Manfred Baumgartner • Stefan Gwihs • Richard Seidl
Thomas Steirer • Marc-Florian Wendland

Basiswissen

Testautomatisierung

Aus- und Weiterbildung zum
**ISTQB® Advanced Level Specialist –
Certified Test Automation Engineer**

dpunkt.verlag

3., aktualisierte und überarbeitete Auflage

Inhalt

Cover

Über den Autor

Titel

Impressum

Vorwort zur 3. Auflage

Geleitwort zur 3. Auflage

Geleitwort zur 2. Auflage

Inhaltsübersicht

Inhaltsverzeichnis

1 Einführung in die Testautomatisierung und ihre Ziele 1

1.1 Einleitung

1.1.1 Standards und Normen

1.1.2 Der Einsatz von Maschinen

1.1.3 Mengen und Massen

1.2 Was ist unter Testautomatisierung zu verstehen?

1.3 Ziele der Testautomatisierung

1.4 Erfolgsfaktoren für die Testautomatisierung

1.4.1 Testautomatisierungsstrategie

1.4.2 Testautomatisierungsarchitektur

1.4.3 Testbarkeit des SUT

1.4.4 Testautomatisierungsframework

1.5 Exkurs: Teststufen und Projektarten

1.5.1 Testautomatisierung auf unterschiedlichen Teststufen

1.5.2 Einsatzgebiet nach Projektart

2 Vorbereitungen für die Testautomatisierung

2.1 SUT-Faktoren mit Einfluss auf die Testautomatisierung

2.2 Bewertung und Auswahl von Werkzeugen

2.2.1 Verantwortlichkeiten

2.2.2 Exkurs: Evaluierung von Automatisierungswerkzeugen

2.2.3 Exkurs: Evaluieren leicht gemacht

2.2.4 Typische Herausforderungen

2.3 Auslegung auf Testbarkeit und Automatisierung

3 Die generische Testautomatisierungsarchitektur

3.1 Einführung in die generische Testautomatisierungsarchitektur (gTAA)

3.1.1 Warum eine gute Testautomatisierungsarchitektur so wichtig ist

3.1.2 Entwicklung von Testautomatisierungslösungen

3.1.3 Die Schichten der gTAA

3.1.4 Projektmanagement einer TAS

3.1.5 Konfigurationsmanagement einer TAS

3.1.6 Unterstützung des Testmanagements und anderer Zielgruppen

3.2 Der Entwurf einer TAA

3.2.1 Grundlegende Fragestellungen

3.2.2 Welcher Ansatz zur Automatisierung von Testfällen soll unterstützt werden?

3.2.3 Welche technischen Überlegungen zum SUT sind zu beachten?

3.2.4 Überlegungen zu Entwicklungs- und Qualitätssicherungsprozessen

3.3 TAS-Entwicklung

3.3.1 Kompatibilität zwischen TAS und SUT

3.3.2 Synchronisierung zwischen TAS und SUT

3.3.3 Wiederverwendbarkeit in einer TAS

3.3.4 Unterstützung verschiedener Zielsysteme

3.3.5 Exkurs: Realisierung in unterschiedlichen Vorgehensmodellen und Methoden

4 Risiken und Eventualitäten bei der Softwareverteilung

4.1 Auswahl des Testautomatisierungsansatzes und Planung von Verteilung/Rollout

4.1.1 Die Erprobung oder der Pilotversuch

4.1.2 Die Verteilung oder das Deployment

4.2 Strategie für die Bewertung und Begrenzung von Risiken

4.2.1 Spezifische Risiken bei der Erstverteilung

4.2.2 Spezifische Risiken bei der Wartungsverteilung

4.3 Wartung der Testautomatisierung

4.3.1 Auslöser und Arten von Wartungsaktivitäten

4.3.2 Überlegungen zur Dokumentation der automatisierten Testmittel

4.3.3 Der Umfang von Wartungsaktivitäten

4.3.4 Wartung von Fremdkomponenten

4.3.5 Wartung von Schulungsmaterial

4.3.6 Verbesserung der Wartbarkeit

4.4 Exkurs: Einsatzgebiet nach Systemarten

4.4.1 Desktop-Applikationen

4.4.2 Client-Server-Systeme

4.4.3 Webapplikationen

4.4.4 Mobile Applikationen

4.4.5 Webservices

4.4.6 Data Warehouse

4.4.7 Dynamische GUIs: Formularlösungen

4.4.8 Cloud Based Systems

4.4.9 Künstliche Intelligenz und Machine Learning

5 Berichte und Metriken

5.1 Exkurs: Metriken und Validität

5.2 Beispiele für Metriken

5.3 Konkrete Implementierung und Realisierbarkeit in einer TAS

5.3.1 Exkurs: TAS und SUT als Quellen für Protokolle

5.3.2 Exkurs: Zentralisierte Verwaltung und Auswertung von Protokollen

5.3.3 Implementierung der Protokollierung in einer TAS

5.4 Erstellung von Berichten zur Testautomatisierung

5.4.1 Qualitätskriterien für Berichte

6 Überführung des manuellen Testens in eine automatisierte Umgebung

6.1 Kriterien für die Automatisierung

6.1.1 Eignungskriterien für die Umstellung auf automatisierte Tests

6.1.2 Vorbereitung der Umstellung auf automatisierte Tests

6.2 Erforderliche Schritte zur Automatisierung von Regressionstests

6.3 Faktoren bei der Automatisierung des Testens neuer oder geänderter Funktionen

6.4 Faktoren bei der Automatisierung von Fehlernachtests

7 Verifizierung der Testautomatisierungslösung

7.1 Warum die Qualitätssicherung einer TAS wichtig ist

7.2 Verifizieren der Komponenten der automatisierten Testumgebung

7.3 Verifizieren der automatisierten Testsuite

8 Fortlaufende Optimierung

8.1 Möglichkeiten der Optimierung der Testautomatisierung

8.2 Planung und Realisierung der Testautomatisierungsverbesserung

9 Ausblick

9.1 Herausforderungen in der Testautomatisierung

9.1.1 Allgegenwärtige Vernetzung

9.1.2 Testautomatisierung für die IT-Sicherheit

9.1.3 Testautomatisierung für autonome Systeme

9.2 Trends und mögliche Entwicklungen

9.2.1 Agile Softwareentwicklung ohne Testautomatisierung ist nicht denkbar

9.2.2 Neue Outsourcing-Szenarien für die Automatisierung

9.2.3 Die Automatisierung der Automatisierung

9.2.4 Ausbildung und Standardisierung

9.3 Innovation und Weiterentwicklung

Anhang

A Softwarequalitätsmerkmale

A.1 Funktionalität (functional suitability)

A.2 Performanz (performance efficiency)

A.3 Kompatibilität (compatibility)

A.4 Benutzbarkeit (usability)

A.5 Zuverlässigkeit (reliability)

A.6 Sicherheit (security)

A.7 Wartbarkeit (maintainability)

A.8 Übertragbarkeit (portability)

B Last- und Performanztest

B.1 Arten von Last- und Performanztests

B.2 Tätigkeiten im Last- und Performanztest

B.3 Definieren der Performanzziele

B.4 Identifizieren der Transaktionen bzw. Szenarien

B.5 Erstellen der Testdaten

B.6 Erstellung von Testszenarien

B.7 Durchführung der Tests

B.8 Monitoring

B.9 Typische Komponenten von Last- und Performanzwerkzeugen

B.10 Checklisten

C Kriterienkatalog zur Testwerkzeugauswahl

D Glossar

E Abkürzungen

F Quellen

Stichwortverzeichnis

4 Risiken und Eventualitäten bei der Softwareverteilung

Nachdem zuvor dargestellt wurde, welche Vorbereitungen für eine erfolgreiche TAS-Einführung getroffen werden müssen, wie auf Basis der gTAA ein für den Projektkontext sinnvoller TAA-Entwurf erstellt werden kann und auch welche Fragestellungen für die Ableitung der TAS aus dem TAA-Entwurf zu berücksichtigen sind, soll nun die tatsächliche, operative Verteilung und Wartung dieser TAS diskutiert werden, in der Testautomatisierung letztendlich wirksam wird. Hier zeigen sich die Effekte der vorhergehenden Schritte, und damit auch deren Sinnhaftigkeit und Erfolg.

Dabei ist relevant, wie die Verteilung bzw. der Rollout der TAS realisiert wird (siehe Abschnitt 4.1), welche Risiken dabei auftreten können und wie diesen begegnet werden kann (siehe Abschnitt 4.2) und – abschließend – wie die TAS-Wartung nach der initialen Verteilung erfolgt (siehe Abschnitt 4.3).

Als Zusatz wird in Abschnitt 4.4 noch ein Überblick über unterschiedliche Einsatzgebiete von TAS und deren spezifische Besonderheiten gegeben.

4.1 Auswahl des Testautomatisierungsansatzes und Planung von Verteilung/Rollout

Bei der Implementierung einer TAS bzw. der Etablierung einer TAS in Projekten ist ein iteratives Vorgehen empfehlenswert. Dadurch kann mit relativ geringem Risiko sichergestellt werden, dass eine TAS-Einführung auch den geplanten Mehrwert stiftet und welche unvorhergesehenen Herausforderungen dabei zu meistern sind. Dies kann durch einen mehrstufigen Prozess erreicht werden, bei dem erst nach erfolgreicher Erprobung in kleinem Rahmen eine großflächige Verteilung stattfindet. Das Hauptziel der Erprobung ist dabei primär ein Erkenntnisgewinn in unterschiedlichen Dimensionen (z.B. TAS, SUT, technische Probleme beim Zusammenspiel von SUT und TAS) sowie die Validierung von im TAA-Entwurf getroffenen Annahmen und den insgesamt notwendigen Vorbedingungen. Erst wenn im Zuge dieser Aktivitäten deutlich wird, dass eine großflächige Verteilung sinnvoll ist, sollte die tatsächliche Verteilung der TAS im breiten Rahmen geplant und umgesetzt werden. Dabei ist es valide, wenn

- nach der ersten Erprobung Anpassungen am TAA-Entwurf bzw. an der TAS notwendig werden und anschließend eine erneute Erprobung stattfinden soll;
- abhängig von der Komplexität der organisatorischen und technischen Rahmenbedingungen nicht nur eine isolierte Erprobung durchgeführt wird, sondern unterschiedliche Erprobungen parallel stattfinden (z.B. die gleiche TAS wird von verschiedenen Teams auf mehrere unterschiedliche SUT angewendet).

Auswertung der Erprobung

Es ist auch wichtig, darauf hinzuweisen, dass die Auswertung der Erprobung selten trivial ist. Optimalerweise wird im Vorfeld geklärt, nach welchen Kriterien, Metriken oder Verfahren über Erfolg oder Misserfolg der Erprobung entschieden wird. Dabei spielen Faktoren wie der notwendige Aufwand, die Komplexität bei der Implementierung, zusätzlicher Schulungsaufwand, aber auch die erzielbare Stabilität und Zuverlässigkeit der TAS eine Rolle. Allerdings müssen die genaue Auslegung und Gewichtung dieser Faktoren immer abhängig von den vorherrschenden Rahmenbedingungen und dem geplanten Einsatzzweck der TAS gewählt werden.

4.1.1 Die Erprobung oder der Pilotversuch

Um valide Erkenntnisse aus einer Erprobung, im Folgenden auch Pilotversuch genannt, zu generieren, sollte dabei folgender Ablauf verfolgt werden:

1. Ermittlung eines geeigneten Projekts
2. Planen des Pilotversuchs
3. Durchführen des Pilotversuchs
4. Evaluieren des Pilotversuchs

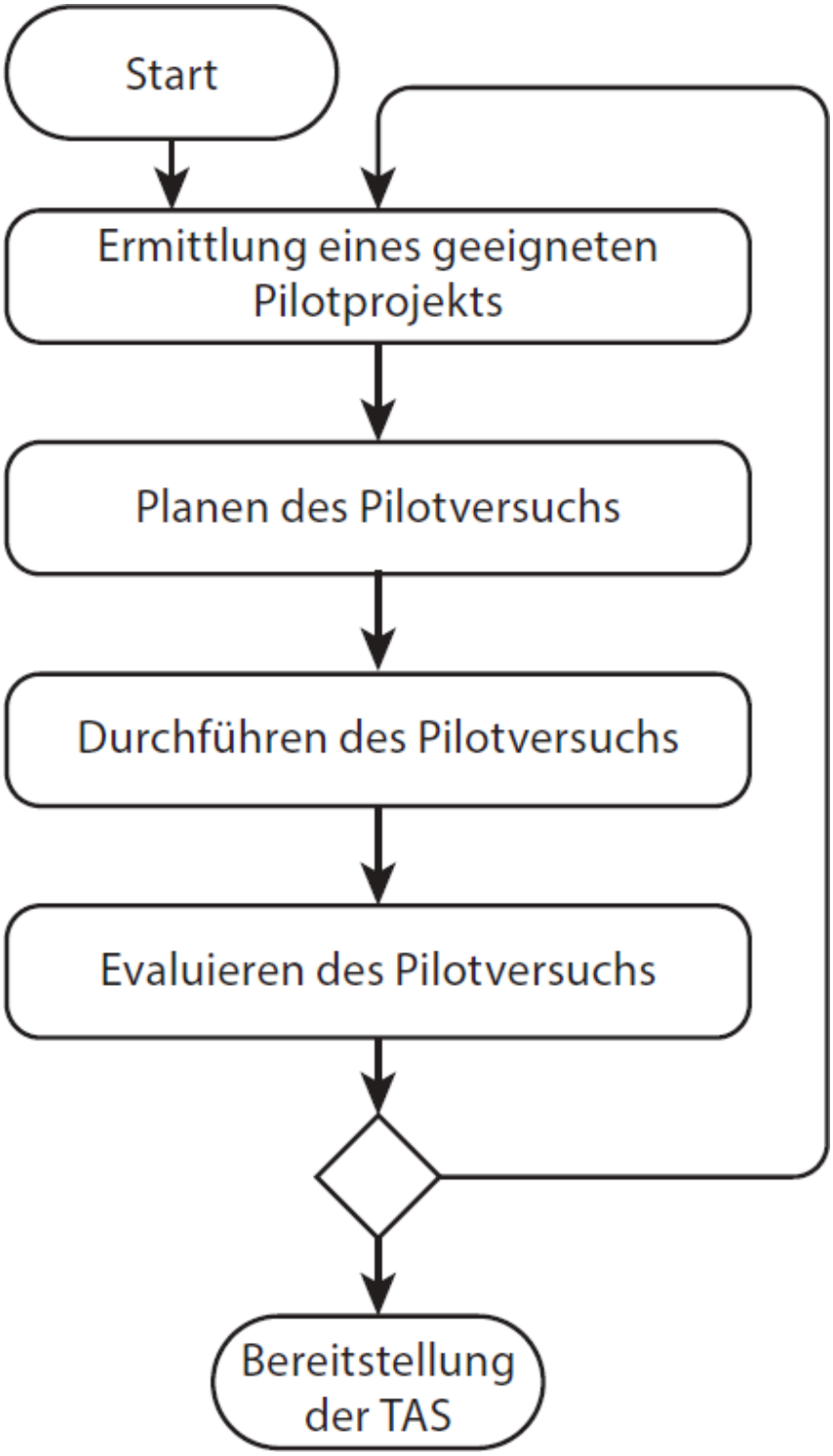

Abb. 4–1
Schematischer Ablauf der Erprobung

Ermittlung eines geeigneten Projekts

Um auf Grundlage der Auswertung eines Pilotversuchs eine valide Evaluierung der TAS durchführen zu können, muss ein geeignetes Projekt bzw. ein geeigneter Rahmen dafür ermittelt werden. Wichtige Faktoren bei dieser Auswahl sind

organisatorische sowie technische Rahmenbedingungen des Projekts. Unter anderem sollte das Projekt folgende Punkte erfüllen:

- Es sollte tatsächlich den Bedarf haben, eine TAS einzuführen bzw. eine neue TAS zu etablieren. Ist das nicht der Fall, kann es dazu führen, dass fälschlicherweise der Gesamtnutzen der TAS infrage gestellt wird, obwohl lediglich im konkreten Projektkontext kein Mehrwert erzielt werden konnte.
- Es sollte nicht gerade eine kritische Projektphase durchlaufen. Die Einführung einer TAS ist häufig mit einer Verzögerung aufgrund unvorhersehbarer Eventualitäten verbunden. Diese Verzögerungen oder Probleme in der Einführung einer TAS dürfen die Lieferqualität des SUT nicht beeinflussen.
- Es sollte repräsentativ für den geplanten Einsatzzweck sein. Das bedeutet, dass es weder zu komplex noch zu trivial ist (fachlich und technologisch), um falsche Ableitungen daraus zu vermeiden. Gerade wenn zu triviale Projekte für die Erprobung herangezogen werden, besteht die Gefahr, dass eine TAS implementiert wird, die der Komplexität der Realität nicht standhält, und dass dieser Umstand durch den Pilotversuch nicht aufgedeckt wird.
- Im Hinblick auf die zu automatisierenden Schnittstellen des SUT sollte das Projekt repräsentativ für andere Projekte in der Organisation sein. Es sollten also nicht nur »exotische« Schnittstellen angesteuert werden, sondern jene, die auch in einer Vielzahl anderer vergleichbarer SUTs anzutreffen sind.
- Außerdem sollte es unter Berücksichtigung der individuellen Bedürfnisse sämtlicher beteiligter Personengruppen (Projektmanagement, Testmanagement) ausgewählt werden.

Planen und Durchführen des Pilotversuchs

Nachdem ein geeignetes Projekt für den Pilotversuch identifiziert wurde, sollte die Planung und Durchführung genauso stattfinden wie für jedes andere Softwareentwicklungsprojekt auch.

Umsetzung und Einführung einer TAS benötigt Zeit.

Dabei ist zu berücksichtigen, dass die Umsetzung und Einführung einer TAS Aufwand und Zeit benötigt. Nicht nur bei der Einführung oder bei der Installation der Werkzeuge, sondern auch bei der Konzeption, der Implementierung, der Auswertung und der Wartung der automatisierten Tests. Aufwand und

Durchlaufzeiten müssen daher eingeplant sein und sich in der Projektplanung niederschlagen. Dies ist in manchen Situationen schwierig, vor allem dann, wenn die Automatisierung nur als beiläufige Tätigkeit oder als »Hobby« eines Testers bzw. Entwicklers wahrgenommen wird oder wenn sie in der ursprünglichen Planung des Rahmenprojekts nicht vorgesehen war. Dann muss umso mehr mit dem Nutzen auch explizit aufgezeigt werden, dass es sich bezahlt macht, für diese Aktivitäten Ressourcen bereitzustellen.

Evaluierung des Pilotversuchs

Nach Abschluss des Pilotprojekts (der mehr oder weniger formal erreicht werden kann) ist die Evaluierung des Pilotversuchs nun der finale Schritt, bevor mit der Bereitstellung einer TAS gestartet werden kann. Wie bereits eingangs erwähnt, geht es hierbei primär um einen Erkenntnisgewinn, und im Zuge der Evaluierung muss nun sichergestellt werden, dass aus dem abgeschlossenen Pilotversuch tatsächlich ausreichend Ableitungen getroffen und die notwendigen Erkenntnisse gezogen werden können. In dieser Phase sollte ein möglichst breiter Kreis an Personengruppen involviert werden, um die spätere Akzeptanz der Evaluierung sicherzustellen und um unterschiedliche Perspektiven und Interpretationen in der Entscheidung berücksichtigen zu können.

Zusammenfassend sollten nach der Evaluierung zumindest für die nachfolgenden Fragen klare Antworten gefunden sein:

Einbeziehung einer breiten Personengruppe verbessert die Akzeptanz der Evaluierung.

- Sind TAA- und TAS-Entwurf mit den bestehenden Prozessen, Abläufen, Werkzeugen und Technologien kompatibel?
- Können Testfälle mit vertretbarem Aufwand automatisiert werden?
- Ist der Wartungsaufwand der TAS für den geplanten Einsatzzweck akzeptabel?
- Welche Optimierungsmaßnahmen sollten in der TAS umgesetzt werden, um einen besseren Nutzen zu erzielen oder um effizienter damit arbeiten zu können?
- Liefert die automatisierte Testdurchführung tatsächlich den erwarteten Mehrwert und mit welchen zusätzlichen Risiken ist das verbunden?
- Welche potenziellen Risiken wurden identifiziert, die die großflächige Einführung der TAS gefährden könnten?

Fehlen diese wichtigen Antworten, kann dies entweder an einer nicht optimalen Auswahl eines geeigneten Projekts oder einer nicht gut geplanten Durchführung liegen. In jedem Fall sollte nicht mit der großflächigen Bereitstellung fortgefahren

werden, sondern ein erneuter Pilotversuch mit anderen Rahmenbedingungen geplant, durchgeführt und bewertet werden.

Aus der Praxis: Best Practice aus einem Projekt

In einer großen Behörde wurde systematisch, aber manuell getestet. Mit der Zeit wurden zwei Probleme immer deutlicher: Zum einen sank durch viele Testzyklen die Testqualität. Zum anderen gab es bedingt durch das Tagesgeschäft oftmals Ressourcenprobleme im Test. Diesen Problemen sollte mit Testautomatisierung entgegengewirkt werden, da mit ihr viele Regressionstests mit gleichbleibender Qualität möglich wären und de facto ein 24h-Arbeitstag zur Verfügung stehen würde. Eine Unsicherheit bestand jedoch in der Machbarkeit der Umsetzung und somit wurde folgendes Vorgehen geplant und realisiert:

- Es wurde ein kompakter Pilotversuch gestartet. Es wurde eine Liste der Top-10-Testfälle erstellt und diese automatisiert.
- Auf Basis der Ergebnisse aus dem Pilotversuch wurde die Entscheidung getroffen, dass Testautomatisierung in diesem Umfeld funktioniert und Nutzen stiftet.
- Die Testautomatisierung wurde auf weitere Testfälle ausgeweitet.

Neben dem Ergebnis der Machbarkeit hatte dieses Vorgehen noch zwei weitere entscheidende Vorteile: Durch die Begrenzung auf die Top-10-Testfälle wurden schnell Resultate sichtbar und verwertbar bereitgestellt und das Testteam konnte sich mit der Testautomatisierung vertraut machen.

4.1.2 Die Verteilung oder das Deployment

Nachdem durch die Auswertung der Erprobung validiert wurde, dass der großflächige Einsatz der TAS zielführend und mehrwertstiftend ist, sollte mit der tatsächlichen Bereitstellung begonnen werden. Damit ist sowohl die initiale Verteilung der TAS gemeint als auch die Verteilung neuer TAS-Versionen (z.B. aufgrund von adaptiven Wartungsaktivitäten), wobei grundsätzlich die gleichen Herausforderungen bzw. Erfolgsfaktoren gelten. Bei der Verteilung von Aktualisierungen der TAS besteht immer auch ein gewisses Risiko (wie grundsätzlich bei jeder Änderung an einem Softwaresystem), dadurch Verzögerungen bzw. Mehraufwände auszulösen. Vergleichbar mit der Entwicklung von generischen Softwaresystemen hat sich auch für die TAS-Entwicklung gezeigt, dass es effizienter ist, dieses Risiko durch häufige, kontinuierliche und inkrementelle Bereitstellung zu verteilen (vgl. Continuous Integration, Continuous Delivery/Deployment [URL: Fowler]). Anstelle einer kontinuierlichen, inkrementellen Verteilung der TAS kann diese auch an Projektmeilensteine des SUT gekoppelt werden (z.B. Projektbeginn, Sprint-Ende, Code-Freeze bei größeren Releases). In diesem Fall muss allerdings berücksichtigt werden, dass auch die Verteilung der TAS im Prinzip einer Integration zwischen SUT und TAS entspricht und daher, abhängig davon, wie die Synchronisierung und Kompatibilität zwischen SUT und TAS sichergestellt wurde (siehe dazu auch Abschnitt 3.3.1 und 3.3.2), zu Problemen, Wartungsaktivitäten bzw. generell Mehraufwand führen

kann. Ganz grundsätzlich haben sich in der Praxis die in Abbildung 4–2 veranschaulichten Empfehlungen für eine erfolgreiche Bereitstellung als hilfreich erwiesen.

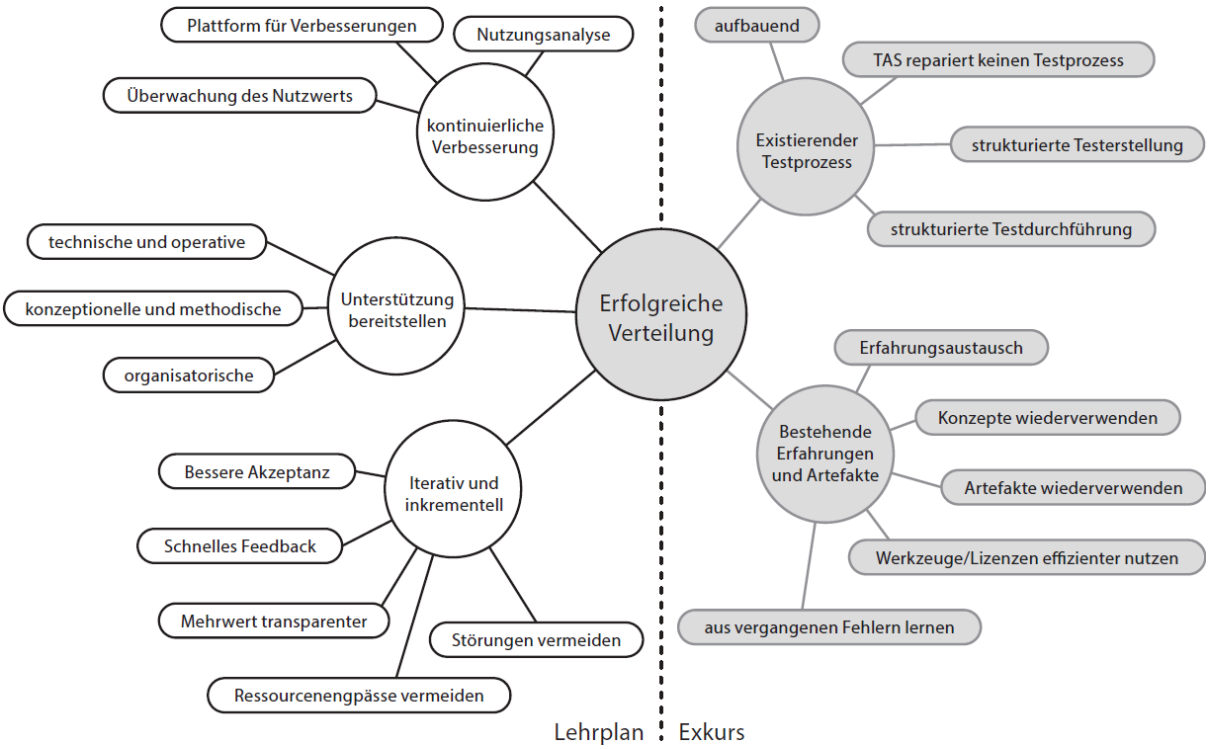

Abb. 4–2
Empfehlungen für eine erfolgreiche Bereitstellung

Iterative und Inkrementelle Verteilungen sind hilfreich

Auch hier ist ein Vergleich mit anderen Softwareentwicklungsprojekten zulässig, bei denen sich ein iteratives und inkrementelles Vorgehen oft als effizienter und effektiver herausgestellt hat [Beck & Andres 04]. Bei einer schrittweisen Einführung einer TAS entstehen weniger Störungen, außerdem ist es bei diesem Vorgehen auch einfacher, notwendige Unterstützung den einzelnen Projekten bereitzustellen, ohne Ressourcenengpässe zu riskieren. Weiterhin kann somit auch vermieden werden, dass Fehler, nicht optimal entworfene Prozesse oder fehlerhaft interpretierte Anforderungen an die TAS zu wesentlichen Widerständen in der Organisation führen. Vielmehr werden diese frühzeitig und in kleinem Rahmen erkannt und können beseitigt werden. Das führt im Allgemeinen zu einer besseren Akzeptanz, höherer Zufriedenheit und einem klar wahrgenommenen Mehrwert, der durch die TAS generiert werden kann. Anpassungen und Optimierungen können dabei sowohl die TAS als auch den TAA-Entwurf betreffen und sollten jedenfalls berücksichtigt werden, bevor ein weiterer Rollout der TAS forciert wird.

Unterstützung bei der Implementierung bereitstellen

Man darf nicht erwarten, dass eine Verteilung ohne aktive Unterstützung der Nutzer erfolgreich ist. Das ist besonders dann wichtig, wenn die Einführung in eine Organisation ohne Erfahrung mit Testautomatisierung passiert, oder auch nach wiederholten Fehlschlägen. Unabhängig davon, sollten bei der Unterstützung jedenfalls folgende Aspekte berücksichtigt werden:

- **Technische und operative Unterstützung** – umfasst sämtliche Informationen oder Maßnahmen, die die Nutzung der TAS erleichtern. Dazu können Checklisten, Schulungsunterlagen, Guidelines oder aktives Mentoring/Coaching sowie Workshops etabliert werden. Wichtig ist, dass diese Maßnahmen nicht nur passiv angeboten, sondern aktiv forciert werden.
- **Konzeptionelle und methodische Unterstützung** – neben der technischen und operativen Unterstützung bei der Arbeit mit der TAS ist bei einer erfolgreichen Verteilung auch der konzeptionelle und methodische Aspekt relevant. Dadurch kann zum Beispiel ein einheitliches und klares Vorgehen bei der Transformation von manuellen Testfällen zu automatisierten Testskripten etabliert werden. Aber auch die Frage nach Kriterien, um entscheiden zu können, unter welchen Umständen Testfälle automatisiert werden sollen, sollte dokumentiert oder diskutiert werden.
- **Organisatorische Unterstützung** – abschließend sollte bei dem Angebot von Unterstützungsmaßnahmen auch die organisatorische Integration als wichtig betrachtet werden. Konkret können dabei unterschiedliche Teamstrukturen bzw. die Integration und Synchronisierung von SUT- und TAS-Entwicklung beschrieben werden und welche Aspekte im konkreten organisatorischen Rahmen besonders berücksichtigt werden müssen.

Kontinuierliche Verbesserung auf Basis von Nutzerfeedback

Nach der Erstverteilung wird leider oft dem laufenden Betrieb einer TAS zu wenig Beachtung geschenkt. In der Praxis führt dieser Umstand häufig dazu, dass eine Lösung nicht optimal genutzt wird, mit fortschreitender Zeit weniger passend für den eigentlichen Anwendungszweck wird und sich der Nutzwert der TAS dadurch kontinuierlich verringert. Um dies zu vermeiden, ist es wichtig, einen Prozess zur kontinuierlichen Verbesserung der TAS zu etablieren. Um diesen Prozess auf Fakten zu basieren, müssen unterschiedliche Quellen für Feedback und Nutzerverhalten berücksichtigt werden:

- Überwachung des kontinuierlich generierten Nutzwerts und der Kosten einer TAS (siehe dazu auch Kap. 5), um einerseits sicherzugehen, dass die

Nutzung und kontinuierliche Investition in die Weiterentwicklung der TAS nach wie vor stattfinden soll. Andererseits aber auch, um festzustellen, welche Änderungen notwendig sind, um diese Ziele weiterhin erreichen zu können.

- Etablierung einer Plattform zur Erhebung und Umsetzung von Verbesserungen, basierend auf Feedback durch Teams, die die TAS tatsächlich nutzen. Durch diese Maßnahme wird erreicht, dass die Weiterentwicklung der TAS mit den praktischen Bedürfnissen der tatsächlichen Nutzer übereinstimmt.
- Eine automatisierte Nachverfolgung der tatsächlichen Nutzung der TAS sollte vor allem bei sehr heterogenen und komplexen Lösungen umgesetzt werden. Auf Basis dieser Informationen zur Nutzung in der Praxis können Module oder Teilkomponenten identifiziert werden, die sehr häufig genutzt werden (und daher eventuell weiter optimiert werden sollten), aber auch welche, die nicht oder nur sehr wenig im Einsatz sind (und daher eventuell außer Betrieb genommen werden sollten). Auf diese Weise ist es auch möglich, Verbesserungsmaßnahmen besser zu priorisieren, um möglichst viel Mehrwert für eine große Anzahl von Nutzern zu generieren und den Nutzwert der TAS insgesamt zu erhöhen.

In Kapitel 8 werden diese und weitere Faktoren zur fortlaufenden Optimierung von Testautomatisierung in höherem Detailgrad beschrieben.

Exkurs: Auf existierende Testprozesse aufbauen

Testautomatisierung ist kein Mittel, um einen schlecht funktionierenden Testprozess zu »reparieren«. Wenn keine Expertise in Testtechniken oder qualifizierte Testressourcen zur Verfügung stehen oder Testen als unstrukturiertes »Probieren« verstanden wird, bringt der Einsatz einer TAS einer Organisation in der Regel keinen oder nur einen sehr geringen Mehrwert. Stattdessen werden Kosten durch die Anschaffung von Werkzeugen sowie durch die Implementierung und Verteilung einer TAS verursacht und im Projekt entstehen zusätzliche Reibungsverluste durch den Zwang, sich damit auseinanderzusetzen, obwohl dadurch kein erkennbarer Mehrwert generiert werden kann.

Das soll nicht bedeuten, dass es notwendig ist, einen bis ins Letzte ausdefinierten Testprozess zu haben. Ein schlanker Prozess, in dem überlegt und strukturiert Testfälle erstellt und durchgeführt werden, reicht im Normalfall aus, um erste nachhaltige Testautomatisierungsschritte umzusetzen.

Exkurs: Auf bestehenden Erfahrungen und Artefakten aufbauen

Gerade in größeren Organisationen gibt es oftmals Automatisierungsprojekte, die in der Vergangenheit erfolgreich umgesetzt, bereits ausgelaufen oder auch erfolglos gescheitert sind. Leider werden diese Expertise, die dabei entworfenen Konzepte oder die implementierten Frameworks oftmals nicht weiterverwendet. Stattdessen kommt es häufig dazu, dass diese Artefakte erneut realisiert werden, was wiederum zu vermeidbarem Mehraufwand und steigender Frustration in einer Organisation führen kann. Dabei liegt in der Wiederverwendung – wo es möglich ist – ein großer Vorteil der Automatisierung. Und auch gescheiterte Ansätze können eine wichtige Entscheidungsgrundlage sein, um Fehler nicht zu wiederholen.

Bei Recherchen im eigenen Unternehmen kommen häufig überraschend viele potenziell wiederverwendbare Ressourcen zutage, die zumindest evaluiert und auf ihre Anwendbarkeit hin geprüft werden sollten. Um diese Herangehensweise zu institutionalisieren, lohnt es sich, gerade in größeren Organisationen zum Beispiel eine Plattform »Erfahrungsaustausch Testautomatisierung« zu etablieren.

Testautomatisierungswerkzeuge bringen in Bezug auf die Wiederverwendung zusätzlich noch die Kostenfrage mit, denn Neuanschaffungen können sehr kostenintensiv sein, obwohl im Unternehmen vielleicht vergleichbare Werkzeuge bereits im Einsatz sind. Abhängig vom geltenden Lizenzmodell kann zum Beispiel bei Floating-Lizenzen die Ausnutzung durch projektübergreifende Verwendung deutlich erhöht werden.

4.2 Strategie für die Bewertung und Begrenzung von Risiken

Mit der Verteilung einer TAS gehen zwangsläufig auch Risiken einher, deren Auftretenswahrscheinlichkeit aufgrund des Einsatzzwecks stark mit der Entwicklung des SUT korreliert. Deren Auswirkung kann oftmals auch eine direkte Konsequenz für Prozesse im SUT haben, z.B. kann eine fehlerhafte Testdurchführung dazu führen, dass ein System nicht für den Abnahmetest freigegeben wird – auch wenn der Fehlschlag nicht zwangsweise auf einen tatsächlichen Fehlerzustand im SUT zurückzuführen ist (also ein falsch positives Ergebnis eines automatisierten Tests). Typische Vertreter dieser Risiken sowie spezifische für die Erst- und Wartungsverteilung relevante Vertreter werden in den nun folgenden Abschnitten beschrieben und mögliche Lösungsansätze dafür vorgestellt.

Typische Risiken

Sehr häufig lassen sich eine Reihe von typischen generischen Risiken identifizieren, die in fast jedem Kontext valide sind. Deren konkrete, spezifische Ausgestaltung mag zwar projekt- oder organisationsabhängig sein, jedoch lassen sie sich grob zu den folgenden Themenkomplexen zusammenfassen:

- Ein zu hoher Grad an Abstraktion erschwert die Arbeit mit automatisierten Testfällen und schreckt potenziell viele Beteiligte ab, sich damit auseinanderzusetzen.
- Statische, konkrete Testdaten werden zu umfangreich oder zu komplex bzw. der Wartungsaufwand für die Pflege dieser Datensets wird zu groß.
- Probleme bei der ausreichenden Bereitstellung von notwendigen Ressourcen (Personal, Zeit, Geld)
- Änderungen oder zusätzliche Entwicklungen in der TAS können den Betrieb stören oder behindern.
- Hoher Aufwand bei der initialen Einführung der TAS
- Hoher Aufwand bei der Anpassung der TAS, die durch Wartungsaktivitäten am SUT ausgelöst wurden (siehe dazu auch Abschnitt 1.1.1)
- Probleme bei der stabilen Interaktion zwischen TAS und SUT

Risiken lassen sich nicht vermeiden.

All diese Risiken lassen sich nicht gänzlich vermeiden, jedoch können deren Auswirkung und Auftretenswahrscheinlichkeit reduziert werden. Außerdem kann der Zeitpunkt, zu dem sie erkannt werden, durch entsprechende Maßnahmen vorgezogen werden. Gerade unerwartet hohe Aufwände für die initiale Einführung und kontinuierliche Wartung können durch einen soliden TAA-Entwurf (siehe Abschnitt 3.2) und durch für den Projektkontext sinnvolle Ansätze zur Testautomatisierung (siehe Abschnitt 3.2.2) vermieden werden. Ebenso lassen sich Probleme bei der stabilen Interaktion zwischen TAS und SUT durch die Wahl einer geeigneten Schnittstelle (siehe die Abschnitte 3.2.3 und 7.2) vermeiden oder zumindest durch den bereits anfangs erwähnten Pilotversuch frühzeitig erkennen. Unzureichend bereitgestellten Ressourcen ist deutlich schwerer zu begegnen, da die Ursache davon oftmals in fehlender Managementunterstützung oder aktivem Widerstand durch unterschiedliche Parteien begründet ist. Begegnen kann man diesem Umstand, indem man kontinuierlich, transparent und kritisch über den Zweck und den erwarteten Nutzen der TAS kommuniziert (siehe Abschnitt 1.1 und Kap. 5).

Häufige Auslöser bzw. verstärkende Faktoren

Wie in Abbildung 4–3 illustriert, treten diese Risiken häufig nicht stetig auf, sondern werden durch bestimmte Projektaktivitäten ausgelöst oder verstärkt. Dazu gehören unter anderem:

- Erst- oder Wartungsverteilung einer TAS in die Zielumgebung
- Migration der TAS in eine andere Umgebung
- Wartungsaktivitäten und Änderungen am SUT

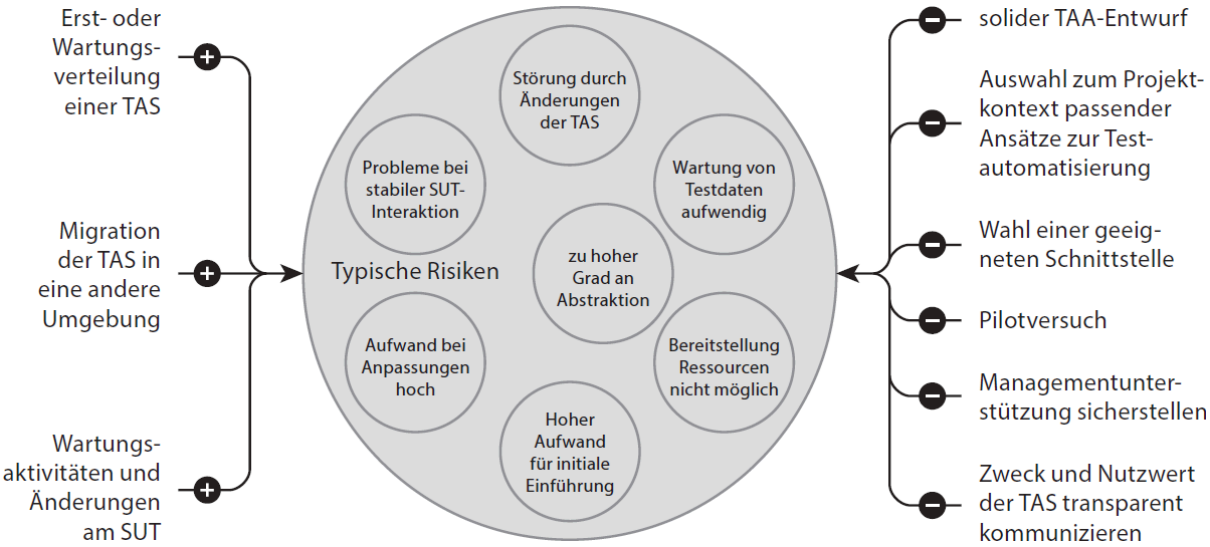

Abb. 4–3

Typische Risiken und deren verstärkende sowie mitigierende Faktoren

Daraus soll allerdings nicht abgeleitet werden, dass diese Projektaktivitäten so gut wie möglich vermieden werden sollen. Vielmehr ist dies ein Hinweis darauf, dass zu diesen Zeitpunkten vermehrt über dadurch hervorgerufene Risiken diskutiert werden muss und mögliche Minderungsstrategien notwendig werden können.

4.2.1 Spezifische Risiken bei der Erstverteilung

Neben den zuvor beschriebenen generischen Risiken, die es zu berücksichtigen gilt, sind gerade mit der Erst- und späteren Wartungsverteilung einige spezifische Herausforderungen verbunden.

Mit der Erstverteilung einer TAS werden jene Schritte bezeichnet, die notwendig sind, um die für die automatisierte Testdurchführung benötigte Testumgebung zu definieren, zu erstellen sowie kontinuierlich weiterzuentwickeln und zu warten. Ebenso zählt dazu die initiale Erstellung sowie die Wartung von Testsuites, die von der TAS ausgeführt werden sollen. Fehlende Erfahrungswerte und Expertise in der Nutzung der TAS können dazu führen, dass es speziell bei der Erstverteilung zu unerwarteten Problemen während dieser Schritte kommt. Einige

häufig anzutreffende Probleme und mögliche Lösungsansätze werden im Folgenden erläutert.

Zu lange Gesamtausführungszeit der Testsuites

Abhängig von dem geplanten Einsatzgebiet einer TAS gibt es unterschiedliche Rahmenbedingungen für die maximale Ausführungszeit einer Testdurchführung. Ein automatisierter Integrationstest als Teil einer CI/CD-Pipeline darf womöglich nicht mehr als 15 Minuten benötigen, wohingegen die Durchführung von umfangreichen Regressionstests im Zuge einer nächtlichen Ausführung durchaus mehrere Stunden dauern darf. So unterschiedlich wie die Rahmenbedingungen sind auch die damit verbundenen Limitierungen, Herausforderungen und entsprechend mögliche Lösungsansätze:

- **Mehr Zeit für die Durchführung bereitstellen** – im einfachsten Fall ist es ausreichend, die längere Ausführungszeit organisatorisch zu berücksichtigen und vom Ergebnis abhängige Projektaktivtäten entsprechend darauf abzustimmen.

Aus der Praxis:

- **Unterschiedlicher Umfang** – oft ist bereits die Ausführung einer Teilmenge der automatisierten Testfälle sinnvoll und führt zu kürzeren Ausführungszeiten. Einerseits kann dies durch Analyse und Selektion von einzelnen Testfällen abhängig von den durchgeführten Änderungen am SUT erreicht werden (Auswirkungsanalyse), andererseits aber auch durch die statische Definition einer Teilmenge für spezifische Zwecke (z.B. Smoke-Tests nach der Verteilung einer neuen Version des SUT). Auch die Reduktion von bestimmten Testfällen entlang einer Risikobewertung ist in der Praxis häufig eine Option, die auch den Wartungsaufwand kürzen kann.
- **Parallelisierung der Testdurchführung** – ist die sequenzielle Durchführung der Tests nicht in einem akzeptablen Rahmen möglich, so kann sie auch parallel erfolgen. Wichtig dabei ist, dass sowohl die automatisierten Testfälle als auch die TAS und die Testumgebung dies erlauben (sie müssen also in beliebiger Reihenfolge und ohne Abhängigkeit untereinander durchgeführt werden können und keine Konflikte durch gemeinsam genutzte Ressourcen oder Testdaten haben).
- **Performanzoptimierung der TAS** – basierend auf umfangreichem Profiling der etablierten TAS sollte auch die Optimierung der TAS-Performanz betrachtet werden. So können beispielsweise oft redundante oder in bestimmten Situationen nicht notwendige Zwischenschritte in Tests reduziert werden.
- **(Teil-)Migration auf andere Schnittstellen** – häufig lassen sich erheblich kürzere Ausführungszeiten erreichen, wenn z. B. neben grafischen Benutzungsschnittstellen auch technische Schnittstellen des SUT genutzt werden (siehe dazu auch Abschnitt 3.2). Es ist nicht notwendig, zwangsweise eine komplette Migration auf andere

Schnittstellen durchzuführen, oft ist es bereits ausreichend, wenn z. B. Testvor- und -nachbereitung auf technische Schnittstellen verlagert werden.

Installations- und Konfigurationsprobleme bei der Testumgebung

Damit eine Testdurchführung gestartet werden kann und valide Ergebnisse liefert, muss im Zuge der Erstverteilung der TAS sichergestellt werden, dass sämtliche dafür notwendigen Vorbedingungen erfüllt sind.

Notwendige Vorbedingungen, um mit einer Testdurchführung starten zu können

Konkret kann das bedeuten, dass notwendige Komponenten und Systeme installiert und konfiguriert sind, die Infrastruktur bereitgestellt ist oder Dienste gestartet werden müssen. Häufig werden diese Aktivitäten manuell, eventuell auf Basis von definierten Checklisten oder dokumentierten Verteilungsverfahren durchgeführt. Dadurch kann eine korrekte und reproduzierbare Installation bzw. Konfiguration nicht garantiert werden, da Menschen Fehlhandlungen begehen und eine angemessene Dokumentation nicht immer vorhanden ist. Dies kann zu einem nicht zu vernachlässigenden Zusatzaufwand führen, der benötigt wird, um die Ursache der nicht aussagekräftigen Testergebnisse zu identifizieren.

Mögliche Maßnahmen, um solche Probleme zu vermeiden, sind:

- Enge Zusammenarbeit und Abstimmung mit Entwicklung und Betrieb, um auftretende Probleme schnell und effektiv zu lösen
- Weitgehende Automatisierung der notwendigen Aktivitäten – durch zeitgemäße Werkzeuge und Methoden (IaC, Chef, Ansible, Cloud-Infrastruktur, Virtualisierung) lassen sich viele manuelle Schritte automatisieren. Manuelle Fehlerquellen und Aufwand können dadurch reduziert, die Reproduzierbarkeit wiederum erhöht werden. Gleichzeitig können die dafür entwickelten Testmittel nicht nur für die Erst-, sondern auch für sämtliche folgenden Wartungsverteilungen wiederverwendet werden. Der kulturelle und technische Rahmen dazu lässt sich gut unter dem Begriff DevOps zusammenfassen, der bereits in Abschnitt 1.5.2 vorgestellt wurde.

4.2.2 Spezifische Risiken bei der Wartungsverteilung

Unter dem Begriff Wartungsverteilung werden all jene Schritte und Aktivitäten zusammengefasst, die notwendig sind, um Änderungen an einer TAS erneut zu verteilen (im Gegensatz zur Erstverteilung, die die TAS an sich von »null weg« bereitstellt). Dazu gehören die Bewertung der Änderungen in der neuen Version der TAS im Vergleich zur alten Version, das Testen der TAS mit Fokus auf neuen

Funktionen sowie potenziellen Regressionen und die Prüfung, ob Testsuites in Abstimmung auf die neue Version der TAS geändert werden müssen. Wie auch bei der Erstverteilung werden im Folgenden einige häufig anzutreffende Risiken und mögliche Lösungsansätze vorgestellt.

Notwendige Änderungen an Testsuites und Testrahmen

Änderungen an der TAS können dazu führen, dass Änderungen an den automatisierten Testfällen oder am Testrahmen notwendig werden. Dadurch entsteht zusätzlicher Aufwand und es kann zu Verzögerungen anderer Projektaktivitäten kommen. Gerade bei sehr umfangreichen Lösungen mit einer großen Anzahl von Testsuiten und Testfällen muss dieser Umstand bei der Planung berücksichtigt werden. Ebenfalls ist es valide, gewisse Änderungen in der TAS nicht bzw. zu einem späteren Zeitpunkt bereitzustellen, wenn ansonsten dadurch wichtige Projektaktivitäten negativ beeinflusst werden.

Ausführliches Testen von umfangreichen Änderungen ist unbedingt notwendig.

Gerade umfangreiche Änderungen sollten ausführlich getestet und validiert werden, bevor sie produktiv bereitgestellt werden. Dazu ist es vorteilhaft, wenn auch für die TAS eine dedizierte Testumgebung genutzt werden kann, die keinen Einfluss auf die produktive Testdurchführung hat. Somit können Fehler frühzeitig erkannt und unmittelbar behoben werden.

Zusätzlich zu ausführlichen Tests der Änderungen führt auch die möglichst kleinteilige, aber in sich konsistente Verteilung dieser Änderungen tendenziell zu weniger Aufwänden bei der Anpassung. Anstelle von einigen wenigen umfangreichen Wartungsverteilung im Jahr sollten diese eher häufig bzw. sogar kontinuierlich erfolgen.

Unerwartete Seiteneffekte durch Änderungen von zentralen TAS-Komponenten

Neben den zuvor erwähnten meist offensichtlich notwendigen Änderungen an Testsuiten und Testrahmen besteht bei Änderungen in der TAS auch immer das Risiko von unerwarteten Seiteneffekten. Eine Änderung an einer Komponente kann zum Beispiel dazu führen, dass die Tests weiterhin durchgeführt werden, aber Fehlerwirkungen im SUT nicht erkannt werden[1]. Die dadurch generierten falsch negativen Testergebnisse führen mittel- und langfristig zu weniger Vertrauen in den Mehrwert der TAS sowie potenziell zu Qualitätsproblemen und hohen Wartungsaufwänden des SUT. Komplett vermeidbar ist dieses Risiko in der Regel nicht, allerdings werden in den Abschnitten 7.2 und 7.3 einige mögliche Herangehensweisen besprochen. So ist es zum Beispiel empfehlenswert, auch für

TAS-Komponenten automatisierte Komponententests sowie statische (manuelle und automatisierte) Testverfahren einzusetzen.

Fehler durch Änderungen an der TAS-Infrastruktur

Auch die TAS-Infrastruktur muss kontinuierlich gewartet, weiterentwickelt und erneuert werden. Durch dabei auftretende Fehler können bestimmte Funktionen der TAS beeinträchtigt oder gar unbrauchbar gemacht werden. Ähnlich wie bei der Erstverteilung kann eine weitgehende Automatisierung der Schritte helfen, diese zumindest teilweise zu vermeiden – allerdings nur in der operativen Durchführung und der Sicherstellung von reproduzierbaren Ergebnissen. Es kann weiterhin dazu kommen, dass notwendige Infrastrukturkomponenten untereinander nicht mehr kompatibel sind (z.B. aufgrund von Änderungen an der Schnittstelle zwischen zwei Systemen) oder dass die Konfigurationsänderungen eine Benutzung von Komponenten unmöglich machen (z.B. aufgrund fehlender Berechtigungen von durch die TAS verwendeten Benutzerkonten). Auch hier empfiehlt es sich daher, Änderungen an der TAS-Infrastruktur (Dokumente, Checklisten, automatisierte Skripte, Konfigurationsvorlagen, …) mit statischen Testverfahren zu verifizieren.

Zusätzliche Fehler oder Performanzprobleme durch die Aktualisierung der TAS

Im Normalfall sollten aktualisierte Versionen einer TAS ausschließlich Verbesserungen (funktional, Performanz) beinhalten und daher auch bedenkenlos verteilt werden können. In der Realität entstehen aber auch bei der TAS-Entwicklung neue Fehler, oder Änderungen verursachen unerwartete Performanzprobleme. Deren Eintrittswahrscheinlichkeit kann zwar wie bereits erwähnt durch unterschiedliche Testverfahren verringert, aber nie vollständig vermieden werden. Sind solche Fehler in neuen TAS-Versionen bekannt, dann muss dies transparent dokumentiert und kommuniziert werden (z.B. in Release Notes), damit auf Basis einer Risiko-Nutzen-Analyse bewertet werden kann, wie damit umzugehen ist. Beispielsweise kann die bekannte Unzulänglichkeit akzeptiert werden, wenn deren Auswirkung vernachlässigbar ist. Eine andere Möglichkeit ist die temporäre Zurückstellung der betreffenden TAS-Version, bis die Fehlerzustände entweder behoben wurden oder ein geeigneter Workaround für den Fehlerzustand bekannt ist.

4.3 Wartung der Testautomatisierung

Das übergeordnete Ziel von Wartungsaktivitäten besteht darin, die Lebens- bzw. Einsatzdauer und die Leistungsfähigkeit einer Testautomatisierung kontinuierlich zu optimieren. Nach der Erstverteilung der Testautomatisierung wird es höchstwahrscheinlich zu mehreren wartungsbedingten Verteilungen kommen,

während die Testautomatisierung bereits produktiv ist. Eine TAS muss sich mit sehr hoher Wahrscheinlichkeit weiterentwickeln, um neuen Zielumgebungen, Schnittstellen, Anforderungen oder gesetzlichen Bestimmungen zu genügen. Ein langfristiger Betrieb einer TAS kann daher nur erfolgreich und rentabel sein, wenn es eindeutige und effektive Wartungsaktivitäten und -prozesse gibt. Der größte Aufwand der Wartungsaktivitäten wird auf die mit der Wartungsverteilung verbundenen Risiken und Eventualitäten fallen.

Wartbarkeit ist für eine Testautomatisierung vermutlich von weitaus größerer Bedeutung als für das zu testende System. Während es den Benutzern eines Softwaresystems verständlicherweise egal sein wird, ob die zugrunde liegende Codebasis der Implementierung eine gute Wartbarkeit unterstützt, trifft eine schlechte wartbare Testautomatisierung das Testautomatisierungsteam (in gewisser Weise die Benutzer eines sehr speziellen Softwaresystems) mit voller Härte.

Zumeist wirkt sich schlechte Wartbarkeit einer Testautomatisierung als Verzögerungen der dynamischen Testaktivitäten aus. Je nach Zeitplan kann dies für das Projekt zu kritischen Situationen führen. Eine in der Praxis ebenfalls häufig erlebte Auswirkung ist ein teils erheblicher Vertrauensverlust in die Testautomatisierung. Wenn Testautomatisierungsentwickler regelmäßig den überwiegenden Teil ihrer Arbeitszeit in das »Nachziehen« von automatisierten Tests investieren, damit diese wieder ausführbar sind, dann sinkt ganz zwangsläufig die Zuversicht des Projektmanagements, mit der Testautomatisierung tatsächlich einen Mehrwert für das Projekt zu generieren. Dies kann so weit gehen, dass mitten im Projekt alle Automatisierungsvorhaben durch das Management abgebrochen werden und zurück zu manuellem Testen gewechselt wird. Durch die vergleichsweise hohen Investitionskosten, um eine Testautomatisierung aufzubauen, in Betrieb zu nehmen und zu halten, sind wartungsbedingte Ausfallzeiten der Testautomatisierung in jedem Fall zu minimieren.

4.3.1 Auslöser und Arten von Wartungsaktivitäten

Wartungsaktivitäten werden durch unterschiedliche Ereignisse ausgelöst. So ist in industriellen Fertigungsstraßen vor allem physischer Verschleiß einzelner Bestandteile der Auslöser für die Wartung. Ventile, die nicht mehr schließen, poröse oder leckende Gummidichtungen oder mechanisch blockierende Scharniere – all dies sind wohlbekannte Beispiele physischen Verschleißes. Wenngleich diese Art des Auslösers in Softwaresystemen nicht zu erwarten ist, so gilt auch – und insbesondere – für industrielle Automatisierungsanlagen, dass ein wartungsbedingter Stillstand, so gut es geht, zu minimieren ist.

Dass Software keinem Verschleißprozess unterliegt, ist natürlich dem immateriellen Charakter von Software geschuldet. Eine interessante Analogie zum physischen Verschleiß von Bestandteilen ist die graduelle Verschlechterung der Wartbarkeit der Codebasis eines Softwaresystems nach Modifikationen bis zu einem Punkt, an dem entschieden wird, dass die Codebasis oder Teile davon von Grund auf und »sauber« neu implementiert werden. Dennoch gibt es für eine Testautomatisierung eine ganze Menge von Auslösern, die Wartungsaktivitäten zur Folge haben. Diese Auslöser lassen sich in zwei Kategorien gliedern:

1. Auslöser, die durch das Testteam bzw. die TAS ausgelöst wurden: Zu dieser Kategorie gehören sowohl die Korrektur von Fehlerzuständen in der TAS oder den automatisierten Testmitteln als auch Verbesserungen funktionaler und nicht funktionaler Aspekte der TAS.
2. Auslöser, die durch das Entwicklungsteam bzw. das zu testende System ausgelöst wurden: Zu dieser Kategorie gehören die Modifikation, Migration oder Außerbetriebnahme des zu testenden Systems.

Je nach der zugrunde liegenden Motivation dieser Änderungen besitzen Wartungsaktivitäten einen adaptiven, korrektiven, präventiven oder verbessernden Charakter (vgl. Abb. 4–4). Details der verschiedenen Wartungsarten, ihre Auslöser und Ziele werden in den nachfolgenden Abschnitten erläutert.

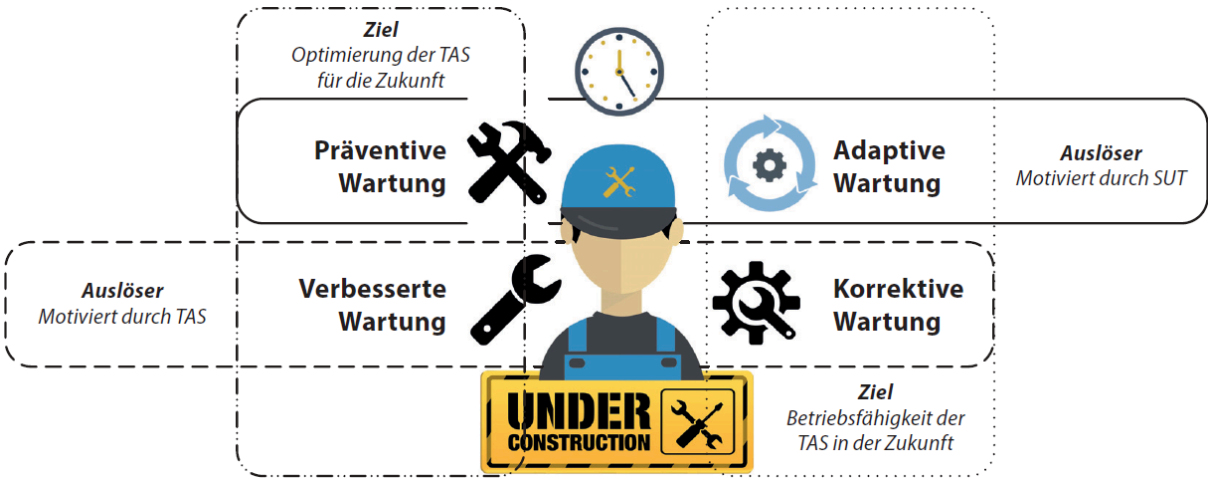

Abb. 4–4
Auslöser und Ziele der Wartungsarten

Adaptive Wartung

Adaptive Wartungsarbeiten werden durch Modifikationen an dem SUT ausgelöst. Sie ziehen üblicherweise notwendige Anpassungen an der Testautomatisierung nach sich. Adaptive Wartungsarbeiten stellen in der Praxis gemeinhin den größten

Anteil aller anfallenden Wartungsaktivitäten dar. Gründe für Modifikationen am zu testenden System gibt es viele, einige davon sind die folgenden:

- Fachliche Änderungen in den Anforderungen oder User Stories des SUT
- Technische Änderungen an den Schnittstellen des SUT
- Gesetzliche Änderungen, die sich auf das SUT auswirken
- Migration des SUT auf neue Zielumgebungen oder Plattformen

Fachliche Änderungen

Fachlich motivierte Modifikationen haben ihren Ursprung so gut wie immer in der Hinzunahme, Außerbetriebnahme oder in Änderungen von Anforderungen, User Stories, Funktionen oder Qualitätsmerkmalen des zu testenden Systems. Jede dieser drei möglichen fachlich motivierten Modifikationen wirkt sich effektiv auf die Testautomatisierung oder die automatisierten Testmittel aus. Bei einer Außerbetriebnahme von Funktionen konzentrieren sich die Wartungsaktivitäten vor allem darauf, die mit den Funktionen in Beziehung stehenden automatisierten Testmittel ebenfalls kontrolliert außer Betrieb zu nehmen. Neben den automatisierten Tests betrifft dies ggf. noch Datenquellen, Konfigurationsdateien, Testbedingungen oder Testmodelle für den automatisierten Testentwurf. Die Außerbetriebnahme der nicht mehr benötigten automatisierten Testmittel sichert ab, dass die Codebasis und sonstige Artefakte der Testautomatisierung möglichst kompakt und konsistent bleiben, es also keine Testmittel mehr gibt, die keine Daseinsberechtigung mehr besitzen. Hilfreich für eine effiziente Außerbetriebnahme ist eine gut etablierte und gepflegte Verfolgbarkeit der Testbasis bis hin zu den automatisierten Testmitteln.

Werden neue Funktionen, Anforderungen oder User Stories hinzugenommen, fallen die üblichen Hauptaktivitäten eines Testprozesses an. Die Testbasis wird analysiert und Testbedingungen werden abgeleitet. Aufbauend auf diesen Testbedingungen erfolgt der Testentwurf (manuell oder automatisiert). Die entworfenen Tests werden abschließend implementiert und für die automatisierte Testdurchführung eingeplant. Oftmals ist die Automatisierung von Tests für neue Funktionalität einfacher als für bestehende Funktionalität, da die Testautomatisierungsentwickler bei einer neuen Funktionalität prinzipiell viel früher in den Prozess einbezogen werden und somit unmittelbar Einfluss auf eine gute Testbarkeit (siehe Abschnitt 2.3) der neuen Funktionalitäten nehmen können.

Die Änderung von bestehenden Anforderungen, User Stories oder Funktionen ist logisch gesehen eine Kombination aus Außerbetriebnahme und Hinzunahme von Funktionen. Zunächst müssen die automatisierten Testmittel identifiziert werden, auf die sich die fachlichen Änderungen auswirken. So müssen beispielsweise Testfälle aus Regressionstestsuiten entfernt werden, da sich die zugrunde liegende Funktionalität geändert hat. Für diesen analytischen Schritt ist

eine lückenlose Verfolgbarkeit immens hilfreich. Eine Möglichkeit, bei fehlender Verfolgbarkeit ganz prinzipiell die von der Änderung betroffenen Testfälle möglichst einfach aufzuspüren, ist die erneute Ausführung der Tests oder einzelner Testsuiten (bspw. Regressionstests) gegen das geänderte SUT. Testfälle, die keine Gültigkeit mehr besitzen oder Anpassungen erfordern, haben ein hohes Potenzial, fehlzuschlagen.

Nachdem die anzupassenden automatisierten Testmittel identifiziert sind, muss analysiert werden, welche Anpassungen notwendig sind. Dazu ist es erforderlich, die geänderte Testbasis zu analysieren, ggf. neue Testbedingungen zu identifizieren, Tests zu entwerfen und zu implementieren, ähnlich zur Hinzunahme neuer Anforderungen, User Stories oder Funktionen. Erst nach der fachlichen Analyse der Testbasis lässt sich entscheiden, ob bestehende Tests obsolet geworden sind und außer Betrieb genommen werden können, eine Anpassung der automatisierten Testmittel möglich oder gar eine komplette Neuentwicklung von automatisierten Testmitteln erforderlich ist.

Technische Änderungen

Technische Änderungen des zu testenden Systems sind in der Praxis allgegenwärtig, da die technischen Voraussetzungen für die Entwicklung bzw. Weiterentwicklung und Optimierung eines Softwaresystems sich ständig ändern. Technische Änderungen können im Zuge von fachlichen Änderungen auftreten, aber auch vollkommen losgelöst von diesen sein. Die Migration eines Systems auf eine Zielumgebung ist eine technische Änderung, der nicht zwingend eine fachliche Änderung zugrunde liegen muss. Weitere Beispiele für rein technisch bedingte Änderungen sind die Unterstützung weiterer Betriebssysteme, Datenbankmanagementsysteme, Webbrowser etc. Die Testautomatisierung muss diese Änderungen reflektieren, damit die automatisierten Tests wieder bzw. überhaupt in der neuen Zielumgebung ausführbar sind.

Eine der wohl am häufigsten auftretenden, rein technischen Änderungen ist die Modifikation von Schnittstellen des zu testenden Systems. Jede der in Kapitel 1 erwähnten Schnittstellen kann technischen Änderungen unterliegen. Ob diese technischen Änderungen im Zuge eines Refactorings oder einer fachlichen Änderung durchgeführt wurden, ist dabei im Grunde unerheblich. In jedem Fall muss die Implementierung der Testadaptierungsschicht für die betroffene Schnittstelle angepasst werden. Je nach Testautomatisierungsansatz muss unter Umständen auch der automatisierte Testfall angepasst werden. Dies sollte jedoch bei rein technischen Änderungen an den Schnittstellen des zu testenden Systems idealerweise nicht erforderlich sein. Es gibt Testautomatisierungsansätze, die inhärent auf einer Trennung von fachlichen und technischen Aspekten eines automatisierten Testfalls basieren wie das schlüsselwortgetriebene Testen oder die prozessgetriebene Skripterstellung. Auch einige Lösungen, die auf der strukturierten Skripterstellung beruhen, unterstützen diese Trennung,

beispielsweise TTCN-3 [URL: TTCN-3]. Durch die Aufteilung von fachlichen und technischen Details in die Testdefinitions- und Testadaptierungsschicht verringern diese Ansätze nachweislich den Wartungsbedarf an der TAS bzw. den automatisierten Testmitteln bei rein technisch bedingten Modifikationen des zu testenden Systems.

Aus der Praxis: Adaptiver Wartungsaufwand als Ausschlusskriterium

In einem »Technical Test Analyst«-Seminar hatte eine Teilnehmerin von ihren Erfahrungen bei der Einführung und dem Betrieb einer TAS in einem traditionell eher manuell ausführenden Testteam berichtet. Die Testautomatisierung stand von Beginn an unter der strengen Beobachtung der Entscheidungsträger, wurde aber auf Drängen der Tester schlussendlich eingeführt. Als Testautomatisierungsansatz wurde die strukturierte Skripterstellung eingesetzt, allerdings gab es keine Trennung von Fachlogik und technischer Anbindung an das zu testende System. Zu Beginn zeigte die Umstellung auf Automatisierung den erhofften Nutzen. Mit zunehmender Zeit jedoch wurde der Wartungsbedarf der zahlreich vorliegenden automatisierten Testmittel immer größer und größer. Eine Vielzahl der Auslöser waren rein technische Änderungen an der Codebasis des zu testenden Systems. Die Tester waren irgendwann überwiegend damit beschäftigt, die Testfälle wieder »gerade zu ziehen«, die durch sich ändernde Schnittstellen des zu testenden Systems nicht mehr ausführbar waren. Gleichermaßen sank das Vertrauen der Entscheidungsträger in die Wirksamkeit der Testautomatisierung, da die Tester nur noch mit Wartungsaktivitäten beschäftigt waren. Die Quintessenz des Vorhabens war, dass das Testautomatisierungsprojekt als gescheitert angesehen und beendet wurde. Die Tester waren gewissermaßen in eine adaptive Wartungsfalle getappt.

Als die Seminarteilnehmerin dann während des Seminars von dem schlüsselwortgetriebenen Ansatz gehört hatte, war sie der Meinung, dass die Testautomatisierung in ihrem Fall vermutlich mit diesem Ansatz nicht gescheitert wäre. Die Tester hatten damals schlichtweg keine Kenntnis von diesem Ansatz gehabt.

Konformität mit Gesetzen und Vorschriften

Gesetze, regulatorische Vorschriften, hersteller- oder branchenspezifische Anforderungen haben mitunter großen Einfluss auf das SUT und die TAS. Eine geänderte Gesetzeslage kann ohne Weiteres einen erheblichen Aufwand für ein Entwicklungsteam bedeuten, um die neuen gesetzlichen Vorgaben im System widerzuspiegeln. Man denke nur an die Einführung der Datenschutz-Grundverordnung (DSGVO) in der Europäischen Union im Jahre 2018. Wird das SUT an die gesetzliche Lage angepasst, wirkt sich dies natürlich auf die Testautomatisierung aus. Testfälle müssen angepasst oder gar neu entworfen werden, um zu validieren, dass das SUT konform zur neuen oder geänderten Gesetzeslage ist. Mitunter müssen auch die automatisierten Testmittel angepasst werden, insbesondere der Schutz personenbezogener oder personenbeziehbarer Daten erschwert den Einsatz von realistischen bzw. Produktivdatensätzen beim Testen. Hier müssen zusätzliche Werkzeuge in die TAS integriert werden, die es ermöglichen, die Daten zu anonymisieren, ohne deren Plausibilität und innere Struktur zu korrumpieren. Eine solche Anforderung wirkt sich direkt auf die TAS und die TAA aus. Ähnlich verhält es sich bei regulatorischen Vorschriften von

Prüfungsbehörden wie beispielsweise dem TÜV. Für viele sicherheitskritische Anwendungen ist die Inbetriebnahme eines Systems an eine Freigabe bzw. Zertifizierung einer Zulassungsbehörde gekoppelt. Diese regulatorischen Stellen haben meist strenge Vorgaben, in welcher Form das Testen zu erfolgen hat, welche Testsprache zu wählen ist, in welcher Form die Testprotokolle zu erfassen und die Berichte abzugeben sind. In einem solchen Szenario ist Konformität zwingend für die Prüfung (oft in Form eines Audits) durch die Behörde einzuhalten. Auch in anderen, nicht zwingend sicherheitskritischen Bereichen ist eine strenge Konformität erforderlich. So veröffentlicht das European Telecommunications Standardisation Institute (ETSI) auf TTCN-3 basierende Konformitätstestsuiten für zahlreiche Kommunikationsprotokolle (darunter auch prominente Vertreter wie LTE), mit denen die Hersteller nachweisen können, dass ihre Implementierungen bzw. Geräte dem jeweiligen Protokollstandard entsprechen. Möchte ein Hersteller nun belegen, dass sein Gerät oder seine Protokollimplementierung konform zu der Spezifikation ist, bleibt ihm keine andere Wahl, als seine TAS mit einer TTCN-3-konformen Testdefinitions- und Testausführungsschicht auszustatten.

Korrektive Wartung

Korrektiver Wartungsbedarf entsteht, wenn in der TAS oder den automatisierten Testmitteln Fehlerzustände identifiziert wurden, die es zu beheben gilt. Korrektive Wartungsauslöser sind also üblicherweise durch das Testteam motiviert. Wir erinnern uns, dass die Entwicklung einer Testautomatisierung wie ein normales Softwareentwicklungsprojekt zu verstehen und zu verwalten ist. Im Grunde steht die Entwicklung einer TAS der Entwicklung eines zu testenden Systems in nichts nach. Eine TAS ist eine spezielle Art eines Softwaresystems, das einzig und allein den Zweck hat, andere Softwaresysteme über deren bereitgestellte Schnittstellen auszuführen und zu testen. Bei der Entwicklung einer TAS oder den automatisierten Testmitteln können Fehlhandlungen begangen werden, die zu Fehlerzuständen führen – ganz genau so, wie das auch bei einem zu testenden System der Fall ist. Idealerweise sind auch die Funktionen einer TAS bzw. die automatisierten Testmittel qualitätsgesichert und getestet (vgl. Kap. 7). Fehlerzustände in einer TAS oder den automatisierten Testmitteln können sich ganz unterschiedlich auswirken. Abstürze der TAS, zu langsame Reaktionen seitens der TAS, (Test-)Datenkorruption, undefinierte Systemzustände bei TAS- oder SUT-seitigem Auftreten von Fehlerwirkungen u.v.m. Eine der offenkundigsten Fehlerzustände in der Testautomatisierung ist jedoch das Auftreten von falsch negativen oder falsch positiven Ergebnissen.

Welche Grundursache einem Fehlerzustand in der Testautomatisierung auch zugrunde liegen mag, sobald erkannt, sollte dieser korrigiert werden. Die korrektiven Wartungsarbeiten an der Testautomatisierung unterscheiden sich wiederum nicht von denen des zu testenden Systems. Fehlerzustände müssen

lokalisiert, korrigiert und getestet werden. Idealerweise sind Fehlermanagementwerkzeuge für die Testautomatisierung im Einsatz, um stets einen Überblick über die gemeldeten Fehler (in Form von Fehlerberichten) zu haben. Nach der Fehlerkorrektur sind mitunter Regressionstests durchzuführen, um abzusichern, dass keine ungewollten Seiteneffekte durch die Fehlerkorrektur in die Testautomatisierung eingebracht wurden.

Weiterführende Details und gute Praktiken, um die TAS bzw. die automatisierten Testmittel zu verifizieren, werden in Kapitel 7 beschrieben.

Aus der Praxis: Wie testet man eigentlich Tests?

Wie wir bereits gelernt haben, sollten sowohl die TAS als auch die automatisierten Testmittel qualitätsgesichert getestet werden. Aber was bedeutet es eigentlich, ein automatisiertes Testmittel zu testen? Müssen dafür ebenfalls Testfälle geschrieben werden, werden diese mit dynamischen Tests getestet?

Die Antwort auf diese Fragen, die zuweilen von Seminarteilnehmern diesbezüglich gestellt wurden, heißt wie so oft: Das hängt davon ab. Für implementierte Komponenten einer TAS sollten dynamische Testfälle geschrieben und durchgeführt werden. Dazu zählen beispielsweise Funktionsbibliotheken einer TAS, Adapterimplementierungen und Komparatoren; im Grunde kann jedes implementierte Artefakt, das Funktionalität bereitstellt, dynamisch getestet werden. Üblicherweise verfährt man dabei so, dass zunächst etablierte, externe Testausführungswerkzeuge verwendet werden. Mit zunehmender Stabilität und Funktionalität einer TAS findet man Szenarien vor, in denen die TAS eingesetzt wurde, um sich selbst zu testen (man spricht auch von dem »Eat your own dogfood«-Prinzip). Dieses reflexive Verfahren wird unter Softwarearchitekten und Programmierern gemeinhin als gute Praktik und als Beleg für die Einsetzbarkeit und Anwendbarkeit einer TAS angesehen.

Eine weitere Möglichkeit, um die korrekte Funktionsweise der TAS nach Änderungen zu überprüfen, ist es, Testfälle mit bekannten Bestanden- oder Fehlgeschlagen-Ergebnissen erneut auszuführen. Ändert sich das Testergebnis eines Testfalls, ohne dass sich das zu testende System oder der automatisierte Testfall geändert haben, ist die Wahrscheinlichkeit groß, dass die Modifikation in der TAS das geänderte Testergebnis bewirkt hat.

Wie verhält es sich aber bei automatisierten Testfällen, diese sind zumeist auch programmiert und liegen in ausführbarer Form vor. Hier kommen zu Qualitätssicherungszwecken überwiegend statische Testverfahren zum Einsatz, insbesondere Reviewtechniken, die den implementierten Testfall auf Konsistenz, Vollständigkeit und Korrektheit überprüfen. Dazu können auch Hilfswerkzeuge wie statische Analysatoren oder Codereview-Werkzeuge eingesetzt werden. Einen dynamischen Testfall zu schreiben, um einen automatisierten Testfall zu testen, ergibt wenig Sinn, da das Ergebnis eines Testfalls ein Testergebnis ist. Ein Testfall für einen Testfall würde demnach einzig überprüfen können, ob das erwartete Testergebnis produziert wurde. Dies würde allerdings keinen wirklichen Mehrwert bringen, da ein solcher dynamische Test für einen automatisierten Test letztlich vergleichbar ist mit der Ausführung des automatisierten Tests.

Verbessernde Wartung

Die Auslöser für verbessernde Wartungsarbeiten haben ihren Ursprung zumeist im Testteam. Bei der verbessernden (oder auch optimierenden) Wartung geht es vor

allem darum, die Anwendbarkeit der TAS effizienter zu gestalten. Es geht also darum, die Testautomatisierung noch leistungsfähiger, benutzerfreundlicher, robuster oder zuverlässiger zu machen. Verbessernde Wartungsarbeiten beziehen sich daher überwiegend auf die nicht funktionalen Eigenschaften einer TAS. So hat beispielsweise die Gebrauchstauglichkeit einer TAS einen hohen Einfluss auf deren Investitionsrendite. Da Tester zumeist täglich mit der TAS interagieren, sollte die Benutzung der TAS möglichst an den Bedarf der Tester optimiert werden.

Üblicherweise werden Optimierungspotenziale während des Einsatzes einer Testautomatisierung in den Projekten durch die Tester bzw. Testautomatisierungsentwickler identifiziert, hoffentlich dokumentiert und schlussendlich in einer Lessons-Learned-Sitzung oder Retrospektive diskutiert, priorisiert und für die Umsetzung terminiert.

Auch wenn die verbessernde Wartung sich im Lehrplan exklusiv auf nicht funktionale Eigenschaften bezieht, so sei an dieser Stelle erwähnt, dass Optimierungspotenziale sich natürlich auch auf funktionale Eigenschaften einer TAS beziehen können. Wird beispielsweise die Funktionalität einer Komponente der TAS verbessert, ohne dass diese Verbesserung durch Änderungen an dem SUT oder einem Fehlerzustand in der entsprechenden Komponente bedingt ist, dann ist dies eine verbessernde Wartungsaktivität, die rein funktionaler Natur ist. Dies sei jedoch nur der Vollständigkeit halber erwähnt. In der Praxis ist es tatsächlich so, dass der überwiegende Anteil der verbessernden Wartungsaktivitäten sich auf nicht funktionale Eigenschaften bezieht.

Präventive Wartung

Mit der vorbeugenden oder präventiven Wartung werden Änderungen bezeichnet, die ihren Auslöser darin haben, die TAS für zukünftige Schnittstellen, Plattformen oder Testarten vorzubereiten. Auch die Unterstützung für den zukünftigen Test technisch unterschiedlicher Versionen des SUT oder für ein komplett anderes SUT kann Auslöser für die präventive Wartung einer TAS sein.

Der Auslöser für präventive Wartungsarbeiten muss aber nicht zwingend mit Änderungen des SUT zusammenhängen. Die Unterstützung eines neuen Betriebssystems (oder einer neuen Betriebssystemversion), mit dem das TAS kompatibel sein muss, ist auch ein Grund für präventive Wartungsarbeiten. Je nach Kontext und Motivation der TAS-Entwicklung kommen präventiven Wartungsarbeiten unterschiedliche Bedeutungen zu. Hersteller einer kommerziellen TAS (bzw. von Werkzeugen, die das automatisierte dynamische Testen unterstützen) entwickeln eine TAS zumeist ohne Bezug zu einem konkreten SUT, sondern zielen darauf ab, entweder eine bestimmte Testart, Schnittstellentechnologie oder einen Testautomatisierungsansatz zu unterstützen. Präventive Wartungsarbeiten sind hier eher als Weiterentwicklungen zu verstehen,

um den Funktionsumfang ihres Produkts zu erweitern. Dort geht es primär um das Ziel, in möglichst vielen verschiedenen Szenarien einsetzbar zu sein. Die Migration einer kommerziellen TAS auf ein neues bzw. weiteres Betriebssystem, die Unterstützung verschiedener Schnittstellentechnologien oder moderner Frameworks erhöhen die Wahrscheinlichkeit, dass das Produkt von vielen Kunden erworben und eingesetzt wird.

Eine TAS-Eigenentwicklung hat zumeist einen sehr engen Bezug zum SUT, für das sie entworfen wurde. Folglich sind die präventiven Wartungsarbeiten hier überwiegend an den Weiterentwicklungsplänen des SUT (oder auch mehrerer SUTs) ausgerichtet. Marktspezifische Vorteile wie die Unterstützung verschiedener Technologien sind hier nicht relevant, sondern nur in dem Maße, in dem diese Technologien beim SUT Einsatz finden.

Aus der Praxis: Adaptive versus präventive Wartung

Mitunter ist es schwierig, zwischen adaptiver und präventiver Wartung zu differenzieren. Wenn beispielsweise feststeht, dass eine Desktop-Applikation im kommenden Jahr für mobile und/oder Webplattformen geöffnet werden soll und der Testautomatisierungsmanager daraufhin veranlasst, die Unterstützung für diese neuen Schnittstellentechnologien durch die TAS zu garantieren, gehören diese Arbeiten eigentlich zur präventiven Wartung, da die Änderung nicht unmittelbar durch Änderungen im SUT motiviert sind. Allerdings wird die Erweiterung der TAS nur deswegen veranlasst, da absehbar ist, dass das SUT diese neuen Technologien zeitnah unterstützen muss. Damit liegen diese Arbeiten wiederum sehr dicht bei den adaptiven Wartungsaktivitäten, denn ohne diese Anpassung des SUT gäbe es keinen Bedarf, die TAS zu ändern. Durch Hinzunahme eines Zeitbezugs lassen sich adaptive und präventive Wartungsaktivitäten jedoch ausreichend gut voneinander abgrenzen. Bei der adaptiven Wartung liegen zunächst Änderungen am SUT vor, die sich nachfolgend auf die TAS auswirken. Bei der präventiven Wartung wird zunächst die TAS modifiziert, um möglichen Änderungen an dem SUT in der Zukunft gerecht zu werden.

4.3.2 Überlegungen zur Dokumentation der automatisierten Testmittel

Die Dokumentation von Entwicklungsartefakten ist in den meisten Projekten ein zweischneidiges Schwert. Unbestritten ist, dass eine gute Dokumentation von Prozessen samt den dazugehörigen Aktivitäten, Architekturen, Abhängigkeiten von Fremdkomponenten, Artefakten und vielem mehr zum Verständnis und zu einem effizienten Betrieb der TAS beiträgt. Zudem konserviert eine gute Dokumentation wichtiges Wissen und macht es langfristig abrufbar. Neue Teammitglieder bekommen die Möglichkeit, sich mit bestehenden Prozessen oder Einschränkungen (bspw. Namenskonventionen, Rechtevergabe im Konfigurationsmanagement) vertraut zu machen und so schneller effektiv etwas beitragen zu können.

Fehlende Dokumentation besse als obsolete Dokumentation

Während das Erstellen einer begleitenden Dokumentation für codebasierte Artefakte von den entsprechenden Werkzeugen, insbesondere von

Entwicklungsumgebungen, bereits halbautomatisch unterstützt wird, liegt bei anderen bereits oben erwähnten Artefakten, wie Architekturen, Fremdkomponenten, Prozessen, zumeist eine rein manuelle Dokumentation vor. Die Dokumentation von Entwürfen, Komponenten, Integrationen, Abhängigkeiten und Verteilungsverfahren erfolgt zumeist nicht werkzeugunterstützt, sodass auch hier manuelle Aktivitäten erforderlich sind, die gleichfalls übersehen werden können.

Allerdings gehen mit der Dokumentation zwei grundlegende und nicht aus der Welt zu schaffende Herausforderungen einher: Jemand muss sie schreiben und jemand muss sie pflegen. Die Dokumentationspflege, die faktisch jede Wartungsaktivität begleitet, wird »aus Zeitgründen« gerne vergessen oder verschoben. Eine veraltete, obsolete Dokumentation ist kritisch, da sie gewissermaßen einem falsch negativen Ergebnis entspricht, dem man zunächst erst einmal vertraut. Im besten Fall sind nur einige wenige Details der Dokumentation veraltet, sodass der Leser die notwendigen Korrekturen selbstständig vornehmen kann, um zum gewünschten Ziel zu gelangen. Im schlimmsten Fall jedoch ist es dem Leser nicht möglich, mit der Dokumentation brauchbare Resultate zu erzielen. Es ist daher unbedingt darauf zu achten, dass die Dokumentation einer Testautomatisierung stets aktuell gehalten wird.

Ziel: angemessene Dokumentation

Anmerkung der Autoren: Während die Inhalte der Dokumentation sich aus dem zu dokumentierendem Artefakt oder Prozess ergeben, gibt es für den Umfang einer guten Dokumentation keine goldene Regel. In dokumentenzentrierten und prozessorientierten Vorgehensmodellen wird in der Regel viel dokumentiert. Mit dem Vormarsch der agilen Vorgehensweisen hielt auch der Trend Einzug, eher weniger zu dokumentieren. Wenn auch nicht explizit in den zwölf Prinzipien des Agilen Manifests [URL: AGILE] erwähnt, so hat sich inzwischen der Begriff einer *angemessenen* Dokumentation etabliert, abgeleitet aus dem zehnten Prinzip des Agilen Manifests: »Einfachheit – die Kunst, die Menge nicht getaner Arbeit zu maximieren – ist essenziell.«

Was bedeutet eigentlich angemessen?

Da der Begriff *angemessen* relativ ist, muss von Fall zu Fall diese Angemessenheit bestimmt werden. So heißt es im Foundation-Lehrplan zum Agile Tester [ISTQB 17] diesbezüglich: »Das Team muss während der Releaseplanung eine Entscheidung darüber treffen, welche Arbeitsergebnisse notwendig sind und bis zu welchem Grad eine Dokumentation der Arbeitsergebnisse erforderlich ist.« Um den Begriff dennoch mit Leben zu füllen, kann man sagen, dass eine Dokumentation *angemessen* ist, wenn sie präzise, konsistent und vollständig die Informationen umfasst, die ein Leser

benötigt, um die inhärente Semantik des dokumentierten Artefakts nachzuvollziehen oder den dokumentierten Prozess durchzuführen.

Eine gute Praktik ist es, die Erstellung und notwendige Anpassungen der Dokumentation als integralen Bestandteil der jeweiligen Aktivität im Entwicklungs- oder Testprozess zu betrachten. Eine Aufgabe gilt erst dann als erledigt, wenn die zugehörige Dokumentation erstellt oder wieder auf dem neuesten Stand ist – und zwar in angemessener Art und Weise.

4.3.3 Der Umfang von Wartungsaktivitäten

Für die Planung von Wartungsaktivitäten ist neben ihren Auslösern und Motivationen vor allem der kontextspezifische Umfang maßgeblich. Der Umfang der Wartungsarbeiten lässt sich durch drei wesentliche Faktoren einschätzen:

- Größe und Komplexität der TAS
- Größe der Änderung
- Risiko der Änderung

Größe und Komplexität der TAS

Die Größe und Komplexität der TAS und der damit verbundenen automatisierten Testmittel sowie der Testumgebung wirken sich natürlich direkt auf den Umfang der Wartungsaktivitäten und insbesondere auf die nachfolgenden Verifikationsaktivitäten aus. Generell ist zu erwarten, dass eine umfangreiche TAS oder eine komplexe Testumgebung eher dazu neigen, unerwünschte Seiteneffekte durch wartungsbedingte Änderungen zu zeigen. Bei der Aufwandsschätzung, um den wahrscheinlichen Wartungsaufwand zu bestimmen, muss dieser Faktor selbstredend miteinbezogen werden.

Größe der Änderung

Die Änderungen an der TAS, der Testumgebung oder den automatisierten Testmitteln variieren von einfachen Fehlerkorrekturen über Aktualisierungen oder Austausch ganzer Komponenten bis hin zur Umstellung auf einen neuen Testautomatisierungsansatz oder der Migration auf eine neue Zielplattform. Wird beispielsweise von der strukturierten Skripterstellung auf einen höheren Ansatz (bspw. schlüsselwortgetriebenes Testen) umgestellt, so ist zu erwarten, dass eine Vielzahl von automatisierten Testmitteln, mindestens jedoch die implementierten Testfälle, auf den neuen Ansatz zu migrieren sind. Dies kann unter Umständen einen erheblichen Aufwand bedeuten.

Größere Änderungen, von denen zu erwarten ist, dass sie sich auf eine Vielzahl von automatisierten Testmitteln auswirken, sollten schrittweise vorgenommen werden. Nach jedem Schritt sollte im Anschluss die Funktionsfähigkeit der TAS

bzw. der automatisierten Testmittel verifiziert werden, sodass bei auftretenden Problemen, die Ursache schnell und präzise bestimmt werden kann.

Risiko der Änderung

Einhergehend mit der Größe der Änderungen, aber nicht zwingend gleichbedeutend damit, ist die Kritikalität einer Änderung. Diese bemisst sich durch die Höhe des Risikos, dass nach Umsetzung der Änderungen an der TAS, der Testumgebung oder den automatisierten Testmitteln die Funktionsfähigkeit nicht mehr gegeben ist. Je kritischer eine Änderung ist, umso sorgfältiger sollte die Umsetzung geplant werden und umso höher ist der zu erwartende Wartungsaufwand. Größe und Risiko stehen oft in einer engen Beziehung zueinander, sind aber gänzlich orthogonale Faktoren. Eine relative kleine Änderung an der TAS kann mit einem großen Risiko verbunden sein und umgekehrt.

Das Risiko einer Änderung bzw. das Risiko, dass die TAS durch die Änderung nicht mehr funktionsfähig ist, variiert auch mit der Wartbarkeit der TAS an sich. Je schlechter die Wartbarkeitsmerkmale einer TAS sind, umso größer ist die Wahrscheinlichkeit, dass Modifikationen zum einen deutlich aufwendiger umzusetzen sind, zum anderen auch eher zu unerwünschten Seiteneffekten führen. Ähnliches gilt für die Testbarkeit einer TAS, der automatisierten Testmittel oder der Testumgebung. Eine schlechte Testbarkeit führt dabei sowohl zu einem größeren Aufwand, um die Korrektheit einer Änderung zu verifizieren, als auch zu einem höheren Restrisiko, da einige Fehlerzustände aufgrund der schlechten Testbarkeit vermutlich unentdeckt bleiben.

Aus der Praxis: Wartungsaufwand und -prozesse sind abhängig vom Kontext

Das wohl am ehesten noch akzeptierbare und bekannteste Beispiel der verschleißbedingten Wartung ist die Inspektion bzw. der Austausch der Verschleißteile beim Auto. Die Wartungsaktivitäten in einer Werkstatt können schon mal einen halben oder ganzen Tag oder mehr in Anspruch nehmen. In anderen Bereichen, in denen Fahrzeuge mit höchster Effizienz funktionieren müssen, haben Fahrzeugingenieure den Wartungsprozess auf wenige Sekunden reduziert: in der Formel 1. Dieses einfache, aber illustrative Beispiel bestätigt in gewisser Weise, dass der jeweilige Kontext die Anforderungen an die Wartungsaktivitäten beeinflusst. Diese Erkenntnis haben wir bereits bei der Planung und Entwicklung einer TAA und TAS kennengelernt – und bereits im Foundation Level lautet eines der sieben Prinzipien des Testens, dass Testen stets kontextabhängig ist [Spillner & Linz 19].

Bewährte Wartungspraktiken einsetzen

Ähnlich wie es sich mit dem Wartungstest eines Softwaresystems verhält, so werden Wartungsaktivitäten an einer sich im Produktivbetrieb befindenden TAS durchgeführt. Da Wartungsaktivitäten stets von anschließenden Verteilungsaktivitäten begleiten werden und diese wiederum verschiedene Herausforderungen und Risiken mit sich bringen (vgl. Abschnitt 4.2), sollten wartungsbedingte Änderungen stets durch Qualitätssicherungsverfahren freigegeben werden. Bereits im Vorfeld dieser Änderungen sollten mittels einer Auswirkungsanalyse die Bereiche der TAS und der automatisierten Testmittel

identifiziert werden, die von der Änderung betroffen sind, und ermittelt werden, wie sich diese Änderungen auswirken können. Dies gilt transitiv auch für die Testmittel, mit denen die TAS qualitätsgesichert wird.

Zu den weiteren bewährten Wartungspraktiken zählen u.a.:

- Die Verteilungsverfahren und die Nutzung der TAS müssen klar dokumentiert sein, siehe Abschnitt 4.1 und 4.2.
- Die Abhängigkeiten von Dritten müssen dokumentiert werden – zusammen mit Nachteilen und bekannten Problemen: Dies gilt sowohl für technische Abhängigkeiten – wenn beispielsweise Fremdkomponenten oder -bibliotheken verwendet werden – als auch für organisatorische oder prozessbedingte Abhängigkeiten, wenn beispielsweise die Wartungsverteilung einer TAS einer Genehmigung bedarf.
- Die TAS muss in einer Umgebung ausgeführt werden, die austauschbar ist oder austauschbare Komponenten hat.
- Wird ein TAF eingesetzt, so müssen die Testskripte von diesem entkoppelt werden: Der Einsatz eines TAF kann die Entwicklung neuer automatisierter Tests erheblich beschleunigen. Allerdings sollte darauf geachtet werden, dass das Testskript nicht mit dem Code des TAF verwoben wird, da so Abhängigkeiten entstehen, die bei einem möglichen Wechsel des TAF nicht mehr aufzulösen sind. Gleichzeitig ist es durchaus denkbar und aus Rentabilitätssicht absolut sinnvoll, ein TAF nicht nur in einem Projekt, sondern in mehreren Projekten einzusetzen. Dies ist ebenfalls nur dann möglich, wenn das TAF und die Testskripte konsequent entkoppelt sind.
- Die TAS muss isoliert von ihrer Entwicklungsumgebung ausgeführt werden, damit Änderungen an der TAS keine Beeinträchtigung der Testumgebung zur Folge haben.
- Die TAS muss zusammen mit Komponenten der Testumgebung, den Testsuiten und Testmitteln unter Konfigurationsmanagement stehen: Das gemeinsame Verwalten aller relevanten Bestandteile einer Testautomatisierung ist eine essenziell wichtige Wartungspraktik. So ermöglicht ein gutes und verlässliches Konfigurationsmanagement die Wiederherstellung der gesamten Testautomatisierung quasi »per Knopfdruck« auf neuen Testrechnern bzw. Zielumgebungen. Dies kann sowohl für neue Mitarbeiter hilfreich und zeitsparend sein als auch für den Wartungstest eines sich im Betrieb befindenden SUT. Letzteres gilt insbesondere für langlaufende SUT, von denen mehrere Versionen zeitgleich im Feld verwendet werden und für die es weiterhin

Wartungsunterstützung geben soll. Dann kann es durchaus vorkommen, dass ein Kunde, der noch eine ältere Version des SUT verwendet, auf einen Fehlerzustand hinweist, den es zwingend zu korrigieren gilt. Durch konsequentes Konfigurationsmanagement ist es der Wartungsabteilung möglich, die für die jeweilige Version gültige Testautomatisierung wiederherzustellen.

4.3.4 Wartung von Fremdkomponenten

In nahezu jeder TAS sind Komponenten oder Bibliotheken »verbaut«, die das TAS-Entwicklungsteam nicht selbst entwickelt hat. Dies können sowohl kommerzielle Komponenten sein (bspw. ein modellbasiertes Testentwurfswerkzeug wie Conformiq [URL: Conformiq] oder die MBTsuite [URL: MBTsuite]) als auch Open-Source-Komponenten (bspw. das schlüsselwortgetriebene Testausführungswerkzeug Robot [URL: Robot]). Auch einzelne Bibliotheken, etwa für einen komfortablen Zugriff auf XML- oder JSON-Datenstrukturen, oder TAF-Implementierungen sind oftmals Bestandteil einer TAS.

Dokumentation von Fremdkomponenten

Um einen genauen Überblick über die Zusammensetzung der TAS zu behalten, empfiehlt es sich dringendst, alle eingesetzten Fremdkomponenten zu dokumentieren und im Konfigurationsmanagement einzubeziehen. Ansprechpartner, Lizenzmodell, Details der Wartungsverträge, all das sind wertvolle und wichtige Informationen für die effiziente Wartung von Fremdkomponenten. Besteht beispielsweise Bedarf, Fehlerzustände in einer Fremdkomponente zu korrigieren oder funktionale Erweiterungen vorzunehmen, muss es einen entsprechenden Prozess dafür geben. Je nach Lizenz der Fremdkomponente fällt dieser wahrscheinlich anders aus. Während bei kommerziellen Produkten üblicherweise der Hersteller für Fehlerbehebungen und Erweiterungen zuständig ist, erlauben Open-Source-Lizenzen üblicherweise die Modifikation der Codebasis durch Dritte. Allerdings unterscheiden sich die Open-Source-Lizenzen teils drastisch. Virale Lizenzen wie GPL erfordern beispielsweise, dass jede Änderung bzw. Anpassung des Open-Source-Codes im kommerziellen Umfeld wiederum quelloffen bereitgestellt werden muss, zusammen mit etwaigem eigens implementiertem Code. Eine klare Dokumentation der Lizenz ist somit erforderlich, um rechtliche Konsequenzen einerseits und den Umfang der erlaubten Änderungen des Quellcodes andererseits zu bestimmen.

Bei kommerziellen Produkten gilt es, die Details des jeweiligen Wartungsvertrags zu dokumentieren. Wartungsverträge sind teils mit hohen Kosten verbunden, umso wichtiger ist es, zu wissen, welche Rechte bzw. Optionen diese Verträge bieten. Werden Fehlermeldungen oder Erweiterungsanfragen per Mail verschickt oder gibt es ein Ticketsystem? Ansprechpartner und

Änderungsprozesse beim Hersteller müssen bekannt sein, um schnell und zielgerichtet agieren zu können.

Aktualisierung von Fremdkomponenten

Eine stets wiederkehrende Aktivität bei der Verwaltung und Wartung von Fremdkomponenten und -bibliotheken ist die Aktualisierung auf neue Versionen sowie das Einspielen von Updates und Upgrades. In der heutigen Zeit werden korrektive Patches und vor allem IT-Sicherheitsupdates regelmäßig über das Internet bereitgestellt. Die Installation der neuesten Version einer Fremdkomponente garantiert stets, aktuell im Hinblick auf Funktionalität, Performanz und IT-Sicherheit zu sein. Es ist eine präventive Maßnahme, deren Aufwand sich langfristig bezahlt macht.

Von besonderer Bedeutung sind die Release und Change Notes eines jeden Updates, da diese sich ggf. auf die Funktionalität der TAS auswirken. Daher sind die Auswirkungen der mit der Aktualisierung verbundenen Änderungen sorgsam zu prüfen. Im Anschluss sollte in jedem Fall ein Smoke- oder Regressionstests gestartet werden, um die Betriebsfähigkeit der TAS zu bestätigen.

4.3.5 Wartung von Schulungsmaterial

Schulungsmaterial wird in der einen oder anderen Form fast immer benötigt. Handelt es sich bei der TAS um eine Eigenentwicklung, die nur innerhalb eines recht kleinen, stabilen Teams verwendet wird, genügt oftmals die angemessene (natürlich! – was denn sonst?!) Dokumentation der TAS. Je mehr Personen jedoch die TAS verwenden, umso größer wird der Bedarf an Schulungsmaterial, um den Einstieg in die TAS zu erleichtern. Handelt es sich bei der TAS um ein kommerzielles Produkt, für das auch noch kommerzielle Schulungen angeboten werden, ist es offensichtlich, dass Schulungsmaterial ein integraler Bestandteil der TAS-Entwicklung ist.

Dokumentation als Basis

Für die Erstellung des Schulungsmaterials kann die Dokumentation der TAS als hilfreiches Ausgangsmaterial herangezogen werden. Allerdings umfasst Schulungsmaterial mehr als die reine Dokumentation der Artefakte und Prozesse. Lernbeispiele, praktische Übungen, Best Practices sowie Tipps und Tricks sind wichtige Bestandteile von Schulungsmaterialien, die sich vermutlich in der Dokumentation nicht finden lassen. Folgende Informationen, die üblicherweise ebenfalls zu den Schulungsmaterialien zählen, können aus einer angemessenen Dokumentation hingegen gut extrahiert und für das Schulungsmaterial wiederverwendet werden:

- Funktionale Spezifikationen der TAS
- Entwurf und Architektur der TAS
- Verteilung und Wartung der TAS

- Nutzung der TAS (Benutzerhandbuch)

Wartung des Schulmaterials

Die Schulungsmaterialien müssen sorgfältig ausgearbeitet und bei jeder Änderung der TAS auf Aktualisierung geprüft werden. Sie unterliegen gewissermaßen den gleichen Anforderungen hinsichtlich Aktualität, wie die Dokumentation der TAS selbst. Nicht aktuelle Schulungsmaterialien sind wenig hilfreich, in einem kommerziellen Umfeld gar kritisch. Der Aufwand für die Wartung des Schulungsmaterials muss erfasst, terminiert und budgetiert werden. Üblicherweise erfolgt die Aktualisierung der Schulungsmaterialien an einem unkritischen Zeitpunkt, beispielsweise am Ende eines Sprints.

Je nach Umfang des Schulungsmaterials und Einsatzzweck bzw. Verbreitung der TAS kann es explizite Trainerrollen für die Durchführung der Schulungen geben. Diese sind dann oftmals auch unmittelbar für die Wartung und Aktualisierung der Unterlagen verantwortlich. Bei Eigenentwicklungen übernehmen zumeist Personen die Trainerrolle, die entweder an der Entwicklung der TAS beteiligt waren oder bereits einen großen Erfahrungsschatz hinsichtlich der Benutzung der TAS besitzen.

4.3.6 Verbesserung der Wartbarkeit

Der Wartbarkeit einer TAS und der dazugehörigen automatisierten Testmittel kommt eine Schlüsselrolle zu, wenn es darum geht, eine Testautomatisierung nachhaltig und verlässlich in einem Unternehmen, einer Abteilung oder einem Projekt zu etablieren und zu betreiben. Wir erinnern uns: Jede Änderung an dem SUT wirkt sich sofort auf die TAS oder die automatisierten Testmittel aus! Um nach den Investitionskosten der TAS, die mit der Anschaffung und Inbetriebnahme einhergegangen sind, auch deren Betriebskosten auf einem akzeptablen Niveau zu halten, ist es unerlässlich sich von Beginn an über eine gute Wartbarkeit Gedanken zu machen. Für diese Überlegungen können die Wartbarkeitsmerkmale des ISO-25010-Standards herangezogen werden – ähnlich wie bei einem *gewöhnlichen* Entwicklungsprojekt.

Bei einem Testautomatisierungsprojekt gibt es neben dem Code der TAS oder einzelner Komponenten noch die automatisierten Testmittel, die stets in Zusammenhang mit der TAS verwendet werden. Dazu zählen insbesondere die automatisierten Testfälle, potenzielle Schlüsselwörter samt Implementierung, in Testbibliotheken zusammengefasste Testschritte sowie Testdaten. Alle diese Artefakte müssen gewartet werden und sind daher auch Gegenstand für Überlegungen zur Wartungsverbesserung.

Namenskonventionen und Standards verbessern

Unter Analysierbarkeit versteht man den Aufwand, der betrieben werden muss, um bei beabsichtigten

Analysierbarkeit.

Änderungen die Auswirkungen auf eine Komponente oder ein System zu bewerten, sie auf Defizite oder Ursachen von Fehlerwirkungen hin zu diagnostizieren oder zu ändernde Teile zu identifizieren. Gute Analysierbarkeit bedingt auch immer gute Lesbarkeit. Lesbarkeit ist nicht nur für Quellcode wichtig, sondern insbesondere auch für die automatisierten Testfälle. Durch den Einsatz von Namenskonventionen und somit einheitlichen Benennungen lässt sich die Lesbarkeit und dadurch die Analysierbarkeit und schlussendlich die Modifizierbarkeit mit einfachsten Mitteln enorm verbessern.

Namenskonventionen können sich auf Variablen und Dateien, Testszenarien, Schlüsselwörter und Schlüsselwortparameter beziehen. Weitere Konventionen betreffen Voraussetzungen und Anschlussmaßnahmen der Testausführung, den Inhalt der Testdaten, die Komponenten der Testumgebung, den Status der Testausführung sowie Ausführungsprotokolle und -berichte. Während die meisten integrierten Entwicklungsumgebungen mit frei definierbaren Quellcodeschablonen aufwarten und eine ganze Reihe von möglichen Programmierrichtlinien, Namenskonventionen und hilfreichen statischen Analysen (bspw. Metriken, um die Komplexität von Quellcode zu berechnen) zur freien Verfügung stehen, ist dies für die meisten automatisierten Testmittel nicht der Fall. Nach welchen Regeln bezeichnet man beispielsweise Schlüsselwörter, wie stellt man einen schnellen Bezug zwischen Schlüsselwörtern und ihren Implementierungen her, wie lassen sich Testfälle am ehesten benennen? All diese Fragen müssen für eine gute Analysierbarkeit beantwortet werden.

Sind solche Namenskonventionen und Richtlinien für ein Projekt, eine Abteilung oder ein Unternehmen definiert und angemessen dokumentiert, leisten sie einen wertvollen Beitrag zur Wartbarkeit. Auswirkungsanalysen können effizienter durchgeführt werden, Modifikationen können zielführender vorgenommen werden. Namenskonventionen und Codierungsregeln lassen sich beispielsweise auch verhältnismäßig einfach mittels Reviews oder statischer Analysatoren überprüfen, sodass Verletzungen mit recht geringem Ressourceneinsatz identifizierbar sind. Ein angenehmer Nebeneffekt von Namenskonventionen ist, dass neue Mitarbeiter sich schneller in die Testautomatisierung einarbeiten können. Das gilt sowohl für die Einarbeitung in die Entwicklung bzw. Weiterentwicklung der TAS als auch für die Implementierung bzw. Anpassung von automatisierten Testmitteln. Damit diese Vorteile vollumfänglich wirksam werden, müssen Namenskonventionen und andere Konventionen bereits bei Beginn eines Testautomatisierungsprojekts vereinbart und dokumentiert werden.

Modularisierung erhöht Wiederverwendbarkeit.

Die Struktur der gTAA ist bereits daraufhin ausgerichtet, dass eine TAS modular implementiert wird. Die architektonischen Schichten trennen

beispielsweise die Testdefinitionen von den Werkzeugen der Testausführungsschicht. Dadurch lassen sich bestimmte Bestandteile einer TAS (theoretisch) unabhängig voneinander austauschen.

Stichwortverzeichnis

A

Abhängigkeitsumkehr 74

Abstraktion 95

Acceptance Test Driven Development 260

Adversarial Testing 201

Agile Softwareentwicklung 8, 27

Aktualität 23

API-Test 10, 21

Architekturebenen 10

Ausbildung 3, 5

Auswahlprozess 48

Außerbetriebnahme 23

Automatische Analyse 202

B

BDD *siehe Behaviour Driven Development*

Behaviour Driven Development 30, 118, 134, 138, 260

Bericht 10, 21, 79, 88, 164, 211, 214, 215

- Berichtsfunktionen 21
- Format 216
- Qualitätsanforderungen 214
- Zielgruppe 215

C

Certified Tester 3

Changemanagement 62

Client-Server 178

CLI-Test 10

Cloud Based Systems 195

Cloud-Services 187
Continuous Deployment 11
Continuous Integration 11, 62
Continuous Testing 11
Cross-Validation 200
Cucumber 93

D

Data Warehouse 190
Datenunabhängigkeit 24
Datenzugriffsroutinen 6
Dependency-Inversion-Prinzip 74
Desktop-Applikationen 177
DevOps 11, 37
DevTestOps 11
Digitalisierung 1
Dokumentation 9, 22, 25, 54, 81, 88, 99, 168, 173, 174, 265, 266, 287, 333, 335, 340
DWH 79, 190
DWH-System 78
Dynamische GUI 193
Dynamischer Test 10

E

Emulation 98
Emulatoren 186
Enhancement Packages 33
Entwicklungszyklus 27
Erprobung 141
Ersetzbarkeit 73
Erweiterbarkeit 51, 72
Evaluierung 50, 53
Evaluierungslizenz 58
Evolution Testing 201
Exploratives Testen 15, 201
Extreme Programming 27

F

Fallstudie 78

Fehlerprotokollierung 283
Fehlersuche 21
Fest definierte Aufgabe 72
Floating 54
Formularlösungen 193
Fremdsoftware 46

G

Gebrauchstauglichkeit 99
Generische Testautomatisierungsarchitektur 49, 69, 75, 77, 78, 80, 89, 92, 135, 141, 176
  Schichten 75, 77, 80, 92
Gherkin 81, 82, 93, 118
Grad der Intrusion 46
Graybox 78
Grenzen der Testautomatisierung 14
gTAA *siehe Generische Testautomatisierungsarchitektur*
GUI-Test 10, 21

H

Hardware 184
Hypothesis Testing 200

I

IaC *siehe Infrastructure as Code*
Infrastructure as Code 39, 130, 155
Inkompatibilität 63
Integration 52
Interface-Segregation-Prinzip 74
Internet of Services 294
Internet of Things 294
Interrupt 184
Intrusion 46
Intuitive Testfallermittlung 201
Investitionskosten 98
IoT *siehe Internet of Things*
ISTQB® 3

K

Kapazität 321

Klasse 27
Kompatibilität 137
Komplexität 3
Komponentenintegrationstest 28
Komponentensegregation 74
Konfigurationsmanagement 77, 79, 87, 136, 172, 179, 245, 261
  einer TAS 87
Kontinuierliche Integration 35
Kriterienkatalog 50
Künstliche Intelligenz 198
L
Last- und Performanztest 18
Liskov-Substitution-Prinzip 73
Lizenzmodell 54
Logik 28
M
Machine Learning 198
Metamorphic Testing 200
Metrik 47, 79, 88, 142, 176, 203, 204, 211, 229, 247
  Beispiele 204
  externe Metrik 204
  interne Metrik 204
Microservices 294
Mobile Applikationen 1, 181
Modularisierung 51
Multi-Layer-Test 185
N
Nachteile der Testautomatisierung 13
Nebeneffekte 8
Netzwerkperformanz 184
Node-Locked 54
Nutzwert 98, 206
O
Objektivität 204
Objektmodell 6, 51

Open-Source-Werkzeuge 56
Open/Closed-Prinzip 72
Optimierung 281
Optische Objekterkennung 297
P
Parallele Verwendung 178
Pilotversuch 143, 144
  Evaluierung 144
  Planen 144
Plattformen 20
Platzhalter 67
Portabilität 65
Priorisierung 8
Produktivität 1
Projektarten 18
Projektmanagement 77, 79, 85, 89, 99, 144, 215, 242, 246, 247
  einer TAS 85
Proof of Concept 53, 64
Protokolle 10, 55, 88, 90, 94, 180, 211, 212, 213, 265, 334
Protokolltest 10
Q
Qualitätsmanagement 1
R
Refactoring 26, 60
Regressionstest 12
Regressionstestsuite 12
Releasemanagement 62
Releasewechsel 33
Reliabilität 204
REST 189
Rückverfolgbarkeit 22
S
Self-Healing 297
Sensitivity Testing 201
Service Oriented Architecture 188, 294

Servicetest 10
Service-Virtualisierung 62
Simulation 98
Simulatoren 186
Single-Responsibility-Prinzip 72
SOA *siehe Service Oriented Architecture*
SOAP 189
Softwareindustrie 2
Softwarequalitätsmerkmale 18
Softwareverteilung 23
SOLID 71, 75
Spezifikationen 6
SUT *siehe System unter Test*
Synchronisierung 138
System unter Test 6, 15, 20, 21, 45, 46, 88, 126, 136, 151, 228, 236
  Architektur 46
  Schnittstellen 45, 47, 126
  Variante 136
Systemintegrationstest 28

T

TAA *siehe Testautomatisierungsarchitektur*
TAF *siehe Testautomatisierungsframework*
TAS *siehe Testautomatisierungslösung*
TDD *siehe Test Driven Development*
Technische Unabhängigkeit 24
Technologien 20
Teilautomatisierte Tests 181
Test Driven Development 27
Test Hook 46
Testadaptierung 79, 94
Testadaptierungsschicht 77, 84
Testarten 18, 48
Testausführung 8, 49, 78, 83, 129, 130, 220, 241, 248, 253, 279, 286
Testausführungsschicht 77, 82
Testautomatisierungsansatz 99

lineare Skripterstellung 101
Mitschnitt-Ansatz 101
modellbasiertes Testen 101
prozessgetriebene Skripterstellung 101
schlüsselwortgetriebener Ansatz 101
strukturierte Skripterstellung 101
Testautomatisierungsarchitektur 18, 19, 21, 69, 70, 236, 261
  Anforderungen 19
  Entwurf 89
  Qualitätsmerkmal 20, 243
Testautomatisierungsentwickler 49
Testautomatisierungsframework 21
Testautomatisierungslösung 9, 11, 18, 69, 70, 71, 131, 161, 227, 235, 239, 243, 263, 265, 275
  Artefakte 134
  Entwicklung 131
  Gebrauchstauglichkeit 267
  Inbetriebnahme 267
  Intrusion 269
  Kompatibilität 260
  Konsistenz 23
  Qualitätssicherung 234
  Wartung 15
Testautomatisierungsmanager 49
Testbarkeit 46, 66
Testdaten 33
  Anonymisierung 41, 331
  Produktionsdaten 41
  synthetische Testdaten 41
  Testdatengenerierung 41
Testdefinition 78, 83, 87, 88, 92, 93, 119, 176
Testdefinitionsschicht 77, 81
Testdurchführung 7
Testentwurfsverfahren 48
Testergebnisse
  falsch negativ 209

falsch positiv 209
valide Testergebnisse 127
Testfallanzahl 7
Testfallbeschreibung 51
Testfallgenerierung 6
Testfallmanagement 62
Testgenerierung 78, 80, 92
Testgenerierungsschicht 77, 80
Testing and Test Control Notation 5, 93, 108, 162, 164, 226
Testklasse 27
Testmanagement 8, 77, 79, 83, 88, 89, 90, 96, 144, 365
Testmittel 10
Testobjekt 26, 34, 71, 75, 92
Testorakel 14
Testplattformen 182
Testskript 22
Teststufen 18
Testsuite 17, 23, 81, 82, 235, 255, 275, 276, 285, 290, 342
Testtreiber 7
Testüberdeckung 8, 27, 134, 193, 202, 210, 285, 294
Testumgebung 22, 33, 129
Treiber 67
TTCN-3 *siehe Testing and Test Control Notation*

U

Überwachung 23
UDDI 189
Umgebungsunabhängigkeit 25
UML 6
Unit Test 26
Unterstützung des Testmanagements 88
Use Case 6

V

Validität 203
Verbrauchsverhalten 321
Verteilung 151

Auslöser 152
Risiken 151
virtualisierte Infrastruktur 130
Virtualisierung 179
Vorbedingungen 10
Vorteile der Testautomatisierung 12

W

Wartbarkeit 11
der Automatisierung 7
Wartemechanismen 283
Wartung 22
Wartungsfreundlichkeit 11
Webapplikation 180
Webservices 188, 294
Werkzeugkompatibilität 49
Wiederherstellung 23
Wiederholbarkeit 12
WSDL 189

Z

Zielgruppe 51
Zielplattformen 186