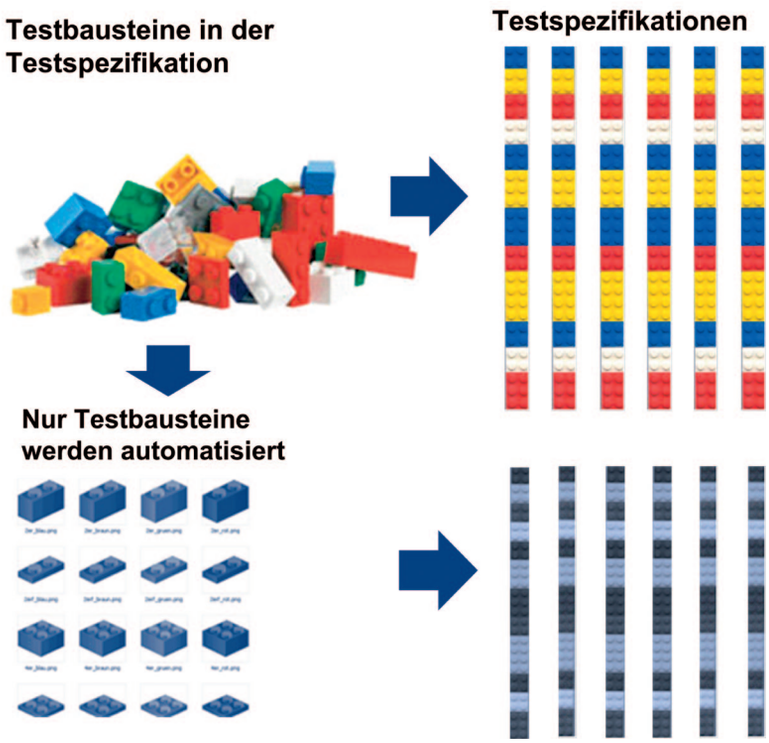
Viele Testautomationen leiden darunter, dass die Tests entweder zu technisch sind und von einem Fachspezialisten nicht mehr verstanden werden oder aber fachlich nicht hinreichend scharf beschrieben vorliegen, um ohne entsprechendes Fachwissen automatisiert zu werden. Mit diesem Defizit macht die gewählte Methode Schluss: Essentiell ist, dass die Keywords mit einem sprechenden Namen für die dazugehörige fachliche Aktion versehen werden und so die Testspezifikation auch vom Fachbereich gut geschrieben werden kann. Und andererseits ist die Testspezifikation so formal und konkret, dass sie alle Informationen für eine unmittelbare Automation enthält.

So weit entspricht das Vorgehen von Adcubum noch der Standardmethodik. Aber auch ein Keyword- und Datadriven-Ansatz ist nur eine Methode, die adaptiert werden muss und man kann auch hier sehr schnell in den Zustand der Unwartbarkeit geraten.

Richtige Testspezifikationssprache als Erfolgsfaktor

Das übliche Aufzeichnen der Keywords führt zwar zu schnellem Erfolg, ist aber längerfristig unflexibel und lässt die Menge an Keywords – also das Vokabular der Testspezifikation – explodieren. Adcubum ist deshalb den Weg gegangen, die Keywords hierarchisch auf Basis generischer Grundaktivitäten eines Benutzers zu definieren; Beispiele sind «Navigation im Menü» oder «Abfüllen aller Daten eines Business-Objektes für die Eingabefelder auf einer Seite, die zu finden sind». In Folge ist bislang nur eine minimale Menge von zirka 300 atomaren Keywords entstanden. Nur diese waren einmalig in der Automation zu implementieren. Ein möglichst kleines Grundvokabular ist entscheidend für die Wartbarkeit.

Entscheidend ist auch eine flexible Grammatik. Um die geforderte Skalierbarkeit zu erreichen, wurden «dynamische Daten» eingeführt. Die Testspezifikation eröffnet beispielsweise die Möglichkeit, Daten in Variablen zu speichern und für Sollergebnisprüfungen gegen Ende wieder zu verwenden, Muster für Vergleiche von Fehlermeldungen zu definieren oder statt der Solldaten, Formeln zu ihrer Berechnung zu hinterlegen. Als besonders nützlich hat sich dabei im Laufe der Zeit die Möglichkeit erwiesen, zur Laufzeit Datenbankabfragen auszuführen, um geeignete Daten (Preconditions) zu suchen, auf denen ein Testfall



Tests spezifizieren ist wie Lego bauen, die Tests sind ad hoc automatisiert, wenn die einzelnen Bausteine automatisiert sind.

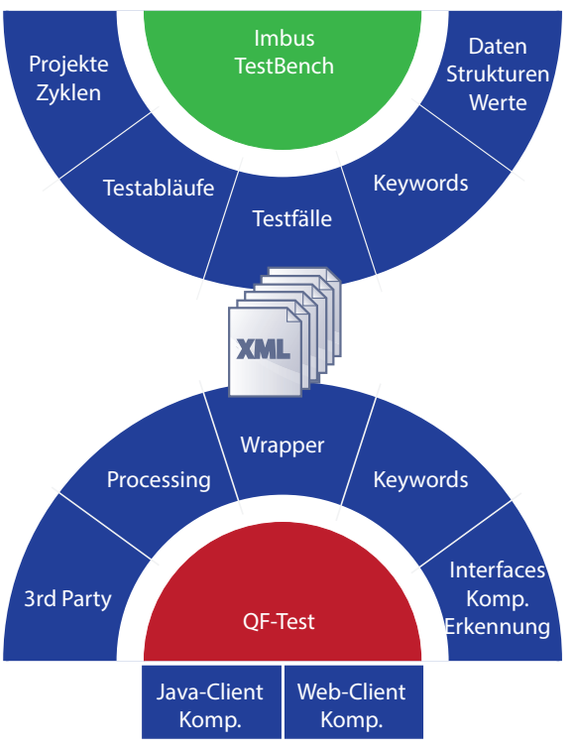
aufsetzen kann. Dieser Mechanismus erlaubt nicht nur, den Datenverbrauch während eines Testlaufes zu kontrollieren, sondern auch sich von synthetischen Daten freizumachen und – mit ein und dem gleichen Testablauf – auf Echtdaten zu operieren. Darüber hinaus ist eine lose Kopplung von einzelnen Testfällen zu ganzen End-to-End-Tests einfach möglich.
Es ist leicht einzusehen, dass die Entwicklung von Vokabular und Grammatik dieser Testspezifikationssprache iterativ entwickelt werden muss. Sie sollte sowohl zum Testobjekt als auch zu den Menschen passen, die die Tests erstellen und pflegen müssen. Die Initialisierungsphase muss entsprechend den Aufbau der ersten Testfälle nutzen, um Vokabular und Grammatik zu entwickeln. Auftretende Designfehler lassen sich frühzeitig korrigieren, bevor nennenswerte Mengengerüste entstanden sind. Bei Adcubum hat diese Phase rund ein Jahr gedauert. Inhaltlich wurde darin ein Grundtest mit rund tausend Testabläufen implementiert. In dieser Phase braucht es zugegeben Kraft und Durchhaltevermögen aller Beteiligten.

Testautomation sollte von Fachlichkeit befreit sein

Durch die Entwicklung einer formalen Testspezifikationssprache ist die Grundlage vorhanden, den Testautomaten auf seine Kernfunktion als Zugriffsschicht für die zu testende Anwendung zu reduzieren – konkret auf die Objekterkennung und die eigentliche Testausführung. So wie eine gute Anwendungsarchitektur die Zugriffsschicht von Fachlichkeit frei hält, gehören fachliche Informationen zu einem Test ausschliesslich in die Testspezifikation.
Um das Automationswerkzeug prinzipiell austauschbar zu machen, hat Adcubum den Weg eingeschlagen, die Implementierung der generischen Keywords aus der Testspezifikation und die Implementierung der Testablaufsteuerung weitestgehend ausserhalb des Werkzeugs durchzuführen. Technologisch wurden dafür auf Java und Groovy gesetzt, mit dem Effekt, dass die Automation weitgehend auf kostspielige Werkzeugspezialisten verzichtet und durch jeden Entwickler unterstützt werden kann.

Es müssen nicht immer die schwergewichtigen Werkzeuge sein

Das Testframework von Adcubum basiert in der Testspezifikation auf Testbench des Lösungsanbieters Imbus und die eigentliche Automation auf QF-Test der Softwareschmiede QFS. Beide Hersteller sind KMU und bieten einen flexiblen Support. Die Lizenzkosten sind gegenüber schwergewichtigen Werkzeugen moderat. Im Gegenzug decken die Werkzeuge auch nur genau das ab, was Adcubum tatsächlich benötigt.



Testspezifikation und Testautomation reduzieren die eingesetzten Werkzeuge auf ihre Grundfunktionalität.

Einhaltung der Grundprinzipien guter Anwendungsarchitekturen garantiert Wartbarkeit

Die bislang automatisierten Tests laufen bei Adcubum täglich über Nacht auf unterschiedlichen Releases und Plattformen. Der Aufwand für die laufende Pflege der Tests hat sich, entgegen allen Befürchtungen, die während der Initialisierungsphase diskutiert wurden – und derer gab es viele, als recht gering herausgestellt. Sie liegen aktuell bei zirka 20 Prozent der

DIE ADCUBUM-GRUPPE

Adcubum ist Hersteller führender Schweizer Standardsoftware für die Assekuranz. Das Kernprodukt Adcubum Syrius ist ein modular aufgebautes Bestandsführungssystem für Kranken-, Unfall- und Sachversicherungen, basierend auf der Java-Enterprise-Edition-Technologie (Java EE). Der Grundstein für die erfolgreiche Unternehmensgeschichte wurde 1998 in St. Gallen gelegt. Seitdem wird das Produkt kontinuierlich entlang der Kunden- und Markterfordernisse weiterentwickelt. Das Unternehmen beschäftigt derzeit mehr als 190 hochqualifizierte Fach- und IT-Experten. Mehrere Millionen Versicherte mit über 30 Millionen Leistungsfällen werden mit der Kernlösung Adcubum Syrius verwaltet – Tendenz steigend.
www.adcubum.com

Neuerstellung pro Jahr. Dies ist letztlich das Ergebnis strikter Designvorgaben und diverser Refactorings, sowohl bei der Testspezifikation wie auch in der Automation. Letztlich handelt es sich in beiden Disziplinen um eine Form der Programmierung und entsprechend gelten auch hier die gleichen Grundprinzipien guten Designs. Wird zu Beginn nicht konsequent auf die Einhaltung der Grundregeln guten Programmierens geachtet, hat dies katastrophale Auswirkungen auf die Wartbarkeit. Dass dies zu Lasten der anfänglichen Automatisierungsgeschwindigkeit geht, führt zugegeben zu einem gewissen Rechtfertigungsdruck.

Ganzheitliche Betrachtung des Business Case

Die Investitionskosten beim Aufbau einer Testautomation sind recht hoch, nach dem Empfinden des Managements zu hoch, da der Test vordergründig nichts produziert, was man verkaufen könnte. Qualitätsschäden und Qualitätsverbesserung sowie Testkosten müssen deshalb ganzheitlich betrachtet werden.

Zudem liegt der ROI üblicherweise nicht im Bereich von 2 bis 3 Jahren, sondern eher bei 4 bis 6 Jahren. Die Erstellung und Erklärung des Business Case ist entsprechend aufwändig und langwierig.

Sicher sind die Business Cases je nach Unternehmenskontext und Startbedingungen ganz unterschiedlich. Rückblickend lässt sich aber sagen, dass folgende Aspekte der Diskussion bei Adcubum vermutlich allgemeine Gültigkeit haben.

1. Der Schaden eines Fehlers muss End to End und abhängig von der Kritikalität betrachtet werden und nicht nur reduziert auf die Fehlerbehebungskosten der Entwicklung oder pauschal. Entsprechend gehören Aufwände im Support und Eskalationen an die Geschäftsleitung ebenso dazu wie Mehraufwand für wiederkehrende Bugs.
2. Der Beitrag einer Testautomation liegt nicht nur in den Fehlern, die sie selbst findet. Sie entlastet die weiteren Testbeteiligten und ermöglicht eine Ausweitung der Testabdeckung insgesamt. Entsprechend muss der Nutzen an der Erhöhung der Fehlerfindungsrate des Tests und die Qualitätssteigerung auch insgesamt gemessen werden. Jeder vor Auslieferung gefundene Fehler trägt dazu bei, egal, ob er im manuellen Test gefunden wurde oder vom Automaten.
3. Die Kosten der Testautomation können nicht isoliert betrachtet werden, sondern müssen in Zusammenhang mit allen weiteren Testkosten gebracht werden. Eine Testautomation reduziert gegebenenfalls den Aufwand manueller Tests oder erhöht den Wirkungsgrad insgesamt.
4. Kein Business Case oder alternative Szenarien. Ein Vergleich von Kosten und Nutzen der Testautomatisierung mit einem Testoutsourcing ist fast schon obligatorisch, wenn auch erstaunlich aufwändig. Für eine ergebnisoffene Diskussion mit potentiellen Serviceanbietern und die Begutachtung der Offerten muss man viel Zeit einplanen.

Die Kostendiskussion ist das Schwierigste und Langwierigste beim Aufbau einer Testautomation. Trotz eines grundlegenden Konsenses war es ein mehrmonatiger Entscheidungsprozess, der von intensiven Nutzendiskussionen geprägt war. Vielen Entscheidern wurden die Gesamtkosten der Tests und die durch Qualitätsmängel hervorgerufenen Schäden das erste Mal durch die ganzheitliche Betrachtung vor Augen geführt. Hinreichender Durchhaltewille ist erforderlich, damit auch solche Diskussionen letztendlich erfolgreich enden. □