Certified Tester
Specialist
Test Automation Strategy (CT-TAS)
Syllabus

v1.0

International Software Testing Qualifications Board

Zur Verfügung gestellt von

Austrian Testing Board, German Testing Board und Swiss Testing Board

Urheberschutzvermerk

Änderungsübersicht

Version	Datum	Änderung
Syllabus v1.0 (english)	2024/05/03	CT-TAS v1.0 GA Release
Syllabus v1.0 (deutsch)	2024/11/27	CT-TAS v1.0 deutsche Übersetzung

Inhaltsverzeichnis

Danksagungen

Dieses Dokument wurde von der Generalversammlung des ISTQB® am 3. Mai 2024 formell freigegeben.

Es wurde von der Test Automation Task Force der Specialist Working Group des International Software Testing Qualifications Board erstellt: Graham Bath (Vorsitzender der Specialist Working Group), Andrew Pollner (stellvertretender Vorsitzender der Specialist Working Group und Vorsitzender der Test Automation Task Force), Péter Földházi, Patrick Quilter, Gergely Ágnecz, László Szikszai. Zu den Gutachtern der Test Automation Task Force gehörten: Armin Beer, Armin Born, Geza Bujdoso, Renzo Cerquozzi, Jan Giesen, Arnika Hryszko, Kari Kakkonen, Gary Mogyorodi, Chris van Bael, Carsten Weise, Marc-Florian Wendland.

Technischer Reviewer: Gary Mogyorodi

Die folgenden Personen haben an dem Review, der Kommentierung und der Abstimmung über diesen Lehrplan teilgenommen:

Horváth Ágota, Laura Albert, Remigiusz Bednarczyk, Jürgen Beniermann, Armin Born, Alessandro Collino, Nicola De Rosa, Wim Decoutere, Ding Guofu, Istvan Forgacs, Elizabeta Fourneret, Sudhish Garg, Jan Giesen, Matthew Gregg, Tobias Horn, Mattijs Kemmink, Hardik Kori, Jayakrishnan Krishnankutty, Ashish Kulkarni, Vincenzo Marrazzo, Marton Matyas, Patricia McQuaid, Rajeev Menon, Ingvar Nordström, Arnd Pehl, Michaël Pilaeten, Daniel Polan, Nishan Portoyan, Meile Posthuma, Adam Roman, Pavel Sharikov, Péter Sótér, Lucjan Stapp, Richard Taylor, Giancarlo Tomasig, Chris Van Bael, Koen Van Belle, Johan Van Berkel, Carsten Weise, Marc-Florian Wendland, Ester Zabar.

ISTQB Working Group Advanced Level Test Automation Engineer (Ausgabe 2016): Andrew Pollner (Vorsitz), Bryan Bakker, Armin Born, Mark Fewster, Jani Haukinen, Raluca Popescu, Ina Schieferdecker.

Die folgenden Personen waren an der deutschen Übersetzung des Lehrplans beteiligt: Manfred Baumgartner, Jürgen Beniermann, Armin Born, Jan Giesen (Leiter der Arbeitsgruppe), Richard Seidl, Carsten Weise, Marc-Florian Wendland.

Am Beta-Review der deutschen Übersetzung waren die folgenden Personen beteiligt: Jessica Heymann, Thomas Letzkus, Uwe Martena, Ralf Pichler und Nishan Portoyan.

Unser herzlicher Dank geht an Ursula Zimpfer für ihre wertvolle Unterstützung bei der Bearbeitung der deutschsprachigen Fassung des vorliegenden Lehrplans.

0 Einleitung

0.1 Zweck dieses Lehrplans

Dieser Lehrplan bildet die Grundlage für die internationale Qualifikation als Certified Tester Test Automation Strategy Specialist (CT-TAS) des Softwaretest-Qualifizierungsprogramms des International Software Testing Qualifications Board (im Folgenden ISTQB® genannt). Das German Testing Board e. V. (im Folgenden GTB® genannt) hat diesen Lehrplan in Zusammenarbeit mit dem Austrian Testing Board (ATB) und dem Swiss Testing Board (STB) in die deutsche Sprache übersetzt. Das GTB® und das ISTQB® stellen den Lehrplan folgenden Adressaten zur Verfügung:

1. Nationalen Mitgliedboards, die den Lehrplan in ihre Sprache(n) übersetzen und Schulungsanbieter akkreditieren dürfen. Die nationalen Mitgliedboards dürfen den Lehrplan an die Anforderungen ihrer nationalen Sprache anpassen und Referenzen hinsichtlich lokaler Veröffentlichungen berücksichtigen.
2. Zertifizierungsstellen zur Ableitung von Prüfungsfragen in ihrer nationalen Sprache, die an die Lernziele dieses Lehrplans angepasst sind.
3. Schulungsanbieter zur Erstellung von Lehrmaterialien und zur Bestimmung angemessener Lehrmethoden.
4. Zertifizierungskandidaten zur Vorbereitung auf die Zertifizierungsprüfung (entweder als Teil einer Schulung oder unabhängig davon).
5. Der internationalen Software- und Systementwicklungs-Community zur Förderung des Berufsbildes des Software- und Systemtesters und als Grundlage für Bücher und Fachartikel.

0.2 Testautomatisierungsstrategie im Softwaretest

Die Qualifikation zum Test Automation Strategy Specialist richtet sich an alle, die mit Softwaretests und Testautomatisierung zu tun haben. Dazu gehören beispielsweise Tester, Testanalysten, Testautomatisierungsentwickler, Testberater, Testarchitekten, Testmanager und Softwareentwickler. Diese Qualifikation eignet sich auch für alle, die ein grundlegendes Verständnis der Testautomatisierung erlangen wollen, wie z. B. Projektmanager, Qualitätsmanager, Softwareentwicklungsmanager, Business-Analysten, IT-Direktoren und Unternehmensberater.

Der Lehrplan Test Automation Strategy Specialist (CT-TAS) stellt mehrere Faktoren vor, die bei der Planung der Testautomatisierung innerhalb einer Organisation eine Rolle spielen. Die Aspekte der technischen Umsetzung von Methoden und Best Practices der Testautomatisierung sind nicht Gegenstand des Lehrplans, da sie im separaten ISTQB®-Lehrplan CTAL Testautomatisierungsentwicklung (@istqb:ct-tae) behandelt werden.

Die Testautomatisierungsstrategie befasst sich mit den Anforderungen an die Testautomatisierung, die über die Herausforderungen der technischen Werkzeugrealisierung und der Integration hinausgehen. Ein strategischer Blick auf die Testautomatisierung bietet eine Vision für eine systematische und konsistente Umsetzung in allen Projekten innerhalb einer Organisation, die letztendlich einen Mehrwert für die Organisation darstellt.

0.3 Karriereweg für Tester und Testautomatisierungsentwickler

Das ISTQB®-Schema unterstützt Testexperten in allen Phasen ihrer Karriere und bietet sowohl eine breite als auch eine tiefe Wissensbasis. Personen, die die ISTQB®-Zertifizierung Test Automation Strategy Specialist erlangen, sind möglicherweise auch an der Qualifikation Testautomatisierungsentwicklung (@istqb:ct-tae) interessiert.

Personen, die die Zertifizierung zum ISTQB® Certified Tester Test Automation Strategy Specialist erlangen, können sich auch für die Core Advanced Levels (Test Analyst, Technical Test Analyst und Testmanager) und danach für die Expert Levels (Test Management oder Improving the Process) interessieren. Für alle, die ihre Fähigkeiten im Bereich des Testens in einer agilen Testumgebung ausbauen möchten, kommen die Zertifizierungen Agile Technical Tester oder Agile Test Leadership at Scale in Frage. Die Spezialisten-Lehrpläne bieten einen tiefen Einblick in Bereiche, die spezifische Testansätze und Testaktivitäten beinhalten, z. B. Testautomatisierungsentwicklung, Performanztests, Sicherheitstests, KI-Tests und Tests mobiler Anwendungen, oder in denen domänenspezifisches Know-how erforderlich ist (z. B. Automotive Software Testing oder Game Testing). Aktuelle Informationen zum ISTQB® Certified Tester Scheme finden Sie unter *www.istqb.org*.

0.4 Geschäftlicher Nutzen

In diesem Abschnitt werden die geschäftlichen Nutzen (Business Outcomes, BO) aufgeführt, die von einem Kandidaten erwartet werden, der die Zertifizierung Testautomation Strategy Specialist erreicht hat.

Ein Certified Tester Test Automation Strategy Specialist kann ...

Kode	Beschreibung
TAS-BO1	Software- und Systemfaktoren verstehen, die den Erfolg der Testautomatisierung beeinflussen
TAS-BO2	Kosten und Risiken der Implementierung einer Testautomatisierungslösung identifizieren
TAS-BO3	Rollen und Verantwortlichkeiten von Personen verstehen, die an der Testautomatisierung beteiligt sind
TAS-BO4	Integration der Testautomatisierung über Teststufen hinweg planen
TAS-BO5	Strategische Überlegungen zur Testautomatisierung in verschiedenen Softwareentwicklungslebenszyklus-Modellen anstellen
TAS-BO6	Anwendbarkeit und Durchführbarkeit von Testautomatisierung verstehen
TAS-BO7	Testautomatisierungslösungen planen, die den organisatorischen Anforderungen entsprechen
TAS-BO8	Einsatzstrategien für die Testautomatisierung verstehen
TAS-BO9	Abhängigkeiten der Testautomatisierung innerhalb der Testumgebung verstehen
TAS-BO10	Kosten für die Einrichtung und Wartung von Testautomatisierung ermitteln
TAS-BO11	Metriken kennen, die die Entscheidungsfindung für eine Testautomatisierung unterstützen
TAS-BO12	Mehrwert erkennen, den eine Testautomatisierung für das Projekt und die Organisation bringt
TAS-BO13	Anforderungen an die Testautomatisierung und die Testberichterstattung identifizieren, um die Bedürfnisse der Stakeholder zu erfüllen
TAS-BO14	Übergangsaktivitäten vom manuellen Testen zu einer Testautomatisierung definieren
TAS-BO15	Eine Testautomatisierungsstrategie definieren, die sicherstellt, dass Projekte Ressourcen und Methoden gemeinsam nutzen, um eine konsistente Umsetzung innerhalb der Organisation zu gewährleisten

0.5 Überprüfbare Lernziele und kognitiver Wissensstand

Die Lernziele (Learning Objectives, LO) unterstützen den geschäftlichen Nutzen und dienen zur Ausarbeitung der Prüfungen zum Certified Tester Test Automation Strategy Specialist.

Im Allgemeinen sind alle Inhalte dieses Lehrplans auf den Stufen K2 und K3 prüfbar, mit Ausnahme der Einleitung und der Anhänge. Das heißt, vom Prüfling kann gefordert werden, einen Schlüsselbegriff oder ein Konzept aus einem der sechs Kapitel wiederzuerkennen, sich daran zu erinnern oder wiedergeben zu können. Die spezifischen Lernziele sind zu Beginn eines jeden Kapitels angegeben und wie folgt klassifiziert:

- K2: Verstehen
- K3: Anwenden

Weitere Einzelheiten und Beispiele für Lernziele finden sich in Anhang A.

Alle Begriffe, die als Schlüsselbegriffe direkt unter den Kapitelüberschriften aufgelistet sind, müssen bekannt sein, auch wenn sie nicht ausdrücklich in den Lernzielen erwähnt werden.

0.6 Die Zertifizierungsprüfung zum Test Automation Strategy Specialist

Die Prüfung für das Zertifikat Test Automation Strategy Specialist basiert auf diesem Lehrplan. Zur Beantwortung einer Prüfungsfrage kann Wissen aus mehreren Abschnitten dieses Lehrplans erforderlich sein. Alle Abschnitte dieses Lehrplans sind prüfungsrelevant, außer der Einführung und den Anhängen. Im Lehrplan sind Standards, Fachbücher und andere ISTQB®-Lehrpläne als Referenzen genannt; deren Inhalt ist jedoch nur insoweit prüfungsrelevant, als er im vorliegenden Lehrplan in zusammengefasster Form enthalten ist.

Weitere Einzelheiten zur Prüfung für das Zertifikat Test Automation Strategy Specialist können im Dokument "Exam Structures and Rules" v1.1 des ISTQB® gefunden werden.

Voraussetzung für die Teilnahme an der Zertifizierung Test Automation Strategy Specialist ist, dass die Kandidaten Interesse an Softwaretests und Testautomatisierung haben. Es wird jedoch dringend empfohlen, dass die Kandidaten auch

- ein Mindestmaß an Erfahrung in der Software- und Systementwicklung, in der Leitung bei der Implementierung von Technologien in einem Unternehmen sowie Erfahrung als Senior Tester, Testleiter oder als Softwareentwickler haben,
- einen Kurs absolvieren, der nach ISTQB®-Standards akkreditiert ist (durch eines der vom ISTQB anerkannten Mitgliedsgremien).

Anmerkung zu den Anforderungen: Das ISTQB®-Zertifikat Certified Tester Foundation Level muss vor der ISTQB®-Zertifizierungsprüfung Certified Tester Test Automation Strategy Specialist erworben werden.

0.7 Akkreditierung

Ein nationales ISTQB®-Mitgliedboard kann Schulungsanbieter akkreditieren, deren Lehrmaterial diesem Lehrplan entspricht. Die Akkreditierungsrichtlinien können bei diesem nationalen Board (in Deutschland: German Testing Board e.V.; in der Schweiz: Swiss Testing Board; in Österreich: Austrian Testing Board)

oder bei einer der Organisationen bezogen werden, die die Akkreditierung im Auftrag des nationalen Boards durchführt. Eine akkreditierte Schulung ist als konform mit diesem Lehrplan anerkannt und darf eine ISTQB®-Prüfung als Teil der Schulung enthalten.

Die Akkreditierungsrichtlinien für diesen Lehrplan folgen den allgemeinen Akkreditierungsrichtlinien, die von der ISTQB-Arbeitsgruppe "Processes Management and Compliance" veröffentlicht wurden.

0.8 Umgang mit Normen und Standards

Im Lehrplan für die Test Automation Strategy wird auf einige Normen verwiesen (z. B. IEEE- und ISO-Normen). Der Zweck dieser Verweise besteht darin, einen Rahmen zu schaffen (wie bei den Verweisen auf *ISO/IEC 25010 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE)* (2023) bezüglich der Qualitätsmerkmale) oder eine Quelle für zusätzliche Informationen zu bieten, falls der Leser dies wünscht. Bitte beachten Sie, dass der Lehrplan die Normdokumente als Referenz verwendet. Die Inhalte der Normen sind nicht prüfungsrelevant. Weitere Informationen zu Normen finden Sie in *Kapitel 11*.

0.9 Auf dem Laufenden bleiben

Die Softwarebranche verändert sich schnell. Um mit diesen Veränderungen umzugehen und den Beteiligten den Zugang zu relevanten und aktuellen Informationen zu ermöglichen, haben die ISTQB®-Arbeitsgruppen auf der Website *www.istqb.org* Links angelegt, die auf unterstützende Dokumente und Änderungen an Standards verweisen. Diese Informationen sind im Rahmen des Lehrplans für Test Automation Strategy nicht prüfungsrelevant.

0.10 Detaillierungsgrad

Der Detaillierungsgrad dieses Lehrplans ermöglicht international einheitliche Kurse und Prüfungen. Um dieses Ziel zu erreichen, besteht der Lehrplan aus:

- Allgemeinen Lehrzielen, die die Absicht des Lehrplans zum Test Automation Strategy Specialist beschreiben.
- Einer Liste von Begriffen, die die Schulungsteilnehmer kennen müssen.
- Lernzielen der einzelnen Wissensgebiete, die die zu erreichenden kognitiven Lernergebnisse beschreiben.
- Einer Beschreibung der Schlüsselkonzepte, einschließlich Verweisen auf Quellen wie anerkannte Literatur oder Normen.

Der Inhalt des Lehrplans ist keine Beschreibung des gesamten Wissensbereichs des Softwaretestens; er spiegelt den Detaillierungsgrad wider, der in den Schulungskursen für Test Automation Strategy Specialist behandelt wird. Der Schwerpunkt liegt auf Testkonzepten und -verfahren, die auf alle Softwareprojekte angewendet werden können, auch auf solche, die nach agilen Methoden durchgeführt werden. Dieser Lehrplan enthält keine spezifischen Lernziele, die sich auf das agile Testen beziehen, aber er diskutiert, wie diese Konzepte in agilen Projekten und anderen Projektarten angewendet werden können.

0.11 Aufbau des Lehrplans

Es gibt sechs Kapitel mit prüfbarem Inhalt. Die Überschrift auf der obersten Ebene jedes Kapitels gibt die Zeit für das Kapitel an. Unterhalb der Kapitelebene werden keine Zeitangaben gemacht. Für akkreditierte Ausbildungskurse verlangt der Lehrplan mindestens 12,75 Unterrichtsstunden, die sich wie folgt auf die sechs Kapitel verteilen:

- Kapitel 1: 45 Minuten - Einführung und Ziele der Testautomatisierungsstrategie
  - Der Tester versteht die Konzepte der Testautomatisierung und lernt die Auswahlkriterien für Kandidatenprojekte kennen.
  - Der Tester versteht die Faktoren, die eine erfolgreiche Implementierung der Testautomatisierung definieren.
- Kapitel 2: 60 Minuten - Ressourcen für die Testautomatisierung
  - Der Tester lernt die verschiedenen verfügbaren Lösungen für die Testautomatisierung und die jeweiligen Investitionskosten kennen.
  - Die Softwarelizenzierung für Werkzeuge zur Testautomatisierung wird behandelt.
  - Der Tester versteht, welche Fähigkeiten für die Testautomatisierung erforderlich sind.
- Kapitel 3: 225 Minuten - Vorbereitungen für die Testautomatisierung
  - Der Tester lernt, wie die Testautomatisierung über die verschiedenen Teststufen und innerhalb von Teststufen eingesetzt wird.
  - Testautomatisierungsstrategien zur angemessenen Verteilung von Tests und zur Erreichung von Shift-Left und Shift-Right werden behandelt.
  - Der Tester lernt, wie Testautomatisierung Standard- und agile Projekte unterstützt.
  - Testautomatisierung im Rahmen von DevOps und kontinuierlichem Testen wird behandelt.
  - Der Tester versteht, wie man Kriterien für den Einsatz von Testautomatisierung definiert, einschließlich Tests, die sich am besten für die Testautomatisierung eignen.
- Kapitel 4: 135 Minuten - Organisatorische Einführungs- und Freigabestrategien für die Testautomatisierung
  - Der Tester erfährt, wie die Testautomatisierung die Markteinführung beschleunigen kann.
  - Der Tester versteht, wie er betriebsrelevante automatisierte Tests und Fehlerberichte entwickeln kann.
  - Die Testautomatisierungsstrategie und Risikominderung werden behandelt.
  - Der Tester lernt die Testautomatisierungsumgebung und ihre Abhängigkeiten kennen.
  - Die Integration von Testautomatisierung und Testdaten in ein System unter Test wird behandelt.
- Kapitel 5: 150 Minuten - Auswirkungsanalyse der Testautomatisierung
  - Metriken und Berichte zur Testautomatisierung, die bei der Entscheidungsfindung helfen, werden behandelt.

  - Ein Tester lernt, wie er eine Investitionsrechnung für die Testautomatisierung durchführen kann.
  - Ziele für eine Organisation und ein Projekt zur Nutzung der Testautomatisierung werden behandelt.
  - Der Tester lernt, wie er Testberichte analysieren und Entscheidungsträger auf klare und verständliche Weise informieren kann.
- Kapitel 6: 150 Minuten - Realisierungs- und Verbesserungsstrategien für die Testautomatisierung
  - Der Tester lernt, wie er von manuellen Tests zur Testautomatisierung und zum kontinuierlichen Testen übergehen kann.
  - Evaluierung der Testautomatisierung zur kontinuierlichen Verbesserung wird behandelt.

1 Einführung und Ziele der Testautomatisierungsstrategie - 45 Minuten (K2)

Schlüsselbegriffe

Testautomatisierungsarchitektur, Testautomatisierungsframework, Testautomatisierungsstrategie

Lernziele für Kapitel 1: Die Lernenden können ...

1.1 Erfolgsfaktoren eines Testautomatisierungsprojekts

TAS-1.1.1 (K2) ... Ziele und Zielsetzungen einer Testautomatisierungsstrategie definieren

TAS-1.1.2 (K2) ... technische Erfolgsfaktoren eines Testautomatisierungsprojekts identifizieren

TAS-1.1.3 (K2) ... geeignete Investitionskriterien bei der Auswahl von Projekten zur Testautomatisierung zusammenfassen

1.1 Erfolgsfaktoren eines Testautomatisierungsprojekts

1.1.1 Ziele und Zielsetzungen einer Testautomatisierungsstrategie definieren

Bei der Definition der Strategie für ein Testautomatisierungsprojekt müssen folgende Punkte berücksichtigt werden:

- Definition des beabsichtigten Zwecks
- Identifizierung der Risiken
- Definition des Umfangs
- Identifizierung der beteiligten Stakeholder
- Auswahl der Werkzeuge für die Automatisierung
- Entwurf der Testautomatisierungsarchitektur (TAA)
- Identifizierung von Umgebungen

Zu den Zielsetzungen der Testautomatisierung können gehören:

- Verbesserte Effizienz der Tests
- Breitere und tiefere Überdeckung
- Verbesserte Gesamtqualität des SUT
- Verringerung der Gesamtkosten und der Markteinführungszeit
- Durchführung von Tests, die für manuelle Tester nicht ausführbar sind
- Zeitlich verkürzte Testdurchführung
- Steigerung der Anzahl an Testdurchführungen

Da die agile Softwareentwicklung schnellere Bereitstellungszyklen ermöglicht und Cloud-Anwendungen weit verbreitet sind, erfordert die Entwicklung ein früheres und schnelleres Feedback über die Qualität des Systems unter Test (SUT). Als Ergebnis dieser Verschiebung hat die Testautomatisierung aufgrund ihrer Art und ihrer Testziele einen größeren Fokus und eine höhere Relevanz in modernen Projekten.

1.1.2 Technische Erfolgsfaktoren eines Testautomatisierungsprojekts identifizieren

Die folgenden Erfolgsfaktoren gelten für Testautomatisierungsprojekte, die sich auf Merkmale fokussieren, die den langfristigen Erfolg eines Projekts beeinflussen. Der Einsatz eines Pilotprojekts hilft bei der Bestimmung der Werkzeuge und der Realisierbarkeit durch die ausgewählten Technologien.

Zu den Erfolgsfaktoren für die Testautomatisierung gehören:

- Testbarkeit des SUT
  - Ermöglicht der Testautomatisierung den Zugriff auf SUT-Schnittstellen.
- Definierte Testautomatisierungsstrategie
  - Die Strategie muss anwendbar und anpassbar sein; ihre Ziele müssen innerhalb von Zeit- und Kostenvorgaben erreichbar sein und sie muss auf dem neuesten Stand gehalten werden.

- TAA
  - Klarheit darüber, was und wie zu implementieren ist.
  - Für weitere Informationen siehe ISTQB-CTAL-TAE (2024), Abschnitt 3.1.1.
- Testautomatisierungsframework (TAF)
  - Ein TAF, das einfach zu benutzen, gut dokumentiert und wartbar ist, unterstützt einen konsistenten Ansatz zur Automatisierung von Tests. Für weitere Informationen siehe ISTQB-CTAL-TAE (2024), Abschnitt 3.1.3.
  - Definierte und implementierte Testberichterstattung
  - Einfache Fehlersuche
  - Geeignete Testumgebung
  - Dokumentierte automatisierte Testfälle
  - Verfolgbarkeit der automatisierten Tests zu Testfalldefinitionen und Anforderungen
  - Einfache Wartung
  - Automatisierte Tests auf aktuellem Stand
  - Ein eindeutiger Einführungsplan (engl. deployment plan)
  - Ein eindeutiger Plan für die Testdurchführung
  - Tests, die nach Bedarf ausgemustert werden
  - Effektive Behandlung von Ausnahmen

Vor Beginn des Testautomatisierungsprojekts ist es wichtig, die Erfolgschancen des Projekts zu analysieren, indem die gegebenen und die fehlenden Faktoren betrachtet werden, wobei die Risiken des gewählten Testansatzes sowie der Projektkontext im Auge behalten werden müssen. Nicht alle Faktoren sind für alle Projekte erforderlich und in der Praxis werden selten alle Faktoren erfüllt.

1.1.3 Geeignete Investitionskriterien bei der Auswahl von Projekten für die Testautomatisierung zusammenfassen

Testautomatisierung in einem Projekt aufzubauen, ist mit Aufwand und in den meisten Fällen auch mit erheblichen Kosten verbunden. Dies sollte zusammen mit der Art des Projekts und der Frage, ob die Testautomatisierung in dem Projekt eingesetzt werden soll, berücksichtigt werden.

Folgende Investitionskriterien sind zu berücksichtigen, bevor Testautomatisierung eingeführt wird:

- Kosten der Einführung: Neben dem Aufwand für den Aufbau der Testautomatisierung werden zusätzliche Kosten für das Projekt anfallen, da möglicherweise neue Testautomatisierungsentwickler (TAEs) eingestellt, neue Hardware gekauft oder Schulungen durchgeführt werden müssen.
- Aktuelle Phase im Softwareentwicklungslebenszyklus (Software Development Life Cycle, SDLC) des Projekts: Es ist besser, mit Testautomatisierung so früh wie möglich zu beginnen, um früher einen größeren Nutzen zu erhalten.

- Erwartete/geplante Dauer des Projekts/der Softwareentwicklung: Bei einem kürzeren Projekt stehen möglicherweise nicht genügend Ressourcen zur Verfügung, um mit der Testautomatisierung zu beginnen, oder es bleibt nicht genug Zeit, um damit einen Mehrwert zu schaffen.
- Kosten für die Wartung: Wenn man bei null anfängt, nimmt die Einrichtung einer Testautomatisierungslösung (Test Automation Solution, TAS) Zeit in Anspruch und sie muss dabei auch gewartet werden.

Wenn der Aufwand für die Einführung der Testautomatisierung in einem Projekt vertretbar ist, kann die Vorbereitung für die Testautomatisierung beginnen. Dazu gehört die Auswahl des richtigen Testansatzes und der Werkzeuge für die Testautomatisierung. Die Hauptverantwortung für diesen Prozess liegt in der strategischen Rolle eines Testarchitekten oder Testmanagers, der über das notwendige Wissen zur Testautomatisierung verfügt, um entsprechende Entscheidungen zu treffen.

2 Ressourcen für die Testautomatisierung - 60 Minuten (K2)

Schlüsselbegriffe

Testautomatisierungsentwickler, Testautomatisierungslösung

Lernziele für Kapitel 2: Die Lernenden können ...

2.1 Kosten und Risiken der Implementierung einer Testautomatisierungslösung

TAS-2.1.1 (K2) ... alternative technische Lösungen im Hinblick auf die Betriebskosten vergleichen

TAS-2.1.2 (K2) ... Überlegungen zum Lizenzmodell für Werkzeuge zur Testautomatisierung erläutern

TAS-2.1.3 (K2) ... Beispiele für Faktoren nennen, die bei der Definition einer Testautomatisierungsstrategie zu berücksichtigen sind

2.2 Rollen und Verantwortlichkeiten innerhalb der Testautomatisierung

TAS-2.2.1 (K2) ... die Rollen und Fähigkeiten zusammenfassen, die für eine erfolgreiche Testautomatisierungslösung notwendig sind

2.1 Kosten und Risiken der Implementierung einer Testautomatisierungslösung

2.1.1 Alternative technische Lösungen im Hinblick auf die Betriebskosten vergleichen

Ein gängiger Ansatz in Bezug auf die Betriebskosten ist eine maßgeschneiderte Inhouse-Lösung auf der Grundlage von Open-Source- oder kommerziellen Werkzeugen. Weitere Alternativen sind nicht-kundenspezifische kommerzielle Lösungen oder der Abschluss eines Vertrags mit einem Outsourcing-Unternehmen, um die Arbeit der TAEs erledigen zu lassen.

Durch eine Inhouse-Entwicklung der TAS wird sichergestellt, dass alle Kosten, Ressourcen, Risiken sowie ihre Steuerung innerhalb der Organisation liegen (z. B. Teammitglieder/Entwickler sowie Hardware- und Software-Ressourcen, die für die eigentliche Lösung erforderlich sind). Dabei ist ein Schlüsselfaktor, dass bei diesem Ansatz und der Reservierung von Zeit für diese Tätigkeit, die TAS ohne Vertrag mit einem Anbieter oder Outsourcing-Unternehmen entwickelt werden kann, da alle erforderlichen TAEs und deren Wissen innerhalb der Organisation verfügbar sind. Für eine erfolgreiche Umsetzung muss die Organisation jedoch über die richtigen TAEs verfügen, die eingestellt oder geschult werden und die Entwicklung der TAS vorantreiben können.

Bei der Arbeit mit einer herstellerbasierten Lösung wird die Verantwortung zwischen dem Kunden und dem Hersteller aufgeteilt, je nach den Einzelheiten des Vertrags. Wenn es in der Organisation bereits ein Testwerkzeug gibt, das pilotiert wurde und alle Anforderungen erfüllt, und kein Bedarf an zusätzlichen Funktionen für das Testwerkzeug besteht, wird dieser Ansatz einfacher umzusetzen sein. Die Tester der Organisation können produktiv arbeiten, sobald sie die erforderliche Schulung vom Anbieter erhalten haben, und in der Regel gibt es innerhalb der Organisation Fachexperten (Subject Matter Experts, SMEs), die als Produktverantwortliche eingesetzt werden. Das Risiko bei diesem Ansatz ist, dass zusätzliche Arbeiten (z. B. Behebung von Fehlerzuständen und weitere Anforderungen an das Testwerkzeug) Zeit in Anspruch nehmen und Verhandlungen mit dem Anbieter erfordern, um sie rechtzeitig und in korrekter Weise zu erledigen, was die Arbeit der Tester, die dieses Testwerkzeug verwenden, beeinträchtigen kann.

Wenn das Unternehmen die Verantwortung für den Aufbau eines Teams nicht übernehmen möchte, ist Outsourcing eine empfehlenswerte Lösung. Bei der Zusammenarbeit mit Outsourcing-Firmen muss der Kunde keine Hardware oder Software anschaffen oder Mitarbeiter mit den erforderlichen Qualifikationen rekrutieren. In der Regel werden der gesamte Aufwand und die zusätzlichen Kosten vom Outsourcing-Unternehmen übernommen, das auch die entsprechenden Werkzeuge zur Verfügung stellt. Im Vertrag müssen messbare Erwartungen und Metriken festgelegt werden, um den Fortschritt transparent und sichtbar zu machen. Dieser Ansatz empfiehlt sich, falls die Organisation aufgrund von kurzen Projektlaufzeiten, Kosten oder anderen Erwägungen nicht in die Einstellung von TAEs investieren möchte.

2.1.2 Überlegungen zum Lizenzmodell für Werkzeuge zur Testautomatisierung erläutern

Die Lizenzierung ist ein wichtiger Aspekt, der bei der Einführung der Testautomatisierung zu berücksichtigen ist. Jedes der unten genannten Lizenzierungsmodelle hat unterschiedliche Auswirkungen auf die Gebrauchstauglichkeit und die Kostenfaktoren, z. B. die Kosten für die Testdurchführung und die Schwierigkeiten bei der Einrichtung der Entwicklungsumgebung.

Zu den Lizenzierungsmodellen gehören:

- **Open Source:** In vielen Fällen setzen Unternehmen Open-Source-Werkzeuge ein, um ihre Testziele bei der Testautomatisierung zu erreichen. Der Hauptgrund dafür ist, dass für die Nutzung der

Testwerkzeuge keine Lizenzkosten oder Gebühren für die Wartung anfallen. Viele Anwender und Organisationen tragen zur Entwicklung von Open-Source-Werkzeugen bei, was die Informationsbeschaffung und den Support erleichtert. Die meisten Open-Source-Lizenzen erlauben es außerdem, die Werkzeuge bei Bedarf zu modifizieren oder sogar ohne nennenswerte Einschränkungen weiterzuveröffentlichen.

- **Lizenz pro Benutzer/Maschine:** Kommerzielle Werkzeuge werden oft pro Benutzer oder Maschine lizenziert. Was die Kosten betrifft, so können die Werkzeuge effizient eingesetzt werden, wenn die Organisation die Anzahl der TAEs kennt, die langfristig an einem bestimmten Projekt arbeiten werden. Je mehr Lizenzen eine Organisation bestellt, desto günstiger ist oft der angebotene Preis.
- **Floating-Lizenz:** Dies ist eine Lizenz, die von mehreren Personen in der Organisation gemeinsam zu unterschiedlichen Zeiten genutzt werden kann. Sie kann sehr nützlich sein, wenn es viele TAEs gibt, die auf verschiedenen Rechnern arbeiten. Die Anzahl der Lizenzen wird anhand der gleichzeitigen Nutzung berechnet, nicht anhand der Gesamtzahl der TAEs oder der spezifischen Rechner, auf denen sie ihre Tests ausführen.
- **Runtime-Lizenz:** Viele kommerzielle Werkzeuge verfügen über Runtime-Lizenzen für die Ausführung der Testautomatisierung. Diese Art von Lizenzen gibt es bei Cloud-Anbietern, die Werkzeuge zur Testautomatisierung, mehrere Betriebssysteme (Operation System, OS) und Browserversionen als Service anbieten. Es gibt auch Cloud-Anbieter, die Zugang zu Gerätefarmen mit einer Vielzahl von mobilen Geräten, Plattformen, Versionen und Mobilfunknetzen bieten. Diejenigen, die diese Dienste in Anspruch nehmen, zahlen nur für die Zeit, die sie für die Ausführung von Tests benötigen, und werden daher auf der Basis einer Runtime-Lizenz abgerechnet.

2.1.3 Beispiele für Faktoren nennen, die bei der Definition einer Testautomatisierungsstrategie zu berücksichtigen sind

Viele Faktoren können Entscheidungen über die Testautomatisierung und deren Strategie beeinflussen.

Die wichtigsten Faktoren sind:

- Zeitliche Beschränkungen
- Erforderliches Fachwissen und Anzahl der TAEs zur Entwicklung der TAS
- Test-Hardware
- Lizenzen für Testwerkzeuge
- Anpassbarkeit
- Wartung
- Unterstützung für verschiedene Plattformen (z. B. Web, Desktop und/oder Mobile)
- Unterstützung für kontinuierliche Integration/kontinuierliche Bereitstellung (CI/CD)
- Einbindung des Testmanagements und der Testberichterstattung

Der erste und wichtigste Kostenfaktor ist der Zeitplan für die durchzuführenden Arbeiten und die Testautomatisierung selbst.

Wenn die Frist (engl. Deadline) kurz ist, werden häufig nur einige wenige qualifizierte TAEs für die Erstellung der TAS eingestellt. Im Hinblick auf eine längere Roadmap und einen längeren Zeitplan kann die

Organisation die Anzahl der erforderlichen TAEs in einem größeren Zusammenhang festlegen. Wenn das SUT wächst, sich verbessert und komplexer wird, kann die Entscheidung getroffen werden, mehr TAEs für die Implementierung und Wartung der TAS und der Tests einzustellen.

Weitere wichtige Kostenfaktoren sind die Werkzeuge und ihre Lizenzen. Das Budget sollte die erforderlichen Testwerkzeuge, Hardware und zusätzliche Schulungen für die TAEs enthalten. Diese Faktoren stehen in engem Zusammenhang mit der Integration in andere Werkzeuge und Systeme, um zusätzliche, nicht zum Testen gehörende Funktionen wie das Hochladen von Testergebnissen in das Testmanagementsystem oder das Auslösen eines Builds im Konfigurationsmanagementsystem auszuführen. Für diese zusätzlichen Funktionen gibt es kostenlose oder kostenpflichtige Werkzeuge und Bibliotheken, die jedoch bei der Erstellung der Testautomatisierungsstrategie berücksichtigt werden müssen und im Budget eingeplant werden sollten.

Wie viele Testumgebungen und Runtime-Agents ein Entwicklungsteam hat, ist immer kontextabhängig. Je größer und komplexer das SUT und der Releaseplan sind, desto mehr Ressourcen benötigt die Organisation, was auch die Kosten erhöht. Ein neuerer Ansatz zur Begrenzung der Ressourcenkosten ist die Nutzung von Cloud-Anbietern und die Bezahlung von Test-Hardware-Ressourcen und Laufzeitlizenzen auf Abruf, was jedoch mit Vorsicht zu genießen ist, da es sich auch nachteilig auswirken kann. Manchmal können Maschinen in Betrieb bleiben und teurer werden als auf ihrer eigenen Hardware. Dieser Ansatz kann die hohen Kosten für Wartung und Betrieb für das TAE-Team reduzieren.

2.2 Rollen und Verantwortlichkeiten innerhalb der Testautomatisierung

2.2.1 Die Rollen und Fähigkeiten zusammenfassen, die für eine erfolgreiche Testautomatisierungslösung notwendig sind

Für eine erfolgreiche TAS benötigen Unternehmen qualifizierte TAEs, die über fundierte Kenntnisse in Programmierung und technischen Architekturen verfügen sollten.

Je nach Größe und Reife des Projekts sollte es außerdem mindestens einen erfahrenen Fachexperten geben (z. B. einen Testleiter, Architekten und/oder Business-Analysten), der die eigentliche Geschäftsdomäne und die Testziele versteht. Die Rolle soll helfen, das Konzept auf der Grundlage der Testziele zu erstellen und voranzutreiben sowie eine Roadmap für die eigentliche TAS festzulegen. Es werden Teammanagement und Soft Skills benötigt, um das Team für die eigentliche Arbeit aufzubauen, zu schulen und zu motivieren (siehe auch ISTQB-CTEL-TM (2011)).

Ein TAE sollte über ausgeprägte technische Fähigkeiten und Kenntnisse sowohl zu verschiedenen SDLCs als auch über die Architektur des SUT und dessen Entwicklungsumgebung verfügen. Abgesehen von den technischen Aspekten sollte ein TAE die Fähigkeit besitzen, mit Testanalysten und anderen Stakeholdern hinsichtlich der Ziele der Testautomatisierung zusammenzuarbeiten. Da vollständiges Testen nicht erreichbar ist, gilt dies auch für die Testautomatisierung: Eine 100-prozentige Überdeckung ist auch mit der Testautomatisierung in der Regel nicht zu erreichen. Da Zeit und Aufwand begrenzt sind, müssen die TAEs in der Lage sein, die aus geschäftlicher und Investitionssicht wichtigsten Testbedingungen zu priorisieren, indem sie zu Risikobewertungen beitragen.

3 Vorbereitungen für die Testautomatisierung - 225 Minuten (K3)

Schlüsselbegriffe

API-Test, Komponententest, Shift-Left, Shift-Right, System unter Test, Testautomatisierungsansatz, Testbedingung, Testdouble, Testpyramide, Teststufe, Vertragstest

Lernziele für Kapitel 3: Die Lernenden können ...

3.1 Integration über Teststufen hinweg

TAS-3.1.1 (K2) ... zwischen verschiedenen Testverteilungsmöglichkeiten der Testautomatisierung unterscheiden

TAS-3.1.2 (K2) ... einen Testautomatisierungsansatz auf der Grundlage der Architektur des Systems unter Test auswählen

TAS-3.1.3 (K3) ... Wege zur Optimierung der Testautomatisierung aufzeigen, um Shift-Left- und Shift-Right-Ansätze zu erreichen

3.2 Strategische Überlegungen zu verschiedenen Modellen des Softwareentwicklungslebenszyklus

TAS-3.2.1 (K2) ... erläutern, wie Testautomatisierungsprojekte mit bestehenden Softwareentwicklungslebenszyklus-Modellen übereinstimmen

TAS-3.2.2 (K2) ... erläutern, wie Projekte zur Testautomatisierung mit den Best Practices der agilen Softwareentwicklung übereinstimmen, die die Testautomatisierung unterstützen

TAS-3.2.3 (K3) ... Testautomatisierungsprojekte mit DevOps Best Practices vorbereiten, um kontinuierliches Testen zu erreichen

3.3 Anwendbarkeit und Durchführbarkeit der Testautomatisierung

TAS-3.3.1 (K2) ... Kriterien zur Bestimmung der Eignung von Tests für die Testautomatisierung erläutern

TAS-3.3.2 (K2) ... Herausforderungen identifizieren, die nur durch Testautomatisierung gelöst werden können

TAS-3.3.3 (K2) ... Testbedingungen identifizieren, die schwierig zu automatisieren sind

3.1 Integration über Teststufen hinweg

3.1.1 Zwischen verschiedenen Testverteilungsmöglichkeiten der Testautomatisierung unterscheiden

Mike Cohn entwickelte das ursprüngliche Konzept einer Testpyramide, die drei Ebenen beschreibt: Unit (das ISTQB® nennt diese Komponenten), Service und UI. Seitdem sind viele Variationen erschienen, die alle die gleichen Grundprinzipien hinsichtlich der Beschreibung der Rolle von Teststufen und der Verteilung der Testfälle auf diese Stufen verwenden (siehe ISTQB-CTFL (2023), Abschnitt 5.1.6 Testpyramide).

In der ursprünglichen Testpyramide von Mike Cohn liegt der Schwerpunkt darauf, welche Testarten von der TAS durchgeführt werden. Die Ebene Service kann in drei Testarten unterteilt werden: Komponentenintegrationstests, Vertragstests und API-Tests.

Unit-Tests (das ISTQB® nennt sie Komponententests) dienen der Validierung einzelner Komponenten und konzentrieren sich auf die Codequalität. Komponentenintegrationstests ermöglichen die Validierung der Benutzungsschnittstelle (UI) und der API durch den Einsatz von Testdoubles wie Mocks und Platzhalter. Der Vertragstest (Contract Testing) ermöglicht die Validierung der Verträge zwischen den Diensten. API-Tests konzentrieren sich auf die funktionale Validierung von bestimmten Diensten mit realen Daten über Dienstverbindungen. UI-Tests sind Ende-zu-Ende-Tests eines Systems über die grafische Benutzerschnittstelle (GUI).

Es ist hilfreich, den aktuellen Stand des Testens als Ausgangsbasis und den Zielzustand festzuhalten. Dies vermittelt ein klares Verständnis davon, was fehlt oder was ein akzeptables Testniveau darstellt. Daraus geht hervor, wie viele Tests auf jeder Teststufe durchgeführt werden und ob es sich um manuelle oder automatisierte Tests handelt. Der Zielzustand hängt auch vom Zeitplan ab und davon, was innerhalb dieses Zeitrahmens erreicht werden kann. Wenn dies machbar ist, sollte die Pyramidenform die Zielform sein.

Beispiele für Testverteilungen:

- **Pyramide:** Hierbei handelt es sich um eine ausgewogene Verteilung der Tests, wobei weniger Tests auf den höheren Teststufen und mehr auf den niedrigeren Teststufen mit stabilen und schnelleren Tests durchgeführt werden. Wenn die Verfügbarkeit von Ressourcen und der Zeitrahmen es zulassen, ist dies oft die Zielzustandspyramide.
- **Eistüte:** Dies ist die umgekehrte Version der Pyramide. Es gibt einen ausgewogenen Anteil an Servicetests, aber das Testen ist stark davon abhängig, die meisten Fehlerzustände in der UI-Teststufe zu finden, die aufgrund ihrer Komplexität in der Regel kostspieliger zu automatisieren ist. Aufgrund des Mangels von Komponententests werden Fehlerzustände erst später im SDLC gefunden.
- **Sanduhr:** Das Testen ist auf der höchsten und niedrigsten Teststufe stark ausgeprägt, während Tests auf Serviceebene meistens fehlen, was zu Fehlerzuständen bei der Integration führt. Wenn die Geschäftslogik durch APIs (d. h. Dienste) bereitgestellt wird, können viele der UI-Tests leicht auf niedrigere Teststufen verlagert werden.

- **Schirm:** Das Testen ist vollständig von kostspieligen UI-Tests abhängig. Dies führt zu einer späten Fehlerbeseitigung, einer kostspieligen Wartung der Testfälle und der TAS. Wenn es technisch nicht möglich ist, Testfälle auf niedrigeren Ebenen zu implementieren, ist eine Abkehr von der Schirmform möglicherweise nicht realisierbar, und der Schwerpunkt sollte auf der Optimierung der UI-Testautomatisierungssuite, der Reduzierung der Testdurchführungsdauer und der Verbesserung der Stabilität liegen.

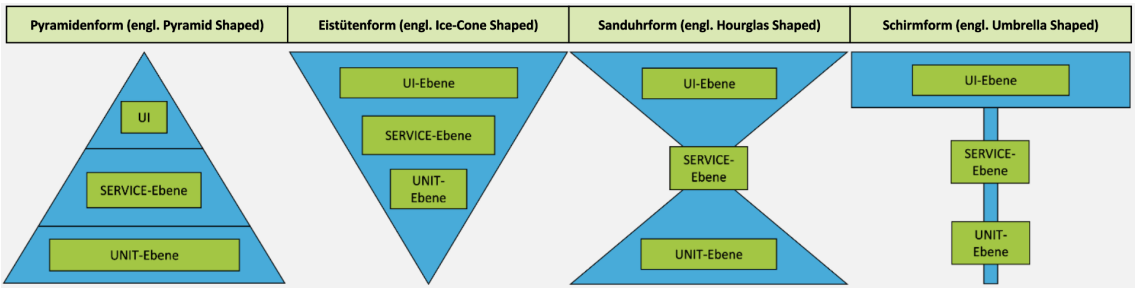

Abbildung 1: Beispiele für Testverteilungen

3.1.2 Einen Testautomatisierungsansatz auf der Grundlage der Architektur des Systems unter Test auswählen

Es gibt viele Möglichkeiten, die Teststufen in einer Teststrategie zu definieren. Welche man wählt, hängt stark von verschiedenen Faktoren ab, wie z. B. der Unternehmenskultur, der allgemeinen Reife des Software-Engineerings, dem befolgten SDLC und ob das SUT eine Mainframe- oder Microservices-Architektur hat.

Obwohl die Pyramidenform als ideale Testverteilung angesehen wird, ist sie nicht immer erreichbar oder es benötigt eine längere Zeit, um zu diesem Endzustand zu gelangen. Es empfiehlt sich, eine realistische Form als Zielzustand festzulegen, z. B. die Umwandlung von der Schirmform in die Sanduhrform. Sobald dies erreicht ist, kann ein neuer Zielzustand festgelegt werden.

Die Wahl der geeigneten Verteilung der Testautomatisierung für Mainframe-Systeme unterscheidet sich von der modernen Softwareentwicklung und hängt wiederum von vielen Faktoren ab, da einige der Stufen möglicherweise überhaupt nicht automatisiert werden können. Die Zugänglichkeit zu Mainframes erfolgt traditionell über Terminal-Emulationen (d. h. Green Screens). Die Validierung der Batch-Prozesse ist möglich, aber begrenzt. Das Testen von Komponenten ist schwieriger zu bewerkstelligen. Da nur wenige oder gar keine Microservices vorhanden sind, ist es nicht immer möglich, die Datenkommunikation zwischen den Schnittstellen mit API-Tests zu validieren. Das Testen über Batch-Jobs, Datenbanken und eine grafische Benutzeroberfläche ist weitaus verbreiteter.

Unternehmen, die ihre Standardlösungen modernisieren und langsam zu einer Microservices-Architektur übergehen, können API-Tests und Vertragstests für die modernisierten Teile des Systems einführen.

3.1.3 Wege zur Optimierung der Testautomatisierung aufzeigen, um Shift-Left- und Shift-Right-Ansätze zu erreichen

Sobald die Verteilung des Istzustands identifiziert und die Verteilung des Sollzustands ausgewählt ist, kann eine Roadmap für Verbesserungen geplant werden. Dabei wird festgelegt, was mit der Automatisierung getestet werden soll (d. h. der Testumfang der Testautomatisierung) und wie getestet werden soll (d. h. die Testautomatisierungsstrategie). Es müssen ein priorisiertes Backlog der zu implementierenden Testelemente und die Eingangskriterien für die Testautomatisierung festgelegt werden.

Bei einer unausgewogenen Verteilung der Testautomatisierung empfiehlt sich ein Bottom-up-Ansatz. Ist die Codeüberdeckung gering, ist dies ein Zeichen für einen Mangel an Testfällen für die Komponenten, und es müssen zusätzliche Testfälle für die Komponenten geschrieben werden. In einem zweiten Schritt muss die Qualität der Komponententests überprüft und ggf. verbessert werden. Das Vorschlagen von Best Practices, die Anwendung eines testgetriebenen Entwicklungsverfahrens und die Durchführung von Testverfahren-Workshops können die Qualität der Testautomatisierung verbessern.

Die Einführung von Komponententests für UI- und API-Tests und der Einsatz von Testdoubles (z. B. Mocks, Platzhalter) ermöglichen einen Shift-Left von teuren, langsamen und unzuverlässigen Tests. Die Aufhebung der Abhängigkeit von realen Diensten und Daten verbessert die Konsistenz der Testdurchführung und liefert frühes Feedback, das sich leicht in CI/CD-Pipelines integrieren lässt.

In den letzten Jahren hat der Vertragstest zunehmend an Bedeutung gewonnen. Durch die Prüfung des Vertrags zwischen einem Erzeuger und einem Konsumenten lassen sich die Grundursachen von Fehlerzuständen leichter und früher finden. Teams sind nicht mehr auf API-Tests oder UI-Tests angewiesen, um Fehlerzustände der zugrunde liegenden Dienste zu erkennen, und können stattdessen die Anzahl solcher Testfälle verringern. Die Erstellung eines Aufrufdiagramms für APIs ist ein intelligenter Weg, um zu verfolgen, welche APIs miteinander verbunden sind und welche die Konsumenten oder Erzeuger einer ausgewählten API sind. So lassen sich potenzielle Schwachstellen eines Systems besser identifizieren.

Während ein Shift-Left-Ansatz das Verschieben von Tests zu einem früheren Zeitpunkt im SDLC vorsieht, verschiebt Shift-Right die Tests zu einem späteren Zeitpunkt, wenn das SUT freigegeben wurde, um die Performanz in einer Testumgebung in der Vorproduktion oder in der Produktion zu bewerten. Durch die Anwendung von Shift-Right können Tester die Anwendungs- und API-Performanz überwachen und gleichzeitig Feedback von den tatsächlichen Benutzern erhalten. Obwohl die Testautomatisierung beim Shift-Left-Ansatz stärker im Vordergrund steht, kann die Testautomatisierung, wenn sie mit Beobachtbarkeit kombiniert wird, dabei helfen, schneller Entscheidungen darüber zu treffen, ob ein vollständiges Release durchgeführt werden kann oder der Releasekandidat zurückgerollt werden muss.

Shift-Right-Testen zielt darauf ab:

- Benutzerpräferenzen zu verstehen.
- Canary Releases und Dark Launches zu unterstützen, um die Benutzerfunktionalität so wenig wie möglich zu stören.
- Fehlerzustände in der Produktion frühzeitig zu erkennen.
- Den Testumfang und den Einsatz der Testautomatisierung auszuweiten.
- Die Überdeckung zu erhöhen.

3.2 Strategische Überlegungen zu verschiedenen Modellen des Softwareentwicklungslebenszyklus

3.2.1 Erläutern, wie Testautomatisierungsprojekte mit bestehenden Softwareentwicklungslebenszyklus-Modellen übereinstimmen

Im Wasserfallmodell folgt die Testphase auf die Phasen der Anforderungsanalyse, des Systementwurfs und der Implementierung. Aufgrund dieser strikten Reihenfolge beginnen die Aktivitäten zur Testautomatisierung später. Längere Zyklen zwischen den Tests bedeuten weniger Möglichkeiten, die Testautomatisierung zu nutzen. Dadurch verzögert sich in der Regel das Feedback aus der Testautomatisierung, was zu einem geringeren Return on Investment (ROI) führt.

Im V-Modell findet die gesamte Testplanung zusammen mit der Vorbereitung der Dokumentation in frühen Phasen statt. So wird zum Beispiel in der Phase des Architekturentwurfs der Testentwurf für die Integration erstellt, wodurch das Testen früher als im Wasserfallmodell beginnt. Oft erfolgt die eigentliche Testautomatisierung erst zu einem späteren Zeitpunkt im SDLC, und obwohl der ROI der Testautomatisierung im Vergleich zum Wasserfallmodell höher ist, bleibt er immer noch hinter modernen agilen Softwareentwicklungspraktiken zurück.

3.2.2 Erläutern, wie Projekte zur Testautomatisierung mit den Best Practices der agilen Softwareentwicklung übereinstimmen, die die Testautomatisierung unterstützen

Eines der Ideale der agilen Softwareentwicklung ist die Automatisierung von Tests innerhalb der Sprints. Das bedeutet, dass alle notwendigen Aktivitäten zur Testautomatisierung als Teil der Endekriterien für jede User-Story festgelegt werden. Dazu gehören die Definition von Testfällen, die Implementierung von automatisierten Testfällen, Aktualisierungen des TAF und in einigen Fällen die Integration in eine CI/CD-Pipeline. Organisationen, die agile Praktiken nicht richtig anwenden, nehmen keine Schätzung des Testaufwands vor und verwalten diesen entweder in einem separaten Ticket oder überwachen ihn überhaupt nicht. Durch eine Testautomatisierung im Sprint stellen agile Teams sicher, dass sie den vereinbarten Umfang innerhalb oder bis zum Ende jedes Sprints bereitstellen können. Wenn ein Team aus agiler Sicht noch nicht reif ist, sollte das erste Ziel die Durchführung von In-Sprint-Tests sein, da die Automatisierung um einen Sprint zurückliegen würde. Davon ausgehend kann sich ein Team an die Testautomatisierung im Sprint herantasten.

3.2.3 Testautomatisierungsprojekte mit DevOps Best Practices vorbereiten, um kontinuierliches Testen zu erreichen

Bei der agilen Softwareentwicklung liegt der Schwerpunkt auf der Arbeitsorganisation, während DevOps für die durchgängige Bereitstellung der Software verantwortlich ist. Erreicht wird dies durch die Automatisierung von Build-, Integrations-, Test-, Bereitstellungs- und Produktionsaktivitäten. Dies erleichtert kontinuierliches Testen durch Feedbackschleifen, die eine fortlaufende Verbesserung gewährleisten.

Der Schwerpunkt liegt eher auf der Implementierung von automatisierten Testfällen der unteren Stufen, einschließlich Komponenten-, Komponentenintegrations- und Vertragstests. Durch die geringere Abhängigkeit von realen Daten und Diensten werden die Tests in kürzerer Zeit ausgeführt. Wenn möglich, sollte die Testautomatisierung in derselben Pipeline ausgeführt werden, in der das SUT erstellt wird. Verschiedene Testsuiten werden nach jeder Entwicklungs- und Build-Phase ausgelöst (z. B. Lokalisierung, Pull Request, Merge und Deployment).

Wenn die Tester die Kapazität haben, größere UI- und API-Testsuiten zu erstellen, werden diese separat ausgeführt, um einen zusätzlichen Nutzen zu bieten. Als Ergänzung zur Testautomatisierung sollten manuelle Tests sich auf explorative Tests und Feedbacks von Endbenutzern konzentrieren.

3.3 Anwendbarkeit und Durchführbarkeit der Testautomatisierung

3.3.1 Kriterien zur Bestimmung der Eignung von Tests für die Testautomatisierung erläutern

Die Auswahl von Testfällen für die Testautomatisierung erfolgt in der Regel durch einen TA (Testanalyst), der weiß, welche Testfälle automatisiert werden können, oder durch einen TAE, der über das notwendige Fachwissen verfügt, um solche Entscheidungen zu treffen.

Bei der Auswahl und Priorisierung von Testfällen für die Testautomatisierung werden die folgenden Punkte berücksichtigt:

- Ist es technisch möglich, die Testfälle automatisiert zu implementieren?
- Gibt es technische Herausforderungen, die sich auf die Durchführung von automatisierten Testfällen auswirken? Ist das Team bereit und gut ausgebildet, um die Implementierungsarbeiten durchzuführen?
- Bietet der Codierungsaufwand einen angemessenen ROI (siehe *Kapitel 5.1.1*)?
- Ist es sinnvoll, die Testfälle häufig auszuführen?
- Handelt es sich um einen funktionalen oder nicht-funktionalen Test? Ist er Teil der Smoke-Test-Suite, der Regressionstest-Suite oder der Fehlernachtest-Suite?
- Ist der Testfall wiederholbar?
- Ist der Testfall einfach zu warten, wenn sich das SUT aufgrund von Updates ändert?
- Deckt der Testfall häufig verwendete Geschäftsabläufe ab?
- Gibt es funktionale Überschneidungen zwischen den Tests, die eine Wiederverwendbarkeit der Testschritte und Testdaten ermöglichen?

3.3.2 Herausforderungen identifizieren, die nur durch Testautomatisierung gelöst werden können

Es gibt bestimmte Tests, die nur mit Testautomatisierung durchgeführt werden können. Das trifft in folgenden Fällen zu:

1. Die manuelle Testdurchführung nimmt mehr Zeit in Anspruch, als angemessen ist.
2. Die Ausführung von Testfällen muss synchronisiert werden.
3. Die Testergebnisse müssen in einer Pipeline verfügbar sein.
4. Große Protokolldateien müssen auf Fehlerzustände analysiert werden.
5. Präzision in der Zeitplanung der Tests ist gefordert.

6. Test-Permutationen sind über mehrere Betriebssysteme, Browser, Geräte, Standorte oder Konfigurationen hinweg erforderlich.
7. Eine große Anzahl von Testausführungen und/oder Dateneingaben ist erforderlich, um die Überdeckung zu maximieren.
8. Nicht-funktionale Tests erfordern eine automatisierte Überwachung und Analyse oder die Eingabe einer großen Anzahl von Benutzern, wie z. B. Stresstests oder Zuverlässigkeitstests.

3.3.3 Testbedingungen identifizieren, die schwierig zu automatisieren sind

Es gibt bestimmte Testbedingungen, die das Automatisieren von Testfällen zu einer schwierigen oder sogar unmöglichen Aufgabe machen. Es gibt viele Beispiele aus der Praxis, einige davon sind im Folgenden aufgeführt:

- Validierung der Anforderungen an das Design, einschließlich der Konsistenz der Benutzeroberfläche auf verschiedenen Plattformen. Das allgemeine Erscheinungsbild einer Software ist subjektiv und erfordert ein Review durch einen Menschen.
- Jeder Testfall, der zu viel menschliche Interaktion erfordert. Im Finanzsektor wäre ein gutes Beispiel ein Kreditantrag. In diesem Prozess kann es Fälle geben, in denen bestimmte Vorschriften oder Bedingungen nicht eingehalten werden (z. B. zu geringes Einkommen und falsche persönliche Angaben). In solchen Situationen müssen die Sachbearbeiter den eigentlichen Kreditantrag noch einmal überprüfen und manuell entscheiden oder den Kunden kontaktieren.
- Technische Schwierigkeiten, die der Testautomatisierung im Wege stehen, z. B. Einschränkungen auf Betriebssystemebene. Ein Beispiel ist native Betriebssystemsoftware, die Textnachrichten an die Benutzer sendet. Interaktionen oder Validierungen dieser Textnachrichten sind mit Werkzeugen zur UI-Testautomatisierung nicht möglich, da das Betriebssystem keine Interaktionen außerhalb des Ziel-SUT zulässt.
- Testbedingungen mit einer langen Zeitabhängigkeit würden eine Testautomatisierung ineffizient machen und sind im Vergleich zur manuellen Testdurchführung oft schwierig einzurichten. Ein Beispiel: Der Tester muss sich am SUT anmelden, den Inhalt der Homepage aktualisieren und zwei Stunden warten, bis sich das SUT aufgrund eines Session-Timeouts automatisch abmeldet. Wenn die verstrichene Zeit nicht manuell geändert werden kann (z. B. durch Mocks, serverseitige Antworten und Zeitänderungen auf Betriebssystemebene), ist es nicht ratsam, solche Testfälle automatisiert zu implementieren.

4 Organisatorische Einführungs- und Freigabestrategien für die Testautomatisierung - 135 Minuten (K2)

Schlüsselbegriffe

Fehlernachtest, Komponente, Quality Gate

Lernziele für Kapitel 4: Die Lernenden können ...

4.1 Testautomatisierungslösung planen

TAS-4.1.1 (K2) ... identifizieren, wie Testautomatisierung eine kürzere Markteinführungszeit unterstützt

TAS-4.1.2 (K2) ... identifizieren, wie Testautomatisierung dabei unterstützt, gemeldete Fehlerzustände gemäß den Anforderungen zu verifizieren

TAS-4.1.3 (K2) ... Ansätze definieren, die die Entwicklung von betriebsrelevanten Szenarien für die Testautomatisierung ermöglichen

4.2 Einführungsstrategien für die Testautomatisierung

TAS-4.2.1 (K2) ... eine Einführungsstrategie für die Testautomatisierung definieren

TAS-4.2.2 (K2) ... Risiken der Testautomatisierung bei der Einführung identifizieren

TAS-4.2.3 (K2) ... Ansätze zur Risikominderung beim Einsatz von Tests definieren

4.3 Abhängigkeiten innerhalb der Testumgebung

TAS-4.3.1 (K2) ... Komponenten für die Testautomatisierung in der Testumgebung definieren

TAS-4.3.2 (K2) ... Infrastrukturkomponenten und Abhängigkeiten der Testautomatisierung identifizieren

TAS-4.3.3 (K2) ... Anforderungen an die Daten und Schnittstellen der Testautomatisierung für die Integration in das System unter Test definieren

4.1 Testautomatisierungslösung planen

4.1.1 Identifizieren, wie Testautomatisierung eine kürzere Markteinführungszeit unterstützt

Die Testautomatisierung trägt dazu bei, Software schneller auf den Markt zu bringen, da sie die Zeit für den Testzyklus verkürzt. Das Testen vor einer Softwarefreigabe erfordert ein hohes Maß an Verifizierung und Validierung. Dies kann besonders bei Regressionstests von Bedeutung sein, da diese Testart die Wiederverwendung fördert, die Kosten senkt und für Konsistenz zwischen den Testausführungen sorgt.

Die Testautomatisierung trägt dazu bei, den manuellen Testaufwand zu verringern und den Entwicklern ein schnelles Feedback zu geben, während sie denselben Testumfang abdeckt. Die Automatisierung von Komponententests und Komponentenintegrationstests, die in der Regel nicht manuell durchgeführt werden, ermöglicht außerdem das Testen zu einem früheren Zeitpunkt im Softwareentwicklungslebenszyklus.

Ein Quality Gate ist eine in den Prozess integrierte Maßnahme, die die Software erfüllen muss, bevor sie in die nächste Phase übergehen kann. Die Einrichtung von Quality Gates auf der Grundlage von Testautomatisierung ermöglicht eine beschleunigte Einführung (engl. Deployment) in eine Vorproduktions- oder Produktionsumgebung. Durch den Einsatz dieser Quality Gates können Fehlerzustände früher gefunden werden, was zu einer Verkürzung der Markteinführungszeit führt. Die Zeit für die Testdurchführung kann durch parallele Testdurchführung und einen Shift-Left-Ansatz weiter verkürzt werden. Ein Shift-Left-Ansatz fördert eine Kultur des Qualitätsbewusstseins und ermutigt zum Testen in einem früheren Stadium des Softwareentwicklungslebenszyklus. Dazu können mehrere unabhängige Testfälle oder browserübergreifende Tests gehören.

Weitere Informationen können in *Kapitel 3.3.3*, *Kapitel 6.1.2* und *Kapitel 6.2.1* gefunden werden.

4.1.2 Identifizieren, wie Testautomatisierung dabei unterstützt, gemeldete Fehlerzustände gemäß den Anforderungen zu verifizieren

Fehlernachtests, die nach einer Codekorrektur durchgeführt werden, können die korrekte Behebung eines Fehlerzustands überprüfen. Ein Tester führt in der Regel die Testschritte durch, die zum Nachstellen des Fehlerzustands erforderlich sind, um zu überprüfen, dass der Fehler nicht mehr existiert.

Fehlerzustände können in späteren Versionen wieder auftreten (z. B. aufgrund eines Problems beim Konfigurationsmanagement oder bei der Verwaltung des Code-Repositorys). Daher sind Fehlernachtests geeignete Kandidaten für die Testautomatisierung und können der bestehenden Testsuite für Regressionstests hinzugefügt werden.

Ein automatisierter Fehlernachtest hat in der Regel einen geringen Funktionsumfang. Die Implementierung kann zu jedem beliebigen Zeitpunkt erfolgen, sobald ein Fehlerzustand gemeldet wird und die zur Nachstellung erforderlichen Testschritte bekannt sind.

Die Nachverfolgung automatisierter Fehlernachtests ermöglicht die Erstellung von Testberichten über die Zeit und die Anzahl der Testzyklen, die für die Behebung von Fehlerzuständen aufgewendet wurden.

Mithilfe der Testautomatisierung reduziert sich der Zeitaufwand für die Verifizierung von Fehlerkorrekturen über mehrere Plattformen, Geräte, Browser und Betriebssystemversionen hinweg erheblich.

4.1.3 Ansätze definieren, die die Entwicklung von betriebsrelevanten Szenarien für die Testautomatisierung ermöglichen

Betriebliche Abnahmetests stellen die Einsatzfähigkeit von Software in Produktionssystemen sicher und werden in der Regel unmittelbar vor dem Release durchgeführt. Dieser Aufwand dient der abschließenden Validierung der Systeme, Komponenten und der sonstigen Infrastruktur des SUT und dem Testen der Produktionsreife. Dies ist notwendig, da es trotz aller Bemühungen eines TAE keine Garantie dafür gibt, dass sich das SUT außerhalb der Testumgebung und in der Produktion gleich verhält.

Ein gutes Testkonzept beinhaltet Tests für Zuverlässigkeit, Fehlertoleranz, Integrität und Wartbarkeit. Die folgenden Testautomatisierungsansätze können dazu verwendet werden:

- Statische Codeanalyse - Automatisierte Werkzeuge, die den Code auf IT-Sicherheit und Schwachstellen analysieren. Testergebnisse werden für Trendanalysen über die Zeit gespeichert.
- Ende-zu-Ende-Tests - Automatisierte Testskripte, die Ende-zu-Ende-Szenarien ausführen und alle Aspekte der Testumgebung testen, um Fehlerzustände aufzudecken.
- Ausfallsicherungstests - Automatisierte Testskripte, die speziell testen, was passiert, wenn die Hardware einer Anwendung ausfällt. Zum Beispiel, wie die Anwendung wiederhergestellt wird, wenn physische Server, Cloud-Server, Netzwerke, Computerfestplatten und andere Hardwarekomponenten ausfallen. Testskripte werden entwickelt, um die Fähigkeit des SUT zu messen, sich automatisch wiederherzustellen. Dies ist besonders wichtig für Organisationen, die Chaos Engineering umsetzen.
- Backup- und Restore-Tests - Automatisierte Testskripte testen den Erfolg eines Backups der aktuellen Version und eines anschließenden Rollbacks zurück zu einem früheren Zeitpunkt (d. h. einer älteren Version der Software).
- Testen der Performanz-Effizienz (z. B. Lasttests) - Automatisierte Testskripte, mit denen die gleichzeitige Nutzung des SUT durch viele Benutzer simuliert werden kann. Testergebnisse werden zu Vergleichszwecken gespeichert und über die Zeit ausgewertet.
- Review der Betriebsdokumentation - Automatisierte Testskripte können für diese Aktivität verwendet werden, um Versionen der SUT-Dokumentation zu vergleichen und den Entwicklungsteams zu signalisieren, ob eine Aktualisierung aufgrund neuer Funktionen in einer bestimmten Version erforderlich ist.
- IT-Sicherheitstests - Automatisierte Testskripte können in Kombination mit branchenüblichen Sicherheitstest-Werkzeugen erstellt werden, um das SUT statisch und dynamisch zu testen. Testergebnisse werden für Vergleiche und Trendanalysen über einen längeren Zeitraum gespeichert.
- Überwachung auf der Grundlage des Service Level Agreements eines Unternehmens - Automatisierte Testskripte, die während des SDLC verwendet werden, können zur Überwachung des Produktionsbetriebs wiederverwendet werden und Warnmeldungen senden, wenn ein automatisierter Test fehlgeschlagen ist. Dies ist eine proaktive Maßnahme, um Produktionsausfälle abzufangen, bevor echte Benutzer sie erleben.

Testautomatisierung für die betriebliche Validierung von Software kann immense Vorteile bieten, insbesondere wenn die automatisierten Tests wiederholt und in mehreren Umgebungen ausgeführt werden können. Dedizierte automatisierte Testsuiten können erstellt werden, so dass Testergebnisse generiert und mit früheren Testausführungen verglichen werden können. Dies gewährleistet Konsistenz, wenn neue Versionen des SUT veröffentlicht werden. Eine zuverlässige automatisierte Testsuite mit betriebsspezifischen Testbedingungen kann die Kosten im Laufe der Zeit senken.

4.2 Einführungsstrategien für die Testautomatisierung

4.2.1 Eine Einführungsstrategie für die Testautomatisierung definieren

Eine gute Strategie für die Testautomatisierung berücksichtigt unter anderem die gesamte Testumgebung, die verfügbaren Testwerkzeuge, den Zugriff auf das SUT, die Speicherung von Testskripten und andere Abhängigkeiten sowie die Bereitstellung von Testdaten. Mit diesen allgemeinen Überlegungen kann ein TAE beginnen, eine Strategie für die Entwicklung und den Einsatz der TAS zu erstellen.

Testumgebung - Die TAEs sollten sich überlegen, wie sie in den verschiedenen Testumgebungen auf das SUT zugreifen wollen. Wenn sie mit der Entwicklung ihrer Testmittel für die Testautomatisierung beginnen, egal ob sie einen schlüsselwortgetriebenen Ansatz oder eine andere Lösung verwenden, müssen sie überlegen, wie diese Lösung in verschiedenen Testumgebungen laufen soll. Ein Testskript sollte beispielsweise so entwickelt werden, dass es sowohl in einer Testumgebung als auch mit minimalen Änderungen in einer Vorproduktionsumgebung ausgeführt werden kann. In der Regel genügt es, einen Uniform Resource Locator (URL) für eine webbasierte Anwendung zu ändern, und schon läuft das Testskript auf dieselbe Weise, egal in welcher Testumgebung es sich befindet.

Werkzeuge - TAEs müssen sich auch überlegen, welche Werkzeuge sie für die Erstellung der TAS verwenden. Wenn es sich um ein kommerzielles Werkzeug handelt, müssen sie sich über dessen Lizenzierung informieren. Der TAE wird möglicherweise feststellen, dass seine Testumgebung Zugriff auf den Lizenzserver des Werkzeugs haben muss. Dies ist zu bedenken, wenn das Testskript in mehreren Testumgebungen verwendet werden soll. Nur weil der Lizenzserver in der Testumgebung verfügbar ist, bedeutet dies nicht unbedingt, dass er auch in der Vorproduktionsumgebung zur Verfügung steht.

Softwarezugang - Es müssen Überlegungen darüber angestellt werden, wie auf das SUT zugegriffen werden kann. Testskripte können so entworfen werden, dass sie Parameter akzeptieren, so dass bestimmte Benutzer oder Berechtigungsnachweise mit speziellem Zugang schnell aktualisiert werden können, wenn sich die SUT-Endpunkte ändern. Dies ist auch wieder sehr wichtig, wenn man zwischen verschiedenen Testumgebungen wechselt und die URL geändert werden muss. Darüber hinaus kann es notwendig sein, je nach Testumgebung unterschiedliche Zugangsdaten zu verwenden (z. B. über Benutzer- und Testkonten, Biometrie und Smartcards). Ein guter Entwurf für ein automatisiertes Testskript bietet genügend Flexibilität, so dass einfache Parameter gesetzt werden können und die Testskripte unabhängig von der Testumgebung wie erwartet ausgeführt werden.

Speicherung der Testskripte - Der TAE sollte sich für einen zentralen Ort zur Speicherung und Verwaltung der automatisierten Testskripte entscheiden. Eine gute Strategie wäre ein Quellcode-Repository, das Konfigurationsmanagement nutzt. Auf diese Weise kann von mehreren Testumgebungen aus auf die Testskripte zugegriffen werden, solange jene Zugriff auf das Quellcode-Repository haben, und es können Versionen für die jeweilige Version des SUT erstellt werden. Alle Konfigurationen und Abhängigkeiten können im gleichen Repository wie die Testskripte gespeichert werden und sorgen so für optimale Übertragbarkeit. Kurz gesagt, die TAS, das TAF und alle Testfälle können in Repositorys gespeichert und verwaltet werden.

Datenbereitstellung - Es ist wichtig zu verstehen, ob ein automatisiertes Testskript von bestimmten Testdaten abhängig ist, die bereits in der Testumgebung, in der das SUT getestet wird, vorhanden sind. Ein guter Testskriptentwurf wird statische Daten (d. h. fixe Datensätze) so weit wie möglich vermeiden. Es wird jedoch Situationen geben, in denen dies nicht machbar ist. In diesen Fällen müssen die TAEs eine Lösung finden, um die Testdaten bereitzustellen oder die Testautomatisierung zu nutzen, um Testskripte zu erstellen, die die notwendigen Testdaten erzeugen, bevor die eigentlichen Testskripte laufen und das

SUT testen. Beide Ansätze haben ihre Vor- und Nachteile. Auf der einen Seite ist es bequem, wenn ein Administrator die Testdaten bereitstellt. Allerdings sind die TAEs dann auf die Verfügbarkeit einer anderen Ressource angewiesen, die sie unterstützt. Ist man mit einem Testskript autark, dann ist man nicht mehr auf eine andere Ressource angewiesen. Die Erstellung dieser Testskripte nimmt jedoch Zeit in Anspruch und wird zu einem weiteren Teil der Lösung, der gewartet werden muss.

Durch die Berücksichtigung aller oben genannten Elemente ist die Testautomatisierungsstrategie in der Lage, eine angemessene Planung und Steuerung der Testautomatisierungsaktivitäten zu gewährleisten.

4.2.2 Risiken der Testautomatisierung bei der Einführung identifizieren

Technische Probleme können zu Produktrisiken und Projektrisiken führen (für weitere Informationen siehe ISTQB-CTFL (2023), Abschnitt 5.2.2). Typische technische Probleme sind:

- Zu viele Abstraktionen können zu Schwierigkeiten führen, zu verstehen, was der Code der Testautomatisierung wirklich tut (z. B. mit Schlüsselwörtern im schlüsselwortgetriebenen Ansatz).
- Testdatentabellen können zu groß/komplex/umständlich werden, um sie in andere Testumgebungen zu migrieren, was zu inkonsistenten Statusergebnissen führt.
- Die TAS kann von der Verwendung bestimmter Betriebssystembibliotheken oder anderer Komponenten abhängig sein, die möglicherweise nicht in allen Testumgebungen des SUT verfügbar sind.

Typische Projektrisiken bei der Einführung sind:

- Personalprobleme: Es kann schwierig sein, die richtigen Mitarbeiter für die Testautomatisierung zu finden.
- Ungeplante Wartung der TAS aufgrund von SUT-Updates, die zu einem fehlerhaften Betrieb der TAS führen.
- Verzögerungen bei der Einführung der Testautomatisierung
- Verzögerungen bei der Aktualisierung der TAS basierend auf den Änderungen am SUT
- Die TAS kann keine nicht standardisierten UI-Objekte erfassen.
- Veraltete Testfälle verbleiben in der Testsuite und vergeuden Zeit für die Testdurchführung.

Potenzielle Fehlerquellen des TAS-Projekts sind:

- Migration in eine andere Testumgebung
- Einführung in eine Produktionsumgebung
- Vergessen, dass Automatisierung eine Software ist, die auch getestet werden muss.

Es ist wichtig zu erkennen, dass verschiedene technische Probleme, Projektrisiken und potenzielle Fehlerquellen den Erfolg von Projekten zur Testautomatisierung gefährden können. Zu den häufigen technischen Problemen gehören die Komplexität, die sich aus übermäßiger Abstraktion ergibt, Herausforderungen beim Testdatenmanagement und Abhängigkeiten von bestimmten Komponenten. Zu den Projektrisiken gehören auch Schwierigkeiten bei der Personalbesetzung, Wartungsprobleme aufgrund von Systemaktualisierungen, Verzögerungen bei der Bereitstellung und das Vorhandensein veralteter Testfälle. Darüber hinaus sollten potenzielle Fehlerquellen wie die Migration auf andere Umgebungen und das Missachten der Notwendigkeit, die Automatisierungssoftware selbst zu testen, durch eine angemessene

Planung berücksichtigt werden. Insgesamt werden Überwachung und proaktive Maßnahmen den Erfolg von Projekten zur Testautomatisierung sicherstellen.

4.2.3 Ansätze zur Risikominderung beim Einsatz von Tests definieren

Es gibt viele Strategien zur Risikominderung, die zur Bewältigung dieser Risikobereiche eingesetzt werden können. Einige davon werden im Folgenden erläutert.

Die TAS hat einen eigenen SDLC, unabhängig davon, ob es sich um eine intern entwickelte oder eine erworbene Lösung handelt. Dabei ist zu beachten, dass die TAS wie jede andere Software auch einem Konfigurationsmanagement unterliegt und ihre Funktionen dokumentiert werden müssen. Andernfalls wird es äußerst schwierig, einzelne Teile davon einzuführen und dafür zu sorgen, dass sie zusammenarbeiten oder in bestimmten Testumgebungen funktionieren.

Außerdem muss es ein dokumentiertes, klares und leicht zu befolgendes Verfahren für den Einsatz geben. Da dieses Verfahren versionsabhängig ist, muss es auch in das Konfigurationsmanagement einbezogen werden.

Beim Einsatz einer TAS gibt es zwei verschiedene Fälle:

1. Erstmaliger Einsatz
2. Wartung - Die TAS ist bereits vorhanden, und es muss eine Aktualisierung eingespielt werden.

Im Zusammenhang mit dem erstmaligen Einsatz können folgende Risiken auftreten:

- Die Gesamtdauer der Testdurchführung der Testsuite kann länger sein als die geplante Dauer der Testdurchführung für den Testzyklus. In diesem Fall ist es wichtig, genügend Zeit einzuplanen, damit die Testsuite vollständig ausgeführt werden kann, bevor der nächste geplante Testzyklus beginnt.
- Es gibt Probleme bei der Installation und Konfiguration von Testumgebungen (z. B. Einrichtung der Datenbank und anfängliches Laden sowie Start/Stopp von Diensten). Die TAS benötigt eine Testvorrichtung (d. h. einen vordefinierten Datensatz), um die notwendigen Vorbedingungen für die Ausführung automatisierter Testfälle in der Testumgebung zu schaffen.

Bei der Wartung sind zusätzliche Aspekte zu berücksichtigen. Die TAS selbst muss weiterentwickelt werden und die Aktualisierungen dafür müssen in der Produktion eingesetzt werden. Bevor eine aktualisierte Version der TAS in der Produktion eingesetzt werden kann, muss sie getestet werden. Es ist daher notwendig, die neuen Funktionen zu prüfen, um sicherzustellen, dass eine Testsuite auf der aktualisierten TAS ausgeführt werden kann, Testberichte gesendet werden können und keine Fehlerzustände oder andere Qualitätsprobleme vorliegen. In einigen Fällen muss möglicherweise die gesamte Testsuite geändert werden, um sie an die neueste Version der TAS anzupassen.

4.3 Abhängigkeiten innerhalb der Testumgebung

4.3.1 Komponenten für die Testautomatisierung in der Testumgebung definieren

Komponenten für die Testautomatisierung bestehen in der Regel aus Werkzeugen, virtuellen Maschinen, Testskripten, Containern und Konfigurationen. Zur Testumgebung gehört auch das SUT. Die Komponenten für die Testautomatisierung in der Testumgebung können wie folgt definiert werden:

SUT - Dies ist ein offensichtlicher Teil der Testumgebung. Das SUT kann als Ganzes getestet oder in Unterkomponenten aufgeteilt werden (z. B. API, webbasierte Schnittstelle und Datenbank).

Plattform - Die Plattform beschreibt, wo die Komponenten der Testautomatisierung gehostet werden. Dazu gehören die Cloud-Infrastruktur, das Netzwerk, virtuelle Maschinen und Container, die verwendet werden können, um die Testautomatisierung effizient und übertragbar zu machen.

Testfälle und Testsuiten - Hier werden einzelne Testfälle beschrieben, die sich aus Testschritten zusammensetzen, die sowohl automatisierte als auch manuelle Aktionen steuern. Die Testsuiten beschreiben eine logische Gruppierung von Testfällen, damit sie gemeinsam effizient ausgeführt werden können.

Werkzeuge - Verschiedene Werkzeuge zur Testautomatisierung werden aus unterschiedlichen Gründen eingesetzt. Eine Sammlung von Werkzeugen umfasst solche für die UI-Testautomatisierung, das Testen von API-Endpunkten, die Datengenerierung, die Überwachung, das Anforderungsmanagement, das Fehlermanagement, die Testprotokollierung und die Testberichterstattung sowie Werkzeuge für die Erstellung von Trends auf der Grundlage von Metriken.

TAF - Dazu gehören alle Elemente, die das TAF-Design ausmachen. TAFs beinhalten Treiberskripte, allgemeine Bibliotheken, Vorlagen für automatisierte Testfälle, Skripte zum Laden/Bereitstellen von Daten, Dokumentation zur Verwendung von TAF-Komponenten und Tutorien, die TAEs bei der Nutzung des TAFs unterstützen. Weitere Informationen lassen sich im ISTQB-CTAL-TAE (2024), Abschnitt 3.1 finden.

Die Wartung der Komponenten ist ein wichtiger Aspekt, denn eine zu komplizierte Testumgebung kann dazu führen, dass man stundenlang mit der Behebung von Fehlerzuständen im TAF beschäftigt ist, anstatt von der Lösung zu profitieren. Es ist wichtig, die richtige Mischung aus Werkzeugen, Konfiguration und Übertragbarkeit der Plattform zu finden, um die Komponenten so wiederverwendbar wie möglich zu machen.

4.3.2 Infrastrukturkomponenten und Abhängigkeiten der Testautomatisierung identifizieren

Es gibt eine Reihe von Infrastrukturkomponenten und Abhängigkeiten, die bei der Zusammenstellung der Testautomatisierung beachtet werden müssen. Zusammen decken sie alle Voraussetzungen ab, die für den Betrieb einer TAS erforderlich sind. Zu den wichtigsten Komponenten und Abhängigkeiten gehören:

Hosts - Dabei kann es sich um virtuelle Maschinen, physische Server, Laptops und Geräte (z. B. Tablets und Mobilgeräte) handeln. Auf ihnen ist die Software zur Testautomatisierung installiert und dort werden die Testskripte erstellt und ausgeführt.

Netzwerk - Ermöglicht der TAS den Zugang zum SUT. Es kann auch die Vernetzung mehrerer Hosts beinhalten, um eine parallele Ausführung von automatisierten Tests zu ermöglichen. Normalerweise müssen sich die Hosts im selben Netzwerk befinden und so konfiguriert sein, dass sie miteinander kommunizieren können.

Plattform - Testautomatisierung kann wie jede andere Software auf Cloud-Plattformen laufen oder für die Ausführung in Containern konzipiert sein. Solange die Plattform Berechtigungen und Zugriff auf das zugrunde liegende Betriebssystem bietet, können alle erforderlichen Werkzeuge und Abhängigkeiten installiert werden.

Softwareabhängigkeiten - Damit die Testautomatisierungswerkzeuge korrekt funktionieren, müssen alle anderen Abhängigkeiten, die für die Testautomatisierung gelten, verstanden werden. So kann es beispielsweise sein, dass für ein bestimmtes Testautomatisierungswerkzeug zunächst die neueste Version einer Programmiersprache auf dem Host installiert werden muss. TAEs müssen diese Abhängigkeiten vor und nach der Auswahl eines Werkzeugs berücksichtigen.

SUT - Nachdem die Komponenten, alle Abhängigkeiten sowie die gesamte Infrastruktur berücksichtigt und richtig konfiguriert wurden, ist der letzte Schritt die Sicherstellung des Zugangs zum SUT. Zusätzlich zum Netzwerk muss auch beachtet werden, wie die Schnittstelle zum SUT aussehen soll. Bei einer webbasierten Anwendung muss zum Beispiel ein Browser auf dem Host installiert werden. Der Typ des Browsers und die Version müssen berücksichtigt werden.

4.3.3 Anforderungen an die Daten und Schnittstellen der Testautomatisierung für die Integration in das System unter Test definieren

Zwei Überlegungen, die TAEs vor der Entwicklung von Testautomatisierungsskripten anstellen sollten, sind, wie sie die Schnittstelle zum SUT planen und welche Datenabhängigkeiten die Testfälle haben. Nachfolgend einige Beispiele:

API - Wenn das SUT eine API verwendet, kann diese auf der Serviceebene getestet werden, anstatt eine Benutzeroberfläche für die Schnittstelle auf der Anwendungsebene zu verwenden. Dies kann ein tieferes Verständnis der Endpunkte und der Datenkommunikation erfordern, die durch eine Benutzeroberfläche verdeckt werden. Die TAEs müssen verstehen, in welcher Reihenfolge APIs aufgerufen werden müssen, um einen Geschäftsprozess zu erstellen, und wie die Daten korreliert werden müssen, damit der Testfall intakt bleibt. APIs, die im Internet verfügbar sind, werden auch als Web-APIs oder Web-Services bezeichnet.

Datenbankschnittstelle - Einige Werkzeuge zur Testautomatisierung haben die Möglichkeit, direkt mit der zugrunde liegenden Datenbank des SUT zu kommunizieren. Es können Testskripte geschrieben werden, die Daten innerhalb von Spalten und Zeilen überprüfen, um sicherzustellen, dass gespeicherte Prozeduren und andere Datenbankregeln richtig konfiguriert sind. Für Zuverlässigkeitstests kann es erforderlich sein, dass bestimmte Datenwerte bereits in der Datenbank vorhanden sind.

Kompatibilität der Schnittstellen - Der Einsatz von Vertragstests (engl. Contract Testing) stellt sicher, dass zwei getrennte Systeme (z. B. zwei Microservices) kompatibel sind und miteinander kommunizieren können. Das Testen von Verträgen kann vom Konsumenten (consumer-driven) oder vom Erzeuger (producer-driven) ausgehen oder bidirektional erfolgen. Weitere Einzelheiten sind in ISTQB-CTAL-TAE (2024), Abschnitt 5.1.3 zu finden.

Darüber hinaus müssen TAEs die Datenabhängigkeiten innerhalb der Testfälle verstehen und sorgfältig überlegen, wie sie die Schnittstellen zum SUT handhaben. Unabhängig davon, ob das Testen über APIs, Datenbankschnittstellen oder die Sicherstellung der Kompatibilität von Schnittstellen zwischen Systemen erfolgt, ist die Beachtung dieser Überlegungen für eine robuste und effektive Testautomatisierung unerlässlich. Durch die Berücksichtigung dieser Faktoren können TAEs die Zuverlässigkeit und Effizienz ihrer Automatisierungsbemühungen verbessern und letztendlich zur Qualität und zum Erfolg des SUT beitragen.

5 Auswirkungsanalyse der Testautomatisierung - 150 Minuten (K3)

Schlüsselbegriffe

Testbericht, Überdeckung

Lernziele für Kapitel 5: Die Lernenden können ...

5.1 Investitionen in den Aufbau und die Pflege der Testautomatisierung

TAS-5.1.1 (K3) ... den Return on Investment für den Aufbau einer Testautomatisierungslösung aufzeigen

5.2 Metriken zur Testautomatisierung

TAS-5.2.1 (K2) ... Metriken für die Testautomatisierung klassifizieren

5.3 Der Wert der Testautomatisierung auf Projekt- und Organisationsebene

TAS-5.3.1 (K3) ... organisatorische Überlegungen zum Einsatz der Testautomatisierung identifizieren

TAS-5.3.2 (K3) ... Projektmerkmale analysieren, die zur Bestimmung der optimalen Testziele für die Testautomatisierung beitragen

5.4 Entscheidungen anhand von Berichten zur Testautomatisierung treffen

TAS-5.4.1 (K2) ... Testberichtsdaten zur Entscheidungsfindung analysieren

5.1 Investitionen in den Aufbau und die Pflege der Testautomatisierung

5.1.1 Den Return on Investment für den Aufbau einer Testautomatisierungslösung aufzeigen

Es ist wichtig, die Einrichtungs- und Wartungskosten für die Testautomatisierung abzuschätzen, bevor mit der Projektrealisierung begonnen wird. Zusätzlich zu den Vorteilen, die die Automatisierung einem Projekt bringen kann, ist es hilfreich, die ROI-Berechnung für das Projekt zu verstehen.

Die ROI-Berechnung kann zu jedem Zeitpunkt des Projektlebenszyklus ein aussagekräftiges Feedback liefern, indem sie die Rendite einer Aktivität für den investierten Aufwand aufzeigt. Die ROI-Berechnung kann weiter angepasst werden, indem sowohl die Kosten für manuelle Tester als auch die TAE-Kosten durch Multiplikation der aufgewendeten Zeit mit den jeweiligen Arbeitskosten einbezogen werden.

Um den ROI zu berechnen, muss man die Investition in die Testautomatisierung (d. h. Zeit und Kosten) und die damit erzielten Einsparungen ermitteln:

ROI = Einsparungen / Investition

Es ist wichtig zu beachten, dass die Einsparungen und die Investition unter Berücksichtigung verschiedener Metriken und Daten in verschiedenen Messungen und Einheiten berechnet werden können. Im Rahmen dieses Lehrplans wird ein einfaches Modell verwendet, um den Ansatz mit Zeiteinheiten und nicht mit Kosten zu demonstrieren. Wenn bestimmte Aktivitäten nur in Kosten gemessen werden, kann dieser Betrag mit einem projektspezifischen Satz in Zeit umgerechnet werden.

Die Einsparungen durch die Testautomatisierung beruhen im Allgemeinen darauf, dass dieselben Tests in wesentlich kürzerer Zeit ausgeführt werden können, als es bei manueller Durchführung der Fall wäre. Das bedeutet auch, dass sie häufiger ausgeführt werden können. Daher kann die Anzahl der ausgeführten Tests erhöht werden.

Um die Einsparungen zu berechnen, müssen die folgenden Metriken berücksichtigt werden:

- Zeit für die manuelle Ausführung eines Testfalls
- Zeit für die Ausführung eines automatisierten Testfalls
- Anzahl der Testfälle
- Anzahl der Testläufe

Zur Berechnung der Investitionen müssen die folgenden Metriken berücksichtigt werden:

- Zeit für die Einrichtung der Testautomatisierung
- Durchschnittliche Zeit für die Entwicklung automatisierter Testskripte
- Anzahl der implementierten automatisierten Testskripte
- Durchschnittliche Wartungszeit für ein automatisiertes Testskript
- Zeit für die Ausführung eines automatisierten Testskripts
- Prozentsatz der fehlgeschlagenen automatisierten Testskripte
- Anzahl der definierten Testfälle
- Anzahl der Testläufe

Wenn man das skizzierte Modell auf die agile Softwareentwicklung anwendet, lassen sich die Sprints eines Projekts vorhersagen, wie in der folgenden Grafik dargestellt. Anhand des Diagramms lässt sich ermitteln, ab welchem Sprint sich die Testautomatisierung amortisiert hat.

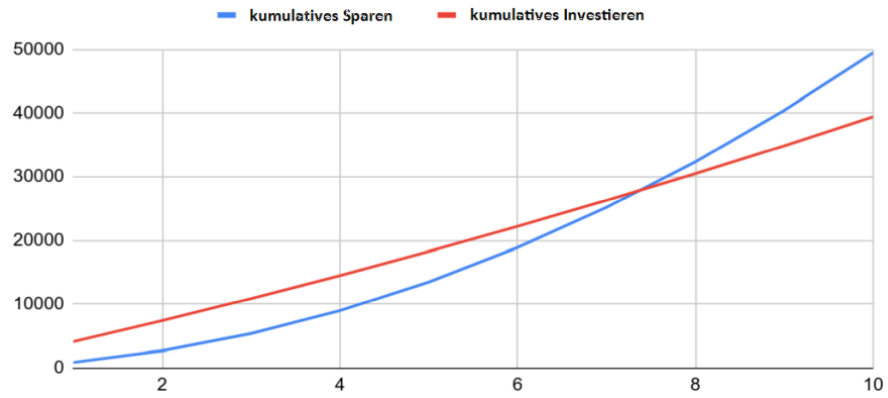

Abbildung 2: Beispiel für eine ROI-Berechnung, die den Zeitpunkt des Return on Investment anzeigt

Bestimmte Verbindungen zwischen den Metriken, die zur Berechnung des ROI gemessen werden, sind zu beachten, z. B.:

- Wenn die geplante Projektdauer unter dem Wendepunkt des ROI liegt, lohnt es sich nicht, die Testautomatisierung einzuführen. In diesem Fall werden durch die manuelle Ausführung der Tests Zeit und Aufwand gespart.
- Da die Investition maßgeblich von der Ausführungszeit der automatisierten Tests abhängt, kann die Anwendung der Testpyramide und die Implementierung von Testfällen auf der richtigen Teststufe diese Ausführungszeit reduzieren und den ROI verbessern.

5.2 Metriken zur Testautomatisierung

5.2.1 Metriken für die Testautomatisierung klassifizieren

Die Analyse von Trends auf der Grundlage von Fakten hilft bei der Entscheidungsfindung. Durch das Sammeln von Metriken einer TAS sind die Tester in der Lage, die TAS zu bewerten und Entscheidungen zu treffen:

- Eignung der TAS für das Projekt
- Anpassbarkeit der TAS für erweiterte Funktionalität bei neuen Testbedingungen
  - Änderungen an Benutzerabläufen im SUT
  - Änderungen an der Art der Testdurchführungen
- Wartbarkeit der Testautomatisierung aufgrund von in der TAS gefundenen Fehlerzuständen

Die Kosten für die Messung sollten so gering wie möglich sein, dies kann oft durch die Automatisierung der Erfassung und Berichterstattung von Metriken erreicht werden. Einige Beispiele sind unten aufgeführt.

Pass-Fail-Quote

Hierbei handelt es sich um eine gängige Metrik, die das Verhältnis zwischen den bestandenen automatisierten Tests und den fehlgeschlagenen automatisierten Tests, die nicht das erwartete Ergebnis erzielt haben, erfasst. Die Pass-Fail-Quote ist die Anzahl der bestandenen Tests / die Anzahl der fehlgeschlagenen Tests.

Verhältnis von Fehlerwirkungen zu Fehlerzuständen

Ein häufiges Problem bei automatisierten Tests ist, dass viele von ihnen aus demselben Grund fehlschlagen, d. h. wegen eines einzigen Fehlerzustands im SUT. Die Messung der Anzahl der automatisierten Tests, die bei einem bestimmten Fehlerzustand fehlgeschlagen sind, kann Aufschluss darüber geben, wo dies ein Problem sein könnte. Zusätzlich kann man die Anzahl der Fehlerwirkungen pro Fehlerzustand erfassen und den Grund für die Fehlerwirkung verfolgen: Fehlerzustand im Subsystem, im gesamten System, Probleme mit den Testdaten oder der Testinfrastruktur.

Ausführungszeit der Testautomatisierung

Eine der einfach zu ermittelnden Metriken ist die Zeit, die für die Ausführung der automatisierten Tests benötigt wird. Diese Metrik beinhaltet auch die Erstellungszeit der TAS. Zu Beginn der TAS ist diese Metrik vielleicht noch nicht so wichtig, aber mit zunehmender Anzahl der automatisierten Testfälle kann sie sehr wichtig werden.

Anzahl der automatisierten Testfälle

Diese Metrik kann verwendet werden, um den Fortschritt des Projekts zur Testautomatisierung aufzuzeigen. Es ist jedoch zu beachten, dass die Anzahl der automatisierten Testfälle nicht viel aussagt, z. B. nichts über den Grad der Überdeckung.

Funktionale Überdeckung der Testautomatisierung

Die Überdeckung gibt an, wie viel Prozent der funktionalen Anforderungen durch automatisierte Testfälle abgedeckt werden.

Codeüberdeckung

Die Codeüberdeckung gibt an, wie viele Codezeilen durch Tests auf niedrigerer Ebene (d. h. Komponenten) überprüft werden.

Es gibt keinen absoluten Prozentsatz, der eine angemessene Überdeckung anzeigt, und eine 100%ige Codeüberdeckung ist oft nur bei einfachster Software zu erreichen. Es besteht jedoch allgemein Einigkeit darüber, dass eine höhere Überdeckung besser ist, da sie das Vertrauen in das SUT erhöht. Je mehr automatisierte Tests auf niedriger Ebene hinzugefügt werden, desto höher ist die Codeüberdeckung.

5.3 Der Wert der Testautomatisierung auf Projekt- und Organisationsebene

5.3.1 Organisatorische Überlegungen zum Einsatz von Testautomatisierung identifizieren

Bevor man mit der Testautomatisierung in einem Projekt beginnt, müssen die folgenden Punkte innerhalb der Organisation geklärt werden:

Vorgaben und Praktiken für die Softwareentwicklung

Es ist zu prüfen, wie die Entwicklungsteams arbeiten und welche Art von Dokumentation vorhanden ist. Jede verfügbare Dokumentation kann von Vorteil sein, um herauszufinden, wie die Testautomatisierung in die Prozesse der Entwicklungsteams eingebunden werden kann. Die Dokumentation kann die technische Spezifikation des SUT, die verwendete Software und die Entwicklungswerkzeuge oder alle verfügbaren Richtlinien zur Entwicklung, wie z. B. Richtlinien zum Code-Review, definierte Programmierstandards und Merge-Prozesse, beinhalten.

Bestehende aktive Projekte zur Testautomatisierung und deren Status

Im Falle eines laufenden Entwicklungsprojekts kann es ein oder mehrere laufende TAS-Entwicklungsprojekte von verschiedenen Teams geben. Eine allgemeine Empfehlung für die Entscheidungsträger besteht darin, diese bestehenden Projekte und ihren Status zu überprüfen, um festzustellen, ob eines von ihnen zu den definierten Testzielen der neuen TAS passt. Durch die Analyse der Lösung lässt sich feststellen, ob eine der bestehenden TAS wiederverwendet werden kann oder eine neue TAS auf der Grundlage des aktuellen Bedarfs erstellt werden soll.

Fachexperten der Organisation für Testautomatisierung

Es wird empfohlen, nach Fachexperten (SMEs) zu suchen, die bei der Einführung einer neuen TAS innerhalb der Organisation helfen können. Diese Fachexperten können über ihre Erfahrungen mit einer TAS und deren Einführung berichten. Daraus lassen sich verschiedene Risiken ableiten, die für eine erfolgreiche TAS-Einführung berücksichtigt oder vermieden werden müssen.

Verfügbarkeit von Testumgebungen

Bei großen Organisationen empfiehlt es sich generell, Informationen über die vorhandenen Testumgebungen, ihre Nutzung und Verfügbarkeit zu sammeln. Diese Informationen sind von entscheidender Bedeutung, um zu vermeiden, dass die Arbeit eines anderen Teams durch die Einführung einer neuen TAS und die Nutzung des Systems ohne jegliche Benachrichtigung oder Absprache beeinträchtigt wird. Oftmals erfordern die Anforderungen eines Projekts eine neue Testumgebung, so dass die bestehenden Umgebungen als Grundlage für die Erstellung einer neuen Umgebung verwendet werden können und die Arbeit leichter von der Hand geht, wenn die verantwortlichen Teams und Personen bekannt und verfügbar sind.

Testwerkzeuge und Lizenzen

Es lohnt sich immer, sich darüber zu informieren, welche Werkzeuge und Lizenzen der eigenen Organisation zum aktuellen Zeitpunkt zur Verfügung stehen. Indem man diese sowie die Kosten und den Zeitrahmen ermittelt, kann der Planungsumfang einer TAS reduziert werden. Zudem sind möglicherweise die Werkzeuge für ein neues Projekt bereits verfügbar. Wenn beispielsweise das Testen in der Cloud über einen bestimmten Anbieter eingerichtet wird und die Lizenzen für das neue Projekt verfügbar sind, ist es

nicht sinnvoll, einen anderen Cloud-Anbieter zu nehmen. Es wird empfohlen, die gleichen Werkzeuge und Lizenzen zu verwenden, um die Projektkosten zu senken.

5.3.2 Projektmerkmale analysieren, die zur Bestimmung der optimalen Testziele für die Testautomatisierung beitragen

Es gibt mehrere wichtige Projektmerkmale, die eine optimale Arbeitsweise definieren und dazu beitragen können, die Ziele der Testautomatisierung zu bestimmen.

Domäne

Es ist wichtig zu verstehen, dass sich jede Domäne entweder durch Vorschriften oder durch Standards unterscheidet. So gibt es beispielsweise im Bereich Tourismus andere Vorschriften und Risiken als im Finanz- oder Gesundheitswesen. Es ist immer empfehlenswert, die verschiedenen Standards und Domäneneinschränkungen zu überprüfen, um sicherzustellen, dass die geplanten Testautomatisierungsziele damit übereinstimmen.

Plattformen

Im Hinblick auf die Testziele der Testautomatisierung ist es auch wichtig zu bewerten, welche Plattformen das Projekt abdeckt und wo eine Testautomatisierung von Vorteil wäre. Die Planung der Testautomatisierung für mehrere Plattformen kann schwieriger sein, da unter Umständen mehrere Lösungen erforderlich sind, um die verschiedenen Plattformen abzudecken. In vielen Fällen, z. B. für Web- und mobile Anwendungen, können die gleichen Werkzeuge verwendet werden. Die Planung bestimmt die Wiederverwendbarkeit der TAS.

Programmiersprache und Technologie-Stack

Die Implementierungsdetails eines Projekts, wie seine Programmiersprache und sein Technologie-Stack, bestimmen auch, welche Art von Testzielen für die Testautomatisierung eine Organisation definieren sollte. Es wird empfohlen, die gleichen Programmiersprachen zu verwenden, die auch von den Entwicklern für ein bestimmtes Projekt eingesetzt werden. Auf diese Weise kann die Zusammenarbeit viel einfacher gestaltet werden, und das gegenseitige Lernen durch die gemeinsame Durchführung von Code-Reviews wird die Qualität des SUT und das Wissen der TAEs noch weiter verbessern.

Reife des Projekts

Durch die Analyse der Reife des Projekts können Informationen abgeleitet werden, um das ideale Testautomatisierungsziel zu entwerfen. Durch die Ermittlung der verschiedenen Faktoren, wie z. B. die Anzahl der Testfälle, vorhandene CI/CD-Pipelines, Anzahl der Tester und TAEs, können verschiedene Testziele und eine Roadmap erstellt werden. Wenn es beispielsweise viele manuelle Testfälle gibt, die eine hohe Priorität haben und deren manuelle Testdurchführung lange dauert, ist es wichtig, diese zuerst zu automatisieren, um Zeit und Aufwand zu sparen. Abgesehen von den oben genannten Faktoren ist auch der Zeitrahmen des Projekts sehr wichtig. Wenn das Projekt kurz ist oder in Kürze endet und nicht mehr viel Zeit zur Verfügung steht, lohnt es sich nicht, eine große und robuste TAS zu planen. Handelt es sich jedoch um ein Projekt auf der grünen Wiese, rentiert es sich, eine schrittweise, auf dem Minimum Viable Product (MVP) basierende inkrementelle Roadmap zu erstellen.

Beteiligung der Stakeholder

In vielen Fällen wird die Testautomatisierung nicht genutzt, weil die wichtigsten Stakeholder sie nicht unterstützen. Dies kann auf die Befürchtung zurückzuführen sein, dass die Geschwindigkeit unter ihren

Zielwert fällt, auf knappe Fristen, auf die Budgetierung oder auf andere Bedenken. Nach der Analyse der Reife des Projekts muss ein strategischer Stakeholder die Produktrisiken ermitteln, die Vorteile der Testautomatisierung auflisten und einen entsprechenden Plan aufstellen, wenn die Implementierung einer TAS in Frage kommt. Dieser Plan muss die Meilensteine aufzeigen, die mit und durch die Testautomatisierung erreicht werden sollen. Außerdem sollte er angeben, wie die Testbarkeit des SUT verbessert werden muss, um eine TAS erfolgreich einzuführen, die die identifizierten Risiken reduziert. Abschließend muss dieser Plan den Stakeholdern vorgestellt werden.

Teamwissen und relevante Erfahrung

Der wichtigste und relevanteste Faktor für eine erfolgreiche Testautomatisierung sind die Fähigkeiten und Erfahrungen der Tester. Es ist von Vorteil, mit den Testern zusammenzuarbeiten und ihr Wissen zu nutzen, um Testziele zu definieren, mit denen sie und die Business-Analysten einverstanden sind. Häufig erstellen Testmanager oder Testleiter eine Kompetenz- und Fähigkeitsmatrix, um das verfügbare Wissen innerhalb der Teams zu ermitteln und gleichzeitig die vorhandenen Lücken zu identifizieren. Diese Lücken können mit verschiedenen Maßnahmen wie Schulungen und zugewiesenen Implementierungsaufgaben geschlossen werden.

Unterstützung durch das Testmanagement und Budgetierung

Je nach Größe und Reife des jeweiligen Projekts sollten bei der Festlegung der Testziele für die Testautomatisierung das verfügbare Budget und die Unterstützung durch das Testmanagement berücksichtigt werden. Es ist wichtig, Testziele zu definieren, die erfüllt werden können, was wiederum die Unterstützung durch das Testmanagement ermöglicht.

Wenn dem Management ein Vorschlag für eine Testautomatisierungsstrategie unterbreitet wird, muss die Dokumentation kurz und prägnant sein und klar die gegenwärtigen Lücken oder zukünftigen Kosten für die korrekte Umsetzung der Testautomatisierung definieren. Eine Liste von Empfehlungen und ihren Vorteilen, die den Geschäftswert und die Kostenreduzierung aufzeigt, wird das Testmanagement dazu ermutigen, die Entwicklung oder Verbesserung einer TAS zu genehmigen.

Qualitätsmerkmale

Die *ISO/IEC 25010 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE)* (2023) definiert die Qualitätsmerkmale, die in ISTQB-CTFL (2023), Abschnitt 2.2.2 Testarten aufgeführt sind. Diese Qualitätsmerkmale können dann zur Bewertung der aktuellen TAS oder zur Bestimmung von Metriken, die von der TAS gesammelt werden, verwendet werden.

5.4 Entscheidungen anhand von Berichten zur Testautomatisierung treffen

5.4.1 Testberichtsdaten zur Entscheidungsfindung analysieren

Das Format und der Inhalt eines Berichts zur Testautomatisierung kann je nach Stakeholdern, die ihn erhalten, variieren. Er kann für das Management, für betriebliche oder technische Stakeholder erstellt werden. Weitere Informationen sind in ISTQB-CTAL-TAE (2024), Abschnitt 6.1.3 zu finden.

Nach Erhalt eines Berichts zur Testautomatisierung kann dieser in einen umfassenderen Testbericht integriert oder konsolidiert werden und dann innerhalb der Organisationsstruktur verteilt werden.

Verschiedene Stakeholder erwarten unterschiedliche Inhalte in einem solchen Testautomatisierungsbericht. Ein strategischer Ansatz besteht darin, Schlüsselmetriken zu identifizieren, die für die beteiligten Stakeholder wichtig sind, und diese wichtigen Metriken hervorzuheben.

Die mit der Automatisierung gesammelten Daten können bei folgenden Punkten helfen:

- Erkennen von Trends und Durchführung einer Ursachenanalyse
- Verlagerung der Testautomatisierungsaufwände in die Wartung
- Verlagerung des Aufwands für die Testautomatisierung auf Verbesserungen und Weiterentwicklungen eines TAF
- Hinzufügen von Features zur gesamten TAS
- Verbesserung von Shift-Left- und Shift-Right-Ansätzen beim Testen
- Ausweitung der funktionalen Überdeckung der Testautomatisierung in zukünftigen Sprints
- Stärkere Fokussierung auf Fehlercluster
- Empfehlungen an die Entwickler über Bereiche zur Codeverbesserung
- Empfehlungen zu den gesamten SDLC-Prozessen
- Änderung des Inhalts und Formats zukünftiger Berichte zur Testautomatisierung

Auf der Grundlage der oben beschriebenen Informationen können die TAEs in Zusammenarbeit mit anderen Stakeholdern Lücken und bestimmte Verbesserungspunkte in der bestehenden Überdeckung der Testautomatisierung und der Testergebnisse identifizieren.

6 Strategien zur Realisierung und Verbesserung der Testautomatisierung - 150 Minuten (K3)

Schlüsselbegriffe

Testsuite, Überdeckung, Vorbedingung

Lernziele für Kapitel 6: Die Lernenden können ...

6.1 Übergang vom manuellen Testen zum kontinuierlichen Testen

TAS-6.1.1 (K2) ... die Faktoren und Planungsaktivitäten beim Übergang vom manuellen Testen zur Testautomatisierung beschreiben

TAS-6.1.2 (K2) ... die Faktoren und Planungsaktivitäten beim Übergang von der Testautomatisierung zum kontinuierlichen Testen beschreiben

6.2 Testautomatisierungsstrategie in der gesamten Organisation

TAS-6.2.1 (K3) ... Verbesserungsbereiche über eine Bewertung der Ressourcen und Praktiken der Testautomatisierung identifizieren

6.1 Übergang vom manuellen Testen zum kontinuierlichen Testen

6.1.1 Die Faktoren und Planungsaktivitäten beim Übergang vom manuellen Testen zur Testautomatisierung beschreiben

Die einfachste Möglichkeit, vom manuellen Testen zur Testautomatisierung überzugehen, ist die gezielte Automatisierung von Regressionstests. Eine Testsuite für Regressionstests wächst, wenn die funktionalen und nicht-funktionalen Tests von heute zu den Regressionstests von morgen werden. Es ist nur eine Frage der Zeit, bis die Anzahl der Regressionstests die Zeit und Verfügbarkeit eines traditionellen manuellen Testteams übersteigt.

Übergangskosten

Während des Übergangs vom manuellen Testen zum automatisierten Testen sollte das Projekt mit erhöhten Kosten rechnen, da sowohl manuelle Tests als auch automatisierte Tests gleichzeitig stattfinden. Sobald man davon ausgehen kann, dass automatisierte Tests die manuellen Testdurchführungstätigkeiten angemessen ersetzen oder ablösen, kann mehr Aufwand in Richtung explorativer Tests und der Definition zusätzlicher Testfälle für die Testautomatisierung verlagert werden, was im Vergleich zu manuellen Regressionstests mit anderen Kosten verbunden ist.

Funktionale Überschneidung

Funktionale Überschneidungen treten auf, wenn die Entwickler von Testskripten genau dieselben Testautomatisierungsschritte in verschiedene Testfälle aufnehmen. Die meisten Testfälle beginnen zum Beispiel mit einer Sequenz von Testschritten für das Login. Dies kann die Eingabe eines Benutzernamens, eines Passworts und die Auswahl einer Login-Taste umfassen. Das Hinzufügen dieser Schritte in jedem Testfall erhöht den Wartungsaufwand. Wenn dem Anmeldeprozess ein zusätzlicher Testschritt hinzugefügt wird, muss die gleiche Änderung in jedem Testfall nachgezogen werden. Ein besserer Ansatz besteht darin, den Anmeldeprozess zu einer wiederholbaren Testautomatisierungskomponente zu machen und alle Testfälle auf diese Funktionalität verweisen zu lassen. Siehe ISTQB-CTAL-TAE (2024), Abschnitt 3.1.5 für Details über das Flow Model Pattern.

Gemeinsame Datennutzung

Tests nutzen häufig Testdaten gemeinsam. Dies kann vorkommen, wenn Tests denselben Satz an Testdaten verwenden, um verschiedene SUT-Funktionalitäten auszuführen. Ein Beispiel hierfür wäre Testfall "A", der die noch verfügbaren Urlaubstage eines Mitarbeiters überprüft, während Testfall "B" überprüft, welche Kurse der Mitarbeiter im Rahmen seiner beruflichen Entwicklung besucht hat. Jeder Testfall verwendet denselben Mitarbeiter, prüft aber unterschiedliche Parameter. In einer manuellen Testumgebung würden die Testdaten des Mitarbeiters typischerweise in jedem manuellen Testfall, der die Testdaten des Mitarbeiters verifiziert, viele Male dupliziert werden. Bei einem automatisierten Test sollten die gemeinsam genutzten Testdaten jedoch, soweit möglich und praktikabel, in einer einzigen Quelle gespeichert und abgerufen werden, um Doppelarbeit oder Fehlhandlungen zu vermeiden.

Abhängigkeit der Tests untereinander

Bei der Durchführung komplexer Regressionstests kann es häufig vorkommen, dass ein Test von einem oder mehreren anderen Tests abhängig ist. Zum Beispiel wird als Ergebnis eines Testschritts eine neue "Auftrags-ID" erstellt. Nachfolgende Tests überprüfen möglicherweise, ob a) die neue Bestellung im System korrekt angezeigt wird, b) Änderungen an der Bestellung möglich sind oder c) das Löschen der Bestellung erfolgreich ist. In jedem Fall muss der "Auftrags-ID"-Wert, der im ersten Test dynamisch erzeugt

wird, für die Wiederverwendung durch spätere Tests festgehalten werden. Dies kann je nach Gestaltung der TAS unterschiedlich behandelt werden. Wenn die "Auftrags-ID" von nachfolgenden Testfällen nicht gefunden werden kann, schlagen diese fehl.

Vorbedingungen für die Testdurchführung

Allzu oft beginnen TAEs sofort mit der Entwicklung von Testskripten, ohne sich über die Vorbedingungen im Klaren zu sein, die sicherstellen, dass ein Testfall zuverlässig ausgeführt werden kann. Dies kann zu einer großen Herausforderung werden, wenn man versucht, einen Testfall gegen dasselbe SUT in mehreren Testumgebungen auszuführen. Beispiele für Vorbedingungen sind Benutzernamen, Konto-Rollen, Datentabellenwerte und andere spezifische Dateneinträge, die den Testfall wiederholbar machen. Eine gute Strategie beinhaltet eine Vorabanalyse, um zu ermitteln, welche Informationen im SUT vorhanden sein müssen, bevor ein bestimmter Testfall automatisiert werden kann. Zusätzlich kann die Strategie verbessert werden, indem die Erstellung von Vorbedingungen vor der Ausführung der eigentlichen Testfälle automatisiert wird, um Zeit zu sparen. Dies kann die Ausführung von Testskripten mit Vorbedingungen beinhalten, die das SUT von der Benutzeroberfläche aus befüllen, oder automatisierte Batch-Prozesse, die Testdaten in eine Datenbank laden.

Funktionale Überdeckung

Die Überdeckung kann verbessert werden, indem funktionale Lücken in Tests identifiziert werden, die, wie in Kapitel 3 erläutert, für eine Testautomatisierung in Frage kommen. 100% der manuellen Testfälle, die automatisiert werden, stellen nicht 100% aller möglichen Testfälle dar, die mit Testwerkzeugen automatisiert werden können. Je mehr Tests automatisiert werden, desto mehr Zeit gewinnen die Tester für die Testdurchführung zurück, die sie nutzen können, um zusätzliche SUT-Tests zu identifizieren und so die Überdeckung zu erhöhen.

Durchführbare Tests

Bevor ein manueller Regressionstest in einen automatisierten Regressionstest umgewandelt wird, ist es wichtig zu überprüfen, ob der manuelle Regressionstest korrekt funktioniert. Dies ist der richtige Ausgangspunkt, um eine erfolgreiche Umwandlung in einen automatisierten Regressionstest zu gewährleisten. Wenn der manuelle Regressionstest nicht korrekt ausgeführt wird, kann das daran liegen, dass er schlecht geschrieben wurde, ungültige Testdaten verwendet, veraltet oder nicht mit dem aktuellen SUT synchronisiert ist oder dass ein Fehlerzustand des SUT vorliegt. Die Automatisierung des Tests, bevor die Grundursache der Fehlerwirkung verstanden und/oder behoben wurde, führt zu einem nicht funktionierenden automatisierten Regressionstest, der unwirtschaftlich und unproduktiv ist. Es ist wichtig, die gleichwertige Funktionalität der neuen automatisierten Tests zu demonstrieren, um Vertrauen in die automatisierten Tests zu schaffen, die die alten manuellen Tests ersetzen werden.

6.1.2 Die Faktoren und Planungsaktivitäten beim Übergang von der Testautomatisierung zum kontinuierlichen Testen beschreiben

Beim kontinuierlichen Testen werden Testressourcen weitaus häufiger eingesetzt als in traditionellen SDLCs. Dies geschieht, indem Testsuiten unmittelbar nach der Durchführung von Codeänderungen und der Bereitstellung in den Testumgebungen kontinuierlich ausgeführt werden. Dies ermöglicht eine unmittelbare Rückmeldung und reduziert die Fehlerkosten, da Fehler früher im Prozess erkannt werden. Dies kann mit ausgefeilten Entwicklungswerkzeugen und der Bereitschaft, das Testen früher in den Build-Prozess zu verlagern, erreicht werden.

Die Anpassung der TAS für kontinuierliches Testen stellt folgende Anforderungen:

- Die Testsuiten müssen geändert werden, damit sie in der aktualisierten TAS laufen: Vor dem Einsatz in der TAS sind die erforderlichen Änderungen an den Testsuiten vorzunehmen und zu testen.
- Platzhalter, Treiber und Schnittstellen, die beim Testen verwendet werden, müssen an die aktualisierte TAS angepasst werden: Vor der Bereitstellung in der TAS sind die erforderlichen Änderungen am Testrahmen vorzunehmen und zu testen.
- Die Infrastruktur muss geändert werden, um der aktualisierten TAS gerecht zu werden: Eine Bewertung der Infrastrukturkomponenten, die geändert werden müssen, muss durchgeführt werden.
- Die aktualisierte TAS weist zusätzliche Fehlerzustände oder Performanzmängel auf: Eine Analyse der Risiken und des Nutzens muss durchgeführt werden. Wenn die entdeckten Fehlerzustände eine Aktualisierung der TAS unmöglich machen, ist es möglicherweise am besten, die Aktualisierung nicht fortzuführen oder auf eine neue Version der TAS zu warten. Wenn die Fehlerzustände im Vergleich zu den Vorteilen ihrer Behebung vernachlässigbar sind, kann die TAS dennoch aktualisiert werden.
- Es ist wichtig, Versionshinweise für bekannte Fehlerzustände zu erstellen, um die TAEs und andere Stakeholder zu informieren, und um abzuschätzen zu können, wann die Fehlerzustände behoben sein werden.

Alle in diesem Abschnitt genannten Punkte sind besonders wichtig, wenn CI/CD eingesetzt wird. Die Zusammenstellung von Pipelines und die Automatisierung des Build-Prozesses sind eine natürliche Ergänzung zu der zu entwickelnden TAS. Wenn sich das Build-Orchestrierungswerkzeug in der richtigen Testumgebung befindet, kann die Pipeline um automatisierte Tests erweitert werden, um das SUT direkt nach der Bereitstellung zu verifizieren. Dies setzt voraus, dass das Tool zur Testautomatisierung richtig konfiguriert ist und auf das SUT sowie auf das Code-Repository und die Skripte zur Datenbereitstellung zugreifen kann.

6.2 Testautomatisierungsstrategie in der gesamten Organisation

6.2.1 Verbesserungsbereiche über eine Bewertung der Ressourcen und Praktiken der Testautomatisierung identifizieren

Wie bei jeder anderen Entwicklungstätigkeit ist es wichtig, die Möglichkeit zu haben, die TAS-Entwicklung zu unterbrechen, um die Lösung zu überarbeiten. Dafür müssen auch folgende Bereiche überwacht werden: die anfängliche Implementierung, die Wartung und die Fähigkeit, die Lösung unter dem Gesichtspunkt der Wiederholbarkeit zu bewerten. Einige gute Kategorien, die man im Auge behalten sollte, sind die Anzahl der Stunden, die für die Entwicklung der TAS aufgewendet werden, die Anzahl der Stunden, die für die Korrektur der TAS benötigt werden, und die Zeitersparnis, die die Tester im Vergleich zum manuellen Testen erzielen. Ein Indikator dafür, dass etwas nicht stimmt, ist, wenn die Zeit für die Wartung der TAS die Zeit für das manuelle Testen übersteigt.

Vor dem ersten Einsatz einer TAS muss sichergestellt werden, dass sie in ihrer eigenen Umgebung laufen kann, dass sie von zufälligen Änderungen abgeschottet ist und dass die Testfälle aktualisiert und verwaltet werden können. Sowohl die TAS als auch ihre Infrastruktur müssen gewartet werden. Bei der erstmaligen Bereitstellung sind die folgenden grundlegenden Schritte erforderlich:

- Werkzeuge zur Codeüberdeckung zeigen an, welcher Anteil des Codes durch die Komponententestsuite geprüft wird und wo Lücken bestehen, die durch zusätzliche Komponententests abgedeckt werden können.
- Die funktionale Überdeckung des SUT kann durch die Erstellung einer Anforderungs-Verfolgbarkeitsmatrix ermittelt werden, die aufzeigt, welche Funktionalität noch nicht durch Testfälle abgedeckt ist.
- Definition einer konsistenten Nutzung der Infrastruktur der TAS über Projekte oder die gesamte Organisation hinweg.
- Entwicklung eines konsistenten Konfigurationsmanagements für die Testsuiten.
- Erstellung gemeinsamer TAS-Entwicklungsrichtlinien unter Nutzung der in ISTQB-CTAL-TAE (2024), Abschnitt 3.1.4 beschriebenen Best Practices.
- Implementierung von Vorbedingungen. Oft kann ein Test nicht ausgeführt werden, bevor nicht Vorbedingungen festgelegt wurden. Zu diesen Vorbedingungen kann die Auswahl der richtigen Datenbank oder der Testdaten gehören, mit denen getestet werden soll, oder die Festlegung von Anfangswerten oder Parametern. Viele dieser Initialisierungsschritte, die zur Festlegung der Vorbedingungen eines Tests erforderlich sind, können automatisiert werden. Dies ermöglicht eine zuverlässigere Lösung, wenn diese Schritte vor der Ausführung der Tests nicht mehr übersehen werden können. Wenn Regressionstests in die Testautomatisierung überführt werden, müssen diese Vorbedingungen Teil des Testautomatisierungsprozesses sein.

Bei inkrementellen TAS-Updates für neue Funktionen oder zu Wartungszwecken sollte Folgendes beachtet werden:

- Prüfung von Updates für Testwerkzeuge oder von anderen neueren Testwerkzeugen, die erweiterte Funktionen für die TAS bieten können.
- Prüfung von Möglichkeiten zur weiteren Optimierung der Funktionen und der Leistung der TAS.
- Identifizierung von Möglichkeiten, Testskripte weiter zu zerlegen und zu modularisieren, um die Wiederverwendbarkeit zu verbessern.
- Sicherstellung der Kenntnisse und des Bewusstseins für wiederverwendbare Komponenten und deren konsequente Nutzung.
- Sammeln von Erkenntnissen über potenzielle Verbesserungsbereiche, um Empfehlungen zu deren Nutzen zu geben.
- Bewertung und Korrektur von Bereichen mit funktionalen Überschneidungen. Bei der Automatisierung bestehender Regressionstests empfiehlt es sich, funktionale Überschneidungen zwischen den Testfällen zu identifizieren und, wenn möglich, bereits entwickelte Komponenten zur Testautomatisierung wiederzuverwenden.
- Bewertung zusätzlicher manueller Testfälle im Hinblick auf ihre Automatisierung und die Erstellung von Backlog-Elementen für die Implementierung.
- Aufzeigen von Möglichkeiten zur Verbesserung des Testentwurfs und des Testdatenmanagements.
- Sicherstellen, dass alle bestehenden Testsuiten an die neueste Version der TAS angepasst werden.

- Wenn die Testautomatisierung zu einer Warteschlange in der Pipeline führt, wird der Umfang der integrierten Tests auf die kritischsten Tests reduziert, d. h., es wird eine Smoke-Test-Suite erstellt. Die größere Testsuite für Regressionstests kann separat ausgelöst oder bei Bedarf ausgeführt werden.

7 Anhänge A – Lernziele/kognitiver Wissensstand

Die nachfolgenden Lernziele sind für diesen Lehrplan definiert. Jedes Thema des Lehrplans wird anhand des jeweiligen Lernziels untersucht.

Die Lernziele beginnen mit einem Aktionsverb, das dem jeweiligen kognitiven Wissensstand entspricht, wie unten aufgeführt.

Stufe 2: Verstehen (K2)

Der Kandidat kann die Gründe oder Erklärungen für Aussagen zum Thema auswählen und kann das Konzept des Testens zusammenfassen, vergleichen, einordnen und Beispiele nennen.

Aktionsverben: Klassifizieren, vergleichen, differenzieren, unterscheiden, erklären, Beispiele nennen, schließen (auf), zusammenfassen

Beispiele	Anmerkungen
Klassifizieren Sie Testwerkzeuge nach ihrem Zweck und den Testaktivitäten, die sie unterstützen.	
Vergleichen Sie die verschiedenen Teststufen.	Kann verwendet werden, um nach Ähnlichkeiten, Unterschieden oder beidem zu suchen.
Differenzieren Sie zwischen Testen und Debugging.	Sucht nach Unterschieden zwischen Konzepten.
Unterscheiden Sie zwischen Projektrisiken und Produktrisiken.	Ermöglicht die separate Klassifizierung von zwei (oder mehr) Konzepten.
Erklären Sie den Einfluss des Kontexts auf den Testprozess.	
Nennen Sie Beispiele dafür, warum Testen notwendig ist.	
Schließen Sie aus einem gegebenen Profil von Fehlerzuständen auf die Grundursache von Fehlern.	
Fassen Sie die Aktivitäten des Reviewprozesses für Arbeitsergebnisse zusammen.	

Stufe 3: Anwenden (K3)

Der Kandidat kann ein Verfahren ausführen, wenn er mit einer vertrauten Aufgabe konfrontiert wird, oder das richtige Verfahren auswählen und es auf einen gegebenen Kontext anwenden.

Aktionsverben: Anwenden, umsetzen, vorbereiten, nutzen

Beispiele	Anmerkungen
Wenden Sie die Grenzwertanalyse an, um Testfälle aus gegebenen Anforderungen abzuleiten.	Sollte sich auf ein Verfahren / eine Technik / einen Prozess etc. beziehen.
Setzen Sie Methoden zur Erfassung von Metriken um, um technische und Managementanforderungen zu unterstützen.	
Bereiten Sie eine Testautomatisierungsumgebung vor.	
Nutzen Sie die Verfolgbarkeit, um den Testfortschritt auf Vollständigkeit und Konsistenz mit den Testzielen, der Teststrategie und dem Testkonzept zu überwachen.	Könnte in einem LO verwendet werden, in dem der Kandidat in der Lage sein soll, eine Technik oder ein Verfahren zu nutzen. Ähnlich wie "anwenden".

Referenz (Für die kognitiven Ebenen von Lernzielen)

Anderson, L. W. und Krathwohl, D. R. (Hrsg.) (2001) A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives, Allyn & Bacon.

8 Anhänge B – Matrix zur Verfolgbarkeit des geschäftlichen Nutzen (Business Outcomes) mit Lernzielen (Learning Objectives)

Dieser Abschnitt listet die Anzahl der Lernziele des vorliegenden Lehrplans auf, die mit dem geschäftlichen Nutzen in Verbindung stehen, sowie die Verfolgbarkeit zwischen dem geschäftlichen Nutzen des Lehrplans und den Lernzielen des Specialist Test Automation Strategy.

Geschäftlicher Nutzen: Specialist Test Automation Strategy		TAS-BO1	TAS-BO2	TAS-BO3	TAS-BO4	TAS-BO5	TAS-BO6	TAS-BO7	TAS-BO8	TAS-BO9	TAS-BO10	TAS-BO11	TAS-BO12	TAS-BO13	TAS-BO14	TAS-BO15
TAS-BO1	Software- und Systemfaktoren verstehen, die den Erfolg der Testautomatisierung beeinflussen	3														
TAS-BO2	Kosten und Risiken der Implementierung einer Testautomatisierungslösung identifizieren		3													
TAS-BO3	Rollen und Verantwortlichkeiten von Personen verstehen, die an der Testautomatisierung beteiligt sind			1												
TAS-BO4	Integration der Testautomatisierung über Teststufen hinweg planen				3											
TAS-BO5	Strategische Überlegungen zur Testautomatisierung in verschiedenen Softwareentwicklungslebenszyklus-Modellen anstellen					3										
TAS-BO6	Anwendbarkeit und Durchführbarkeit von Testautomatisierung verstehen						3									
TAS-BO7	Testautomatisierungslösungen planen, die den organisatorischen Anforderungen entsprechen							3								

Geschäftlicher Nutzen: Specialist Test Automation Strategy		TAS-BO1	TAS-BO2	TAS-BO3	TAS-BO4	TAS-BO5	TAS-BO6	TAS-BO7	TAS-BO8	TAS-BO9	TAS-BO10	TAS-BO11	TAS-BO12	TAS-BO13	TAS-BO14	TAS-BO15
TAS-BO8	Einsatzstrategien für die Testautomatisierung verstehen								3							
TAS-BO9	Abhängigkeiten der Testautomatisierung innerhalb der Testumgebung verstehen									3						
TAS-BO10	Kosten für die Einrichtung und Wartung von Testautomatisierung ermitteln										1					
TAS-BO11	Metriken kennen, die die Entscheidungsfindung für eine Testautomatisierung unterstützen											1				
TAS-BO12	Mehrwert erkennen, den eine Testautomatisierung für das Projekt und die Organisation bringt												2			
TAS-BO13	Anforderungen an die Testautomatisierung und die Testberichterstattung identifizieren, um die Bedürfnisse der Stakeholder zu erfüllen													1		
TAS-BO14	Übergangsaktivitäten vom manuellen Testen zu einer Testautomatisierung definieren														2	
TAS-BO15	Eine Testautomatisierungsstrategie definieren, die sicherstellt, dass Projekte Ressourcen und Methoden gemeinsam nutzen, um eine konsistente Umsetzung innerhalb der Organisation zu gewährleisten															1

Geschäftlicher Nutzen: Specialist Test Automation Strategy		TAS-BO1	TAS-BO2	TAS-BO3	TAS-BO4	TAS-BO5	TAS-BO6	TAS-BO7	TAS-BO8	TAS-BO9	TAS-BO10	TAS-BO11	TAS-BO12	TAS-BO13	TAS-BO14	TAS-BO15
LO-Nummer	Lernziel (K-level)															
1	Einführung und Ziele der Testautomatisierungsstrategie - 45 Minuten (K2)															
1.1	Erfolgsfaktoren eines Testautomatisierungsprojekts															
TAS-1.1.1	... Ziele und Zielsetzungen einer Testautomatisierungsstrategie definieren (K2)	X														
TAS-1.1.2	... technische Erfolgsfaktoren eines Testautomatisierungsprojekts identifizieren (K2)	X														
TAS-1.1.3	... geeignete Investitionskriterien bei der Auswahl von Projekten zur Testautomatisierung zusammenfassen (K2)	X														
2	Ressourcen für die Testautomatisierung - 60 Minuten (K2)															
2.1	Kosten und Risiken der Implementierung einer Testautomatisierungslösung															
TAS-2.1.1	... alternative technische Lösungen im Hinblick auf die Betriebskosten vergleichen (K2)		X													
TAS-2.1.2	... Überlegungen zum Lizenzmodell für Werkzeuge zur Testautomatisierung erläutern (K2)		X													
TAS-2.1.3	... Beispiele für Faktoren nennen, die bei der Definition einer Testautomatisierungsstrategie zu berücksichtigen sind (K2)		X													
2.2	Rollen und Verantwortlichkeiten innerhalb der Testautomatisierung															

Geschäftlicher Nutzen: Specialist Test Automation Strategy		TAS-BO1	TAS-BO2	TAS-BO3	TAS-BO4	TAS-BO5	TAS-BO6	TAS-BO7	TAS-BO8	TAS-BO9	TAS-BO10	TAS-BO11	TAS-BO12	TAS-BO13	TAS-BO14	TAS-BO15
TAS-2.2.1	... die Rollen und Fähigkeiten zusammenfassen, die für eine erfolgreiche Testautomatisierungslösung notwendig sind (K2)			X												
3	Vorbereitungen für die Testautomatisierung - 225 Minuten (K3)															
3.1	Integration über Teststufen hinweg															
TAS-3.1.1	... zwischen verschiedenen Testverteilungsmöglichkeiten der Testautomatisierung unterscheiden (K2)				X											
TAS-3.1.2	... einen Testautomatisierungsansatz auf der Grundlage der Architektur des Systems unter Test auswählen (K2)				X											
TAS-3.1.3	... Wege zur Optimierung der Testautomatisierung aufzeigen, um Shift-Left- und Shift-Right-Ansätze zu erreichen (K3)				X											
3.2	Strategische Überlegungen zu verschiedenen Modellen des Softwareentwicklungslebenszyklus															
TAS-3.2.1	... erläutern, wie Testautomatisierungsprojekte mit bestehenden Softwareentwicklungslebenszyklus-Modellen übereinstimmen (K2)					X										
TAS-3.2.2	... erläutern, wie Projekte zur Testautomatisierung mit den Best Practices der agilen Softwareentwicklung übereinstimmen, die die Testautomatisierung unterstützen (K2)					X										
TAS-3.2.3	... Testautomatisierungsprojekte mit DevOps Best Practices vorbereiten, um kontinuierliches Testen zu erreichen (K3)					X										

Geschäftlicher Nutzen: Specialist Test Automation Strategy		TAS-BO1	TAS-BO2	TAS-BO3	TAS-BO4	TAS-BO5	TAS-BO6	TAS-BO7	TAS-BO8	TAS-BO9	TAS-BO10	TAS-BO11	TAS-BO12	TAS-BO13	TAS-BO14	TAS-BO15
3.3	Anwendbarkeit und Durchführbarkeit der Testautomatisierung															
TAS-3.3.1	... Kriterien zur Bestimmung der Eignung von Tests für die Testautomatisierung erläutern (K2)						X									
TAS-3.3.2	... Herausforderungen identifizieren, die nur durch Testautomatisierung gelöst werden können (K2)						X									
TAS-3.3.3	... Testbedingungen identifizieren, die schwierig zu automatisieren sind (K2)						X									
4	Organisatorische Einführungs- und Freigabestrategien für die Testautomatisierung - 135 Minuten (K2)															
4.1	Testautomatisierungslösung planen															
TAS-4.1.1	... identifizieren, wie Testautomatisierung eine kürzere Markteinführungszeit unterstützt (K2)							X								
TAS-4.1.2	... identifizieren, wie Testautomatisierung dabei unterstützt, gemeldete Fehlerzustände gemäß den Anforderungen zu verifizieren (K2)							X								
TAS-4.1.3	... Ansätze definieren, die die Entwicklung von betriebsrelevanten Szenarien für die Testautomatisierung ermöglichen (K2)							X								
4.2	Einführungsstrategien für die Testautomatisierung															
TAS-4.2.1	... eine Einführungsstrategie für die Testautomatisierung definieren (K2)								X							

Geschäftlicher Nutzen: Specialist Test Automation Strategy		TAS-BO1	TAS-BO2	TAS-BO3	TAS-BO4	TAS-BO5	TAS-BO6	TAS-BO7	TAS-BO8	TAS-BO9	TAS-BO10	TAS-BO11	TAS-BO12	TAS-BO13	TAS-BO14	TAS-BO15
TAS-4.2.2	... Risiken der Testautomatisierung bei der Einführung identifizieren (K2)								X							
TAS-4.2.3	... Ansätze zur Risikominderung beim Einsatz von Tests definieren (K2)								X							
4.3	Abhängigkeiten innerhalb der Testumgebung															
TAS-4.3.1	... Komponenten für die Testautomatisierung in der Testumgebung definieren (K2)									X						
TAS-4.3.2	... Infrastrukturkomponenten und Abhängigkeiten der Testautomatisierung identifizieren (K2)									X						
TAS-4.3.3	... Anforderungen an die Daten und Schnittstellen der Testautomatisierung für die Integration in das System unter Test definieren (K2)									X						
5	Auswirkungsanalyse der Testautomatisierung - 150 Minuten (K3)															
5.1	Investitionen in den Aufbau und die Pflege der Testautomatisierung															
TAS-5.1.1	... den Return on Investment für den Aufbau einer Testautomatisierungslösung aufzeigen (K3)										X					
5.2	Metriken zur Testautomatisierung															
TAS-5.2.1	... Metriken für die Testautomatisierung klassifizieren (K2)											X				

Geschäftlicher Nutzen: Specialist Test Automation Strategy		TAS-BO1	TAS-BO2	TAS-BO3	TAS-BO4	TAS-BO5	TAS-BO6	TAS-BO7	TAS-BO8	TAS-BO9	TAS-BO10	TAS-BO11	TAS-BO12	TAS-BO13	TAS-BO14	TAS-BO15
5.3	Der Wert der Testautomatisierung auf Projekt- und Organisationsebene															
TAS-5.3.1	... organisatorische Überlegungen zum Einsatz der Testautomatisierung identifizieren (K3)												X			
TAS-5.3.2	... Projektmerkmale analysieren, die zur Bestimmung der optimalen Testziele für die Testautomatisierung beitragen (K3)												X			
5.4	Entscheidungen anhand von Berichten zur Testautomatisierung treffen															
TAS-5.4.1	... Testberichtsdaten zur Entscheidungsfindung analysieren (K2)													X		
6	Strategien zur Realisierung und Verbesserung der Testautomatisierung - 150 Minuten (K3)															
6.1	Übergang vom manuellen Testen zum kontinuierlichen Testen															
TAS-6.1.1	... die Faktoren und Planungsaktivitäten beim Übergang vom manuellen Testen zur Testautomatisierung beschreiben (K2)														X	
TAS-6.1.2	... die Faktoren und Planungsaktivitäten beim Übergang von der Testautomatisierung zum kontinuierlichen Testen beschreiben (K2)														X	
6.2	Testautomatisierungsstrategie in der gesamten Organisation															

Geschäftlicher Nutzen: Specialist Test Automation Strategy		TAS-BO1	TAS-BO2	TAS-BO3	TAS-BO4	TAS-BO5	TAS-BO6	TAS-BO7	TAS-BO8	TAS-BO9	TAS-BO10	TAS-BO11	TAS-BO12	TAS-BO13	TAS-BO14	TAS-BO15
TAS-6.2.1	... Verbesserungsbereiche über eine Bewertung der Ressourcen und Praktiken der Testautomatisierung identifizieren (K3)															X

9 Anhänge C – Release Notes

Der ISTQB® Test Automation Strategy Syllabus 2024 ist ein neuer ISTQB®-Lehrplan, der strategische Aspekte aus dem vorherigen ISTQB®-Lehrplan Test Automation Engineer aus dem Jahr 2016 mit zusätzlichen Aktualisierungen und aktuellen Best Practices für die Implementierung und Messung des Erfolgs der Testautomatisierung kombiniert. Aus diesem Grund gibt es keine detaillierten Release Notes pro Kapitel und Abschnitt.

10 Anhänge D – Bereichsspezifische Begriffe

Begriffsname	Definition
Canary Release	Eine Bereitstellungs- und Teststrategie zur Verringerung des Risikos und zur Überprüfung neuer Software, indem die Software nur für einige wenige Benutzer freigegeben wird.
Container	Eine Softwareeinheit, die Code und seine Abhängigkeiten verpackt, sodass die Software schnell und zuverlässig in verschiedenen Umgebungen ausgeführt werden kann.
DevOps	Eine Methodik, die die Arbeit von Softwareentwicklung und IT-Betrieb integriert und automatisiert, um den Softwareentwicklungslebenszyklus zu verbessern und zu verkürzen.
Flow Model Pattern	Eine High-Level-Ansicht der Arbeitsdomäne, ihrer Komponenten und der Verbindungen zwischen ihnen.

11 Referenzen

Normen

ISO/IEC 25010 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE): Product quality mode, 2023. Standard. International Organization for Standardization.

ISTQB® Dokumente

ISTQB-CTAL-TAE, ISTQB®, 2024. *Certified Tester Test Automation Engineering Syllabus*. v2.0.

ISTQB-CTEL-TM, ISTQB®, 2011. *Certified Tester Expert Level Test Management Syllabus*. v1.0.

ISTQB-CTFL, ISTQB®, 2023. *Certified Tester Foundation Level Syllabus*. v4.0.

12 Weitere Literatur

BAKER, Paul; DAI, Zhen Ru; GRABOWSKI, Jens; HAUGEN, Oystein; SCHIEFERDECKER, Ina; WILLIAMS, Clay E., 2007. *Model-Driven Testing - Using the UML Testing Profile*. Springer-Verlag Berlin Heidelberg. ISBN 9783540725626.

BINDER, Robert V.; MILLER, Suzanne, 2017. Five Keys to Effective Agile Test Automation for Government Programs. *Software Engineering Institute, Carnegie Mellon University*. Auch verfügbar unter: https://resources.sei.cmu.edu/asset_files/Webinar/2017_018_101_503516.pdf.

CIO, DoD, 2023. Modern Software Practices - DevSecOps Fundamentals Guidebook: Activities and Werkzeuge. Jg. v2.2. Auch verfügbar unter: https://dodcio.defense.gov/Portals/0/Documents/Library/DevSecOpsActivitesToolsGuidebookTables.pdf.

DUSTIN, Elfriede; GARRETT, Thom; GAUF, Bernie, 2009. *Implementing Automated Software Testing: How to Save Time and Lower Costs While Raising Quality*. Addison-Wesley Professional. ISBN 0321580516.

DUSTIN, Elfriede; RASHKA, Jeff; PAUL, John, 1999. *Automated Software Testing: Introduction, Management, and Performance*. Addison-Wesley Professional. ISBN 9780201432879.

GRAHAM, Dorothy; FEWSTER, Mark, 1999. *Software Test Automation: Effective Use of Test Execution Tools*. Addison-Wesley Professional. ISBN 9780201331400.

GRAHAM, Dorothy; FEWSTER, Mark, 2012. *Experiences of Test Automation: Case Studies of Software Test Automation*. Addison-Wesley Professional. ISBN 9780321754066.

ISO/IEC/IEEE 29119-1 Software and systems engineering — Software testing: Part 1: General Concepts, 2022. Standard. International Organization for Standardization.

ISTQB-CT-PT, ISTQB®: Certified Tester Performance Testing Syllabus, 2023. v1.0.

ISTQB-CT-SEC, ISTQB®, 2016. *Certified Tester Security Tester Syllabus*. v1.0.

ISTQB-GLOSSARY, ISTQB®, [o. D.]. *https://glossary.istqb.org*.

JAYACHANDRAN, Naveen, 2005. Understanding ROI Metrics for Software Test Automation. Auch verfügbar unter: https://digitalcommons.usf.edu/cgi/viewcontent.cgi?article=3937&context=etd.

JOSE, Boby, 2021. *Test Automation: A Manager's Guide*. BCS. ISBN 9781780175478.

MCCAFFREY, James D., 2006. *.NET Test Automation Recipes: A Problem-Solution Approach*. APRESS. ISBN 9781590596630.

MOSLEY, Daniel J.; POSEY, Bruce A., 2002. *Just Enough Software Test Automation*. Prentice Hall. ISBN 9780130084682.

PESTAK, Thomas; ROWELL, William, 2018. Automated Software Testing Practices and Pitfalls. Auch verfügbar unter: https://www.afit.edu/stat/statcoe_files/Automated%20Software%20Testing%20Practices%20and%20Pitfalls%20Rev%201.pdf.

POLLNER, Andrew; SIMPSON, Jim; WISNOWSKI, Jim, 2018. Automated Software Testing Implementation Guide for Managers and Practitioners. Auch verfügbar unter: https://www.afit.edu/stat/statcoe_files/Automated_Software_Testing_Implementation_Guide.pdf.

ROSENTHAL, Casey, 2020. *Chaos Engineering: System Resiliency in Practice*. O'Reilly. ISBN 9781492043867.

WILLCOCK, Colin; DEISS, Thomas; TOBIES, Stephan; KEIL, Stefan; ENGLER, Federico; SCHULZ, Stephan, 2011. *An Introduction to TTCN-3*. Wiley. ISBN 9780470663066.

13 Abbildungsverzeichnis

14 Index