Sebastian Stephan

(sstephan@proficom.de)
leitet seit 2014 die Softwareentwicklung der profi.com AG. Die tägliche Herausforderung des Diplom-Informatikers besteht darin, den Lebenszyklus von Anwendungen von Anfang bis Ende zu betrachten und mit den entsprechenden Werkzeugen das Leben von Entwicklern und Testern leichter zu machen.

Rückenwind für Entwickler und Tester
Ein praktischer Ratgeber zur Testautomatisierung

Open-Source- oder Enterprise-Lösung? Funktionsvielfalt versus minimalistischer Umfang? Kostenfrei beziehungsweise lizenzpflichtig? Wozu überhaupt Softwaretests automatisieren? Der Artikel vergleicht wesentliche Features unterschiedlichster Testautomatisierungswerkzeuge und gibt Empfehlungen, für welchen Einsatzzweck sich welches Tool eignet.

Softwareentwicklung *ohne* Testautomatisierung ist wie Surfen *ohne* Wind. Etwas Entscheidendes fehlt und man kommt nicht richtig vorwärts. Dabei besitzen automatisierte Tests nicht nur den Charme der maschinellen Abarbeitung im Vergleich zu manuellen Tests; vielmehr noch zählt das Potenzial für deutliche Effizienzsteigerungen. Egal ob innovatives Start-up-Unternehmen oder global agierender Konzern – der Einstieg in die Testautomatisierung lohnt sich jederzeit.

Testautomatisierung in der Softwareentwicklung ist mittlerweile ein akzeptierter Standard und wird heute mehr denn je eingefordert.

Bei den Testinvestitionen ist und bleibt der Fokus auf der Testautomatisierung, wie der „Testing Trends & Benchmarks Report 2013“ der SwissQ Consulting AG [SwissQ13] herausgefunden hat. Denn nirgendwo sonst lassen sich vermeintlich leichter Kosten einsparen. Im Vergleich zu manuellen Softwaretests bis zu 73 Prozent:

„Automating the testing process delivered significant benefits to JetBlue, allowing the company to reduce its testing costs by 73 %, and its post-production failures by 80 %.“ Quelle: Hewlett Packard Case Study [hp13]

Kurze und zur Entwicklung synchrone Testzyklen sind die gebräuchlichste Methode, um über stetige Iterationen die Qualität der zu entwickelnden Software zu steigern. Das kann manuell erfolgen – mit einer wahren Heerschar an Testern, wie zum Beispiel beim Crowd-Testing, oder eben automatisiert. Besonders bei agilen Entwicklungsmethoden wie Scrum, Kanban oder Feature Driven Development (FDD) sind automatisierte Tests Pflicht, um die versprochenen Geschwindigkeitsvorteile in der Entwicklung zu erreichen.

Ist Testautomatisierung also die eierlegende Wollmilchsau? Die Antworten darauf sind zwiespältig: Ja und nein. Je nachdem. Kommt ganz darauf an.

Die Qual der Wahl

Mehrere etablierte Automatisierungs-Tools buhlen um die Aufmerksamkeit der Entwickler und Qualitätssicherungsmanager. Und vielversprechende Newcomer versuchen, den Markt mit kostenfreien Werkzeugen zu erobern. Derzeit existieren mehr als 50 unterschiedliche Lösungen, vom hilfreichen Add-on über kleine spezialisierte Open-Source-Tools bis zu mächtigen vollintegrierten Werkzeugen, wie ein aktueller Tool-Überblick ([Grei13]) zeigt. Eine umfassende Marktbetrachtung kann und soll es an dieser Stelle nicht geben, sondern einen Vergleich von ausgewählten Werkzeugen für Testautomatisierung mit dem .NET-Framework von Mircosoft.

Thomas Jähnig, Leiter des Geschäftsbereichs Applications bei der profi.com AG, sagt: „Aus der Vielzahl miteinander konkurrierender Testautomatisierungs-Tools dasjenige herauszufinden, das zu den Rahmenbedingungen des Projektes passt, gleicht oft der sprichwörtlichen Suche nach der Nadel im Heuhaufen.“ Doch nach welchen Kriterien kann eine entsprechende Evaluierung erfolgen?

Worauf Sie Wert legen sollten

Mit Testautomatisierung erreichen Unternehmen eine höhere Testbreite und -tiefe in derselben Zeit bei gleichzeitig geringeren Kosten gegenüber manuellen Überprüfungen. Daher werden schon in der Anforderungsanalyse die Grundlagen für die spä-

Abb. 1: Diese vier Werkzeuge wurden verglichen

tere Testautomatisierung gelegt, indem Testszenarien aus den Anforderungen abgeleitet werden – kleinteilig, modular und wiederverwendbar. Der Vorteil: Es werden Testfälle erstellt, die spätere Regressionstests einfacher machen, indem sie neue Fehler in bereits getesteten Teilen der Software vermeiden. Hier gilt die einfache Regel: Tests mit niedrigem Aufwand und hohem Wert sollten zuerst automatisiert werden. Umgekehrt bedeutet dies, dass Tests mit hohem Aufwand und geringem Nutzen erst spät, beziehungsweise gar nicht, automatisiert werden sollen.

Im vorliegenden Vergleich wurden die spezifischen Anforderungen und Erwartungen von Testern als Grundlage genommen. Dazu zählen die einfache Bedienung und Aufzeichnungsmöglichkeiten genauso wie effektive Automatisierungsroutinen zur Wiederverwendung der Testfälle. Deshalb wurden die Automatisierungswerkzeuge in Bezug auf folgende vier Kriterien untersucht und bewertet.

Capture & Replay bezeichnet das einfache Aufzeichnen von Abläufen zur späteren Verwendung als Test.

Code-Generierung erzeugt aus den aufgezeichneten Tests „sauberen" Code, der noch angepasst und erweitert werden kann. Als Maßstäbe für sauberen Code wurden unter anderem eine statische Code-Analyse mit dem kompletten Microsoft-Regelsatz sowie eine subjektive Messung nach dem internen Regelsatz der profi.com AG durchgeführt und die Häufigkeit der nicht eingehaltenen Regeln gezählt.

Wiederverwendbarkeit vereint mehrere Aspekte:

- *GUI-Mapping* ist die Ansammlung von Oberflächenelementen, die für den Test zur Verfügung stehen,
- *Testschritte* sind das Zusammenfassen kleinerer wiederkehrender Abläufe,
- *Parametrisierung* bedeutet, dass sich Testfälle mit verschiedenen Daten mehrfach ausführen lassen,

Build-Integration zeigt die mögliche Verknüpfung mit verschiedenen Build-Systemen allgemein, nicht nur die empfohlenen Build-Server.

Die Kandidaten

Der Vergleich von vier sehr unterschiedlichen Lösungen soll einen Weg aus dem Tool-Labyrinth aufzeigen, der es Entwicklern und Testern ermöglicht, auf bereits vorhandenen Umgebungen aufzusetzen oder sich freier Werkzeuge zu bedienen. Bei der – zugegebenermaßen subjektiven – Selektion wurden Tools ausgewählt, die das weit verbreitete .NET-Framework unterstützen. Folgende Testautomatisierungswerkzeuge (siehe **Abbildung 1**) werden verglichen:

- *Microsoft Coded UI* ist Bestandteil der Microsoft Visual Studio Premium Version.
- *HP Unified Functional Testing (UFT)* ist eine Stand-Alone-Anwendung mit Integrationsmöglichkeiten in die HP-Qualitätssicherungs-Plattform (HP ALM)
- *Ranorex* ist ein Stand-Alone-Werkzeug, das jedoch in die Entwicklungsumgebung MS Visual Studio integriert werden kann.
- *FitNesse* ist eine Stand Alone Library und Freeware, bei der Testfälle Wiki-basiert gepflegt und Testmethoden beziehungsweise Testadapter über Namen aufgerufen werden.

Das Ergebnis

Der besseren Übersicht halber lassen sich anhand einer einfachen Bewertungsmatrix die spezifischen Vor- und Nachteile der ausgewählten vier Tools aufzeigen – immer gemessen an den Hauptkriterien (siehe **Tabellen 1 bis 6**). Das Spektrum reicht von „sehr gut" (++) über „gut" (+) auf der einen Seite bis zu „durchschnittlich" (o) und „schlecht" beziehungsweise „nicht vorhanden" (-) am anderen Ende der Skala.

Zeit für Empfehlungen

Sie werden es vielleicht schon ahnen: Es gibt keinen eindeutigen Sieger in diesem

Tool	Capture	Replay
Microsoft Coded UI	**+** - ohne zusätzliche Tools - breite Unterstützung verschiedenster Technologien und Anwendungen (WPF, WinForms, XAML, Firefox, IE...)	**o** - Referenzen werden über wenige Eigenschaften der GUI-Elemente aufgezeichnet - weitere Eigenschaften zur Identifizierung können ausgewählt werden
HP Unified Functional Testing (UFT)	**+** - knapp besser als MS Coded UI, da es mehr Oberflächenframeworks unterstützt - keine zusätzlichen Tools nötig - breite Unterstützung verschiedenster Technologien und Anwendungen - mit kostenpflichtigen Add-ons auch für mobile Apps und Browser geeignet	**+** - Referenzen werden über wenige Eigenschaften der GUI-Elemente aufgezeichnet - weitere Eigenschaften zur Identifizierung können ausgewählt werden - Smartidentification kann genutzt werden, falls das Objekt im ersten Anlauf nicht gefunden wird
Ranorex	**++** - sehr breite Unterstützung, auch von mobilen Apps und Browsern - teilweise sind Plug-ins nötig (z. B. für Firefox), die jedoch kostenfrei mitgeliefert werden	**+** - Referenzierung per xPath erlaubt sichere Objektidentifikation
FitNesse	**-** - keine Capture-Funktion enthalten	**-** - keine Replay-Funktion enthalten

Tab. 1: Capture & Replay

Tool	Code-Generierung
Microsoft Coded UI	o - es wird kein sauberer Code erstellt - kaum Regelverletzungen bei statischer Code-Analyse mit Microsoft-Regeln - sehr viele Verletzungen bei statischer Code-Analyse mit internem Regelsatz der profi.com AG - Methoden pro Testschritt - ein GUI-Objekte-Repository - erstellt eine Klassenbibliothek, die über einen Testrunner gestartet wird
HP Unified Functional Testing (UFT)	- - das VBScript ist nur in HP UFT ausführbar - das Script wird zur Testlaufzeit interpretiert und ist dadurch im Allgemeinen langsamer als kompilierter Code - kein direkter Bezug zu den Oberflächenelementen, lediglich der Key des GUI-Mappings wird benutzt - aufgezeichnete Validierungen nur im Object-Repository einsehbar
Ranorex	o - es wird kein sauberer Code erstellt - kaum Regelverletzungen bei statischer Code-Analyse mit Microsoft-Regeln - sehr viele Verletzungen bei statischer Code-Analyse mit internem Regelsatz der profi.com AG - eine Methode für den ganzen Test - ein GUI-Objekte-Repository - erstellt eine Konsolenanwendung, der Testrunner startet automatisch
FitNesse	- - keine Capture-Funktion bedeutet auch, dass keine Code-Generierung möglich ist

Tab. 2: Code-Generierung

Tool	GUI-Mapping
Microsoft Coded UI	o - die Aufzeichnung einzelner Mappings ist möglich - eine Map beinhaltet auch Testmethoden sowie Testparameter - kleinste Einheit: Mapping + Testmethoden + Parameter basierend auf dem UI-Mapping - Methoden der Map im Test aufrufen - teilweise umsetzbar, Aktionen sind an das Mapping gebunden
HP Unified Functional Testing (UFT)	+ - Aufzeichnung von Shared Object Repositories (gemeinsame GUI-Mappings) ist möglich - Actions (eine Art Methode im Test) können mit weiteren Repositories assoziiert werden und deren GUI-Elemente nutzen - gute Modularisierung und Wiederverwendung möglich
Ranorex	+ - die Aufzeichnung mehrerer Mappings ist möglich - es sind beliebige Mappings pro Test nutzbar
FitNesse	o - einzelne Mappings sind frei erstellbar - Mappings können in den Test eingebunden werden (dafür muss jedoch ein eigener Adapter programmiert werden)

Tab. 3: Wiederverwendbarkeit – GUI-Mapping

Vergleich! Aber es gibt klare Empfehlungen, welches Werkzeug sich für welchen Einsatz eignet oder auch nicht. Grundsätzlich gibt es für jedes Werkzeug ideale Einsatzvoraussetzungen. Dennoch soll hier der Versuch einer Einordnung vorgenommen werden.

Microsoft Coded UI ist eine gute Wahl, falls im Unternehmen schon eine Microsoft-Plattform für kollaborative Softwareprojekte – hier der MS Team Foundation Server (TFS) - existiert. Das Werkzeug erfreut die Softwareentwickler mit vielen Features, dennoch sind crossfunktionale Fähigkeiten gefragt. Sprich, Tester sollten programmieren können oder besser - Programmierer sollten auch testen können. Eine explizite Empfehlung erhält MS Coded UI für Großprojekte, die mit MS TFS realisiert werden. Die Einbindung von Endbenutzern ist ohne technische Kenntnisse möglich und leicht umsetzbar. Für die Premium-Tools halten sich die Kosten im Rahmen, denn die Funktionalitäten sind in der Visual Studio Premium-Lizenz enthalten und die Lizenz für den TFS ist Inhalt der MSDN. Microsoft bietet auch den kostengünstigen Wechsel von Visual Studio Professional auf Visual Studio Premium, wobei die Kosten recht überschaubar sind. (Voraussetzung: Visual Studio Professional mit MSDN).

Für *HP Unified Functional Testing* spricht, dass kaum Programmierkenntnisse für die Bedienung nötig sind. Das spricht reine Testteams an, die eine strukturelle Aufteilung und Gliederung der Tests wünschen. Mit der Qualitätssicherungs-Plattform HP Application Lifecycle Managment (HP ALM) ist das Tool auch für Großprojekte geeignet. Eine Einbindung der späteren User ist über die „Keyword-driven"-Testerstellung möglich. Auch hier gilt: Premium-Anspruch kostet.

Ranorex ist eine kostengünstige Alternative oder Ergänzung zu Microsoft Coded UI. Durch seine tiefe Integration in die Entwicklungsumgebung sind entsprechende Kenntnisse nötig. Wenn mobile Anwendungen im Vordergrund stehen, ist Ranorex eine echte Empfehlung. Zudem hält sich der finanzielle Aufwand im Vergleich zu den Premium-Anbietern in Grenzen.

Dank *FitNesse* können auch kleinere Teams in effektive Testautomatisierung investieren. Das kostenlose Open-Source-Werkzeug ist ideal für die durch Akzeptanztests getriebene Softwareent-

wicklung geeignet, denn die graphische Benutzerschnittstelle kann z. B. als Testfall beschrieben werden, ohne vorhanden zu sein. Mit kleineren Einschränkungen wie dem fehlenden Capture & Replay muss man leben, dafür ist mit geringem technischem Verständnis eine gute Einbindung der User erreichbar. Allerdings: FitNesse ist nur in kleinen Projekten sinnvoll, da es sonst im Wiki schnell sehr unübersichtlich werden kann.

Wir hoffen, mit diesem Ratgeber eine Entscheidungshilfe zu geben, die einige Grundlagen der Testautomatisierung erklärt und so dazu beiträgt, dass sich mehr Unternehmen trauen, in eine sinnvolle Testautomatisierungsstrategie zu investieren. Denn eines bleibt, egal welches Tool letztlich angeschafft wird: Entwickler und Tester freuen sich über den zusätzlichen Rückenwind durch Testautomatisierung und surfen dadurch schneller und mit mehr Elan zum nächsten Release. ■

Tipps aus der Testautomatisierungspraxis

Testautomatisierung eignet sich besonders für Regressionstests, da diese häufig ausgeführt werden und die Qualität kontinuierlich gemessen wird (Continuous Testing/Continuous Integration).

Testautomatisierung eignet sich auch zum Nachstellen von Fehlern, bevor diese behoben sind.

Neben klar definierten Testfällen sollten immer auch manuelle explorative Tests durch Kenner der Anwendung erfolgen, um möglichst viele potenzielle Schwachstellen vor dem produktiven Einsatz Ihrer Software aufzuspüren.

Literatur & Links

[Grei13] G. Greiter, Test Automation Tool 2013, siehe: http://greiterweb.de/spw/test_werkzeuge.htm

[hp13] http://h20195.www2.hp.com/V2/GetPDF.aspx/4AA2-4856ENW.pdf

[SwissQ13] SwissQ Consulting AG, Testing Trends & Benchmarks Schweiz 2013, siehe http://www.tricentis.com/sites/default/files/pdf/analysts/Testing-Trends-und-Benchmarks-2013-Web.pdf

Tool	Testschritte
Microsoft Coded UI	+ - zusammengehörige Schritte sind als separater UI-Test möglich - die Testreihe kombiniert mehrere Tests - eine Kombination mit anderen Unit-Tests ist möglich (z. B. als „Vorbedingung“)
HP Unified Functional Testing (UFT)	o - Aufzeichnung und Wiederverwendung von Business-Komponenten (nur mit HP ALM) und Aktionen - beinhalten jeweils mehrere Schritte - über Action Libraries können Testschritte in ein gemeinsames Skript ausgelagert werden
Ranorex	+ - zusammengehörige Schritte können als separates Recording aufgezeichnet werden - eine Testsuite besteht aus mehreren Testfällen - der Testfall kombiniert mehrere Recordings
FitNesse	++ - einzelne Methoden können überall genutzt werden - zusammengehörige Schritte können als Wiki-Seite abgelegt und in die Tests eingebunden werden

Tab. 4: Wiederverwendbarkeit – Testschritte

Tool	Testfälle und Parameter
Microsoft Coded UI	o - Parameter können direkt am Test gepflegt werden
HP Unified Functional Testing (UFT)	+ - Parameterwerte können über Daten-Tabellen oder statisch eingebunden werden - der Test wird für jeden Datensatz ausgeführt
Ranorex	o - Variablen können pro Recording oder Mapping angelegt werden - trotzdem ist ein Recording pro Testfall nötig - alternativ lassen sich Parameter über Quellcode umsetzen
FitNesse	+ - die Testseite wird mit Parametern versehen - Modularisierung und Wiederverwendung erfolgt manuell

Tab. 5: Wiederverwendbarkeit – Testfälle und Parameter

Tool	Integration
Microsoft Coded UI	+ - der Testrunner kann über die Console gestartet werden
HP Unified Functional Testing (UFT)	+ - Tool muss auf der Build-Maschine installiert werden - über die Konsole lässt sich UFT mit Parameter „Test“ starten
Ranorex	+ - Ranorex muss auf der Build-Maschine installiert werden - dann wird einfach die .exe-Datei im Build ausgeführt
FitNesse	+ - es kann ein Msbuild-Task erstellt werden, der die Tests startet - anschließend wird der Build-Task eingebunden

Tab. 6: Integration