Ausgabe 2013 / 1

Erscheinungsart: ca. 4 x jährlich in elektronischer Form

Die Kunst der Testautomatisierung über die GUI

DI Johannes Hochrainer - Software Quality Lab

Automatisieren Sie so wenig Testfälle wie möglich über die grafische Benutzeroberfläche (GUI)!

Im ersten Abschnitt dieses Artikels werden die Gründe für diese - für viele vielleicht ungewöhnliche - Behauptung aufgezeigt und im Anschluss daran wird Ihnen eine gute Strategie für Testautomatisierung vorgestellt. Wenn schon die GUI für automatisierte Testfälle verwendet wird, dann sollte dies angemessen durchgeführt werden: Es werden in diesem Knowledge Letter verschiedene Testautomatisierungsarchitekturen präsentiert und dargestellt, welche dieser Architekturen jene mit dem meisten Potential für wartbare und robuste Testfälle ist.

Herausforderungen

Automatisierte Tests können verschiedene Schnittstellen verwenden, um mit dem zu testenden System zu interagieren. Die sehr oft favorisierte aber meist auch die schwierigste Schnittstelle bei System- und Abnahmetests ist die grafische Benutzeroberfläche (GUI). Die damit verbundenen Herausforderungen können in folgende 3 Gruppen unterteilt werden:

1) Verzögerte Information bei Änderungen am zu testenden System: Testwerkzeuge verwenden Techniken wie Reflection, Window Messages oder Bilderkennung, um mit dem GUI zu interagieren. Folglich kann das Testwerkzeug bzw. der Compiler Änderungen im GUI nicht erkennen. Erst während der Testdurchführung treten Probleme auf.

Ein weiterer Aspekt dabei ist, dass die GUI-Testautomatisierung oft zeitlich und organisatorisch weit entfernt von der Entwicklung stattfindet. Um diesen Problempunkt zu reduzieren ist ein gut funktionierendes Change-Management erforderlich. Auch moderne Prozessansätze wie die agile Entwicklung versuchen, diese Distanz zu mindern.

2) Technische Herausforderungen: Testwerkzeuge funktionieren meist gut mit Standardsteuerelementen. Vielfach sind aber auch selbst entwickelte oder von Drittanbietern erworbene Steuerelemente im Einsatz. Will man nicht auf unrobuste Workarounds ausweichen, muss man die Steuerelemente durch Anpassung des Quellcodes testbar machen. Drittanbieter stellen jedoch häufig den Quellcode nicht zur Verfügung.

Auch die Synchronisation zwischen GUI und automatisierten Testfällen stellt ein Problem dar. Denn manchmal dauert es etwas länger, bis eine Tabelle vollständig mit Daten gefüllt ist oder das Testwerkzeug zum gewünschten Steuerelement vordringen kann.

3) Aufwand für robuste Testfälle: Ein unerwarteter, für die Testbedingung aber unwesentlicher Dialog (z.B. ein Update-Hinweis des Antivirenprogramms), hat schon so manchen Testautomatisierer zur Verzweiflung gebracht. Ein oft noch größeres Problem ist es, für jeden Testfall wieder einen definierten Anfangszustand herzustellen. Auch nach einem Fehler muss für den folgenden Testfall dieser Zustand garantiert werden. GUI-Tests sind überwiegend Systemtests. Folglich ist es meist nicht so leicht, sich von Fremdsystemen, Sensoren oder Datenbanken zu entkoppeln. In niedrigeren Teststufen wäre das durch „Mocks“ einfacher umzusetzen.

Strategie für die Testautomatisierung

Lisa Crispin und Janet Gregory stellen in ihrem vielfach verkauften Buch „Agile Testing – A Practical Guide for Testers in Agile Teams“ eine Testpyramide vor. Daran angelehnt ist die nachfolgende Testpyramide:

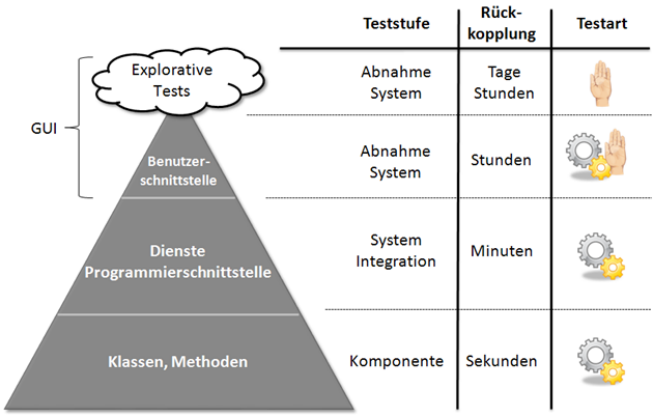

Abbildung 1 - Testpyramide

Komponententests (engl. Unit Tests – also Tests von Klassen und Methoden) bilden die breite Basis aller Tests. Implementiert der Entwickler die Tests während oder im besten Fall vor dem Code (testgetriebene Entwicklung) entsteht automatisch testbarer Code. Diese Vorgehensweise verringert den Aufwand für die Testautomatisierung beträchtlich. Ein weiterer Vorteil ist die rasche und häufige Rückkopplung zum Entwickler mit Informationen über mögliche Fehler.

Dienste (z. B. Web Services) und Programmierschnittstellen (API) bilden die zweite Schicht in der Testpyramide. Sie sind deswegen überwiegend leicht automatisierbar, da es meist keinen Medienbruch zwischen getesteter Applikation und Testautomatisierung gibt. Die betrachteten Systemteile lassen sich zumeist auch leichter von der Außenwelt isolieren.

Das Ganze ist mehr als die Summe seiner Teile. Diesen Lehrsatz von Aristoteles muss auch bei der Teststrategie berücksichtigt werden. Folglich sind auch System- und Abnahmetests nötig. Wie die Testpyramide zeigt, sollen die Tests über Benutzerschnittstellen aber nur einen geringen Anteil am Gesamtumfang der Tests einnehmen.

Auswahl geeigneter Tests

Besonders bei der Testautomatisierung über die GUI sollte selektiv bei der Auswahl geeigneter Testfälle vorgegangen werden. Es kann folgendes einfaches aber effektives Modell angewendet werden: Dabei ermittelt man für jeden Testfall den Wert und den Aufwand für die Testautomatisierung und stellt die beiden Attribute anschließend in einem Diagramm gegenüber.

Unter anderem können folgende Faktoren verwendet werden, um den **Wert** eines Testfalls zu ermitteln:

- **Risiko**: Das Risiko ist das Produkt aus Wahrscheinlichkeit des Auftretens eines Ereignisses und der Schwere der Auswirkungen.
- **Verwendungshäufigkeit**: Eine für den Tagesbetrieb notwendige Funktion ist häufig wichtiger, als eine alle paar Monate benötigte Funktion.
- **Fehlerhäufigkeit in der Vergangenheit**: Fehler sind wie Pilze: Sie treten meist in Gruppen auf und der größte Teil befindet sich unsichtbar „im Boden“. Testen Sie also Funktionen, die in der Vergangenheit fehlerträchtig waren, intensiver und häufiger.
- Einfluss des getesteten Prozesses auf den Erfolg der Organisation

Der **Aufwand** für die Testautomatisierung ist die zweite Messlatte für die Auswahl geeigneter Tests. Einfluss auf diese Faktoren haben etwa folgende Aspekte:

- Testumgebung bereitstellen
- Testdaten bereitstellen
- Anzahl der schwer testbaren Steuerelemente
- Test automatisieren
- Ergebnisse analysieren

Sind Wert und Aufwand für einen Test ermittelt, kann aufgrund folgenden Diagramms gesagt werden, ob er automatisiert werden soll:

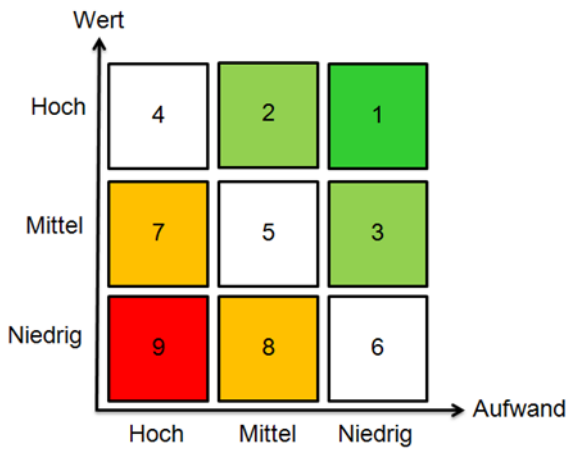

Abbildung 2 - Auswahl geeigneter Testfälle

Die Nummern in den Quadraten des Diagramms veranschaulichen die empfohlene Reihenfolge der Testautomatisierung. Demnach sollen Tests mit hohem Wert und niedrigem Aufwand als erstes automatisiert werden. Tests mit niedrigem Wert und hohem Aufwand sollen erst zum Schluss bzw. gar nicht automatisiert werden.

Eine nachvollziehbare Auswahl der Testfälle ist auch aufgrund eines anderen Aspekts wichtig: Menschen suchen nach Herausforderungen. Folglich sind herausfordernde Testfälle, also jene mit hohem Aufwand, für die Testautomatisierer meist interessanter als einfache 08/15-Testfälle. Den Testautomatisierern sollte also eine klare Reihenfolge vorgegeben werden.

Architekturen

Nachdem dargestellt wurde, dass die GUI diejenige Schnittstelle mit den größten Herausforderungen ist und gezeigt wurde, wie geeignete Testfälle ermittelt werden können, wird in diesem Abschnitt die Testautomatisierung näher betrachtet. Dazu gibt es verschiedene Architekturen. Die Wahl der richtigen Architektur hat beträchtliche Auswirkungen auf den Erfolg der Testautomatisierung.

Automatisierte Testfälle sollen folgende Qualitätsattribute erfüllen:

Robust: Kleine Änderungen in Steuerelementen (z.B. ID, Position) dürfen sich nicht auf die automatisierten Tests auswirken. Des Weiteren sollen Tests trotz unerwarteter, aber für den Test unwesentlicher Ereignisse, weiterlaufen. Jeder Test muss am Beginn der Durchführung auf einen definierten Systemzustand vertrauen können.

Wartbar: Änderungen in der GUI dürfen nur einen geringen Wartungsaufwand verursachen.

Lesbar: Testspezifikation und Testprotokolle sollen für die Zielgruppe lesbar sein.

Um diese Qualitätsattribute umsetzen zu können, unterscheidet man zwischen drei Abstraktionsschichten, die zwischen der Testspezifikation und dem getesteten System liegen:

Die unterste Schicht ist die „**Technik**“. Sie befasst sich mit der Identifikation von grafischen Steuerelementen und wie diese Informationen abgelegt und wiederverwendet werden können.

Workflows bilden die mittlere Schicht. Wiederkehrende Abläufe sollen in möglichst vielen Testfällen Anwendung finden.

Die oberste Schicht bildet die **Präsentation** der Testfälle. Sie wird nur benötigt, wenn automatisierte Tests auch von Personen zusammengestellt werden sollen, die nicht programmieren können.

Vollständiger Knowledge Letter Zugang

Wir freuen uns, dass Sie an diesem Thema Interesse haben und den Knowledge Letter von Software Quality Lab bis hierher gelesen haben.

Dieser Knowledge Letter ist eine Vorschau (gekürzte Version des gesamten Artikels).

Wenn Sie den ungekürzten Knowledge Letter lesen möchten, registrieren Sie sich bitte unter
http://www.software-quality-lab.com/download/knowledge-letter/anfrage-knowledge-letter/

Sie erhalten nach der Registrierung vollen Zugang zu allen bisherigen Knowledge Lettern von Software Quality Lab und erhalten automatisch künftige Knowledge Letter per E-Mail.