GUI Testautomatisierung – 50% Erstellungsaufwände sparen, wenn man Anforderungen analysiert

Version: 21. November 2018
Jörg Sievers
PONTON GmbH
E-Mail: sievers@ponton.de

Abstract: Wenn man heute Tests für Weboberflächen automatisieren möchte, ist die Antwort meistens Selenium. Es ist lizenzkostenfrei und weit verbreitet. Das bekannte Vorgehen, ein technisches Mittel anhand der Anforderungen zu wählen, wird einfach weggewischt, da der Lizenzkostenfaktor in den Vordergrund der Überlegungen gestellt wird. In einem Projekt haben wir verschiedene GUI Test-Automatisierungstools geprüft und ermittelt welches zu dem Entwicklungsvorgehen, den handelnden Personen und deren gestellten Aufgaben am besten passen könnte und kamen zu einem anderen Entschluss, der dem Projekt ca. 50% der Aufwände gegenüber Selenium spart und dazu geführt hat, das das Tool nun unternehmensweit Anklang gefunden hat.

Im Unternehmen sind ausreichend Kompetenzen in Sachen GUI-Testautomatisierung mit Selenium vorhanden, jedoch waren die Aufwände, die für die Erstellung und Wartung der Tests aufgewendet wurden, zu hoch. Die Entwickler empfanden die Tests nicht als Unterstützung zur qualitativen Beurteilung und Nachweis ihrer Arbeit, sondern eher als „Klotz am Bein“. Auch andere Projekte inner- und außerhalb der Firma klagten über ähnliche Probleme, die der Einsatz von Selenium mit sich bringt:

- deterministische Ausführung auf unterschiedlichen Systemen (lokal beim Entwickler und auf dem kontinuierlichen Buildservern (CIS) teilweise mit nicht nachvollziehbaren, immer unterschiedlichen Ergebnissen
- AJAX-Behandlung musste oft mit Hilfsfunktionen durch die Entwickler realisiert werden (u.a. Nutzung der sog. *impliziten*, *expliziten* und *fluent wait()*-Funktionen [1])
- Apache *Wicket* generiert eigene ID’s, die es dem Entwickler, der nicht unbedingt der Ersteller der Funktionalität war, nur mittels Zusatztools wie *Firebug* die „Accessoren“ (*CSS*- oder *XPATH*-Pfade) ausfindig zu machen.

Im Rahmen eines Praktikums erstellte ein Student aus dem Fachbereich Informatik der Universität Hamburg im Sommer 2015 einen Anforderungskatalog und ermittelte wie andere Tools im Vergleich zu Selenium abschneiden. Er setzte dabei eine Bewertungsmatrix aus Erfüllungsgraden und deren Gewichtung (1 (schlecht, unwichtig) bis 10) ein. In die Anforderungskriterien flossen Dinge ein, wie z.B.

- Generelle Konfigurierbarkeit des Tools
- Möglichkeit zur Verwaltung der Testfälle
- Höhe der Anforderung an Tester / Entwickler um das Tool effizient zu nutzen
- *Bamboo*-, *Maven*- *GIT*-Integration
- Erkennungsrate von *HTML*- und *JavaScript*-Objekten
- Unterstützung von *AJAX*, *jQuery*
- Robustheit / Stabilität der Tests
- Unabhängigkeit der Code-Struktur von der Anwendung
- uvm.

Am Ende kamen zwei Tools in die engere Auswahl, die Selenium sehr deutlich überlegen waren (Selenium erzielte 1043, die anderen beiden Tools 1750 und 1676 Bewertungspunkte), wovon wir eines der Tools schon in einem anderen Projekt einsetzten und der Zuschlag dann für eine Pilotphase an dieses Tool fiel.

Drei Jahre später wird das damals auserkorene Tool nun durchgängig in weiteren Projekten eingesetzt und ist auf dem Weg von den Entwicklern in den generellen unternehmensweiten Entwicklungsprozess eingebunden zu werden, wie zuvor noch kein anderes Testtool dieser Art.

Zur Bewältigung der Aufgabe „die Qualität sicherzustellen“ wird das Testtool als Hilfe verstanden und nicht als notwendiges Übel. In dem ersten Projekt, wo auch die Anforderungsanalyse stattfand, wurde das Tool sogar beim Kunden für kleinere Tagesgeschäftsaufgaben verwendet, da es zuverlässig seine Aufgaben erfüllte und auch von den dortigen

Fachabteilungen schnell verstanden wurde. Wodurch ist diese positive Trendwende entstanden?

- **Lernkurve:** Selbst für neu hinzugekommene Mitarbeiterinnen und Mitarbeiter ist die Einarbeitungszeit sehr gering, da das Tool wurde aus QA- und nicht aus Entwicklersicht entwickelt worden ist. Anwenderszenarien lassen sich sehr gut abbilden, da alltägliche Dinge, wie Up- und Download, Vergleich von *Microsoft Excel*- oder *CSV*-Dateien sehr einfach zu lösen sind.
- **Wiederverwendbarkeit**: Mir ist keine Version bekannt, die eine komplette Umstellung der Testbasis nach sich gezogen hätte. In der Regel kann eine neue Version des Testtools sehr schnell, nach einer Prüfung bestehender Tests, eingesetzt werden.
- **Wartbarkeit**: Dadurch das nicht *XPATH*- oder *CSS*-Pfade zur Accessoren-Ermittlung, sondern eine Eigenentwicklung, unabhängig vom ausgelieferten Source, benutzt werden, können Umbauten, die keine fachlichen Konsequenzen („es muss alles so funktionieren wie vorher“) nach sich ziehen, durchgeführt werden. Beispiel: Das Tool betrachtet eine Tabelle weiterhin als eine Tabelle, auch wenn sie jetzt evtl. anders in der Seite verankert wurde.
- **Übertragbarkeit**: Ein Ärgernis von Selenium sind die unterschiedlichen Ergebnisse auf kontinuierlichen Integrationssystemen (*CIS*) und auf der lokalen Entwicklermaschine. Das ausgewählte Tool erzielt auf dem *CIS* zumeist dieselben Ergebnisse, wie auch auf dem lokalen Rechner. Das Analysieren von Fehlern ist so deutlich zeitsparender.

In dem Pilotprojekt wurden die Zeiten, die für die Erstellung von automatisierten GUI-Testfällen aufgewendet worden sind, um ca. 50% reduziert. Selbst komplexeste Aufgabenstellungen, wie das komplette Rücksetzen einer Datenbank auf den definierten Testzustand, wurden nach der Pilotphase im automatischen Integrationsprozess gelöst. Wiederkehrende, bei jeder Abnahme manuell getestete Anwendungsszenarien, die mit Selenium aufgrund nicht-deterministischer Ausführung und der hohen Erstellungs- und Wartungsaufwände, nie automatisiert wurden, sind mithilfe des neuen Tools in akzeptabler Zeit umgesetzt worden und reduzieren die Auslieferungszeit.

Unser Praktikant schrieb: Im Buch [2] „(…) werden alle wichtigen Grundsätze und Prinzipien der Testautomatisierung und des entwickeln von Testskripten erläutert.“ [3] Graham und Fewster haben vor knapp 20 Jahren schon in ihrem Kapitel 10 „Choosing a tool for automate testing“ [2] beschrieben, wie man ein Tool auswählt und das nicht der Tool-Markt, sondern die Anforderungen des Projektes darüber entscheiden sollten, welches Tool man einsetzt.

Nicht weil ein Tool lizenzkostenfrei ist, muss es die beste Lösung für das jeweilige Projekt sein. Zudem war das Lizenzkostenmodell des ausgewählten Tools fair und eindeutig. Da in unserer Firma die Projekte allesamt *B2B*-Projekte und die meisten sehr ähnlich gelagert sind, kann es u.U. auch dazu führen, dass ein Tool unternehmensweit Akzeptanz findet.

Literaturverzeichnis

[1] A. Ghahrai, „WebDriver Implicit, Explicit and Fluent Wait Examples,“ 23 10 2018. [Online]. Available: https://www.testingexcellence.com/webdriver-explicit-implicit-fluent-wait/. [Zugriff am 19 11 2018].

[2] D. Graham und M. Fewster, Software Test Automation, Addison-Wesley, 1999.

[3] P. Bischof, „Praktikumsbericht,“ 2015.