Der Glaubenskrieg über Testautomatisierung

Matthias Hamburg[1], Stephan Weißleder[2]

Abstract: Mit fortschreitender Technologie mehren sich die Erfolgsberichte über Testautomatisierung, z.B. im Rahmen von DevOps und Continuous Delivery. Protagonisten des explorativen Testens setzen entgegen, dass nur die kreative menschliche Arbeit das Wort „Testen" verdient, und jede Art von Automatisierung einen kreativen Prozess unmöglich macht. Die Autoren untersuchen deshalb die Kosten und den Nutzen der Automatisierung der Aktivitäten des dynamischen Testens. Sie gehen auch darauf ein, welche Orientierung Lehrpläne und Glossar des ISTQB® bei dieser Frage bieten. Daraus leiten sie ihre Antwort auf die Streitfrage ab, ob die Testautomatisierung eine neue Wunderlösung oder ein Irrglaube ist.

Keywords: Testautomatisierung, modellbasiertes Testen, Continuous Delivery, Verifizierung,Validierung.

1 Einleitung

In der aktuellen Fachliteratur werden grundsätzlich gegensätzliche Meinungen zur Automatisierung des Testens vertreten. Einerseits lautet das bekannte Mantra von DevOps „automatisiere alles, was du mehr als zweimal ausführst". Insbesondere die Befürworter der Continuous Delivery setzen als qualitätssichernde Maßnahme auf automatisierte Tests, die vor jedem Deployment durchgeführt werden. Andererseits setzen Kritiker entgegen, dass jede Art von Automatisierung eine standardisierte Wiederholung voraussetzt, die einen kreativen Prozess unmöglich macht. Entsprechend würde das bedeuten, dass „das einzig wahre Testen" kreativ sein muss und deshalb unbedingt manuell durchzuführen sei. Alles andere sei eher eine technische Nachprüfung („checking").

Automatisierung lohnt sich grundsätzlich für wohldefinierte, klar abgegrenzte Aktivitäten, wenn sie häufig wiederholt werden, und nicht durch manuelle Tätigkeiten unterbrochen werden. Außerdem muss der Nutzen die Investition und Wartung der Automatisierung überwiegen. Ausgangspunkt für die Entwicklung eines validen Standpunktes in dieser Kontroverse soll deshalb eine genauere Betrachtung der zu automatisierenden Testaktivitäten sein. Dabei ist die Kontroverse auf die Kernaktivitäten des dynamischen Testens fokussiert. Testmanagementaktivitäten wie Planung und Steuerung des Testens, Einrichtung und Verwaltung der Testumgebung oder Fehlermanagement sind weniger umstritten und werden an dieser Stelle ausgeklammert.

[1] German Testing Board e.V., Koldestrasse 8b, 91052 Erlangen, m.hamburg@gtb.de
[2] German Testing Board e.V., Koldestrasse 8b, 91052 Erlangen, s.weissleder@gtb.de

2 Hauptaktivitäten des dynamischen Testens

Unser Ausgangspunkt ist das Domänenmodell der Testentwicklung in Abb. 1, das von der Arbeitsgruppe Glossar des ISTQB® entwickelt wurde [Ha19].

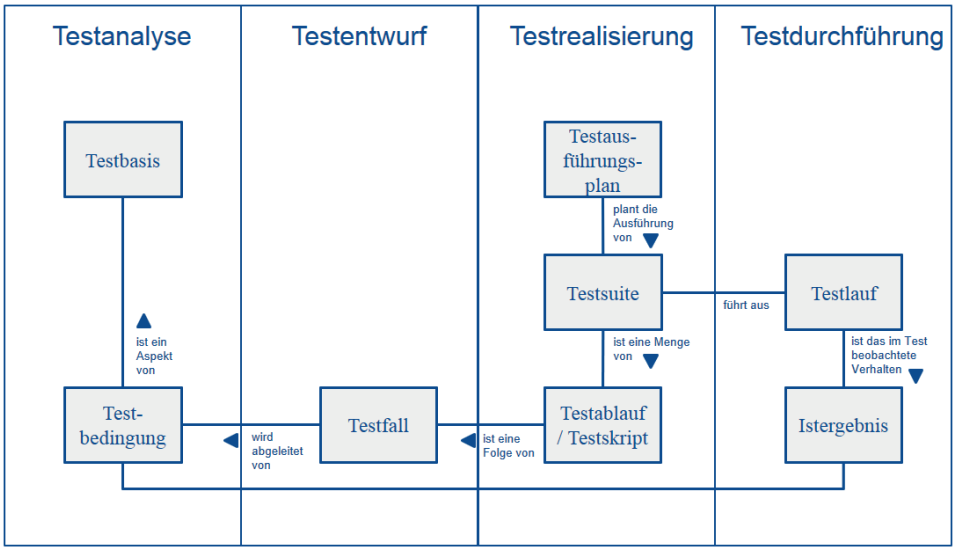

Abb. 1: Dynamische Testaktivitäten und ihre Ergebnisse

Dieses Modell in Abb. 1 zeigt die Beziehung verschiedener Artefakte in den vier Hauptaktivitäten des dynamischen Testens nach ISTQB® [CT19]: Testanalyse, Testentwurf, Testrealisierung und Testdurchführung. Im Folgenden betrachten wir die Automatisierung in den drei Aktivitäten Testentwurf, Testrealisierung und Testdurchführung. Wieweit können wir die Erstellung dieser Arbeitsergebnisse automatisieren, welche Kosten sind damit grundsätzlich verbunden, und was ist der Nutzen? Während der Nutzen mit den Arbeitsergebnissen in Abb. 1 verbunden ist, hilft beim Einschätzen der Kosten die generische Testautomatisierungsarchitektur des ISTQB® [CT16].

Wir beginnen mit der Automatisierung bei der letzten Aktivität der Kette und schreiten in Richtung früherer Aktivitäten voran, um unnötige Effizienzverluste zu vermeiden. Wenn man zum Beispiel Testfälle durch automatischen Testentwurf aus Testbedingungen generieren kann, aber danach die Testrealisierung (z.B. die Bereitstellung von Testdaten, der Entwurf von Testskripten) manuell erfolgt, kann es zu hohen Wartungsaufwänden kommen. Nach einer kleinen Änderung im Verhaltensmodell können die generierten abstrakten Testfälle radikal anders aussehen als vorher, und die manuelle Realisierung müsste fast komplett wiederholt werden.

3 Nur die Testdurchführung automatisieren

Die Automatisierung der Testdurchführung ist laut ISTQB-Glossar „Die Verwendung einer Software, um die Ausführung von Tests zu steuern, tatsächliche mit erwarteten Ergebnissen zu vergleichen, die definierten Vorbedingungen herzustellen sowie weitere Testüberwachungs- und Berichtsfunktionen durchzuführen“ [CT21a]. Sie ist oftmals der erste und intuitive Ansatz zur Automatisierung von wiederholten Aktivitäten des Testens.

Kosten

Die Architektur der automatischen Testdurchführung enthält eine Testausführungsschicht, welche die Steuerung der Ausführung der Testskripte, inkl. der Behandlung von Abweichungen, die Protokollierung der Ausführung, und die Generierung von Testberichten umfasst [CT16]. Darüber hinaus enthält die Architektur eine Testadaptierungsschicht, die die relativ abstrakten Testskripte in die Zielsprache der verschiedenen Komponenten, Konfigurationen oder Schnittstellen des Testobjekts übersetzt. Die Kosten dieser Schichten hängen wesentlich davon ab, wie verbreitet die verwendeten Technologien sind.

Eine weitere Herausforderung sind Unterbrechungen durch die manuelle Untersuchung der Abweichungen des Istergebnisses vom erwarteten Ergebnis. Neuerdings wird an dieser Stelle künstliche Intelligenz zur Automatisierung eingesetzt [Za21], deren Kosten ebenfalls berücksichtigt werden müssen.

Nutzen

Automatisiert man die Durchführung von Testsuiten, so entsteht eine Kostenersparnis bei der wiederholten Durchführung ein und derselben Testsuite. Solche Wiederholungen schaffen ein Vertrauen in die Lauffähigkeit des Testobjekts. Zum Finden von Fehlerwirkungen durch neue oder geänderte Teile des Testobjekts sind sie allerdings weniger geeignet.

Effizienz

Die Automatisierung der Testdurchführung allein lohnt sich nur unter günstigen Umständen. Dazu gehören eine gängige oder standardisierte Testumgebung auf der Kostenseite sowie eine stark inkrementelle Entwicklung mit zahlreichen Regressionstests auf der Nutzenseite.

Missionare des explorativen Testens kritisieren also die hohen Kosten für Architektur und manuelles Skripten einerseits und den beschränkten Nutzen andererseits nicht ganz zu Unrecht. Auf der anderen Seite muss man auch den Missionaren der Automatisierung Recht geben, dass menschliches Ad-Hoc Testen das Vertrauen in die Lauffähigkeit immer größerer Softwaresysteme nicht begründen kann.

Um durch die Testautomatisierung mehr Nutzen zu erzielen, müssen die ausgeführten Testskripte flexibler werden. Dies führt zum Bedarf nach der Automatisierung der Testrealisierung.

4 Zusätzlich die Testrealisierung automatisieren

Die Testrealisierung bereitet auf Basis von Testanalyse und -entwurf die Testmittel für die Testdurchführung vor [CT21a]. Ausgangspunkt sind die Testbedingungen und Testfälle.

Kosten

Um diese Aktivität zu automatisieren, sollten Testdaten aus Datenmodellen und den Inhalten der Testfälle (Eingaben und erwartete Ergebnisse) generiert werden. Testskripte können aus einem Verhaltensmodell des Testobjekts und aus den Testfallinhalten generiert werden. Entsprechend braucht eine generische Testautomatisierungsarchitektur auch eine Testdefinitionsschicht, die die Testrealisierung durch Definition von Testsuiten und/oder Testfällen unterstützt, z.B. durch Anbieten von Vorlagen bzw. Richtlinien.

Nutzen

Diese Automatisierungsstufe entlastet die Tester von der manuellen Pflege der Testskripte und Testsuiten, ohne auf die Dokumentation dieser Arbeitsergebnisse verzichten zu müssen. Darüber hinaus können Testdaten für den gleichen abstrakten Testfall leicht variiert werden. Dies ist insbesondere bei variantenreichen Anwendungen nützlich, die entweder auf zahlreichen Plattformen oder mit zahlreichen Umgebungseinstellungen laufen. Die auszuführenden abstrakten Testfälle in den Testsuiten können dynamisch ausgewählt werden.

Effizienz

Die Mehrkosten für die Entwicklung und Wartung der Testdefinitionsschicht sind im Allgemeinen gering. Der Zusatznutzen aus Skriptgenerierung und mehr Flexibilität der Testfälle ist es allerdings auch. Die mehrfache Realisierung und Ausführung des gleichen abstrakten Testfalls hat relativ wenig Chancen, neue Fehlerzustände aufzudecken. Nur in bestimmten Situationen – z.B. wenn die Dokumentation und Verfolgbarkeit der Testskripte vorgeschrieben ist, oder bei variantenreichen oder komplexen datenorientierten Anwendungen mit komplexen Testskripten – bringt dieser Automatisierungsschritt einen wesentlichen Effizienzgewinn.

Kritiker der Testautomatisierung greifen oft das Problem der Ineffizienz des „geskripteten Testens“ auf. Wenn man aber die (abstrakten) Testfälle von den Testskripten unterscheidet (Abb. 1), wird dieser Einwand durch die Automatisierung der Testrealisierung entkräftet. Auf der anderen Seite wird das Ziel einer effektiven Fehlerfindung durch diese Ausbaustufe der Automatisierung noch nicht ausreichend unterstützt. Dies legt die zusätzliche Automatisierung des Testentwurfs nahe.

5 Zusätzlich den Testentwurf automatisieren

Beim Automatisieren des Testentwurfs spricht man oft von modellbasiertem Testen (MBT). Das ISTQB widmet den Grundlagen des MBT einen eigenen Lehrplan [CT15]. Im Grunde ist aber modellbasiertes Testen ein sehr allgemeiner Begriff, definiert als Testen, das auf Modellen basiert oder diese involviert [CT21a].

Kosten

Diese vergleichsweise umfassende Automatisierung des Testens erfordert es, alle Informationen in Modellen zu pflegen, aus denen alle anderen Arbeitsergebnisse des dynamischen Testens generiert werden können. Das ergibt eine gleichfalls komplexe und mächtige Grundlage. Die für den Testentwurf passende Schicht einer generischen Testautomatisierungsarchitektur ist die Testgenerierungsschicht, die den manuellen oder automatisierten Entwurf von Testsuiten und/oder Testfällen unterstützt.

Wollen wir z.B. Black-Box-Tests generieren, so brauchen wir ein Verhaltensmodell des Testobjekts als Basis. Auch die Testbedingungen (wie z.B. 2-Switch Zustandsübergänge) müssen dem Modell angefügt werden. Um konsistente abstrakte Testfälle generieren zu können, kann auch ein Modell der Testdaten oder der Struktur des Testobjekts hilfreich sein (z.B. ein UML-Klassendiagram). Für konkrete Testfälle können weitere mögliche UML-Objektmodelle hinzukommen.

Nutzen

Modellbasiertes Testen ist nicht neu und birgt viel Potenzial [Wi09]. Durch die erreichte volle Flexibilität bei der Generierung abstrakter Testfälle erreichen wir eine effektive Überdeckung und Fehlerfindung. Es erleichtert auch den Einsatz von Testverfahren wie paarweises Testen oder n-Switch Überdeckung [CT21b], die höhere Fehlerfindungsraten bei einer relativ geringen Anzahl von Testfällen versprechen, also geringe Kosten für die Testrealisierung und –durchführung, die jedoch ohne werkzeuggestützte Testfallgenerierung einen unpraktikabel hohen Aufwand im Testentwurf bedeuten. Darüber hinaus ist die Ableitung der Testmodelle aus der Testbasis eine frühe Qualitätssicherungsmaßnahme [CT15].

Effizienz

Die Automatisierung des Testentwurfs bringt einen hohen Zusatznutzen in der Erreichung der Testziele. Die damit verbundene frühe Prüfung der Testbasis auf Testbarkeit erhöht zusätzlich die Effizienz des Testens. Die Zusatzkosten sind allerdings auch hoch. Der wichtigste Engpass ist in der Praxis die Qualifikation der Tester. Bei guter Qualifikation (insbesondere Zertifizierungen nach [CT15] und [CT16]) kann hiermit in der Zukunft die höchste Effizienzstufe erreicht werden. Für die Protagonisten der Continuous Delivery wäre damit das eigentliche Ziel erreicht: so weit wie möglich automatisieren. Auf Kritiker der Testautomatisierung kann man soweit eingehen, dass die strukturierten (Black-Box oder White-Box) Testverfahren durch erfahrungsbasiertes Testen ergänzt werden sollten.

6 Schlussfolgerung

Unsere Antwort an die Befürworter der vollständigen Testautomatisierung ist: Testentwurf, -realisierung und -durchführung lassen sich im Rahmen der Verifizierung vollständig automatisieren. Die heutige Technologie ermöglicht das alles. Anwender sollten dabei Kosten- und Nutzenfaktoren der Automatisierung der einzelnen Testaktivitäten im Auge behalten. Unter passenden Rahmenbedingungen kann automatisches dynamisches Testen der effektive und effiziente Kern der Qualitätssicherung sein. Zu den günstigen Rahmenbedingungen gehören idealerweise eine verbreitete Standardtechnologie mit umfangreichen verfügbaren Testbibliotheken für eine vollumfängliche Testschnittstelle des Testobjekts, eine langlebige Anwendung mit häufigen und schnellen Änderungszyklen, gegebenenfalls Variantenreichtum, und nicht zuletzt ein hochqualifiziertes Testteam, mit Testanalysten die in abstrakten Modellen denken können, sowie mit Automatisierungsingenieuren, die eine komplexe Automatisierungsarchitektur entwickeln und pflegen.

Unsere Antwort an die Skeptiker der Testautomatisierung lautet: Softwaretester müssen klar zwischen Verifizierung und Validierung unterscheiden können. Beide sind unverzichtbar. Das Ziel der Verifizierung ist der Nachweis, dass festgelegte Anforderungen erfüllt worden sind. Hingegen ist das Ziel der Validierung, die Eignung für den beabsichtigten Gebrauch zu bestätigen [CT21a]. Die Verifizierung kann man unter günstigen Umständen sehr weitgehend automatisieren, und seine Effektivität durch Automatisierung sogar verstärken. Das wesentlich informellere Validieren hingegen nicht, hierbei ist Kreativität gefordert. Deswegen ist aber nicht das eine das „wahre Testen“ und das andere „etwas für Roboter“. Beide sind sehr wichtig und müssen sich sinnvoll ergänzen.

Literaturverzeichnis

[CT15] ISTQB®, Certified Tester Model-Based Tester Syllabus, 2015.

[CT16] ISTQB®, Certified Tester Test Automation Engineer Syllabus, 2016.

[CT19] ISTQB®, Certified Tester Foundation Level Syllabus v3.1, 2019.

[CT21a] ISTQB®/GTB Standardglossar der Testbegriffe, https://glossary.istqb.org/de, Stand: 13.06.2021.

[CT21b] ISTQB®, Certified Tester Advanced Test Analyst Syllabus v3.1, 2021.

[Ha19] Hamburg, M.: Ein Domänenmodell des Testens. German Testing Magazin 01/19, S. 10-11, 2019.

[Wi09] Winter, M.: Modellbasiertes Testen – Alter Wein in neuen Schläuchen? Proc. 10. SQC-Kongress, Düsseldorf, 28. Mai 2009.

[Za21] Zax, M.: Self-Healing Tests – Der heilige Gral der Testautomation. Proc. German Testing Day, 2021