**ISTQB FRAGEN**

**3. Die generische Testautomatisierungsarchitektur –**

**### K2-Frage (Verstehen von Konzepten):**

\*\*Frage:\*\* Was ermöglicht die generische Testautomatisierungsarchitektur (gTAA) im Entwicklungsprozess einer Testautomatisierungslösung (TAS)?

\*\*Antwortoptionen:\*\*

a) Sie definiert spezifische Technologien und Werkzeuge für die Implementierung.

b) Sie bietet eine anbieterneutrale Struktur und unterstützt verschiedene Softwareentwicklungsansätze.

c) Sie schreibt vor, dass alle TAS-Komponenten in einer einzigen Programmiersprache geschrieben werden müssen.

d) Sie eliminiert die Notwendigkeit von Wartung und Weiterentwicklung der TAS.

\*\*Richtige Antwort:\*\* b) Sie bietet eine anbieterneutrale Struktur und unterstützt verschiedene Softwareentwicklungsansätze.

**### K3-Frage (Anwendung von Konzepten):**

\*\*Frage:\*\* In einem Projekt wird die gTAA verwendet, um eine Testautomatisierungslösung (TAS) zu entwickeln. Welche Maßnahmen sollten ergriffen werden, um die Wartbarkeit und Erweiterbarkeit der TAS zu gewährleisten?

\*\*Antwortoptionen:\*\*

a) Alle TAS-Komponenten sollten manuell getestet werden, ohne auf Testautomatisierungswerkzeuge zurückzugreifen.

b) Die TAS-Komponenten sollten fest definierte Aufgaben haben und offen für Erweiterungen, aber geschlossen gegenüber Modifikationen sein.

c) Die TAS-Komponenten sollten keine Abhängigkeiten von externen Schnittstellen haben.

d) Alle TAS-Komponenten sollten in einer einzigen, zentralen Datei gespeichert werden.

\*\*Richtige Antwort:\*\* b) Die TAS-Komponenten sollten fest definierte Aufgaben haben und offen für Erweiterungen, aber geschlossen gegenüber Modifikationen sein.

**### K4-Frage (Analyse und Bewertung):**

\*\*Frage:\*\* Sie sind verantwortlich für die Entwicklung einer TAS, die auf der gTAA basiert, und sollen sicherstellen, dass die TAS für verschiedene Softwareproduktreihen wiederverwendbar ist. Welche Ansätze und Prinzipien sollten Sie berücksichtigen, um dieses Ziel zu erreichen?

\*\*Antwortoptionen:\*\*

a) Implementierung einer starren Architektur, die für jedes Produkt angepasst werden muss.

b) Nutzung von Komponenten, die durch spezifische automatisierte Testszenarien definiert sind.

c) Anwendung von Prinzipien wie Komponentensegregation und Abhängigkeitsumkehr, um eine modulare und flexible Architektur zu schaffen.

d) Vermeidung der Dokumentation von TAS-Artefakten, um Entwicklungszeit zu sparen.

\*\*Richtige Antwort:\*\* c) Anwendung von Prinzipien wie Komponentensegregation und Abhängigkeitsumkehr, um eine modulare und flexible Architektur zu schaffen.

### Fragen für den gegebenen Abschnitt

#### K2-Frage (Verstehen von Konzepten):

\*\*Frage:\*\* Welches Prinzip besagt, dass jede TAS-Komponente für eine bestimmte Aufgabe zuständig sein sollte und diese vollständig in der Komponente gekapselt sein muss?

\*\*Antwortoptionen:\*\*

a) Erweiterbarkeit

b) Ersetzbarkeit

c) Komponentensegregation

d) Fest definierte Aufgabe

\*\*Richtige Antwort:\*\* d) Fest definierte Aufgabe

#### K3-Frage (Anwendung von Konzepten):

\*\*Frage:\*\* Sie müssen eine TAS entwerfen, die sowohl wiederverwendbar als auch wartbar ist. Welche Prinzipien sollten Sie bei der Gestaltung der TAS-Komponenten berücksichtigen, um diese Ziele zu erreichen?

\*\*Antwortoptionen:\*\*

a) Verwenden Sie eine einzige allgemeine Komponente für alle Aufgaben, um Komplexität zu reduzieren.

b) Stellen Sie sicher, dass jede Komponente offen für Erweiterungen, aber geschlossen für Modifikationen ist, und trennen Sie die Komponenten nach ihren spezifischen Aufgaben.

c) Binden Sie die Komponenten direkt an spezifische automatisierte Testszenarios.

d) Fügen Sie alle Funktionalitäten in eine zentrale Komponente ein, um die Verwaltung zu erleichtern.

\*\*Richtige Antwort:\*\* b) Stellen Sie sicher, dass jede Komponente offen für Erweiterungen, aber geschlossen für Modifikationen ist, und trennen Sie die Komponenten nach ihren spezifischen Aufgaben.

#### K4-Frage (Analyse und Bewertung):

\*\*Frage:\*\* Sie sind verantwortlich für die Entwicklung einer TAS, die auf der gTAA basiert, und soll sicherstellen, dass die Testautomatisierung für verschiedene Softwareproduktreihen wiederverwendbar ist. Welche Maßnahmen und Prinzipien sollten Sie umsetzen, um die langfristige Wartbarkeit und Erweiterbarkeit der TAS zu gewährleisten?

\*\*Antwortoptionen:\*\*

a) Entwickeln Sie alle TAS-Komponenten ohne Rücksicht auf zukünftige Erweiterungen und Modifikationen, um die Entwicklungszeit zu verkürzen.

b) Implementieren Sie die TAS-Komponenten nach dem Open-Closed-Prinzip und sorgen Sie für Komponentensegregation, um unnötige Abhängigkeiten zu vermeiden.

c) Verwenden Sie ein festes Set von Technologien und Werkzeugen für alle TAS-Komponenten, um die Konsistenz zu gewährleisten.

d) Entfernen Sie alle Testschnittstellen nach der ersten Testphase, um die Sicherheitsrisiken zu minimieren.

\*\*Richtige Antwort:\*\* b) Implementieren Sie die TAS-Komponenten nach dem Open-Closed-Prinzip und sorgen Sie für Komponentensegregation, um unnötige Abhängigkeiten zu vermeiden.

### Fragen für den gegebenen Abschnitt

#### K2-Frage (Verstehen von Konzepten):

\*\*Frage:\*\* Welche der folgenden Funktionen gehört zur Testgenerierungsschicht?

\*\*Antwortoptionen:\*\*

a) Einrichten und Aufräumen von Testsuites

b) Verknüpfen von Testfällen mit Testzielen oder SUT-Anforderungen

c) Validieren der Reaktionen des SUT

d) Steuerung des Testrahmens

\*\*Richtige Antwort:\*\* b) Verknüpfen von Testfällen mit Testzielen oder SUT-Anforderungen

#### K3-Frage (Anwendung von Konzepten):

\*\*Frage:\*\* Sie sind verantwortlich für die Implementierung einer Testautomatisierungslösung (TAS). Welche Maßnahmen sollten Sie in der Testdefinitionsschicht ergreifen, um sicherzustellen, dass die Testfälle auf abstrakter und konkreter Ebene festgelegt werden können?

\*\*Antwortoptionen:\*\*

a) Nur konkrete Testfälle definieren, um die Ausführung zu erleichtern.

b) Testdaten partitionieren und parametrisieren, um unterschiedliche Testfälle zu spezifizieren.

c) Die Testgenerierungsschicht verwenden, um Testdaten zu erfassen.

d) Testfälle manuell ohne Nutzung von Testbibliotheken spezifizieren.

\*\*Richtige Antwort:\*\* b) Testdaten partitionieren und parametrisieren, um unterschiedliche Testfälle zu spezifizieren.

#### K4-Frage (Analyse und Bewertung):

\*\*Frage:\*\* Sie sollen eine TAS entwerfen, die eine effektive Validierung der Reaktionen des SUT ermöglicht und dabei verschiedene Technologien und Umgebungen unterstützt. Welche Aspekte und Komponenten der Testausführungsschicht und der Testadaptierungsschicht sollten besonders berücksichtigt werden, um dieses Ziel zu erreichen?

\*\*Antwortoptionen:\*\*

a) Konzentrieren Sie sich nur auf die automatische Ausführung von Testfällen und die Protokollierung der Ergebnisse.

b) Implementieren Sie technologiespezifische Adapter in der Testadaptierungsschicht, um die Interaktion mit dem SUT zu ermöglichen und sicherzustellen, dass das SUT zur Testausführung richtig eingerichtet wird.

c) Verwenden Sie ausschließlich manuelle Tests, um die Reaktionen des SUT zu überprüfen.

d) Stellen Sie sicher, dass die Testgenerierungsschicht für die Modellierung des SUT und seiner Umgebung verantwortlich ist.

\*\*Richtige Antwort:\*\* b) Implementieren Sie technologiespezifische Adapter in der Testadaptierungsschicht, um die Interaktion mit dem SUT zu ermöglichen und sicherzustellen, dass das SUT zur Testausführung richtig eingerichtet wird.

### Fragen für den gegebenen Abschnitt

#### K2-Frage (Verstehen von Konzepten):

\*\*Frage:\*\* Was sollte das Konfigurationsmanagement einer TAS umfassen, um sicherzustellen, dass sie mit den verschiedenen Versionen des SUT kompatibel ist?

\*\*Antwortoptionen:\*\*

a) Nur die Testskripte und Testdaten

b) Testmodelle, Testdefinitionen, Testskripte, Testadapter, Simulatoren, Testergebnisse und Testberichte

c) Nur die Maschine für die Testausführung und die ergänzenden Werkzeuge

d) Nur die Testberichte und Testergebnisse

\*\*Richtige Antwort:\*\* b) Testmodelle, Testdefinitionen, Testskripte, Testadapter, Simulatoren, Testergebnisse und Testberichte

#### K3-Frage (Anwendung von Konzepten):

\*\*Frage:\*\* Sie leiten ein Projekt zur Entwicklung einer TAS. Welche Maßnahmen sollten Sie ergreifen, um sicherzustellen, dass Statusinformationen (Metriken) während der Entwicklung einfach extrahiert und dem Projektmanagement automatisch übermittelt werden können?

\*\*Antwortoptionen:\*\*

a) Die Entwicklungsumgebung der TAS so auslegen, dass sie keine Metriken erfasst.

b) Nur manuelle Berichte verwenden, um den Projektstatus zu dokumentieren.

c) Die Entwicklungsumgebung der TAS so auslegen, dass Statusinformationen (Metriken) automatisch extrahiert und dem Projektmanagement übermittelt werden können.

d) Alle Metriken in einer separaten Datenbank speichern, ohne sie zu integrieren.

\*\*Richtige Antwort:\*\* c) Die Entwicklungsumgebung der TAS so auslegen, dass Statusinformationen (Metriken) automatisch extrahiert und dem Projektmanagement übermittelt werden können.

#### K4-Frage (Analyse und Bewertung):

\*\*Frage:\*\* Sie sind verantwortlich für das Konfigurationsmanagement einer TAS, die in mehreren Iterationen entwickelt wird. Wie würden Sie sicherstellen, dass ältere Versionen der TAS verwendet werden können, um Fehler im Produktivbetrieb mit älteren SUT-Versionen zu reproduzieren?

\*\*Antwortoptionen:\*\*

a) Implementieren Sie ein Versionierungssystem, das alle Testmittel (einschließlich Testmodelle, Testskripte, Testadapter und Simulatoren) verwaltet und sicherstellt, dass jede Version der TAS mit der entsprechenden Version des SUT kompatibel ist.

b) Verwenden Sie eine feste Version der TAS für alle SUT-Versionen, um die Komplexität zu reduzieren.

c) Führen Sie keine Versionsverwaltung durch, sondern aktualisieren Sie die TAS kontinuierlich ohne Rücksicht auf ältere Versionen.

d) Löschen Sie ältere Versionen der TAS, um Speicherplatz zu sparen.

\*\*Richtige Antwort:\*\* a) Implementieren Sie ein Versionierungssystem, das alle Testmittel (einschließlich Testmodelle, Testskripte, Testadapter und Simulatoren) verwaltet und sicherstellt, dass jede Version der TAS mit der entsprechenden Version des SUT kompatibel ist.

### Fragen für den gegebenen Abschnitt

#### K2-Frage (Verstehen von Konzepten):

\*\*Frage:\*\* Welche der folgenden Aktivitäten gehört zur Erfassung der Anforderungen für eine Testautomatisierungsarchitektur (TAA)?

\*\*Antwortoptionen:\*\*

a) Auswahl des Testausführungswerkzeugs

b) Entscheidung für die manuelle oder automatisierte Testgenerierung

c) Bestimmen, welche Teststufe unterstützt werden soll

d) Auswahl der Styleguides und Richtlinien für die Definition hochwertiger Tests

\*\*Richtige Antwort:\*\* c) Bestimmen, welche Teststufe unterstützt werden soll

#### K3-Frage (Anwendung von Konzepten):

\*\*Frage:\*\* Sie sind verantwortlich für den Entwurf einer Testautomatisierungsarchitektur (TAA) und müssen die Testgenerierungsschicht festlegen. Welche Faktoren sollten Sie berücksichtigen, um eine geeignete Strategie für die Testgenerierung zu entwickeln?

\*\*Antwortoptionen:\*\*

a) Auswahl des Testausführungswerkzeugs und der Implementierungstechnologie

b) Entscheidung für die manuelle oder automatisierte Testgenerierung und Auswahl von Testgenerierungsstrategien wie Modellüberdeckung oder Szenarioüberdeckung

c) Auswahl der Notation für die Testdefinition und der Styleguides für die Definition hochwertiger Tests

d) Auswahl der Testschnittstellen zum SUT und der Werkzeuge zur Überwachung des SUT

\*\*Richtige Antwort:\*\* b) Entscheidung für die manuelle oder automatisierte Testgenerierung und Auswahl von Testgenerierungsstrategien wie Modellüberdeckung oder Szenarioüberdeckung

#### K4-Frage (Analyse und Bewertung):

\*\*Frage:\*\* Sie sind beauftragt, eine TAA für ein neues Testautomatisierungsprojekt zu entwerfen. Die Anforderungen umfassen die Unterstützung von funktionalen Tests, Konformitätstests und Interoperabilitätstests für verschiedene Softwareproduktreihen. Wie sollten Sie vorgehen, um eine robuste und skalierbare Testautomatisierungsarchitektur zu entwickeln?

\*\*Antwortoptionen:\*\*

a) Konzentrieren Sie sich auf die Auswahl eines einzigen Testausführungswerkzeugs und implementieren Sie alle Testfälle damit.

b) Entwickeln Sie eine TAA, die sowohl datengetriebene als auch schlüsselwortgetriebene Testdefinitionen unterstützt und wählen Sie die geeigneten Testgenerierungs- und Testausführungsstrategien basierend auf den spezifischen Anforderungen jeder Testart.

c) Definieren Sie die Testgenerierung ausschließlich auf Basis von szenariobasierten Ansätzen und ignorieren Sie die anderen Testgenerierungsstrategien.

d) Wählen Sie eine einzige Notation für die Testdefinition, um Konsistenz zu gewährleisten, und vermeiden Sie die Verwendung mehrerer Notationen.

\*\*Richtige Antwort:\*\* b) Entwickeln Sie eine TAA, die sowohl datengetriebene als auch schlüsselwortgetriebene Testdefinitionen unterstützt und wählen Sie die geeigneten Testgenerierungs- und Testausführungsstrategien basierend auf den spezifischen Anforderungen jeder Testart.

### Fragen für den gegebenen Abschnitt

#### K2-Frage (Verstehen von Konzepten):

\*\*Frage:\*\* Welche Vorteile bietet die Abstraktion in einer Testautomatisierungsarchitektur (TAA)?

\*\*Antwortoptionen:\*\*

a) Reduzierung der Entwicklungskosten

b) Erhöhung der Portabilität der Testartefakte und Anbieterneutralität

c) Vereinfachung der manuellen Testdurchführung

d) Erhöhung der Abhängigkeit von spezifischen Technologien

\*\*Richtige Antwort:\*\* b) Erhöhung der Portabilität der Testartefakte und Anbieterneutralität

#### K3-Frage (Anwendung von Konzepten):

\*\*Frage:\*\* Sie sind verantwortlich für die Implementierung einer TAA und müssen entscheiden, welche Schnittstellen über die gesamte Lebensdauer der TAA hinweg externalisiert, formal definiert und stabil gehalten werden sollen. Welche Schritte sollten Sie unternehmen, um diese Entscheidung zu treffen?

\*\*Antwortoptionen:\*\*

a) Bestimmen Sie, welche Testarten automatisiert werden sollen, und entwickeln Sie eine einfache TAA ohne Berücksichtigung von Schnittstellen.

b) Arbeiten Sie mit den an der Softwareentwicklung, Qualitätssicherung und Testdurchführung beteiligten Personen zusammen, um den geeigneten Abstraktionsgrad und die relevanten Schnittstellen festzulegen.

c) Implementieren Sie alle Schnittstellen der Testadaptierungs- und Testausführungsschicht ohne Rücksprache mit anderen Beteiligten.

d) Wählen Sie nur die Testausführungsschicht für die Automatisierung und ignorieren Sie die anderen Schichten.

\*\*Richtige Antwort:\*\* b) Arbeiten Sie mit den an der Softwareentwicklung, Qualitätssicherung und Testdurchführung beteiligten Personen zusammen, um den geeigneten Abstraktionsgrad und die relevanten Schnittstellen festzulegen.

#### K4-Frage (Analyse und Bewertung):

\*\*Frage:\*\* Sie sollen eine TAA entwickeln, die sowohl die Flexibilität für zukünftige Entwicklungen als auch die Wartbarkeit und Erweiterbarkeit berücksichtigt. Welche Kompromisse müssen Sie bei der Entscheidung über den Abstraktionsgrad der TAA eingehen und wie sollten Sie vorgehen, um die beste Balance zu finden?

\*\*Antwortoptionen:\*\*

a) Entscheiden Sie sich für eine minimale Abstraktion, um die Implementierung zu vereinfachen und die Lernkurve zu reduzieren, auch wenn dies die Flexibilität und Wartbarkeit einschränkt.

b) Wählen Sie einen hohen Abstraktionsgrad, was eine größere Anfangsinvestition und eine steilere Lernkurve erfordert, aber langfristig Flexibilität und Wartbarkeit verbessert.

c) Implementieren Sie eine TAA ohne Abstraktion, um sofortige Kosteneinsparungen zu erzielen, auch wenn dies die langfristige Anpassungsfähigkeit beeinträchtigt.

d) Entwickeln Sie die TAA ausschließlich auf Basis von konkreten Testfällen ohne Berücksichtigung von abstrakten Testdefinitionen oder Testmodellen.

\*\*Richtige Antwort:\*\* b) Wählen Sie einen hohen Abstraktionsgrad, was eine größere Anfangsinvestition und eine steilere Lernkurve erfordert, aber langfristig Flexibilität und Wartbarkeit verbessert.

**Kapitel 4.1: Auswahl des Testautomatisierungsansatzes und Planung von Softwareverteilung/Rollout**

Hauptaktivitäten:

Pilotversuch (Erprobung):

Ermittlung eines geeigneten Projekts: Wichtig für die Auswahl eines Projekts, das repräsentativ und geeignet ist, das TAS zu testen.

Planen des Pilotversuchs: Detaillierte Planung der Schritte und Ziele des Tests.

Durchführen des Pilotversuchs: Umsetzung des Tests im gewählten Projekt.

Evaluieren des Pilotversuchs: Analyse der Ergebnisse und Feststellung der Effektivität des TAS.

Verteilungsphase (Deployment):

Ermitteln der ersten Zielprojekte: Auswahl der ersten Projekte für die Implementierung.

Verteilung der TAS in den ausgewählten Projekten: Einsatz des TAS in diesen Projekten.

Überwachung und Evaluierung der TAS: Überprüfung der Leistung des TAS nach einer bestimmten Zeit.

Rollout im übrigen Unternehmen: Erweiterung des Einsatzes des TAS auf weitere Projekte oder das gesamte Unternehmen basierend auf den gesammelten Erfahrungen.

**Wichtige Aspekte der Werkzeugimplementierung (Pilotprojekt)**

Ziele des Pilotprojekts:

Erkenntnisse gewinnen: Testen, wie das Testautomatisierungssystem (TAS) funktioniert und welche Vorteile es bringt.

Integration in bestehende Prozesse: Prüfen, wie gut sich das TAS in die aktuellen Arbeitsabläufe und Tools einfügt und ob Änderungen nötig sind.

Automatisierungsschnittstelle entwerfen: Entwickeln einer benutzerfreundlichen Schnittstelle für die Tester.

Regeln aufstellen: Festlegen, wie das TAS benutzt wird, wie es verwaltet wird, und wie man die Tests speichert und organisiert. Dazu gehört auch, wie Dateien benannt werden und wie die Tests strukturiert sind.

Leistung messen: Bestimmen, wie man überprüfen kann, ob das TAS gut funktioniert. Das bedeutet, festlegen, wie man misst, ob es leicht zu benutzen, zu warten und zu erweitern ist.

Kosten und Nutzen bewerten: Überprüfen, ob das TAS den gewünschten Nutzen bringt und ob die Kosten dafür angemessen sind. Falls nötig, kann man die Erwartungen anpassen, wenn das TAS eingesetzt wird.

Fähigkeiten im Team: Herausfinden, welche Fähigkeiten die Teammitglieder brauchen, um das TAS zu nutzen. Prüfen, ob diese Fähigkeiten im Team vorhanden sind oder ob Schulungen notwendig sind.

**Wichtige Aspekte der Auswahl eines geeigneten Projekts für das Pilotprojekt**

Nicht-kritisches Projekt:

Warum? Wenn das Testautomatisierungssystem (TAS) anfangs Zeit braucht und es zu Verzögerungen kommt, darf das keine ernsthaften Folgen haben.

Beispiel: Ein weniger wichtiges Projekt, bei dem Verzögerungen nicht kritisch sind.

Nicht triviales Projekt:

Warum? Ein einfaches Projekt gibt nicht genug Erkenntnisse für komplexere Projekte.

Beispiel: Ein Projekt, das repräsentativ für typische Herausforderungen ist, aber nicht zu einfach.

Einbeziehung aller Beteiligten:

Warum? Damit das Pilotprojekt gut ausgewählt wird, sollten alle relevanten Personen (einschließlich Management) mitentscheiden.

Beispiel: Meetings oder Besprechungen mit dem gesamten Team und dem Management.

Gute Referenz für andere Projekte:

Warum? Das System im Pilotprojekt (SUT) sollte typisch für andere Projekte sein, um relevante Erkenntnisse zu gewinnen.

Beispiel: Das SUT sollte z. B. repräsentative grafische Benutzeroberflächen (GUI) haben, die automatisiert werden sollen.

**Wichtige Aspekte der Planung des Pilotversuchs**

Pilotprojekt wie ein reguläres Entwicklungsprojekt behandeln:

Plan machen: Detaillierter Plan für das Pilotprojekt erstellen.

Budget und Ressourcen reservieren: Genügend Geld und Arbeitsmittel bereitstellen.

Fortschritt informieren: Regelmäßig über den Fortschritt berichten.

Meilensteine definieren: Wichtige Etappen im Projekt festlegen.

Zeit für die Bereitstellung sicherstellen:

Warum? Die Personen, die am TAS arbeiten, brauchen genug Zeit, auch wenn sie noch an anderen Projekten arbeiten.

Beispiel: Zeitmanagement und Prioritäten festlegen.

Unterstützung des Managements:

Warum? Besonders wichtig bei geteilten Ressourcen, damit das Projekt nicht behindert wird.

Beispiel: Regelmäßige Updates und Meetings mit dem Management.

Einbeziehung der Entwickler:

Warum? Wenn das TAS im eigenen Haus entwickelt wird, müssen die Entwickler in den Bereitstellungsprozess eingebunden sein.

Beispiel: Entwickler arbeiten eng mit dem Bereitstellungsteam zusammen.

**Wichtige Aspekte der Bereitstellung des Testautomatisierungssystems (TAS)**

Schrittweiser Rollout:

Warum? Damit die Einführung des TAS kontrolliert und in kleinen Schritten erfolgt, um Engpässe zu erkennen und zu beseitigen.

Wie? Einführung in Wellen, damit neue Benutzer nach und nach unterstützt werden.

Anpassung und Optimierung von Prozessen:

Warum? Unterschiedliche Benutzer und Prozesse müssen auf das TAS abgestimmt werden, um reibungslos zu funktionieren.

Wie? Kleine Anpassungen an Prozessen oder dem TAS selbst vornehmen.

Schulungen und Coaching/Mentoring:

Warum? Neue Benutzer müssen lernen, wie man das TAS benutzt.

Wie? Schulungen und Workshops anbieten, bevor die Benutzer mit dem TAS arbeiten.

Definieren der Nutzungsrichtlinien:

Warum? Um Fragen und Probleme zu vermeiden.

Wie? Erstellung von Richtlinien, Checklisten und FAQs für die Benutzer.

Erfassung von Informationen zur Nutzung:

Warum? Um zu wissen, wie das TAS tatsächlich verwendet wird und welche Funktionen genutzt werden.

Wie? Automatisierte Tools zur Datenerfassung implementieren.

Überwachung der Nutzung, des Nutzwertes und der Kosten:

Warum? Um festzustellen, ob das TAS effektiv genutzt wird und um die Wirtschaftlichkeit zu überprüfen.

Wie? Regelmäßige Überwachung und Auswertung der Nutzung.

Unterstützung für Test- und Entwicklungsteam:

Warum? Damit das Team bei Problemen schnell Hilfe bekommt.

Wie? Bereitstellung eines Support-Teams.

Sammeln von Erkenntnissen:

Warum? Um aus den Erfahrungen der Teams zu lernen und das TAS zu verbessern.

Wie? Regelmäßige Evaluierungs- und Bewertungssitzungen durchführen.

Ermitteln und Umsetzen von Verbesserungen:

Warum? Um das TAS basierend auf Feedback und Überwachung zu optimieren.

Wie? Verbesserungsmaßnahmen identifizieren und umsetzen, und das den Beteiligten klar mitteilen.

**4.1.3 Verteilung der TAS innerhalb des Softwarelebenszyklus**

Die Verteilung einer TAS hängt in großem Maß von der Entwicklungsphase des Softwareprojekts ab, das von

der TAS getestet wird.

Eine neue TAS oder eine neue Version von ihr wird in der Regel entweder zu Projektbeginn oder bei Erreichen

eines Meilensteins (z. B. Code-Freeze oder Ende eines Sprints) bereitgestellt. Grund dafür ist, dass die Verteilungsaktivitäten mit allen damit einhergehenden Tests und Modifikationen Zeit und Aufwand erfordern. Zudem ist das eine gute Möglichkeit, das Risiko zu begrenzen, dass die TAS nicht funktioniert und Unterbrechungen im Testautomatisierungsprozess bewirkt. Wenn es bei der TAS kritische Probleme gibt, die behoben werden müssen oder wenn eine Komponente in der Umgebung, in der sie läuft, ausgetauscht werden muss, dann erfolgt die Verteilung jedoch unabhängig von der Entwicklungsphase des SUT.

**Wichtige Aspekte der Strategien für die Bewertung und Begrenzung von Risiken**

Typische technische Probleme:

Zu starke Abstraktion: Kann es schwierig machen, die tatsächlichen Abläufe zu verstehen.

Datengetriebene Probleme: Datentabellen können zu groß oder komplex werden.

Abhängigkeit von bestimmten Betriebssystem-Komponenten: Diese sind nicht immer in allen Zielumgebungen verfügbar.

Typische Risiken eines Softwareverteilungsprojekts:

Personalmangel: Schwierigkeit, die richtigen Leute für die Wartung der Codebasis zu finden.

Fehlerhafte SUT-Arbeitsergebnisse: Können den Betrieb der TAS stören.

Verzögerungen bei der Einführung und Aktualisierung der TAS: Können die Automatisierung behindern.

Nicht erkennbaren Objekte: Die TAS kann spezielle Objekte möglicherweise nicht erfassen.

Potentielle Fehlerpunkte des TAS-Projekts:

Migration in eine andere Umgebung: Probleme beim Wechsel der Umgebung.

Verteilung in der Zielumgebung: Schwierigkeiten beim Verteilen des TAS.

Neue Lieferung aus der Entwicklung: Neue Versionen oder Updates können Probleme verursachen.

Strategien zur Risikominimierung:

Versionskontrolle: TAS sollte wie jede andere Software unter Versionskontrolle stehen.

Dokumentation: Alle Funktionen und Prozesse der TAS sollten dokumentiert sein.

Klares Verteilungsverfahren: Ein einfach zu befolgendes und dokumentiertes Verfahren zur Verteilung der TAS.

Zwei prägnante Fälle der Verteilung einer TAS:

Erstverteilung:

Definieren und Erstellen der Infrastruktur, in der die TAS ausgeführt wird.

Verfahren für Wartung der TAS und ihrer Testsuite erstellen.

Risiken: Gesamtausführungszeit der Testsuite, Installations- und Konfigurationsprobleme.

Wartungsverteilung:

TAS existiert bereits und muss gewartet werden.

Durchführung einer Bewertung der Änderungen, Testen der neuen Funktionen, Prüfen der Testsuite auf Kompatibilität.

Risiken: Notwendige Änderungen an der Testsuite, Anpassungen von Treibern und Schnittstellen, Infrastrukturänderungen, mögliche neue Defekte oder Performanzprobleme.

Maßnahmen zur Begrenzung von Risiken bei Updates:

Änderungen an der Testsuite: Vor der Verteilung testen.

Anpassung von Treibern und Schnittstellen: Änderungen vor der Verteilung testen.

Infrastruktur-Bewertung: Änderungen an der Infrastruktur vor der Verteilung testen.

Risiko-Nutzen-Analyse: Bewertung, ob die Aktualisierung der TAS sinnvoll ist, ggf. auf nächste Version warten.

**Wichtige Aspekte der Wartung der Testautomatisierung**

Grundlegende Anforderungen:

Modular, skalierbar, verständlich, zuverlässig und testbar: Diese Eigenschaften machen eine gute Testautomatisierungslösung aus.

Weiterentwicklung: Testautomatisierungslösungen müssen sich weiterentwickeln, um zuverlässig und effizient zu bleiben.

Wartung: Ein wichtiger Aspekt der Entwicklung und Anpassung an neue Anforderungen und Umgebungen.

**Arten der Wartung**:

Präventive Wartung:

Was? Änderungen, um die TAS für mehr Testarten und neue Systeme vorzubereiten.

Warum? Um zukünftige Anforderungen besser zu erfüllen.

Beispiel: Unterstützung für neue Schnittstellen oder neue Versionen des zu testenden Systems (SUT).

Korrektive Wartung:

Was? Behebung von Fehlern in der TAS.

Warum? Um die Zuverlässigkeit und Funktionsfähigkeit der TAS zu gewährleisten.

Wie? Regelmäßige Wartungstests durchführen.

Verbessernde Wartung:

Was? Optimierung der TAS, um nicht-funktionale Fehler zu beheben.

Warum? Um die Leistung, Benutzbarkeit, Robustheit oder Zuverlässigkeit der TAS zu verbessern.

Beispiel: Verbesserung der Benutzerfreundlichkeit oder der Verarbeitungsgeschwindigkeit.

Adaptive Wartung:

Was? Anpassung der TAS an neue Softwaresysteme oder gesetzliche Anforderungen.

Warum? Um sicherzustellen, dass die TAS mit neuen Umgebungen und Vorschriften kompatibel bleibt.

Beispiel: Unterstützung neuer Betriebssysteme oder Einhaltung neuer gesetzlicher Vorschriften.

**4.3.2 Umfang und Ansatz**

Wartung ist ein Prozess, der alle Schichten und Komponenten einer TAS betreffen kann. Ihr Umfang hängt von

folgenden Faktoren ab:

Größe und Komplexität der TAS: Je größer und komplexer die TAS, desto umfassender die Wartung.

Größe der Änderung: Große Änderungen erfordern umfassendere Wartungsarbeiten.

Risiko der Änderung: Änderungen mit hohem Risiko müssen gründlicher getestet und überwacht werden.

Wartungsprozess:

Auswirkungsanalyse:

Warum? Um zu verstehen, wie sich Änderungen auf die TAS auswirken.

Wie? Schrittweise Änderungen vornehmen und nach jedem Schritt Tests durchführen.

Bewährte Vorgehensweisen:

Dokumentation: Verteilungsverfahren und Nutzung der TAS müssen klar und dokumentiert sein.

Modularität: TAS sollte modular aufgebaut sein, um Teile leicht ersetzen zu können.

Isolierte Ausführung: TAS sollte getrennt von der Entwicklungsumgebung ausgeführt werden, um Unabhängigkeit zu gewährleisten.

Konfigurationsmanagement: TAS, Testumgebung, Testsuite und Testmittel müssen unter Konfigurationsmanagement stehen.

Wartung von Fremdkomponenten und -bibliotheken:

Dokumentation: Alle Fremdkomponenten müssen dokumentiert und unter Konfigurationsmanagement stehen.

Plan für Änderungen: Es muss einen Plan geben, wie auf Änderungen oder Fehler in Fremdkomponenten reagiert wird.

Lizenzdokumentation: Informationen über die Lizenzbedingungen für Fremdkomponenten müssen vorliegen.

Aktualisierungen: Fremdkomponenten und -bibliotheken sollten regelmäßig auf Updates und neue Versionen überprüft und aktualisiert werden.

Benennungsstandards und Konventionen:

Warum? Um die Testsuite und TAS einfach lesbar, verständlich und wartbar zu machen.

Standards: Benennungsstandards für Variablen, Dateien, Testszenarien, Schlüsselwörter und Parameter.

Dokumentation: Standards und Konventionen müssen zu Beginn eines Projekts festgelegt und dokumentiert werden.

Dokumentation:

Notwendigkeit: Gute und aktuelle Dokumentation ist entscheidend für die Wartung und Nutzung der TAS.

Praxis: Schreiben der Dokumentation sollte Teil des Entwicklungsprozesses sein.

Schulungsmaterial:

Basis: Dokumentation der TAS dient als Basis für Schulungsmaterial.

Inhalt: Funktionale Spezifikationen, Entwurf und Architektur der TAS, Verteilung und Wartung, Nutzung (Benutzerhandbuch), praktische Beispiele und Tipps.

Wartung: Regelmäßige Überprüfung und Aktualisierung des Schulungsmaterials durch benannte Ausbilder.

**Kapitel 5-Auswahl von TAS-Metriken**

Was versteht man unter Metriken?

Metriken sind Zahlen oder Messgrößen, die uns helfen, Dinge zu bewerten und zu verstehen, wie gut etwas funktioniert. Stell dir vor, es sind wie Noten in der Schule, aber für ein Computersystem.

Auswahl von TAS-Metriken

Warum brauchen wir TAS-Metriken?

Überwachung der Strategie: Metriken helfen uns zu sehen, ob unsere Pläne für die Testautomatisierung gut funktionieren.

Überwachung der Wirksamkeit: Sie zeigen, wie gut unser Testautomatisierungssystem (TAS) arbeitet und ob es seine Aufgaben erfüllt.

**Unterschied zu anderen Metriken:**

TAS-Metriken: Diese messen, wie gut das System arbeitet, das die Tests automatisiert.

SUT-Metriken: Diese messen, wie gut das eigentliche System funktioniert, das getestet wird (SUT = Software Under Test).

**Wer kümmert sich um TAS-Metriken?**

Testmanager: Wählt die wichtigen Metriken aus, die wir für das Projekt brauchen.

Testautomatisierungsmanager (TAM) und Testautomatisierungsentwickler (TAE): Nutzen diese Metriken, um den Fortschritt zu verfolgen und zu sehen, wie Änderungen am System wirken.

**Arten von TAS-Metriken:**

Externe Metriken:

Was ist das? Messen, wie die Testautomatisierung andere Arbeiten beeinflusst.

Beispiel: Wenn die Automatisierung hilft, Zeit bei manuellen Tests zu sparen.

Interne Metriken:

Was ist das? Messen, wie gut das Testautomatisierungssystem selbst funktioniert.

Beispiel: Wie viele Tests das System automatisch erfolgreich durchführt oder wie lange es dauert, diese Tests auszuführen.

Folgende TAS-Metriken werden in der Regel ermittelt:

• externe TAS-Metriken

• Nutzwert der Automatisierung

• Aufwand für die Erstellung automatisierter Tests

• Aufwand für die Analyse der bei automatisierten Tests ermittelten Abweichungen

• Aufwand für die Wartung automatisierter Tests

• Verhältnis von fehlgeschlagenen Tests zu Fehlern im SUT

• Ausführungszeit automatisierter Tests

• Anzahl der automatisierten Testfälle

• Anzahl der positiven und negativen Ergebnisse

• Anzahl der falsch-negativen und falsch-positiven Ergebnisse

• Überdeckungsgrad des Quellcodes

• interne TAS-Metriken

• Skriptmetriken

• Fehlerdichte des Automatisierungscodes

• Geschwindigkeit und Effizienz der TAS-Komponenten

Diese Metriken werden nachstehend beschrieben

**1-Nutzwert der Automatisierung**

Warum den Nutzwert messen?

Sichtbarkeit: Zeigt, welchen Nutzen das TAS bringt.

Dokumentation: Auch Personen außerhalb des Testprojekts können den Wert der Automatisierung verstehen.

Was wird gemessen?

Ziele der TAS: Die Nutzenmessung hängt davon ab, was das Ziel der TAS ist, z.B. Zeitersparnis, mehr Tests oder weniger Fehler.

Mögliche Kennzahlen:

Zeitersparnis bei manuellen Tests: Wie viele Stunden durch Automatisierung gespart wurden.

Schnellere Regressionstests: Verkürzte Zeit für wiederkehrende Tests.

Mehr Testzyklen: Anzahl zusätzlicher Testdurchläufe.

Mehr Tests in Prozent: Wie viel Prozent mehr Tests durchgeführt wurden.

Automatisierungsanteil: Anteil der automatisierten Testfälle im Vergleich zur Gesamtanzahl der Testfälle.

Erhöhung des Überdeckungsgrades: Mehr Anforderungen, Funktionen oder Code-Strukturen abgedeckt.

Frühzeitige Fehlerfindung: Anzahl der Fehler, die frühzeitig gefunden wurden, und die dadurch eingesparten Kosten.

Fehler, die manuell nicht gefunden wurden: Fehler, die nur durch die Automatisierung entdeckt wurden.

Indirekter Nutzen:

Manueller Aufwand: Zeit, die durch Automatisierung eingespart wird, kann für andere manuelle Tests genutzt werden, wie z.B. explorative Tests.

Zusätzliche Fehler: Fehler, die durch diese zusätzlichen manuellen Tests gefunden werden, sind ein indirekter Nutzen der Automatisierung.

**2-Aufwand für die Erstellung automatisierter Tests**

Wesentliche Punkte:

Kosten für die Automatisierung:

Höher als manuelle Tests: Die Kosten für die Automatisierung eines Tests sind oft höher als die manuelle Durchführung desselben Tests.

Warum? Diese Kosten können ein Argument gegen die Ausweitung der Testautomatisierung sein.

Faktoren, die die Kosten beeinflussen:

Test selbst: Komplexe oder umfangreiche Tests kosten mehr Zeit und Geld zu automatisieren.

Automatisierungsansatz: Unterschiedliche Methoden der Automatisierung können verschiedene Kosten verursachen.

Vertrautheit mit Werkzeugen: Wenn die Tester die Werkzeuge gut kennen, kann die Automatisierung schneller und kostengünstiger erfolgen.

Umgebung und Fähigkeiten: Die Umgebung, in der die Tests laufen, und die Fähigkeiten der Testautomatisierungsentwickler (TAE) spielen eine Rolle.

Zeitaufwand:

Komplexe Tests dauern länger: Umfangreichere oder kompliziertere Tests benötigen mehr Zeit zur Automatisierung.

Durchschnittliche Entwicklungszeit: Es ist sinnvoll, die durchschnittliche Zeit für die Entwicklung eines automatisierten Tests zu berechnen.

Kostenberechnung:

Durchschnittliche Kosten pro Testgruppe: Man kann die Kosten auch für spezifische Gruppen von Tests berechnen (z.B. Tests für eine bestimmte Funktion).

EMTE (Equivalent Manual Test Effort): Die Entwicklungskosten können als ein Vielfaches des manuellen Testaufwands dargestellt werden. Zum Beispiel könnte die Automatisierung eines Tests den doppelten Aufwand der manuellen Durchführung erfordern (2x EMTE).

**3-Aufwand für die Analyse von SUT-Fehlern**

Wichtige Punkte:

Komplexität der Fehleranalyse:

Automatisierte Tests vs. Manuelle Tests: Bei automatisierten Tests ist die Analyse von Fehlern oft schwieriger, da der Tester die Abläufe, die zu einem Fehler führen, nicht direkt kennt.

Lösung: Das Problem kann auf der Entwurfsebene und der Berichterstattungsebene reduziert werden.

Kennzahlen für Fehleranalyseaufwand:

Mittelwert pro fehlgeschlagenen Testfall: Durchschnittlicher Aufwand für die Analyse eines fehlgeschlagenen Tests.

EMTE (Equivalent Manual Test Effort): Darstellung des Analyseaufwands als Vielfaches des manuellen Testaufwands, besonders bei variierenden Testkomplexitäten.

Rolle der Protokollierung:

Wichtigkeit: Die Protokollierung von SUT (System Under Test) und TAS (Testautomatisierungssystem) ist entscheidend für eine effiziente Fehleranalyse.

Anforderungen an die Protokollierung:

Synchronisierte Protokollierung: SUT- und TAS-Protokolle müssen gleichzeitig erfasst werden.

Erwartetes und tatsächliches Verhalten: TAS muss sowohl das erwartete als auch das tatsächliche Verhalten protokollieren.

Ausgeführte Aktionen: TAS muss die durchgeführten Aktionen dokumentieren.

Protokollierung im SUT:

Aktionen protokollieren: SUT muss alle Aktionen aufzeichnen, unabhängig davon, ob sie manuell oder automatisiert ausgelöst wurden.

Interne Fehler: Alle internen Fehler müssen dokumentiert werden.

Absturz-Speicherabbilder und Stack Traces: Diese müssen verfügbar sein, um tiefergehende Fehleranalysen durchzuführen.

**4-Aufwand für die Wartung automatisierter Tests**

Warum ist Wartungsaufwand wichtig?

Hoher Aufwand: Es kann viel Arbeit kosten, automatisierte Tests an neue Versionen des Systems (SUT) anzupassen.

Probleme vermeiden: Wenn der Aufwand zu hoch ist, können Automatisierungsprojekte scheitern.

Überwachung des Wartungsaufwands:

Warum? Um sicherzustellen, dass der Aufwand nicht außer Kontrolle gerät.

Wie? Regelmäßig prüfen, wie viel Arbeit für die Wartung nötig ist.

Wie misst man den Wartungsaufwand?

Gesamtzahl der Tests: Wie viele automatisierte Tests müssen bei jeder neuen Version des SUT gewartet werden?

Durchschnitt pro Test: Wie viel Arbeit kostet es im Durchschnitt, einen Test zu aktualisieren?

Vergleich mit manuellem Testaufwand (EMTE): Wie viel mehr Aufwand ist es im Vergleich zu einem manuellen Test?

Wartungsbedarf-Metrik:

Anzahl der Tests mit Wartungsbedarf: Wie viele Tests brauchen Wartung?

Warum sind diese Informationen wichtig?

Entscheidungen treffen: Diese Informationen helfen zu entscheiden, ob neue Funktionen hinzugefügt oder Fehler behoben werden sollen.

Änderungen berücksichtigen: Bei Änderungen am SUT sollte der Aufwand für die Wartung der Tests berücksichtigt werden.

**5-Verhältnis von fehlgeschlagenen Tests zu Fehlern im SUT**

Wichtige Punkte:

Häufiges Problem bei automatisierten Tests:

Mehrere fehlgeschlagene Tests, ein Fehler: Viele automatisierte Tests können wegen des gleichen Fehlers in der Software fehlschlagen.

Warum ein Problem? Das führt zu Ressourcenverschwendung, da die Analyse jedes einzelnen fehlgeschlagenen Tests aufwendig sein kann.

Zweck der Tests:

Fehler entdecken: Tests sollen Fehler in der Software aufdecken.

Effizienz: Es ist ineffizient, wenn viele Tests denselben Fehler finden.

Messung und Analyse:

Anzahl fehlgeschlagener Tests: Zählen, wie viele automatisierte Tests wegen eines bestimmten Fehlers fehlschlagen.

Problemfälle identifizieren: Diese Messung hilft zu erkennen, wo das Problem liegt.

Lösung:

Testentwurf: Verbesserung des Designs der automatisierten Tests.

Testauswahl: Bessere Auswahl der Tests, die ausgeführt werden sollen, um Redundanz zu vermeiden.

**6-Ausführungszeit automatisierter Tests**

Wichtige Punkte:

Was ist die Ausführungszeit?

Definition: Die Zeit, die benötigt wird, um alle automatisierten Tests auszuführen.

Anfangs nebensächlich:

Warum? Bei wenigen Tests ist die Ausführungszeit oft nicht wichtig.

Wichtigkeit mit der Zeit:

Warum? Mit zunehmender Anzahl automatisierter Tests kann die Ausführungszeit eine bedeutende Rolle spielen.

Problem: Längere Ausführungszeiten können Verzögerungen verursachen und die Effizienz beeinträchtigen.

Warum ist die Ausführungszeit wichtig?

Effizienz: Längere Testzeiten können die Entwicklungszeit verlängern und die Bereitstellung neuer Softwareversionen verzögern.

Ressourcen: Längere Ausführungszeiten können mehr Rechenressourcen erfordern, was zu höheren Kosten führt.

**7-Anzahl der automatisierten Testfälle**

Wichtige Punkte:

Was bedeutet die Anzahl der automatisierten Testfälle?

Definition: Die Gesamtzahl der Tests, die automatisiert wurden.

Fortschrittsanzeige:

Warum wichtig? Zeigt den Fortschritt des Testautomatisierungsprojekts an.

Einschränkungen der Metrik:

Nicht aussagekräftig allein: Die Anzahl der automatisierten Testfälle allein sagt nichts über die Qualität oder Effektivität der Tests aus.

Testüberdeckung: Diese Metrik gibt nicht an, ob mehr Bereiche der Software abgedeckt werden.

Warum sollte man die Anzahl der automatisierten Testfälle messen?

Fortschritt: Um zu sehen, wie viel des Testens automatisiert wurde.

Planung: Hilft bei der Planung weiterer Automatisierungsaufgaben.

**8-Anzahl der positiven und negativen Ergebnisse**

Wichtige Punkte:

Was bedeutet die Metrik?

Definition: Sie zeigt, wie viele automatisierte Tests erfolgreich bestanden wurden (positive Ergebnisse) und wie viele fehlgeschlagen sind (negative Ergebnisse).

Bedeutung der Metrik:

Erfolgsmessung: Gibt an, ob die Tests die erwarteten Ergebnisse erreicht haben.

Fehleranalyse: Hilft festzustellen, wie viele Tests nicht erfolgreich waren.

Analyse der Fehlschläge:

Warum wichtig? Um die Ursache der Fehlschläge zu verstehen.

Mögliche Ursachen:

Fehler im SUT: Fehler in der Software, die getestet wird.

Externe Probleme: Probleme mit der Testumgebung oder der Testautomatisierung selbst.

Warum sollte man die Anzahl der positiven und negativen Ergebnisse messen?

Qualitätssicherung: Um die Qualität und Zuverlässigkeit der Software und der Tests zu überprüfen.

Problembehebung: Um schnell auf Probleme reagieren und sie beheben zu können.

**9-Anzahl der falsch-negativen und falsch-positiven Ergebnisse**

Wichtige Punkte:

Was bedeutet die Metrik?

Falsch-positive Ergebnisse: Tests schlagen fehl, obwohl das SUT korrekt funktioniert. Das Problem liegt in der TAS oder dem Testfall, nicht im SUT.

Falsch-negative Ergebnisse: Tests bestehen, obwohl das SUT Fehler hat. Der Fehler im SUT wird nicht erkannt.

Warum sind falsch-positive Ergebnisse problematisch?

Frustration und Aufwand: Fehlalarme kosten Zeit und sind frustrierend, da die Analyse keinen Fehler im SUT findet.

Vertrauensverlust: Zu viele Fehlalarme können das Vertrauen in die Testautomatisierung mindern.

Warum sind falsch-negative Ergebnisse gefährlich?

Übersehene Fehler: Ein Fehler im SUT bleibt unentdeckt, was zu größeren Problemen führen kann.

Ursachen: Falsche Verifizierung des Ergebnisses, ungültiges Testorakel oder fehlerhafte Erwartung des Testfalls.

Ursachen für Fehlalarme:

Fehler im Testquellcode: Fehler im Automatisierungscode selbst.

Instabiles SUT: Unvorhersehbares Verhalten des SUT, z.B. durch Überschreitung der Ausführungszeit.

Test Hooks: Intrusive Testmethoden können ebenfalls Fehlalarme auslösen.

Maßnahmen zur Minimierung:

Fehlerdichte im Automatisierungscode reduzieren: Sorgfältige Erstellung und Wartung des Testcodes.

Stabilität des SUT verbessern: Sicherstellen, dass das SUT konsistent und vorhersehbar reagiert.

Verifizierung verbessern: Verifizierungsmethoden und Testorakel korrekt anwenden.

**10-Codeüberdeckung (Code Coverage)**

Wichtige Punkte:

Was bedeutet Codeüberdeckung?

Definition: Der Anteil des Codes im System (SUT), der durch Testfälle abgedeckt wird.

Bedeutung der Codeüberdeckung:

Information: Gibt Aufschluss darüber, wie viel vom Code durch die Tests überprüft wird.

Beispiel: Codeüberdeckung der Regressionstestsuite zeigt, wie gut die Tests wiederkehrende Fehler abdecken.

Prozentwert der Codeüberdeckung:

Kein absoluter Wert: Es gibt keinen festen Prozentsatz, der eine ausreichende Codeüberdeckung definiert.

100% Überdeckung: In der Praxis nur bei sehr einfachen Softwareanwendungen möglich.

Höhere Überdeckung ist besser: Eine höhere Überdeckung reduziert das Risiko, unentdeckte Fehler in der gelieferten Software zu haben.

Änderungen in der Codeüberdeckung:

Sinkende Codeüberdeckung: Kann darauf hinweisen, dass neue Funktionen hinzugefügt wurden, ohne die Testsuite entsprechend zu erweitern.

Warum ist Codeüberdeckung wichtig?

Risikoreduktion: Höhere Codeüberdeckung bedeutet, dass mehr Teile des Codes getestet wurden, was das Risiko von Fehlern reduziert.

Aktivitätsanzeige: Änderungen in der Codeüberdeckung können auf Aktivitäten im SUT hinweisen, wie z.B. hinzugefügte Funktionen.

**11-Skriptmetriken**

Wichtige Punkte:

Was sind Skriptmetriken?

Definition: Metriken, die verwendet werden, um die Entwicklung und Qualität von Automatisierungsskripten zu überwachen.

Ähnlichkeit zu Quellcode-Metriken:

Vergleichbar mit SUT-Metriken: Viele Skriptmetriken sind ähnlich wie die Metriken, die für den Quellcode des Systems unter Test (SUT) verwendet werden.

Wichtige Skriptmetriken:

Lines of Code (LOC): Anzahl der Codezeilen in einem Skript. Zu viele Zeilen können auf zu große oder komplexe Skripte hinweisen.

Zyklomatische Komplexität: Misst die Anzahl der unabhängigen Pfade durch den Code. Eine hohe Komplexität kann auf schwer verständliche und wartbare Skripte hinweisen.

Verhältnis von Kommentaren zu Anweisungen: Zeigt, wie gut ein Skript dokumentiert ist. Ein höheres Verhältnis deutet auf eine bessere Dokumentation hin.

Konformitätsverstöße: Anzahl der Verstöße gegen Skripterstellungsstandards. Viele Verstöße deuten darauf hin, dass die Standards nicht eingehalten werden.

Warum sind Skriptmetriken wichtig?

Qualitätskontrolle: Helfen dabei, die Qualität und Wartbarkeit der Automatisierungsskripte zu sichern.

Optimierung: Identifizieren von Skripten, die zu groß oder zu komplex sind und daher überarbeitet werden sollten.

Dokumentation: Sicherstellen, dass Skripte gut kommentiert und verständlich sind.

Standardkonformität: Überwachen, ob die Erstellungsstandards für Skripte eingehalten werden

**12-Fehlerdichte des Automatisierungscodes**

Wichtige Punkte:

Was ist Fehlerdichte?

Definition: Anzahl der Fehler pro Codezeile im Automatisierungscode.

Automatisierungscode und SUT-Code:

Gleiche Bedeutung: Automatisierungscode ist genauso wichtig wie der Code des Systems unter Test (SUT).

Warum? Beide enthalten Fehler und müssen sorgfältig behandelt werden.

Gute Kodierungspraktiken:

Bedeutung: Gute Praktiken und Standards müssen auch im Automatisierungscode angewendet werden.

Ziel: Sicherstellen, dass der Automatisierungscode von hoher Qualität ist.

Überwachung der Fehlerdichte:

Metriken: Die Fehlerdichte kann mit Metriken überwacht werden.

Werkzeuge: Konfigurationsmanagementsysteme helfen dabei, die notwendigen Daten zu erfassen.

Warum ist die Fehlerdichte wichtig?

Qualitätssicherung: Eine niedrige Fehlerdichte bedeutet, dass der Code weniger Fehler enthält und zuverlässiger ist.

Wartbarkeit: Weniger Fehler bedeuten, dass der Code leichter zu warten und zu erweitern ist.

**13-Geschwindigkeit und Effizienz der TAS-Komponenten**

Wichtige Punkte:

Unterschiedliche Ausführungszeiten:

Problemindikator: Wenn die gleichen Testschritte in derselben Umgebung unterschiedlich lange dauern, kann das ein Problem im System unter Test (SUT) anzeigen.

Variabilität: Unterschiede in der Ausführungszeit können auf Schwankungen im System hinweisen, die nicht akzeptabel sind und sich bei höherer Last verschlimmern könnten.

Stabilität der TAS:

Wichtigkeit: Die Testautomatisierungslösung (TAS) muss stabil sein und die Leistungsfähigkeit des SUT nicht beeinträchtigen.

Leistungsanforderungen: Wenn das SUT hohe Leistungsanforderungen hat, muss die TAS so gestaltet sein, dass sie diese Anforderungen unterstützt.

Warum ist Geschwindigkeit und Effizienz wichtig?

Problemidentifikation: Unterschiedliche Ausführungszeiten helfen, Probleme im SUT zu erkennen.

Systemstabilität: Eine stabile TAS trägt zur Zuverlässigkeit und Leistungsfähigkeit des SUT bei.

**14-Trend-Metriken**

Wichtige Punkte:

Was sind Trend-Metriken?

Definition: Trend-Metriken betrachten, wie sich Messwerte im Laufe der Zeit verändern, anstatt nur einen einzelnen Messwert zu einem bestimmten Zeitpunkt zu betrachten.

Warum sind Trends aufschlussreicher?

Langfristige Einsichten: Veränderungen und Entwicklungen über die Zeit bieten tiefere Einblicke als einzelne Momentaufnahmen.

Beispiel: Wenn die durchschnittlichen Wartungskosten pro automatisiertem Test in den letzten Versionen des SUT gestiegen sind, zeigt das ein potenzielles Problem.

Nutzen von Trend-Metriken:

Problemidentifikation: Trends können auf Probleme hinweisen, die sofortige Maßnahmen erfordern.

Prozessverbesserung: Erkennen von Mustern hilft, die zugrunde liegenden Ursachen zu finden und den Prozess zu verbessern.

Kosten für das Messen:

Effizienz: Die Kosten für das Sammeln von Daten sollten niedrig gehalten werden.

Automatisierung: Automatisierung der Datenerfassung und Berichterstattung hilft, die Kosten zu minimieren und die Genauigkeit zu erhöhen.

Warum sind Trend-Metriken wichtig?

Früherkennung: Trends helfen, Probleme frühzeitig zu erkennen und rechtzeitig Maßnahmen zu ergreifen.

Besseres Verständnis: Sie bieten ein umfassenderes Bild der Systemleistung und -entwicklung über die Zeit.

**5.2 Implementierung der Messwerterfassung**

Implementierung der Messwerterfassung

Wichtige Punkte:

Erweiterung der Testmittel:

Informationen aufzeichnen: Automatisierte Testmittel können erweitert werden, um Nutzungsinformationen zu erfassen, wie z.B. Start- und Endzeit der Testausführung.

Abstraktion und Struktur: Diese Erweiterungen können auf alle Tests angewendet werden, um einheitliche Messungen zu ermöglichen.

Unterstützung durch Skriptsprachen:

Messung und Berichterstattung: Viele Testwerkzeuge unterstützen das Messen und Erstellen von Berichten über Mechanismen, die Informationen vor, während und nach der Testausführung aufzeichnen.

Analysefunktion:

Ergebnisse vergleichen: Berichte müssen in der Lage sein, Ergebnisse früherer Testläufe zu berücksichtigen und Trends zu erkennen.

Erfolgsquote: Analyse von Veränderungen in der Erfolgsquote der Tests.

Automatisierung der Testverifizierung:

Vergleich mit erwarteten Ergebnissen: Testwerkzeuge vergleichen Testergebnisse mit vordefinierten erwarteten Ergebnissen und protokollieren den Teststatus (z.B. bestanden, fehlgeschlagen).

Fehlschläge analysieren: Bei Fehlschlägen sind detaillierte Informationen über die Ursache notwendig (z.B. Screenshots).

Integration mit Drittanbieterwerkzeugen:

Kompatibilität: Informationen aus automatisierten Testfällen sollten in Formaten bereitgestellt werden, die von Drittanbieterwerkzeugen genutzt werden können (z.B. Excel, Word, HTML).

Rückverfolgbarkeit: Diese Integration hilft bei der Rückverfolgung und Berichterstattung.

Visualisierung der Ergebnisse:

Grafische Darstellung: Testergebnisse sollten grafisch aufbereitet werden, z.B. durch Dashboards, Diagramme und Ampelsysteme.

Managementfreundlich: Grafische Zusammenfassungen helfen dem Management, Testergebnisse auf einen Blick zu verstehen.

Warum ist die Messwerterfassung wichtig?

Effizienz und Genauigkeit: Automatisierte Messwerterfassung und Berichterstattung erhöhen die Effizienz und Genauigkeit der Testprozesse.

Trendanalyse: Trends in den Testergebnissen zu erkennen, hilft bei der Verbesserung der Teststrategien.

Transparenz: Detaillierte und grafisch aufbereitete Berichte bieten Transparenz und erleichtern Entscheidungsfindungen.

**5.3 Protokollierung von TAS und SUT**

Wichtige Punkte:

Warum ist Protokollierung wichtig?

Fehleranalyse: Testprotokolle sind essenziell, um potenzielle Probleme zu analysieren und zu beheben.

TAS-Protokollierung:

Testfall-Informationen: Protokollierung des aktuellen Testfalls, einschließlich Start- und Endzeit.

Teststatus: Status der Testausführung („bestanden“, „fehlgeschlagen“, „TAS-Fehler“).

Wichtige Schritte und Zeitangaben: Protokollierung auf hoher Ebene.

Dynamische Informationen: Erfassung von Informationen wie Speicherlecks.

Ausführungszyklen: Zählung der Zyklen bei Zuverlässigkeits- und Stresstests.

Zufallsparameter: Protokollierung zufälliger Elemente im Testfall.

Aktionen des Testfalls: Protokollierung aller Aktionen zur Reproduzierbarkeit.

Screenshots: Speicherung visueller Dokumentationen für die Fehleranalyse.

Fehlerinformationen: Sicherstellung, dass alle nötigen Informationen für die Fehleranalyse gespeichert werden.

Farbcodierung: Verwendung von Farben zur Unterscheidung verschiedener Arten von Informationen.

SUT-Protokollierung:

Problembehebung: Alle Informationen, die zur Analyse von Problemen benötigt werden, protokollieren (Datums- und Uhrzeitstempel, Fehlermeldungen, Quellstandort).

Benutzerinteraktionen: Protokollierung von Interaktionen über die UI und Netzwerkschnittstellen.

Konfigurationsdaten: Protokollierung der Konfigurationsdaten bei Inbetriebnahme des Systems.

Durchsuchbarkeit und Synchronisierung:

Einfach durchsuchbar: Protokolldaten müssen leicht durchsuchbar sein.

Synchronisierung: Verwendung von Zeitstempeln zur Synchronisierung verschiedener Protokolle erleichtert die Zuordnung der Ereignisse bei Fehlern.

Warum ist Protokollierung wichtig?

Fehlerbehebung: Hilft bei der Analyse und Behebung von Fehlern.

Reproduzierbarkeit: Ermöglicht die Reproduzierbarkeit von Problemen, was für die Lösung wichtig ist.

Transparenz: Bietet eine klare Übersicht über den Testverlauf und -status.

**5.4 Erstellung von Berichten zur Testautomatisierung**

Warum sind Berichte wichtig?

Überblick: Testprotokolle bieten detaillierte Informationen, aber Berichte geben einen klaren Überblick über die gesamten Testergebnisse.

Notwendigkeit: Ein präziser Bericht muss nach jeder Ausführung der Testsuite erstellt und veröffentlicht werden.

Inhalt der Berichte:

Zusammenfassung: Ein Überblick über die Testergebnisse, das getestete System und die Testumgebung.

Fehlgeschlagene Tests: Welche Tests fehlgeschlagen sind und warum.

Testverlauf und Verantwortliche: Informationen über den Verlauf der Testausführung und die verantwortliche Person für den Test.

Fehlerdiagnose: Unterstützung bei der Diagnose von Fehlern in den TAF-Komponenten.

Verantwortlichkeiten:

Analyse: Die verantwortliche Person untersucht die Ursache für das Fehlschlagen, meldet Probleme und verfolgt deren Behebung.

Veröffentlichung der Berichte:

Zugänglichkeit: Berichte sollten für alle Interessierten zugänglich sein.

Verbreitungswege: Upload auf eine Website, Versand über Mailingliste oder Export in ein Testmanagementwerkzeug.

Benachrichtigungen: Bereitstellung der Berichte per E-Mail oder über einen Bezugsmechanismus, um die Wahrscheinlichkeit zu erhöhen, dass die Berichte gelesen und analysiert werden.

Berichtshistorie:

Statistische Analyse: Eine Berichtshistorie kann helfen, problematische Teile des SUT zu identifizieren und häufige Regressionen zu erfassen.

Warum sind Berichte zur Testautomatisierung wichtig?

Transparenz: Sie bieten eine klare und verständliche Übersicht über die Testergebnisse.

Problembehebung: Erleichtern die Identifikation und Behebung von Fehlern.

Nachvollziehbarkeit: Verfolgen den Verlauf und die Verantwortlichkeit der Tests.

**6. Überführung des manuellen Testens in eine automatisierte**

**Umgebung –**

Kriterien für die Automatisierung

Wichtige Punkte:

Bewertung der manuellen Tests:

Manuelle Tests analysieren: Bevor auf automatisierte Tests umgestellt wird, muss der aktuelle Zustand der manuellen Tests bewertet werden.

Geeignet für Automatisierung? Die vorhandene Struktur eines manuellen Tests kann geeignet sein oder muss eventuell neu geschrieben werden.

Umstellung auf Automatisierung:

Neuschreiben von Tests: In einigen Fällen kann ein vollständiges Neuschreiben der Tests notwendig sein.

Wiederverwendung von Komponenten: Eingabewerte, erwartete Ergebnisse und Navigationspfade aus manuellen Tests können für die Automatisierung verwendet werden.

Manuelle Teststrategie:

Berücksichtigung der Automatisierung: Die manuelle Teststrategie sollte so gestaltet sein, dass sie eine spätere Automatisierung erleichtert.

Nicht alle Tests müssen automatisiert werden:

Manuelle und automatisierte Tests: Einige Tests können oder müssen manuell bleiben, insbesondere in der ersten Iteration.

Besondere Testarten: Tests wie Zuverlässigkeitstests, Stresstests und Performanztests sind oft effektiver, wenn sie automatisiert werden.

Automatisierung von Tests ohne UI:

Schnittstellentests: Automatisierte Tests ermöglichen das Testen über Schnittstellen, was manuell unpraktisch ist.

Frühzeitiges Testen: Automatisierung ermöglicht das frühzeitige Entdecken von Fehlern, bevor manuelles Testen möglich ist.

Eignungskriterien für die Automatisierung:

Verwendungshäufigkeit: Wie oft wird der Test verwendet?

Komplexität der Automatisierung: Wie schwierig ist es, den Test zu automatisieren?

Werkzeugunterstützung: Gibt es geeignete Werkzeuge zur Automatisierung?

Reifegrad des Testprozesses: Wie gut sind die Testprozesse entwickelt?

Lebenszyklusphase: Passt die Automatisierung in die aktuelle Phase des Softwareprodukt-Lebenszyklus?

Nachhaltigkeit: Ist die automatisierte Umgebung langfristig haltbar?

Steuerbarkeit: Wie gut kann das SUT durch die Automatisierung gesteuert werden?

Warum sind diese Kriterien wichtig?

Effizienz: Sicherstellen, dass die Automatisierung sinnvoll und durchführbar ist.

Qualität: Gewährleisten, dass die Tests effektiv und zuverlässig sind.

Kosten-Nutzen: Abwägen, ob der Nutzen der Automatisierung die Kosten rechtfertigt.

**1-Verwendungshäufigkeit**

Wichtige Punkte:

Bedeutung der Verwendungshäufigkeit:

Wie oft wird ein Test ausgeführt? Die Häufigkeit der Testausführung ist ein entscheidender Faktor für die Automatisierung.

Regelmäßige Tests: Tests, die regelmäßig im Rahmen von Major- oder Minor-Release-Zyklen durchgeführt werden, sind bessere Kandidaten für die Automatisierung.

Nutzwert automatisierter Tests:

Häufigkeit und Nutzen: Je häufiger ein Test ausgeführt wird, desto größer ist der Nutzen einer Automatisierung.

Regelmäßige Updates: Automatisierte Tests, die regelmäßig durchgeführt werden, bieten eine höhere Investitionsrendite (ROI).

Regressionstests:

Funktionale Tests: Automatisierte funktionale Tests können in späteren Releases für Regressionstests verwendet werden.

ROI und Risikominderung: Automatisierte Regressionstests bieten eine hohe Investitionsrendite und reduzieren das Risiko für die bestehende Codebasis.

Gelegentliche Tests:

Selten ausgeführte Tests: Wenn ein Testskript nur einmal im Jahr ausgeführt wird und sich das SUT ändert, ist die Automatisierung möglicherweise nicht praktikabel.

Effizienz: In solchen Fällen ist es effizienter, die Tests manuell durchzuführen, da der Aufwand für die jährliche Anpassung des automatisierten Tests den Nutzen übersteigen könnte.

Warum ist die Verwendungshäufigkeit wichtig?

Effizienz: Regelmäßige Tests profitieren mehr von der Automatisierung.

Kosten-Nutzen-Verhältnis: Häufig ausgeführte Tests bieten eine höhere Investitionsrendite und lohnen sich mehr, automatisiert zu werden.

**2-Komplexität der Automatisierung**

Wichtige Punkte:

Vorteile der Automatisierung bei komplexen Systemen:

Reduzierung der manuellen Belastung: Automatisierung kann helfen, komplexe und wiederholte Schritte zu vermeiden, die für manuelle Tester langwierig und fehleranfällig sind.

Herausforderungen bei der Automatisierung:

Schwierigkeit und Kosten: Manche Testskripte sind schwer oder sehr kostspielig zu automatisieren.

Faktoren, die die Automatisierungskomplexität beeinflussen:

Inkompatibles SUT: Ein System unter Test (SUT), das nicht mit bestehenden automatisierten Testlösungen kompatibel ist.

Umfangreicher Programmcode: Notwendigkeit, umfangreichen Code zu schreiben und API-Aufrufe für die Automatisierung zu entwickeln.

Systemvielfalt: Vielfalt von Systemen, die während der Testausführung angesprochen werden müssen.

Externe Schnittstellen: Interaktion mit externen Schnittstellen oder proprietären Systemen.

Benutzbarkeitstests: Bestimmte Aspekte von Benutzbarkeitstests sind schwer zu automatisieren.

Testzeit: Zeitaufwand für das Testen der Automatisierungsskripte.

Warum ist die Komplexität der Automatisierung wichtig?

Effizienz und Kosten: Automatisierung kann die Effizienz steigern, aber bei hoher Komplexität auch hohe Kosten verursachen.

Entscheidungsfindung: Verständnis der Komplexität hilft bei der Entscheidung, ob eine Automatisierung sinnvoll und durchführbar ist.

**3-Werkzeugunterstützung**

Wichtige Punkte:

Vielfalt der Plattformen und Werkzeuge:

Plattformen: Es gibt viele verschiedene Plattformen für die Entwicklung von Anwendungen.

Werkzeuge: Tester müssen wissen, welche Werkzeuge für die jeweilige Plattform verfügbar sind und wie gut diese Plattform unterstützt wird.

Arten von Testwerkzeugen:

Kommerzielle Werkzeuge: Diese bieten in der Regel bezahlten Support und haben oft ein Netzwerk von Experten, die bei der Implementierung helfen können.

Open-Source-Werkzeuge: Unterstützung erfolgt meist über Online-Foren, wo man Informationen finden und Fragen stellen kann.

Selbst entwickelte Werkzeuge: Unterstützung wird vom internen Personal des Unternehmens bereitgestellt.

Ressourcen zur Unterstützung:

Unternehmen: Jedes Unternehmen hat unterschiedliche Bedürfnisse in Bezug auf die Unterstützung von Testwerkzeugen.

Support: Die Art des Supports (kommerziell, open-source, intern) ist entscheidend für den Erfolg der Implementierung.

Bedeutung der Werkzeugunterstützung:

Kompatibilität: Es ist wichtig zu wissen, wie gut die Testwerkzeuge mit dem System unter Test (SUT) kompatibel sind.

Risiko des Scheiterns: Unkenntnis über die Unterstützung und Kompatibilität kann zum Scheitern des Testautomatisierungsprojekts führen.

Kritische Tests: Selbst wenn die meisten Tests automatisiert werden können, ist es wichtig sicherzustellen, dass dies auch für die kritischen Tests gilt.

Warum ist Werkzeugunterstützung wichtig?

Erfolgreiche Implementierung: Gute Unterstützung und Kompatibilität der Werkzeuge sind entscheidend für den Erfolg eines Testautomatisierungsprojekts.

Fehlervermeidung: Unzureichende Unterstützung kann zu Problemen führen, die den gesamten Testprozess beeinträchtigen.

4-Reifegrad des Testprozesses

Wichtige Punkte:

Struktur und Regelkonformität:

Strukturierter Prozess: Ein Testprozess muss klar strukturiert sein, um automatisiert werden zu können.

Regelkonform: Der Prozess muss den festgelegten Regeln und Standards entsprechen.

Wiederholbarkeit:

Konstante Ergebnisse: Ein wiederholbarer Prozess liefert bei jeder Durchführung konsistente Ergebnisse, was für die Automatisierung entscheidend ist.

Automatisierungsprozess:

Kompletter Entwicklungsprozess: Die Automatisierung bringt einen vollständigen Entwicklungsprozess in den bestehenden Testprozess ein.

Management des Automatisierungscodes: Erfordert die Verwaltung von Automatisierungscode und zugehörigen Komponenten.

Warum ist der Reifegrad des Testprozesses wichtig?

Effizienz: Ein reifer, strukturierter und regelkonformer Testprozess ist leichter zu automatisieren.

Qualität: Ein wiederholbarer Testprozess stellt sicher, dass automatisierte Tests zuverlässige und konsistente Ergebnisse liefern.

Verwaltung: Der Automatisierungscode und seine Komponenten müssen effektiv gemanagt werden, um eine erfolgreiche Automatisierung zu gewährleisten.

5-Eignung der Automatisierung für die Phase des Softwareprodukt-Lebenszyklus

Wichtige Punkte:

Produktlebenszyklus des SUT:

Dauer: Ein System unter Test (SUT) kann einen Produktlebenszyklus haben, der sich über Jahre oder Jahrzehnte erstreckt.

Veränderungen: In den frühen Entwicklungsphasen ändern sich Systeme schnell, um Fehler zu beheben und Nutzeranforderungen zu erfüllen.

Frühe Entwicklungsphasen:

Dynamische Änderungen: Bildschirm-Layouts und Bedienelemente werden ständig optimiert.

Herausforderung: Kontinuierliche Nachbesserung der Automatisierung ist ineffizient und ineffektiv.

Vergleich: Wie das Wechseln eines Reifens bei einem fahrenden Auto.

Stabilisierte Systeme:

Stabilität: Der beste Zeitpunkt für die Implementierung automatisierter Tests ist, wenn das System stabil ist und einen Kern an wichtigen Funktionen enthält.

Effizienz: Automatisierung ist sinnvoller, wenn das System weniger Änderungen erfährt.

Ende des Produktlebenszyklus:

Nicht ratsam: Automatisierung lohnt sich nicht für Systeme, die kurz vor dem Ende ihres Lebenszyklus stehen oder bald außer Betrieb genommen werden.

Umbau und Wiederverwendung: Bei einer Umgestaltung des Systems unter Beibehaltung der Funktionalität ist eine automatisierte Testumgebung nützlich. Testdaten können wiederverwendet werden, und die automatisierte Umgebung kann an die neue Architektur angepasst werden.

Warum ist die Phase des Softwareprodukt-Lebenszyklus wichtig für die Automatisierung?

Effizienz: Automatisierung ist effizienter und effektiver in stabilen Phasen des Lebenszyklus.

Ressourcen: Ressourcen sollten nicht auf Automatisierung verschwendet werden, wenn das System sich schnell ändert oder kurz vor dem Ende seines Lebenszyklus steht.

**6-Nachhaltigkeit der Umgebung**

Wichtige Punkte:

Flexibilität und Anpassungsfähigkeit:

Wichtig: Eine Testumgebung für die Automatisierung muss flexibel sein und sich an Änderungen im System unter Test (SUT) anpassen können.

Anforderungen an eine nachhaltige Testumgebung:

Schnelle Diagnose und Behebung: Probleme müssen durch Automatisierung schnell diagnostiziert und behoben werden können.

Einfache Wartung: Automatisierungskomponenten müssen leicht zu warten sein.

Erweiterbarkeit: Es muss möglich sein, neue Funktionen und Unterstützung in die automatisierte Umgebung einzubinden.

Integraler Bestandteil des Designs:

Gesamtentwurf: Diese Attribute sind wesentliche Teile des Gesamtentwurfs und der Implementierung der generischen Testautomatisierungsarchitektur (gTAA).

Warum ist die Nachhaltigkeit der Umgebung wichtig?

Langfristige Effizienz: Eine nachhaltige Testumgebung bleibt auch bei zukünftigen Änderungen im SUT effizient.

Kosteneffizienz: Wartungs- und Erweiterungsfähigkeiten reduzieren langfristig die Kosten und den Aufwand.

Zuverlässigkeit: Schnellere Problemlösungen und weniger Ausfälle erhöhen die Zuverlässigkeit der Testumgebung.

**7-Steuerbarkeit des SUT (Vorbedingungen, Setup und Stabilität)**

Wichtige Punkte:

Steuerungs- und Visibilitätsmerkmale:

Ermittlung: Der Testautomatisierungsentwickler (TAE) muss die Steuerungs- und Sichtbarkeitsmerkmale des Systems unter Test (SUT) ermitteln.

Warum? Diese Merkmale helfen bei der Entwicklung effektiver automatisierter Tests.

Interaktionen mit dem UI:

Einschränkungen: Automatisierungslösungen, die nur auf Interaktionen mit der Benutzeroberfläche (UI) basieren, sind weniger gut wartbar.

Alternative: Eine umfassendere Steuerung des SUT bietet mehr Stabilität und Wartbarkeit.

Bezug zur Testbarkeit und Automatisierung:

Auslegung auf Testbarkeit: Das SUT sollte so gestaltet sein, dass es leicht testbar und automatisierbar ist (siehe Abschnitt 2.3).

Beispiele für Steuerbarkeit: Vorbedingungen, Setups und die Stabilität des Systems sollten gut kontrollierbar sein.

Warum ist die Steuerbarkeit des SUT wichtig?

Effektive Tests: Gut steuerbare Systeme ermöglichen präzisere und stabilere automatisierte Tests.

Wartbarkeit: Tests, die nicht nur auf UI-Interaktionen basieren, sind leichter zu warten und anzupassen.

Stabilität: Eine stabile und steuerbare Umgebung reduziert Fehler und verbessert die Zuverlässigkeit der Tests.

**8-Technische Planung zur Unterstützung der ROI-Analyse**

Wichtige Punkte:

Nutzen der Testautomatisierung:

Analyse des Nutzens: Bevor man den Aufwand für die Entwicklung automatisierter Tests auf sich nimmt, muss der potenzielle Gesamtnutzen analysiert und bewertet werden.

Investitionsrendite (ROI): Es ist wichtig, den Nutzen zu kalkulieren und die Kosten zu berechnen, um die ROI zu bestimmen.

Maßnahmen zur Umsetzung:

Plan definieren: Nach der Analyse müssen Maßnahmen zur Umsetzung des Automatisierungsplans festgelegt werden.

Kosten kalkulieren: Die damit verbundenen Kosten müssen ermittelt werden.

Vorbereitung der Umstellung:

Verfügbarkeit von Werkzeugen: Sicherstellen, dass geeignete Werkzeuge für die Testautomatisierung in der Testumgebung vorhanden sind.

Genauigkeit der Testdaten und Testfälle: Testdaten und Testfälle müssen präzise und verlässlich sein.

Umfang der Arbeiten: Den Umfang der Arbeiten zur Testautomatisierung abschätzen.

Schulung und Rollenverteilung:

Schulung des Testteams: Das Testteam muss auf den Paradigmenwechsel zur Automatisierung vorbereitet und geschult werden.

Rollen und Zuständigkeiten: Klar definierte Rollen und Zuständigkeiten sind wichtig für den Erfolg.

Kooperation und paralleles Arbeiten:

Kooperation: Enge Zusammenarbeit zwischen den Entwicklungsteams und den Testautomatisierungsteams ist notwendig.

Paralleles Arbeiten: Die Entwicklung und die Automatisierung müssen parallel ablaufen können.

Berichterstellung:

Berichte: Erstellen von Berichten zur Testautomatisierung, um den Fortschritt und die Ergebnisse zu dokumentieren.

Warum ist technische Planung wichtig?

Effizienz: Sorgt für eine strukturierte und gut geplante Umstellung auf die Testautomatisierung.

Kosten-Nutzen-Verhältnis: Hilft, die Investitionsrendite zu maximieren und unnötige Ausgaben zu vermeiden.

Erfolg: Eine gründliche Planung erhöht die Erfolgsaussichten des Automatisierungsprojekts

**9-Verfügbarkeit von Werkzeugen in der Testumgebung für die Testautomatisierung**

Wichtige Punkte:

Installation und Prüfung:

Installation: Ausgewählte Testwerkzeuge müssen installiert werden.

Funktionsprüfung: Die Werkzeuge müssen in der Testlaborumgebung auf ihre Funktionsfähigkeit geprüft werden.

Wichtige Schritte:

Herunterladen von Service Packs und Updates: Aktuelle Versionen und notwendige Updates der Werkzeuge herunterladen.

Auswahl der richtigen Installationskonfiguration: Die richtige Konfiguration der Werkzeuge und eventuell benötigter Add-Ins auswählen.

Unterstützung des SUTs: Sicherstellen, dass die Werkzeuge das System unter Test (SUT) unterstützen.

Gewährleistung der ordnungsgemäßen Funktion:

Testlaborumgebung: Die Werkzeuge müssen in der Testlaborumgebung ordnungsgemäß funktionieren.

Umgebung für die Automatisierungsentwicklung: Die Testautomatisierungslösung (TAS) muss auch in der Entwicklungsumgebung stabil und funktionsfähig sein.

Warum ist die Verfügbarkeit von Werkzeugen wichtig?

Effektive Testautomatisierung: Die richtigen Werkzeuge sind entscheidend für eine effektive und effiziente Testautomatisierung.

Stabilität und Zuverlässigkeit: Sicherstellung, dass die Testwerkzeuge stabil und zuverlässig in der Testumgebung arbeiten.

Unterstützung des SUTs: Die Werkzeuge müssen das SUT vollständig unterstützen, um aussagekräftige Testergebnisse zu liefern.

**10-Genauigkeit der Testdaten und Testfälle**

Wichtige Punkte:

Bedeutung der Genauigkeit:

Warum? Die Genauigkeit und Vollständigkeit manueller Testdaten und Testfälle ist entscheidend für die Vorhersagbarkeit der Ergebnisse in der automatisierten Umgebung.

Anforderungen an automatisierte Tests:

Explizite Daten: Automatisierte Tests benötigen klar definierte und präzise Daten.

Eingabe: Daten, die in das System eingegeben werden.

Navigation: Schritte zur Navigation innerhalb des Systems.

Synchronisierung: Daten zur Synchronisierung der Tests.

Validierung: Daten, die zur Überprüfung der Testergebnisse verwendet werden.

Warum ist die Genauigkeit wichtig?

Vorhersagbare Ergebnisse: Nur genaue und vollständige Testdaten und Testfälle liefern konsistente und vorhersagbare Ergebnisse in automatisierten Tests.

Effektivität: Genauigkeit stellt sicher, dass die Tests die gewünschten Bereiche abdecken und valide Ergebnisse liefern.

Zuverlässigkeit: Verlässliche Testdaten und -fälle sind die Grundlage für aussagekräftige und nützliche Testergebnisse.

**11-Umfang der Arbeiten zur Testautomatisierung**

Wichtige Punkte:

Begrenzter Anfangsumfang:

Warum? Mit einem begrenzten Umfang beginnen, um frühe Erfolge zu erzielen und technisches Feedback zu sammeln.

Pilotprojekt: Ein Pilotprojekt kann sich auf einen repräsentativen Bereich des Systems konzentrieren.

Erkenntnisse aus dem Pilotprojekt:

Schätzung des Aufwands: Hilft bei der genaueren Schätzung des zukünftigen Zeitaufwands und der Planung.

Technische Ressourcen: Identifiziert Bereiche, die spezialisierte technische Ressourcen erfordern.

Auswahl der Testfälle:

Mit Umsicht wählen: Testfälle mit geringem Automatisierungsaufwand, aber hohem Mehrwert auswählen.

Beispiele: Automatische Regressions- oder Smoke-Tests, die häufig ausgeführt werden.

Vorteile der Automatisierung:

Zuverlässigkeitstests: Einfach zu implementieren und liefern schnell Ergebnisse, indem sie Probleme aufdecken, die manuell schwer zu finden sind.

Strategische Prioritäten:

Größter Nutzen: Tests priorisieren, die anfänglich den größten Nutzen bieten.

Technisch komplexe Tests: Anfangs ausklammern, da sie hohen Aufwand erfordern und weniger vorzeigbare Ergebnisse liefern.

Warum ist der Umfang der Arbeiten wichtig?

Früher Erfolg: Ein begrenzter Anfangsumfang zeigt schnelle Erfolge und sichert die Unterstützung durch das Management.

Effizienz: Sorgfältige Auswahl der Testfälle maximiert den Nutzen bei minimalem Aufwand.

Strategische Planung: Hilft, die Automatisierung sinnvoll zu erweitern und langfristig erfolgreich zu machen.

**12-Schulung des Testteams im Hinblick auf den Paradigmenwechsel**

Wichtige Punkte:

Vielfalt im Testteam:

Unterschiedliche Arten von Testern: Einige sind Fachexperten aus der Endbenutzer-Community oder Wirtschaftsanalytiker, andere haben ausgeprägte technische Kompetenzen.

Breites Kompetenzspektrum: Für effektives Testen ist ein vielfältiges Team mit unterschiedlichen Kompetenzen wichtig.

Spezialisierung durch Automatisierung:

Spezialisierte Rollen: Mit der Umstellung auf Automatisierung werden die Rollen innerhalb des Testteams spezialisierter.

Änderung der Teamzusammensetzung: Das Team muss entsprechend der neuen Anforderungen umgestellt werden.

Frühzeitige Aufklärung:

Bedeutung: Das Team muss frühzeitig über die geplante Umstellung informiert werden.

Ziel: Befürchtungen zerstreuen, insbesondere Ängste bezüglich Rollenveränderungen oder der Gefahr, überflüssig zu werden.

Positive Einstellung und konstruktiver Wandel:

Richtige Steuerung: Wenn der Prozess gut gesteuert wird, werden die Teammitglieder die Umstellung positiv aufnehmen.

Konstruktiver Beitrag: Alle Mitglieder des Teams können sich konstruktiv in den organisatorischen und technischen Wandel einbringen.

Warum ist die Schulung des Testteams wichtig?

Effektivität: Ein gut geschultes Team kann die Vorteile der Automatisierung besser nutzen und effektiver arbeiten.

Motivation: Frühzeitige Aufklärung und Schulung helfen, Ängste zu zerstreuen und die Motivation zu steigern.

Erfolgreiche Umstellung: Ein positiv eingestelltes und gut informiertes Team trägt maßgeblich zum Erfolg der Automatisierung bei.

**13-Rollen und Zuständigkeiten**

Wichtige Punkte:

Beteiligung aller Teammitglieder:

Beitrag aller: Jeder im Team sollte zur Testautomatisierung beitragen können.

Unterschiedliche Rollen: Obwohl alle beitragen können, hat nicht jeder dieselbe Rolle.

Technische Aufgaben:

Entwicklung und Wartung: Die Entwicklung, Implementierung und Wartung einer automatisierten Testumgebung sollten technisch versierten Mitarbeitern überlassen werden.

Programmierkenntnisse: Diese Aufgaben erfordern gute Programmierkenntnisse und technischen Hintergrund.

Nutzung der Umgebung:

Für alle nutzbar: Die automatisierte Testumgebung muss von technisch versierten Personen und Laien genutzt werden können.

Maximierung des Wertes:

Fachwissen: Personen mit Fachwissen im Testen sind notwendig, um geeignete Testskripte zu entwickeln.

Testdaten: Erstellen der entsprechenden Testdaten ist entscheidend.

Überprüfung und Effizienz:

Geschäftsexperten: Prüfen Berichte, um den Funktionsumfang der Anwendung zu bestätigen.

Technische Experten: Stellen sicher, dass die automatisierte Umgebung richtig und effizient funktioniert.

Entwickler: Entwickler mit Interesse am Testen können das Automatisierungsframework oder die Testbibliotheken konzipieren und warten.

Wartbarkeit und Implementierung:

Wartbare Software: Erfahrung in der Softwareentwicklung ist Voraussetzung für das Konzipieren wartbarer Software.

Tester: Die Implementierung der Testfälle sollte den Testern überlassen werden.

Warum sind klare Rollen und Zuständigkeiten wichtig?

Effizienz: Klare Rollenverteilungen sorgen für eine effiziente Nutzung der Ressourcen und Fähigkeiten.

Qualität: Technische Experten stellen sicher, dass die Automatisierungsumgebung stabil und leistungsfähig ist.

Wartbarkeit: Entwickler mit Erfahrung sorgen dafür, dass die Software wartbar bleibt, was für die langfristige Nutzung entscheidend ist.

**14-Kooperation zwischen den Entwicklungsteams und den Testautomatisierungsteams**

Wichtige Punkte:

Einbindung beider Teams:

Zusammenarbeit: Eine erfolgreiche Testautomatisierung erfordert die enge Zusammenarbeit von Softwareentwicklungsteam und Testteam.

Rollen: Entwickler und Tester müssen gemeinsam arbeiten, um technische Informationen und Unterstützung bereitzustellen.

Unterstützung und technische Informationen:

Entwickler: Stellen Support und technische Informationen zu ihren Methoden und Werkzeugen bereit.

Tester: Nutzen diese Informationen, um effektive automatisierte Tests zu entwickeln.

Bedenken hinsichtlich der Testbarkeit:

Testbarkeit: TAEs (Testautomatisierungsentwickler) können Bedenken zur Testbarkeit von Systementwürfen und Entwicklercode äußern.

Standards: Probleme entstehen, wenn Standards nicht eingehalten werden oder wenn ausgefallene, selbst entwickelte oder neue Bibliotheken/Objekte verwendet werden.

Kompatibilität von Werkzeugen:

Fremdanbieter-Komponenten: Entwickler könnten GUI-Steuerungen von Fremdanbietern verwenden, die möglicherweise nicht mit dem gewählten Automatisierungswerkzeug kompatibel sind.

Projektmanagement und Rollenverständnis:

Klares Verständnis: Das Projektmanagementteam muss ein klares Verständnis der benötigten Rollen und Zuständigkeiten für eine erfolgreiche Automatisierung haben.

Warum ist die Kooperation wichtig?

Effizienz: Enge Zusammenarbeit verbessert die Effizienz und Effektivität der Testautomatisierung.

Qualität: Gemeinsames Arbeiten stellt sicher, dass alle technischen Anforderungen und Standards erfüllt werden.

Problemlösung: Durch Kooperation können potenzielle Probleme frühzeitig erkannt und behoben werden.

**15-Paralleles Arbeiten**

Wichtige Punkte:

Zusätzliches Team für Automatisierung:

Paralleler Prozess: Viele Unternehmen bilden ein weiteres Team, um den Prozess der Umstellung von manuellen auf automatisierte Tests parallel zu beginnen.

Integration: Die neuen, automatisierten Skripte werden in die bestehenden Tests eingebunden und ersetzen die manuellen Skripte.

Verifizierung der Automatisierung:

Gleicher Test: Vor der Ersetzung sollte überprüft werden, dass das automatisierte Skript denselben Test wie das manuelle Skript durchführt.

Effizienz: Sicherstellen, dass die automatisierten Skripte effizient und korrekt arbeiten.

Bewertung und Umstrukturierung:

Manuelle Skripte bewerten: Vor der Umstellung wird oft eine Bewertung der manuellen Skripte durchgeführt.

Umstrukturierung: Diese Bewertung kann ergeben, dass die bestehenden manuellen Testskripte umstrukturiert werden müssen, um die automatisierte Variante effizienter und wirkungsvoller zu gestalten.

Warum ist paralleles Arbeiten wichtig?

Kontinuität: Es ermöglicht die kontinuierliche Durchführung von Tests, während die Automatisierung eingeführt wird.

Qualität: Sicherstellen, dass die automatisierten Skripte korrekt arbeiten und die manuellen Tests effektiv ersetzen.

Effizienz: Umstrukturierung der Testskripte kann die Effizienz und Wirkung der automatisierten Tests verbessern.

**16-Berichterstattung über die Automatisierung**

Wichtige Punkte:

Automatische Berichte:

Arten von Berichten: Die Testautomatisierungslösung (TAS) kann verschiedene Berichte automatisch erzeugen, z.B.:

Bestanden-/Fehlgeschlagen-Status einzelner Skripte oder Schritte innerhalb eines Skripts.

Statistiken zur Testausführung.

Gesamtperformanz der TAS.

Einblick in den Betrieb der TAS:

Ordnungsgemäßer Betrieb: Es ist wichtig, sicherzustellen, dass die TAS korrekt funktioniert.

Präzision und Vollständigkeit: Gemeldete Ergebnisse müssen präzise und vollständig sein.

Verifizierung der TAS:

Kapitel 7: Das Kapitel 7 des Lehrplans behandelt die Verifizierung der TAS und stellt sicher, dass die Automatisierung ordnungsgemäß funktioniert.

Warum ist die Berichterstattung wichtig?

Überblick: Automatische Berichte bieten einen klaren Überblick über den Status und die Leistung der Tests.

Fehlererkennung: Berichte helfen, Fehler schnell zu identifizieren und zu beheben.

Transparenz: Einblick in den ordnungsgemäßen Betrieb der TAS gewährleistet präzise und vollständige Testergebnisse.

**6.2 Erforderliche Schritte zur Automatisierung von Regressionstests**

Wichtige Punkte:

Häufigkeit der Testausführung:

Warum wichtig? Tests, die häufig im Rahmen von Regressionstests ausgeführt werden, sind die besten Kandidaten für eine Automatisierung.

Vorteile: Diese Tests sind bereits entwickelt, führen bekannte Funktionen des Systems unter Test (SUT) aus und profitieren von der Automatisierung durch eine deutlich geringere Ausführungszeit.

Testausführungszeit:

Wichtiger Parameter: Die Zeit, die benötigt wird, um einen Test oder eine komplette Testsuite auszuführen, ist entscheidend für die Bewertung des Nutzens der Automatisierung.

Ansatz: Beginnen Sie mit der Automatisierung zeitaufwändiger Tests. Das ermöglicht eine schnellere und effizientere Testausführung und erhöht die Anzahl der Ausführungszyklen automatisierter Regressionstests.

Vorteile: Häufigeres Feedback zur SUT-Qualität und geringeres Verteilungsrisiko.

Funktionsüberschneidung:

Ermittlung und Verringerung: Es ist bewährte Praxis, funktionale Überschneidungen zwischen Testfällen zu ermitteln und in den automatisierten Tests zu verringern.

Neue Struktur: Automatisierte Tests nutzen oft wiederverwendbare Komponenten und gemeinsam genutzte Daten-Repositories, was zu einer neuen Struktur führen kann.

Zerlegen und Zusammenführen: Bestehende manuelle Tests können in kleinere automatisierte Tests zerlegt oder mehrere manuelle Tests zu einem größeren automatisierten Test zusammengeführt werden.

Bewertung: Manuelle Tests sollten einzeln und als Gruppe bewertet werden, um eine effektive Umsetzungsstrategie zu entwickeln.

Gemeinsame Datennutzung:

Datenzentralisierung: Speichern Sie gemeinsam genutzte Daten an einem zentralen Ort.

Vermeidung von Duplikationen: Reduziert Fehler und vereinfacht die Verwaltung.

Gegenseitige Abhängigkeit von Tests:

Abhängigkeiten: Berücksichtigen Sie Abhängigkeiten zwischen Tests, wie z.B. die Verwendung von dynamisch erstellten Werten.

Datenweitergabe: Erfassen und Wiederverwenden von Werten in nachfolgenden Tests.

Vorbedingungen für Tests:

Initialisierung: Automatisieren Sie die Initialisierungsschritte, um die Vorbedingungen für Tests sicherzustellen.

Zuverlässigkeit: Sicherstellen, dass Tests nur ausgeführt werden, wenn alle Vorbedingungen erfüllt sind.

Erforderliche Schritte zur Automatisierung von Regressionstests

Wichtige Punkte:

SUT-Überdeckung:

Breite und Tiefe: Tests sollten so gestaltet sein, dass sie das System unter Test (SUT) möglichst breit und tief abdecken.

Codeüberdeckungs-Werkzeuge: Diese Werkzeuge überwachen die Ausführung automatisierter Tests und quantifizieren deren Wirksamkeit.

Zusätzliche Tests: Automatisierte Regressionstests können im Laufe der Zeit zu einer höheren Überdeckung führen.

Ausführbare Tests:

Vor der Automatisierung: Überprüfen, ob manuelle Tests richtig funktionieren, bevor sie in automatisierte Tests umgewandelt werden.

Fehlerbehebung: Sicherstellen, dass alle Fehler im manuellen Test oder SUT behoben sind, um einen funktionierenden automatisierten Test zu gewährleisten.

Vermeidung von Ressourcenverschwendung: Nicht funktionierende manuelle Tests sollten nicht automatisiert werden, da dies Ressourcen verschwendet.

Große Regressionstestsuites:

Parallele Ausführung: Große Testsuites können parallel ausgeführt werden, wenn mehrere SUTs verfügbar sind.

Ressourcenbeschränkungen: Bei begrenzten und/oder teuren SUTs ist parallele Ausführung möglicherweise nicht machbar.

Teilweise Ausführung: Eine Option ist, nur Teile der Testsuite auszuführen und die Tests über einen längeren Zeitraum zu verteilen.

Risikoanalyse: Entscheidung, welche Teile der Regressionstestsuite ausgeführt werden sollen, basierend auf einer Risikoanalyse (z.B. welche Teile des SUT wurden kürzlich geändert).

**Faktoren bei der Automatisierung des Testens neuer Funktionen**

Wichtige Punkte:

Frühe Automatisierung:

Neue Funktionen: Es ist oft einfacher, Testfälle für neue Funktionen zu automatisieren, bevor ihre Implementierung abgeschlossen ist.

Zusammenarbeit: Testentwickler können Entwicklern und Architekten erläutern, was für eine effektive Testautomatisierung benötigt wird.

Entwicklung neuer Tests:

Neue Anforderungen: Wenn ein SUT um neue Funktionen erweitert wird, müssen neue Tests für diese Funktionen und Anforderungen entwickelt werden.

Feedback und Analyse: Der TAE muss Feedback von Testentwicklern einholen und prüfen, ob die aktuelle TAS den neuen Anforderungen genügt.

Änderungen an der TAS:

Dokumentation: Änderungen an der TAS müssen vollständig dokumentiert werden, um bestehende Funktionen nicht zu beeinträchtigen.

Prüfung: Änderungen oder Ergänzungen müssen mit vorhandenen Testmitteln geprüft werden.

Neue Objektklassen und Testmittel:

Aktualisierungen: Neue Objektklassen können Aktualisierungen oder Ergänzungen der Testmittel erfordern.

Kompatibilität: Die Kompatibilität mit bestehenden Testwerkzeugen muss überprüft und alternative Lösungen müssen ermittelt werden.

Schlüsselwortgetriebene Ansätze:

Schlüsselwörter: Möglicherweise müssen neue Schlüsselwörter entwickelt oder bestehende Schlüsselwörter modifiziert werden, um neue Funktionen zu unterstützen.

Evaluierung neuer Testwerkzeuge:

Unterstützung neuer Umgebungen: Neue Testwerkzeuge könnten erforderlich sein, um neue Umgebungen zu unterstützen.

Beispiel: Ein neues Testwerkzeug könnte benötigt werden, wenn das bestehende Werkzeug nur HTML unterstützt.

Auswirkungen auf bestehende Tests:

Überprüfung: Vor Änderungen sollten bestehende automatisierte Tests am neuen SUT ausgeführt werden, um Änderungen zu ermitteln.

Abhängigkeiten: Ermittlung und Zuordnung wechselseitiger Abhängigkeiten mit anderen Tests.

Aktualisierung bestehender Anforderungen:

Projektplan: Der Aufwand für die Aktualisierung von Testfällen muss Teil des Projektplans sein.

Rückverfolgbarkeit: Anforderungen müssen zu Testfällen zurückverfolgbar sein, um notwendige Aktualisierungen zu identifizieren.

Testbarkeit neuer Funktionen:

Entwurfsphase: Testentwickler sollten sicherstellen, dass neue Funktionen testbar sind, indem Testschnittstellen geplant werden.

Abschnitt 2.3: Siehe Abschnitt 2.3 zur Auslegung auf Testbarkeit und Automatisierung.

**Faktoren bei der Automatisierung von Fehlernachtests**

Wichtige Punkte:

Was sind Fehlernachtests?

Zweck: Fehlernachtests werden durchgeführt, um sicherzustellen, dass ein gemeldeter Fehler nach seiner Behebung nicht mehr auftritt.

Prozess: Ein Tester reproduziert die erforderlichen Schritte, um zu bestätigen, dass der Fehler behoben ist.

Warum Fehlernachtests automatisieren?

Fehlerfortpflanzung: Fehler können sich von Release zu Release fortpflanzen, was auf Probleme im Konfigurationsmanagement hinweist.

Effizienz: Die Automatisierung von Fehlernachtests verkürzt die Ausführungszeit.

Erweiterung: Automatisierte Fehlernachtests können zu bestehenden Regressionstests hinzugefügt werden.

Begrenzter Funktionsumfang:

Spezifische Tests: Automatisierte Fehlernachtests haben in der Regel einen eng begrenzten Funktionsumfang.

Zeitpunkt der Implementierung: Die Automatisierung kann erfolgen, sobald ein Fehler gemeldet und die Schritte zur Reproduzierung ermittelt wurden.

Einbindung in Regressionstestsuite:

Standardisierte Suite: Automatisierte Fehlernachtests können in eine standardisierte automatisierte Regressionssuite integriert werden.

Zusammenfassung: Alternativ können sie in bestehenden automatisierten Tests zusammengefasst werden.

Nachverfolgung und Erfassung:

Erfassung: Die Nachverfolgung automatisierter Fehlernachtests ermöglicht die Erfassung der Zeit und der Anzahl der Zyklen, die für die Fehlerbehebung aufgewendet wurden.

Regressionstests und Auswirkungsanalyse:

Nebenwirkungen: Regressionstests sind erforderlich, um sicherzustellen, dass durch die Fehlerbehebung keine neuen Fehler eingeführt wurden.

Auswirkungsanalyse: Zur Bestimmung des angemessenen Umfangs der Regressionstests kann eine Auswirkungsanalyse notwendig sein.

**7. Verifizieren der TAS**

Verifizieren der Komponenten der automatisierten Testumgebung

Wichtige Punkte:

Zweck der Verifizierung:

Sicherstellung der Funktionalität: Das Testautomatisierungsteam muss prüfen, ob die automatisierte Testumgebung wie vorgesehen arbeitet, bevor das automatisierte Testen beginnt.

Schritte zur Überprüfung der Komponenten:

Installation, Einrichtung, Konfiguration und Anpassung von Testwerkzeugen:

Komponenten der TAS: Die Testautomatisierungslösung (TAS) besteht aus vielen Komponenten, die zur zuverlässigen und wiederholbaren Ausführung beitragen.

Herzstück: Dazu gehören ausführbare Komponenten, funktionale Bibliotheken, unterstützende Daten und Konfigurationsdateien.

Konfiguration der TAS:

Automatisierte Skripte: Konfiguration kann durch automatisierte Installationsskripte erfolgen.

Manuelle Ablage: Alternativ durch manuelles Ablegen von Dateien in entsprechenden Ordnern.

Service Packs und Add-Ins: Regelmäßige Updates, um die Kompatibilität mit der SUT-Umgebung zu gewährleisten.

Automatisierte Installation und zentrales Repository:

Vorteile: Gewährleistet, dass Tests an unterschiedlichen SUTs mit derselben Version und Konfiguration der TAS durchgeführt werden.

Upgrades: Upgrades der TAS können über ein zentrales Repository erfolgen.

Standardisierte Prozesse: Die Nutzung des Repositorys und der Upgrade-Prozess sollten identisch mit den standardmäßigen Entwicklungswerkzeugen sein.

Warum sind diese Schritte wichtig?

Konsistenz: Automatisierte Installation und zentrale Repositories sorgen für konsistente Testumgebungen.

Zuverlässigkeit: Regelmäßige Updates und korrekte Konfiguration stellen sicher, dass die TAS zuverlässig und kompatibel mit der SUT-Umgebung ist.

Effizienz: Automatisierte Prozesse sparen Zeit und minimieren Fehler, die bei manuellen Abläufen auftreten können.

**Verifizieren der Komponenten der automatisierten Testumgebung (Fortsetzung)**

Testskripte mit bekannten Bestanden- und Fehlgeschlagen-Ergebnissen

Ermittlung grundlegender Probleme:

Bestandene Tests: Wenn Testfälle, die bestehen sollten, fehlschlagen, liegt ein grundlegendes Problem vor, das sofort behoben werden muss.

Fehlgeschlagene Tests: Wenn Testfälle bestehen, die hätten fehlschlagen sollen, müssen die fehlerhaften Komponenten identifiziert werden.

Überprüfung von Protokolldateien und Performanzmetriken:

Protokollierung: Überprüfen, ob Protokolldateien korrekt erstellt wurden.

Performanzmetriken: Sicherstellen, dass Performanzmetriken richtig erfasst wurden.

Automatisches Setup und Teardown: Testfälle bzw. Testskripte müssen automatisch eingerichtet und außer Betrieb genommen werden.

Durchführung weiterer Tests:

Verschiedene Testtypen: Durchführen von Tests anderer Typen und Ebenen (z.B. funktionale Tests, Performanztests, Komponententests) zur zusätzlichen Validierung.

Framework-Ebene: Diese Überprüfungen sollten auch auf Framework-Ebene erfolgen, um sicherzustellen, dass das gesamte Testautomatisierungsframework ordnungsgemäß funktioniert.

Warum sind diese Schritte wichtig?

Fehlererkennung: Schnellere Erkennung und Behebung grundlegender Probleme.

Verlässlichkeit: Sicherstellung, dass die Testautomatisierungsumgebung verlässlich und korrekt arbeitet.

Umfassende Validierung: Zusätzliche Tests auf verschiedenen Ebenen erhöhen die Sicherheit, dass die Automatisierung korrekt funktioniert.

**Wiederholbarkeit der Einrichtung/Außerbetriebnahme der Testumgebung**

Einfach erklärt:

TAS auf verschiedenen Systemen:

Die Testautomatisierungslösung (TAS) wird auf verschiedenen Computern und Servern installiert.

Systematischer Ansatz:

Damit die TAS überall richtig funktioniert, braucht man einen klaren Plan, wie man sie installiert und wieder deinstalliert.

Gleiche Ergebnisse:

Es sollte keinen Unterschied machen, wie oft man die TAS einrichtet oder neu einrichtet – sie muss immer gleich gut funktionieren.

Konfigurationsmanagement:

Ein gutes System zur Verwaltung der TAS-Einstellungen sorgt dafür, dass man die TAS immer zuverlässig einrichten kann.

Warum ist das wichtig?

Zuverlässigkeit: Damit die TAS immer richtig funktioniert, egal wo sie installiert wird.

Einfachheit: Damit die Einrichtung einfach und fehlerfrei ist.

Konsistenz: Damit die Ergebnisse der TAS immer gleich sind, egal wie oft sie eingerichtet wird.

**Konfiguration von Testumgebung und -komponenten**

Einfach erklärt:

Veränderungen im System:

Wenn sich das System unter Test (SUT) ändert, müssen wir wissen, welche Teile der Testautomatisierungslösung (TAS) betroffen sind oder geändert werden müssen.

Komponenten der TAS:

Die verschiedenen Teile der TAS müssen bekannt und dokumentiert sein. Dazu gehören:

Testwerkzeuge: Software, die die Tests ausführt.

Skripte: Die Anweisungen, die die Tests beschreiben.

Daten: Die Informationen, die die Tests verwenden.

Konfigurationsdateien: Einstellungen, die die Tests steuern.

Dokumentation:

Alle Teile der TAS müssen gut dokumentiert sein, damit jeder weiß, was sie tun und wie sie funktionieren.

Warum ist das wichtig?

Verständnis: Damit man weiß, welche Teile der TAS bei Änderungen im SUT angepasst werden müssen.

Effizienz: Damit Anpassungen schnell und korrekt vorgenommen werden können.

Verlässlichkeit: Damit die TAS immer richtig funktioniert, egal welche Änderungen im SUT vorgenommen werden.

**Konnektivität gegenüber internen und externen Systemen/Schnittstellen**

Einfach erklärt:

Installation der TAS:

Die Testautomatisierungslösung (TAS) wird in der Umgebung des Systems unter Test (SUT) installiert.

Kontrollen und Vorbedingungen:

Bevor die TAS gestartet wird, müssen bestimmte Kontrollen durchgeführt und Vorbedingungen erfüllt sein:

Konnektivität: Sicherstellen, dass die TAS mit allen internen und externen Systemen und Schnittstellen verbunden ist.

Warum ist das wichtig?

Funktionalität: Die TAS kann nur richtig funktionieren, wenn sie mit allen benötigten Systemen und Schnittstellen verbunden ist.

Nachweis: Die erfolgreiche Erfüllung der Vorbedingungen zeigt, dass die TAS korrekt installiert und konfiguriert wurde.

**Grad der Intrusion automatisierter Testwerkzeuge**

Einfach erklärt:

Was ist Intrusion?

Intrusion: Wie stark das Testwerkzeug das System unter Test (SUT) beeinflusst.

Enge Verzahnung der TAS und SUT:

Positiv: Hohe Kompatibilität, besonders bei Interaktionen auf GUI-Ebene.

Negativ: Kann das Verhalten und die Performance des SUT beeinflussen, wenn die TAS aktiv ist.

Beispiele für verschiedene Intrusionsgrade:

Niedriger Intrusionsgrad:

Externe Schnittstellen: Testen über physische Schalter, USB-Signale usw.

Vorteile: Beste Simulation der Benutzeraktionen, keine Änderungen am SUT, kein Einfluss auf Verhalten und Timing.

Nachteile: Komplex, benötigt möglicherweise spezielle Hardware.

Mittlerer Intrusionsgrad:

GUI-Schnittstellen: Anpassung der SUT-Umgebung für UI-Befehle und Datenlesen.

Vorteile: Weniger komplex, handelsübliche Werkzeuge können verwendet werden.

Nachteile: Timing des SUT kann beeinflusst werden, was das Verhalten verändert.

Hoher Intrusionsgrad:

APIs und Testschnittstellen: Nutzung von Software-Schnittstellen für Tests.

Vorteile: Einfach und kostengünstig, automatisierte Tests sind leicht durchführbar.

Nachteile: Höhere Wahrscheinlichkeit, das Verhalten des SUT zu beeinflussen, da Schnittstellen genutzt werden, die Endbenutzer nicht verwenden.

Risiken eines hohen Intrusionsgrades:

Fehler beim Testen: Tests können Fehler zeigen, die in der Praxis nicht auftreten.

Vertrauen: Vertrauen in die Testautomatisierungslösung kann sinken, wenn Tests aufgrund der Intrusion fehlschlagen.

Manuelle Reproduktion von Fehlern:

Vorgabe: Entwickler können verlangen, dass Fehler aus automatisierten Tests manuell reproduziert werden, um die Analyse zu unterstützen.

Warum ist das wichtig?

Verlässlichkeit: Niedriger Intrusionsgrad bedeutet, dass Tests zuverlässiger sind und das SUT nicht beeinflusst wird.

Effizienz: Automatisierte Tests sollten das SUT möglichst wenig beeinflussen, um genaue Ergebnisse zu liefern.

Vertrauen: Geringere Intrusion erhöht das Vertrauen in die Ergebnisse der Testautomatisierung

**Testen der Framework-Komponenten**

Einfach erklärt:

Einzelne Tests für Framework-Komponenten:

Warum? Wie bei jeder Software müssen die Teile des Testautomatisierungs-Frameworks (TAS) einzeln geprüft werden, um sicherzustellen, dass sie richtig funktionieren.

Arten von Tests:

Funktionale Tests: Prüfen, ob jede Komponente ihre Aufgaben korrekt ausführt.

Nichtfunktionale Tests: Untersuchen die Leistung, Ressourcennutzung, Benutzbarkeit und andere Aspekte, die nicht direkt mit den Funktionen zusammenhängen.

Beispiele für funktionale Tests:

Objektverifizierung: Komponenten, die auf GUI-Systemen Objekte prüfen, müssen auf vielen verschiedenen Objektklassen getestet werden, um sicherzustellen, dass sie richtig arbeiten.

Fehlerprotokolle und Berichte: Diese müssen genaue Informationen über den Status der Automatisierung und das Verhalten des SUT liefern.

Beispiele für nichtfunktionale Tests:

Leistung: Überprüfen, ob das Framework bei der Ausführung langsamer wird oder mehr Ressourcen verbraucht, was auf Probleme wie Speicherlecks hinweisen kann.

Ressourcennutzung: Sicherstellen, dass das Framework nicht zu viele Systemressourcen (z. B. Speicher, CPU) verbraucht.

Interoperabilität: Testen, ob die Komponenten gut miteinander und mit anderen Systemen arbeiten.

Warum ist das wichtig?

Verlässlichkeit: Einzelne Tests stellen sicher, dass jede Komponente zuverlässig funktioniert.

Leistung: Nichtfunktionale Tests sorgen dafür, dass das Framework effizient arbeitet und keine Systemressourcen verschwendet.

Präzision: Fehlerprotokolle und Berichte müssen genaue Informationen liefern, um Probleme schnell zu identifizieren und zu beheben.

**Verifizieren der automatisierten Testsuite**

Einfache Erklärung:

Warum verifizieren?

Sicherstellen, dass die automatisierte Testsuite vollständig, konsistent und korrekt funktioniert.

Schritte zur Überprüfung:

Ausführung von Testskripten mit bekannten Ergebnissen:

Bekannte Ergebnisse: Testskripte ausführen, die bekanntlich bestehen oder fehlschlagen sollten.

Fehler finden: Wenn Tests fehlschlagen, die bestehen sollten, gibt es ein Problem, das behoben werden muss. Wenn Tests bestehen, die fehlschlagen sollten, muss der Fehler gefunden werden.

Protokolle und Daten: Überprüfen, ob Protokolldateien, Ausführungsdaten, Setup und Teardown korrekt generiert werden.

Verschiedene Tests: Führen Sie auch funktionale Tests, Performanztests und Komponententests durch.

Prüfen der Testsuite:

Vollständigkeit: Überprüfen, ob alle Testfälle erwartete Ergebnisse haben und alle Testdaten vorhanden sind.

Korrekte Version: Sicherstellen, dass die Testsuite die richtige Version des Frameworks und des SUT verwendet.

Verifizieren neuer Tests:

Neue Funktionen: Neue Testfälle, die neue Funktionen des Frameworks nutzen, genau überwachen und überprüfen.

Gewährleisten der Wiederholbarkeit:

Konstante Ergebnisse: Sicherstellen, dass wiederholte Tests immer dasselbe Ergebnis liefern.

Fehleranalyse: Tests, die unzuverlässige Ergebnisse liefern, aus der aktiven Testsuite entfernen und analysieren. Ursachen können im Testfall, Framework oder SUT liegen.

Genug Verifizierungspunkte:

Nachweis der Ausführung: Protokollieren von Start- und Endzeit jedes Testfalls und Aufzeichnung des Testausführungsstatus. Verifizieren, dass die Testsuite wie vorgesehen ausgeführt wurde.

Warum ist das wichtig?

Verlässlichkeit: Sicherstellen, dass die Testsuite korrekt und zuverlässig arbeitet.

Fehlererkennung: Frühzeitige Erkennung und Behebung von Fehlern.

Konsistenz: Gewährleisten, dass Tests bei jeder Ausführung die gleichen Ergebnisse liefern.

Umfassende Prüfung: Alle wichtigen Aspekte des Systems werden überprüft.

**8. Fortlaufende Optimierung**

Möglichkeiten der Optimierung der Testautomatisierung

Einfach erklärt:

Die Testautomatisierungslösung (TAS) kann in vielen Bereichen verbessert werden, um effizienter, benutzerfreundlicher und leistungsfähiger zu sein. Hier sind einige wichtige Möglichkeiten zur Optimierung:

Skripterstellung:

Ansatz ändern: Verwenden Sie komplexere Automatisierungsansätze für neue Tests und passen Sie bestehende Tests an, wenn nötig.

Überlappungen prüfen: Konsolidieren Sie Testschritte, die sich überschneiden, in Funktionen, die in einer Bibliothek gespeichert und wiederverwendet werden können.

Fehlerwiederherstellung: Entwickeln Sie Prozesse, um Fehler im Test oder im SUT zu beheben, damit die Tests weiterlaufen können.

Wartemechanismen verbessern: Verwenden Sie dynamisches Warten oder abonnieren Sie Ereignisse, um Wartezeiten zu optimieren und die Testzeit zu verkürzen.

Testmittel als Software behandeln: Wenden Sie gute Programmierpraktiken an und ziehen Sie gegebenenfalls Softwareentwickler zur Unterstützung heran.

Problematische Skripte überarbeiten: Überarbeiten oder entfernen Sie problematische Skripte, um die Wartung zu erleichtern.

Testausführung:

Parallele Ausführung: Führen Sie Tests auf mehreren Systemen parallel aus, wenn möglich.

Testsuite aufteilen: Teilen Sie große Testsuiten in kleinere Teile auf, die in vordefinierten Zeiträumen ausgeführt werden können.

Duplikate entfernen: Analysieren Sie die Testsuite auf doppelte Tests und entfernen Sie diese, um die Ausführungszeit zu verkürzen.

Verifizierung:

Standardisierte Verifizierungsverfahren: Verwenden Sie einheitliche Verifizierungsmethoden für alle Tests, um doppelte Implementierung zu vermeiden.

Architektur:

Architekturänderungen: Passen Sie die Architektur des SUT oder der TAS an, um die Testbarkeit zu verbessern. Dies kann Investitionen erfordern, aber die Effizienz der Automatisierung erheblich steigern.

Vor- und Nachbearbeitung:

Standardisierte Setup- und Teardown-Aufgaben: Verwenden Sie standardisierte Aufgaben für die Einrichtung und das Aufräumen, um die Implementierung und Wartung zu vereinfachen.

Dokumentation:

Umfassende Dokumentation: Dokumentieren Sie Skripte, die Benutzung der TAS und die von der TAS erzeugten Berichte und Protokolle gründlich.

TAS-Funktionen:

Erweiterung der Funktionen: Fügen Sie neue Funktionen wie detaillierte Berichterstattung oder Integration mit anderen Systemen hinzu, aber nur, wenn diese auch genutzt werden.

TAS-Updates und -Upgrades:

Probetests: Führen Sie Tests durch, bevor Sie die TAS auf neue Versionen aktualisieren, um sicherzustellen, dass bestehende Testfälle nicht beeinträchtigt werden.

Warum ist das wichtig?

Effizienz: Optimierungen reduzieren die Eingriffe durch den Menschen und machen die Tests schneller und zuverlässiger.

Benutzerfreundlichkeit: Verbesserungen machen die TAS einfacher zu bedienen.

Leistungsfähigkeit: Die TAS kann mehr Funktionen bieten und besser auf die Bedürfnisse des Projekts abgestimmt werden.

**8.2 Planung der Realisierung der Testautomatisierungsverbesserung**

Planung der Realisierung der Testautomatisierungsverbesserung

Einfache Erklärung:

Sorgfältige Planung und Analyse:

Änderungen an der Testautomatisierungslösung (TAS) müssen gut durchdacht und geplant sein, da sie die Zuverlässigkeit und Leistung beeinflussen können.

Ermittlung von Änderungen:

Änderungen an Komponenten: Überlegen, welche Teile der TAS geändert oder verbessert werden müssen, z.B. Testsoftware, Funktionsbibliotheken oder Betriebssysteme.

Schrittweise Änderungen: Änderungen sollten schrittweise erfolgen, um ihre Auswirkungen zu messen. Wenn alles gut läuft, können die Änderungen vollständig umgesetzt werden.

Regressionslauf: Führen Sie am Ende einen vollständigen Testdurchlauf durch, um sicherzustellen, dass die Änderungen keine Probleme verursacht haben.

Effizienzsteigerung der TAS-Funktionsbibliotheken:

Neue Techniken: Integrieren Sie neue Techniken und optimierten Code in die zentralen Funktionsbibliotheken, um Aufgaben effizienter auszuführen.

Konsolidierung von Funktionen:

Funktionen zusammenfassen: Fassen Sie mehrere Funktionen, die auf dieselben Steuerelemente wirken, zusammen, um Wartungsaufwand zu reduzieren und die Effizienz zu erhöhen.

Refaktorisieren der TAS:

Anpassung an SUT-Änderungen: Passen Sie die Architektur der TAS an, um Änderungen im System unter Test (SUT) zu berücksichtigen und sicherzustellen, dass die TAS weiterhin das SUT unterstützt.

Benennungskonventionen und Standardisierung:

Einhaltung von Standards: Stellen Sie sicher, dass neue Automatisierungscodes und Funktionsbibliotheken den bestehenden Benennungskonventionen und Standards entsprechen.

Prüfen bestehender Testskripte:

Überarbeitung/Löschung: Überprüfen Sie alte Skripte auf ihre Relevanz. Löschen Sie selten oder nie ausgeführte Tests und teilen Sie komplexe Tests in kleinere, handlichere Teile auf.

Warum ist das wichtig?

Zuverlässigkeit: Sorgfältige Planung und schrittweise Änderungen verhindern unerwartete Probleme und stellen sicher, dass die TAS zuverlässig bleibt.

Effizienz: Neue Techniken und die Konsolidierung von Funktionen machen die TAS schneller und einfacher zu warten.

Anpassungsfähigkeit: Durch die Anpassung der TAS an Änderungen im SUT bleibt die Testautomatisierung effektiv.