Ostfalia
Hochschule für angewandte Wissenschaften

Fakultät Informatik

Hadi Dahnoun, 70486348

Testautomatisierungstool aus Sicht der Requirements-Engineering

Hausarbeit zur Erlangung des Leistungsnachweises
im Modul
Requirements Engineering
an der
Ostfalia Hochschule für angewandte Wissenschaften
– Hochschule Braunschweig/Wolfenbüttel
Fachrichtung: Wirtschaftsinformatik (M. Sc.)

Erste/-r Prüfer/-in: Prof. Dr. Dirk Frosch-Wilke

Eingereicht am 28.07.2024

Diese Arbeit wurde im Rahmen der Lehrveranstaltung Requirements-Engineering erstellt

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit mit dem Titel

„Testautomatisierungstool aus Sicht der Requirements-Engineering“

selbstständig angefertigt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der beim Professor abgegebenen Fassung übereinstimmt.

Hadi Dahnoun

Digitally signed by Hadi Dahnoun
Date: 2024.08.04 19:12:07 +02'00'

Stuttgart, der 28.07.2024

Ort, Datum **Unterschrift**

Aufgabenstellung

Die Praxisarbeit sollte 15-20 Seiten umfassen und die im Modul behandelten Themen anhand eines Ihnen bekannten Projektes -möglichst aus Ihrem beruflichen Umfeld – konkretisieren. Falls dies nicht möglich ist, soll ein bei Ihnen im Unternehmen eingesetzte Software Grundlage der Praxisarbeit sein. In der Praxisarbeit soll die (eigene) Vorgehensweise im Projekt aus Sicht des Requirements Engineering dargestellt werden bzw. im Sinne eines „Reverse Requirements Engineering“ beschrieben werden.

Die Praxisarbeit soll die folgenden Aspekte beinhalten:

a) Beschreibung der **Vision und Ziele** des Systems sowie des **Systemkontextes** (4 Punkte)
b) Beschreibung der wesentlichen **Marktanforderungen** (3 Punkte)
c) **Analyse der Anforderungsquellen** (Details für drei bis vier Quellen (davon mindestens 2 Stakeholder)) (4 Punkte)
d) Ausgehend von den Ergebnissen in Teil c) begründete Auswahl von **Techniken zur Erhebung der Anforderungen** (ggf. Beschreibung der Verwendung der Techniken) (3 Punkte)
e) Umfassende **natürlich-sprachliche Dokumentationen** von mindestens acht Produktanforderungen, die sowohl funktionale Anforderungen als auch Qualitätsanforderungen sowie Randbedingungen umfassen müssen (incl. Klassifikation der Anforderungen). Hierbei soll die Satzschablone des Sophisten-Regelwerkes verwendet werden. Die Dokumentation einer Anforderung soll ferner den diesbezüglichen Qualitätskriterien genügen. (6 Punkte)
f) Für ausgewählte(s) **Systemverhalten, -funktionen und –strukturen** ist jeweils ein semi-formales Anforderungsmodell (also insgesamt drei Modelle) zu erstellen. (6 Punkte)
g) Auswahl von vier im Teil e) beschriebenen Produktanforderungen, die Kriterien zur Priorisierung erfüllen (dies ist zu begründen). Anwendung des **AHP-Verfahrens zur analytischen Priorisierung** auf diese vier Anforderungen. (4 Punkte)

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einführung in das Software-System

1.1 Vorstellung des betrachteten Systems

Bei dem betrachteten System handelt es sich um ein Automatisierungstool für die Ausführung von Softwaretests, das speziell für den Einsatz in der Automobilindustrie entwickelt wird. Es zielt darauf ab, die Ausführung von Testfällen für Fahrerassistenzsysteme (ADAS) mit einem einzigen Klick zu automatisieren und umfassende Testberichte zu erstellen. Das Tool lässt sich nahtlos in bestehende Entwicklungs- und Testumgebungen integrieren und unterstützt daher die Wünsche mehrere Kunden.

Die zu automatisierenden Testfälle können von Projekt zu Projekt variieren. Das Tool kann nach der Testfallerstellung die Ausführung der Testfälle automatisieren und hilft, das Reporting zu standardisieren und klar zu strukturieren und kann weitere Testprobleme lösen. Automatisierte Testfallerstellung auf Basis des Lastenheftes ist keine Funktion des Tools.

Hauptfunktionen des Systems:

- Benutzerfreundliche Testfallerstellung
- Automatisierte Ausführung von Testfällen
- Generierung detaillierter Testberichte
- Kompatibilität mit gängigen ADAS-Entwicklungsumgebungen (dspace, CANoe..)

1.2 Vision des Software-Systems

Die Vision des Projekts besteht darin, einen Industriestandard für die automatisierte Testdurchführung in der Automobilbranche zu setzen.

Effizienzsteigerung wird durch eine Erhöhung der Testtiefe bei gleichzeitiger Reduzierung der für die Durchführung von Softwaretests benötigten Zeit und Kosten erreicht. Darüber hinaus verbessert das Tool die **Sicherheit**, indem es sicherstellt, dass ADAS-Software den höchsten Sicherheitsstandards entspricht mit Berücksichtigung der Erfüllung der Compliance-Anforderungen. Eine Vision ist es, die **Marktführerschaft** zu erlangen und das Tool als bevorzugte Lösung für die Testautomatisierung in der Automobilindustrie zu etablieren. Durch innovative Lösungen (wie KI-Schnittstellen), die dieses Tool enthalten kann, wird es für unterschiedliche große Kunden besonders attraktiv.

1.3 Ziele des Projekts mit Berücksichtigung der S.M.A.R.T Kriterien

1. **Testfall-Automatisierung:** Ziel ist es, eine zuverlässige Plattform bereitzustellen, die mit einem einzigen Klick automatisch Testfälle für ADAS-Software ausführen kann. Durch die vollständige Automatisierung soll die manuelle Arbeit minimiert und die Wahrscheinlichkeit menschlicher Fehler verringert werden. Dies garantiert eine höhere Qualität und Effizienz bei der Durchführung von Tests. Durch die Integration von

KI (ML-Technologien) sollen Algorithmen zur Testfallerstellung und Priorisierung entwickelt werden. Das Tool soll automatisch Parameter variieren oder Testfälle erweitern auf der Grundlage früherer Erfahrungen, um Lücken im Testprozess zu schließen. Dies führt zu einer dynamischeren und umfassenderen Testabdeckung, die kontinuierlich optimiert wird.

2. **Erstellung von Testberichten:** Das Hinzufügen einer Funktion, die automatisch umfassende Testberichte erstellt, ist entscheidend. Um eine vollständige und verständliche Dokumentation der Testergebnisse jeder Softwarestand zu erhalten, sollten diese Berichte sowohl technische Details als auch Zusammenfassungen für die Stakeholder enthalten. Dadurch wird eine bessere Nachvollziehbarkeit ermöglicht.
3. **Integration und Kompatibilität:** Die Unterstützung von weit verbreiteten Entwicklungsplattformen stellt sicher, dass mehrere Kunden aus der Automobilindustrie davon profitieren können. Die breite Kompatibilität stellt sicher, dass die Werkzeuge effektiv und anpassungsfähig in einer Vielzahl von Umgebungen eingesetzt werden können. **Zielsetzung:** Gewährleistung der nahtlosen Integration des Tools in mindestens drei weit verbreitete Entwicklungsplattformen (Dspace, CANoe, und Matlab), mit einer erfolgreichen Kompatibilitätsbewertung durch Benutzer.
4. **Einhaltung von Sicherheits- und Compliance-Anforderungen:** Einhaltung aller geltenden Sicherheitsstandards und gesetzlichen Anforderungen, wie z. B. ISO 26262. Das Werkzeug muss Funktionen zur Dokumentation und Validierung der Einhaltung dieser Standards bieten, um die Sicherheit und Konformität der getesteten ADAS-Software zu gewährleisten.

Die Funktionalität von den Zielen sollte innerhalb 12 Monaten nach Projektstart lauffähig sein.

1.4 Definition des Systemkontextes

Der Systemkontext beschreibt der Umgebung, in der das automatisierte Testtool für Softwaretests operiert. Diese Umgebung umfasst alle relevanten **Produktive Systeme**, **Personen(-gruppen)**, **Prozesse** und **Ereignisse**, die das Tool beeinflussen oder davon beeinflusst werden.

1.4.1 Externe Systeme und Schnittstellen

Das Testtool interagiert mit verschiedenen externen Systemen und Softwarekomponenten:

1. **Entwicklungsumgebungen** (**u.a Visual Studio)**: Integration zur direkten Ausführung von Tests
2. **Fahrzeugsimulationssoftware wie dSPACE, Matlab, Vector CANoe** bieten Simulierte Testumgebungen für ADAS-Software, um verschiedene Szenarien zu testen oder Funktionalität eines Units nachzubilden.
3. **Cloud-Plattformen wie AWS, Azure oder Google Cloud** bieten Skalierbare Ressourcen für umfangreiche Testausführungen und Speicherung großer Datenmengen (Datenbanken).

1.4.2 Stakeholder

Verschiedene Benutzergruppen (Stakeholder) interagieren mit dem Testtool. Diese umfassen:

1. **Entwickler des Software-Tools:** Zuständig für die Umsetzung der Anforderungen, die für das Testing Tool geplant sind
2. **Testingenieure**: Verantwortlich für die Erstellung, Verwaltung und Ausführung von Testfällen sowie die Analyse der Testergebnisse.
3. **Projekt/Test-manager**: Überwachen den Fortschritt und die Qualität der Softwareentwicklung durch die Analyse von Testberichten-> Koordinieren mit externen Softwarelieferanten.
4. **IT-Administratoren**: Zuständig für die Installation, Konfiguration und Wartung des Testtools sowie die Verwaltung der Infrastruktur.(z.B. Verwaltung von abonnierten Cloud Services)
5. **Potenzielle Externe Softwarelieferanten:** Liefern die zu testende Software und erhalten Feedback basierend auf den Testergebnissen.
6. **Potenzielle Kunde/Auftragsgeber:** oft ein Automobilhersteller oder ein Tier-1-Zulieferer, beauftragt ein spezialisiertes Testunternehmen mit der Validierung der von externen Softwarelieferanten bereitgestellten Software.

1.4.3 Prozesse

Das Testtool wird in verschiedene Entwicklungs- und Testprozesse integriert:

1. **Software-Integrationsprozess:** Empfangen der extern entwickelten Softwareversionen zu definierten Zeitpunkten (z.B. Sprints) und Integration in die Testumgebung.
2. **Testmanagement**: Verwaltung von Testfällen, Planung und Ausführung von Tests sowie Nachverfolgung und Berichtserstellung von Fehlern.
3. **Fehler- und Änderungsmanagement**: Nachverfolgung von Fehlern und Testfalländerungen durch das Tool.

1.4.4 Technische Rahmenbedingungen

Das Tool muss unter bestimmten physischen und technischen Bedingungen arbeiten:

1. **Hardwareanforderungen**: von Softwarekomplexität abhängige leistungsfähige Server oder Cloud-Ressourcen zur Ausführung von Tests und Verarbeitung deren Datenmengen.
2. **Netzwerkanforderungen**: Zuverlässige Netzwerkverbindungen
3. **Sicherheitsanforderungen**: Implementierung von Sicherheitsmechanismen wie Authentifizierung, Autorisierung und Verschlüsselung zum Schutz sensibler Daten und zur Einhaltung von Datenschutzvorschriften.

2 Marktanalyse und Marktanforderungen

Die Problemdefinition ist der erste Schritt im Lebenszyklus eines Softwareprojekts. In dieser Phase dominieren die Marktanforderungen, die sich aus den Geschäftszielen oder -plänen eines Unternehmens ergeben und die Anforderungen von Kunden, Anwendern des Systems und des Gesetzgebers berücksichtigen. Daher finden sich in den Marktanforderungen Informationen über das "Warum?" einer potenziellen Softwaresystementwicklung. Diese Anforderungen sind oft noch sehr unpräzise oder nicht korrekt beschrieben, da die Kunden oder Nutzer eines Systems oft noch nicht genau wissen, wie sie ihre Anforderungen beschreiben sollen, um ihre Problemstellung ausreichend zu verdeutlichen. Es werden in diesem Kapitel die aktuelle trends im Bereich der Software-Testing in der Automobilindustrie kurz erläutert dann werden im Abschnitt 2.2 Marktanforderungen für das betrachtete Software-Produkt definiert.

2.1 Markt-Trends

Viele fortschrittliche Fahrerassistenzsysteme (ADAS) sind im Alltag bereits fest verankert und etabliert. Nach und nach kommen immer komplexere Assistenzfunktionen hinzu. Das zunehmende Komplexität sowie Die Forderung nach schnelleren Release-Zyklen und verbesserter Softwarequalität sind die treibenden Kräfte für neue Trends in der im Bereich des Software-testing.

Der Markt für Steuergerätetests in der Automobilindustrie wird von mehreren wichtigen Trends beeinflusst. Die Funktionalität von Steuergeräten wird in simulierten Umgebungen durch den Einsatz von **Hardware-in-the-Loop- (HiL) und Software-in-the-Loop- (SiL) Testmethoden** validiert, die umfangreiche Szenariotests vor der physischen Integration ermöglichen. Außerdem wird das kontinuierliche Testen während des gesamten Entwicklungszyklus durch die Integration von automatisierten Test-Frameworks in **CI/CD-Pipelines** ermöglicht, was eine schnelle Validierung von Änderungen garantiert. **Der Einsatz von Machine Learning** steigert die Testeffizienz durch vorausschauende Überwachung, intelligente Testfallerstellung und Erkennung von Anomalien. Das Tool *„Functionize“* beispielsweise setzt KI zur Erstellung und Pflege automatisierter Tests ein. Seine Engine (Natural Language Processing) ermöglicht es, Tests in einfachem Englisch zu schreiben, und das Tool nutzt KI, um sich an Änderungen in der Anwendung anzupassen. Der Schwerpunkt auf **Standardisierung und Einhaltung von Industriestandards** wie ISO 26262 gewährleistet, dass die Testprozesse strengen Sicherheits- und Qualitätsanforderungen genügen. Als letztes ermöglicht die Einführung **adaptiver und agiler Testmethoden** einen flexiblen und reaktionsfähigen Testprozess, der schnellere Iterationen und kontinuierliche Verbesserungen ermöglicht. In einem agilen Team arbeiten die Tester beispielsweise eng mit den Entwicklern zusammen, um automatisierte Tests für jede User Story zu erstellen, während diese entwickelt wird, sodass sofortiges Feedback und schnelle Iterationen gewährleistet sind.

2.2 Marktanforderungen

Für das automatisierte Testtool für ADAS-Software lassen sich folgende wesentliche Marktanforderungen identifizieren:

- **Testfall-Automatisierung:** Das Testtool muss in der Lage sein, Testfälle vollständig zu automatisieren, um die Effizienz des Testprozesses zu maximieren und menschliche Fehler zu minimieren.

- **Zuverlässigkeit und Sicherheit:** Das Testtool muss in der Lage sein, die Zuverlässigkeit und Sicherheit von ADAS-Software umfassend zu überprüfen. Es muss strenge Sicherheitsstandards wie ISO 26262 erfüllen und Sicherheitsprüfungen, einschließlich statischer und dynamischer Analysen automatisiert durchführen können.

Die Einhaltung gesetzlicher und regulatorischer Vorgaben ist zwingend erforderlich, um die Marktzulassung und Compliance sicherzustellen. Dazu gehört auch der Schutz projektrelevanter sensibler Daten.

- **Kompatibilität mit verschiedenen Plattformen:** Das Testtool muss mit unterschiedlichen Entwicklungsumgebungen integrieren lassen, die in der Automobilindustrie verwendet werden. (Z.B auch unterschiedliche Simulationsprogrammen wie CANoe oder dspace)

Die Kompatibilität mit verschiedenen Plattformen stellt sicher, dass das Tool von verschiedenen Unternehmen und in unterschiedlichen Projekten verwendet werden kann.

- **Intelligente Testfallerstellung**: Das Testtool soll KI- und ML-Algorithmen nutzen, um basierend auf bisherigen Testergebnissen automatisch neue Testfälle zu generieren und zu priorisieren. Es soll vorausschauende Überwachung, intelligente Testfallerstellung und Anomalieerkennung ermöglichen.

- **Berichterstellung:** Das Testtool muss in der Lage sein, detaillierte Testberichte zu erstellen, die sowohl technische Details als auch Zusammenfassungen für Stakeholder enthalten. Diese Berichte sollen klar und nachvollziehbar sein, um die Ergebnisse und den Status der Tests transparent zu machen.

Klare und umfassende Berichte helfen dabei, Testergebnisse zu verstehen und zu kommunizieren. Diese können auch als wichtige Grundlage für Entscheidungen über die Weiterentwicklung und Freigabe von Software dienen.

3 Anforderungsquellen

Es handelt sich um ein Softwareprodukt, das in erster Linie für die interne Testung und Validierung der von einem externen Lieferanten gelieferten Software bestimmt ist. Der Lieferant der Software ist nicht unbedingt der Auftraggeber. Im Allgemeinen können die Anforderungen an Softwareprodukte je nach ihrem Ursprung in drei Kategorien eingeteilt werden. Diese Kategorisierung hilft dabei, die unterschiedlichen Bedürfnisse und Erwartungen an die Software klar zu erkennen und zu berücksichtigen.

3.1 Interne Stakeholder

Bei der Definition des Systemkontexts wurden die an diesem Projekt beteiligten Stakeholder in Kapitel 1.4 definiert. Interne Stakeholder wie die Entwickler des Tools, die Nutzer des Tools (Tester und Projektmanager) und die IT-Abteilung können einen großen Einfluss auf die Gestaltung der Anforderungen haben.

3.2 Externe Stakeholder

Im betrachteten Projekt sind die Kunden/Auftraggeber auf Seiten des Softwareanbieters die wichtigsten externen Stakeholder, da sie den größten Einfluss auf die Anforderungen an die zu testende Software haben und die Anforderungen an die Entwicklung eines Testwerkzeugs beeinflussen können. So muss das Tool beispielsweise mit der Entwicklungsplattform des Softwareanbieters kompatibel sein, was zu einer vorrangigen Anforderung an das Testtool führt, da es die grundlegende Funktion des Tools, nämlich das Testen der Funktionalität der Software, beeinflusst.

3.3 Dokumentenanalyse

Um zu gewährleisten, dass die ADAS Software Sicherheits-, Qualitäts- und Konformitätskriterien erfüllen, sind regulatorische Quellen für Anforderungen im Automobilsektor von entscheidender Bedeutung. Fahrzeugindustriekonzerne und Regulierungsbehörden haben diese Standards und Regeln festgelegt, um zu gewährleisten, dass Fahrzeugsysteme - einschließlich elektronischer Steuergeräte (ECUs) - sicher und zuverlässig sind und wie vorgesehen funktionieren.

Wichtige Normen wie ISO 26262, ISO/IEC 15504 und ISO/IEC 27001 haben einen erheblichen Einfluss auf die Entwicklung von automatisierten Testtools in der Automobilbranche. ISO 26262 konzentriert sich auf die funktionale Sicherheit elektrischer und elektronischer Systeme in der Automobilindustrie und erfordert strenge Risikobewertungs-, Validierungs- und Verifizierungsprozesse, um die Sicherheit während des gesamten Lebenszyklus zu gewährleisten. Diese Norm wirkt sich auf Testwerkzeuge aus, da sie Funktionen für die Sicherheitsanalyse, Fehlererkennung und Konformitätsberichte vorschreibt. ISO/IEC 15504 (SPICE) bewertet den Reifegrad und die Fähigkeit von Softwareentwicklungsprozessen und betont den Bedarf an Testwerkzeugen, die die Prozessdokumentation, die Erfassung von Messwerten und die kontinuierliche Verbesserung unterstützen. ISO/IEC 27001 befasst sich mit dem Informationssicherheitsmanagement und verlangt von Automobilsystemen, dass sie den Datenschutz und die sichere Kommunikation effektiv verwalten. Die Einhaltung die-

ser Normen gewährleistet, dass Testwerkzeuge die Sicherheits-, Qualitäts- und Schutzanforderungen unterstützen, die für die Entwicklung zuverlässiger und sicherer Automobilsysteme unerlässlich sind.

3.4 Analyse der wichtigsten Anforderungsquellen

Für eine kurze Analyse werden vier wichtigste Anforderungsquellen betrachtet. Für jede Quelle werden die Rolle und Priorität oder der Grad des Einflusses auf das Projekt eingeordnet.

1. **Testingenieure:**
   Rolle: Testingenieure sind beteiligt, den Umfang, die Ziele, die Ressourcen, den Zeitplan und die für das Testen erforderlichen Aktivitäten zu planen. sie entwickeln und führen die entwickelten Testfälle mit Hilfe des automatisierten Testwerkzeugs aus. Nach der Ausführung der Tests analysieren sie die Ergebnisse, um Fehlfunktionen zu ermitteln. Außerdem dokumentieren sie die Ergebnisse, melden Fehler und arbeiten mit den Software-Lieferanten zusammen, um Probleme zu beheben.

   Einfluss auf das Projekt: Als primäre Anwender/Hauptnutzer des Tools haben die Tester einen direkten Einfluss auf die Gestaltung und Funktionalität des Tools. Ihr Beitrag führt zur Integration effektiver Testverfahren und zu schnelleren Testzyklen.

2. **Externe Softwarelieferanten:**
   Rolle: Externe Softwareanbieter sind für die Entwicklung und Bereitstellung der zu testenden Software verantwortlich. Sie definieren, was akzeptable Testergebnisse in den verschiedenen Entwicklungsphasen sind und stellen sicher, dass die Software die vordefinierten Kriterien für Funktionalität, Leistung usw. erfüllt. Die Lieferanten legen Fristen für die Lösung festgestellter Probleme fest und sorgen dafür, dass die Softwareentwicklung im Zeitplan bleibt und die Projektmeilensteine eingehalten werden.

   Einfluss auf das Projekt: Die Anforderungen der Lieferanten haben einen erheblichen Einfluss auf die Teststrategie, einschließlich der Arten der durchzuführenden Tests und der Kriterien für das Bestehen dieser Tests. Kommunikation und Zusammenarbeit mit den Zulieferern in einem Projekt sind wichtig, um Fehler zu beheben jedoch um das zu ermöglichen muss das Testtool die Testberichte so erstellen, dass es für die Entwickler beim Zulieferer verständlich und kompatibel sind

3. **Dokumente wie Normen und Lastenhefte (Siehe Abschnitt 3.3)**
   Rolle: Die Einhaltung dieser Standards ist gesetzlich vorgeschrieben und notwendig, um die Software auf den Markt bringen zu können. Daher für eine vollständige Softwareabsicherung relevant. ISO 26262 erfordert unter anderem die Implementierung von Diagnosefunktionen, die die Nutzung von DIDs (Data Identifiers) im Rahmen des UDS-Protokolls umfassen können.

   Einfluss auf das Projekt: Die Normen bieten einen strukturierten Ansatz für die Softwareprüfung, was zu einer besseren Projektplanung und -durchführung führt. Die Berücksichtigung und Umsetzung der erforderlichen Diagnosefunktionen gemäß ISO

26262 trägt zur Gesamtqualität und Sicherheit des Produkts bei. Die Einhaltung der regulatorischen Normen ist entscheidend für die rechtliche Marktfähigkeit der Software. Das ist für den Auftragsgeber einen Kritischen Punkt wenn er die Testing-Firma beauftragt die Software abzusichern.

Eine weitere bedeutende Quelle sind die Lastenhefte und die verfügbaren Testfälle (Testdokumentation) von laufenden oder früheren Projekten. Überprüfung vom verfügbaren Lastenheften, um eine Einsicht in die typische Testumfänge zu schaffen kann nützlich sein. Die Untersuchung der zu automatisierten Testfälle sehr wichtig , um die Überprüfungskriterien zu verstehen. Dadurch wird sichergestellt, dass alle Anforderungen des Tools realisierbar sind da hier Testfälle für anderen Projekte wiederverwendet sein können.

Als letztes müssen Dokumentationen bzw. API-Dokumentationen der zu verbindenden externen ADAS-Entwicklungsumgebungen analysiert werden.

4. **Auftragsgeber (Kunde)**

Rolle: Der Kunde legt fest, welche Anforderungen die Software erfüllen muss und erwartet umfassende Berichte über die Testergebnisse für das Antriggern der Software-freigabe

Einfluss auf das Projekt: Die Anforderungen des Kunden bestimmen weitgehend die Teststrategie und setzen Prioritäten für besonders sorgfältig zu testende Testfälle. Der Kunde legt fest, welche Testergebnisse akzeptabel sind und welche Kriterien für die Freigabe der Software erfüllt sein müssen, was die Definition der Abnahmetests und der Fehler- und Leistungsmetriken beeinflusst.

Zeitpläne und Fristen, die vom Kunden vorgegeben werden, erfordern eine organisierte Testdurchführung, um sicherzustellen, dass alle Anforderungen rechtzeitig validiert werden. Regelmäßige Rückmeldungen des Kunden zu den Testberichten führen zu Anpassungen im Testprozess, so dass das Testtool kontinuierlich optimiert wird. Außerdem hat der Kunde spezifische Anforderungen an die Dokumentation der Testergebnisse was wiederum Einfluss darauf hat, wie das Testtool die Berichte erstellt und welche Informationen darin enthalten sein müssen.

4 Techniken zur Anforderungserhebung

Ausgehend von der Analyse der Anforderungsquellen in Kapitel 4 werden in diesem Kapitel der Arbeit die Techniken vorgestellt, die zur Erhebung der Anforderungen aus diesen Quellen verwendet werden, und es wird erläutert, warum sie ausgewählt werden.

4.1 Dokumentenanalyse für existierende Artefakte

Eine Artefakt basierte Erhebungstechnik wird sich auf Dokumente aus den Lastenheften sowie die Testfälle eines bereits in älteren Projekt getesteten Software sowie die genannten Standards und Normen basieren.

1. Regulatorischer Dokumente und Normen, insbesondere mit Schwerpunkt auf ISO 26262 und zugehörigen Diagnoseprotokollen.
2. Bereits abgearbeitete Testfälle oder Lastenhefte eines Software-Produkts.
3. API-Dokumetation von CANoe, Dspace, Matlab

Aus Regulatorischer Dokumente und Normen kann eine Checkliste der Compliance-anforderungen erstellt und in den Entwicklungsplan für das Testtool integriert. Darüber hinaus hilft die ISO26262 dabei, die Standardtests für sämtliche Steuergerätesoftware zu definieren (standarisiertes UDS-Protokoll für Steuergerätesoftware) d.h. es können aus diesem Norm Anforderungen extrahiert, die für das Testen jeder ADAS/Software gültig sind. Für andere Testfälle kann dies herstellerspezifisch sein und daher kann ein Beispiel auf der Grundlage älterer Projekte verwendet werden.

Der dritte Punkt ist wichtig, da das Tool eine API-Verbindung nutzt, um die Testfälle automatisch in der Test-/Entwicklungsumgebung zu testen. Daher ist es sehr wichtig zu prüfen, was die API erlaubt und wo die Grenzen liegen.

4.2 Interviews mit Testingenieuren

Einzelgespräche mit einer Auswahl von Testingenieuren führen und eine Reihe offener Fragen vorbereiten, die sich auf ihre täglichen Aufgaben, Herausforderungen und gewünschten Funktionen für das Testtool konzentrieren. Ein persönliches Gespräch mit Testingenieuren kann hier ein tieferes Verständnis für ihre alltäglichen Probleme. Diese Methode ermöglicht ein gründliches, qualitatives Feedback, das ihre Probleme und Verfahren genau beschreiben kann.

Um sicherzustellen, dass die erfassten Bedürfnisse relevant und nützlich sind, können die Interviews so angepasst werden, dass sie sich auf bestimmte Interessenbereiche konzentrieren z.B. die zu automatisierenden Testing-Umfänge kategorisieren in Testplanung, Eingangstests, Dauerlauftests, und Berichtserstellung und da die Interviews interaktiv sind, können hier unklare Bedürfnisse schnell geklärt und Themen detaillierter behandelt werden.

4.3 Workshops mit potenziellen externen Software-Lieferanten

Organisation von Workshops mit wichtigen Vertretern der potenziellen Softwareanbieter. Nutzung von moderierten Diskussionen zur Erfassung und Dokumentation ihrer Anforderungen, Erwartungen und akzeptablen Testergebnisse sowie die Fehlerberichtserstellung.

Diese Methode kann eine Herausforderung darstellen, wenn im Unternehmen keine Projekte mit bekannten Softwareanbietern laufen. Aus diesem Grund wird sich diese Praxisarbeit auf Benchmarking und möglicherweise Prototyping in Kapiteln 4.4 und 4.5 konzentrieren.

4.4 Benchmarking

Durch die Analyse der auf dem Markt vorhandenen Lösungen und Tools lassen sich empfehlenswerte Verfahren und gemeinsame Anforderungen ermitteln. Dies hilft bei der Entwicklung eines Testtools, das wettbewerbsfähig und auf die Bedürfnisse potenzieller Nutzergruppen abgestimmt ist. Dies und das Benchmarking können dazu beitragen, zukünftige Trends und Anforderungen zu antizipieren, um sicherzustellen, dass das Tool langfristig relevant bleibt. Diese Technik ermöglicht es, Anforderungen potenzieller Softwareanbieter zu erfassen, ohne dass direkte Workshops durchgeführt werden müssen.

Implementierung: Analyse der auf dem Markt vorhandenen Testwerkzeuge und Lösungen. Identifizierung von Best Practices, gemeinsamen Funktionen und Anforderungen durch Analyse von Marktberichten und Konkurrenzprodukten.

4.5 Prototyping

Prototyping ist eine passende Technik für komplexe Systeme wie Testtools für Automobilsteuergeräte. Beim Prototyping wird ein frühes Modell des endgültigen Systems erstellt, um dessen Funktionalitäten und Benutzerfreundlichkeit zu veranschaulichen und zu testen.

Zu den Low-Fidelity-Prototypen gehören hier digitale Darstellungen der Benutzeroberfläche und grundlegende Layoutmodelle, die die Struktur und die wichtigsten Komponenten ohne detaillierte Designelemente zeigen. High-Fidelity-Prototypen hingegen umfassen interaktive und funktionale Prototypen. Interaktive Prototypen sind detaillierte digitale Modelle, die die tatsächliche Benutzeroberfläche und Interaktionen simulieren und mit Tools wie Figma, Adobe XD oder Axure erstellt werden. Funktionsprototypen sind frühe Versionen der Software, die tatsächlichen Code und grundlegende Funktionen enthalten und das Testen bestimmter Funktionen oder Arbeitsabläufe in einer realen Umgebung ermöglichen.

Durch eine konkrete Systemdarstellung, die das Verständnis und die Überprüfung der Anforderungen erleichtert, gewährleistet das Prototyping eine frühzeitige Validierung. Durch visuelles Feedback verbessert es die Kommunikation und hilft den Beteiligten, eine gemeinsame Lösung zu finden. Durch die frühzeitige Problemerkennung und Bewertung der Machbarkeit senkt das Prototyping die Risiken. Es dient als wertvolles Hilfsmittel zur Erhebung detaillierter Anforderungen des Systems auf der Grundlage des Feedbacks der genannten Stakeholder.

Implementierung:

1. **Erster Low-Fidelity-Prototyp:** Wireframes für die Benutzeroberfläche des Tools erstellen, um die wichtigsten Bereiche wie die Erstellung von Testfällen, die Ausführung und die Berichterstattung darzustellen dann Feedback von

Testingenieuren zum Layout und den vorgeschlagenen Arbeitsabläufen einholen.

2. **Iterativer High-Fidelity-Prototyp:** Ein interaktiver Prototyp mit Tools wie Figma oder Adobe XD oder Streamlit Package python basiertes Tool entwickeln. Dabei die einfache Interaktionen für die Erstellung und Ausführung von Testfällen einbeziehen. Anschließend ein funktionaler Prototyp implementieren. Die Stakeholder können mit dem High-Fidelity-Prototyp interagieren und Aufgaben wie die Erstellung von Testfällen, die Ausführung von Tests und die Anzeige von Berichten durchführen.

5 Anforderungserhebung und Dokumentation

Damit etwas als Anforderung gelten kann, muss es folgende Qualitätskriterien haben

- Klar und verständlich für alle Beteiligten formuliert sein.
- Eindeutig sein, d.h., es darf keine Mehrdeutigkeiten in der Interpretation geben.
- Verfizierbar sein, sodass festgestellt werden kann, ob die Anforderung erfüllt wurde
- Relevant für das Projektziel sein und einen Wert für Stakeholder darstellen.

In diesem Kapitel werden unter Berücksichtigung der oben genannten Qualitätsmerkmale und unter Verwendung der Vorlage für die Satzschablone des Sophisten-Regelwerkes acht nach ihrer Art klassifizierte Anforderungen für das betrachtete Testtool dokumentiert.

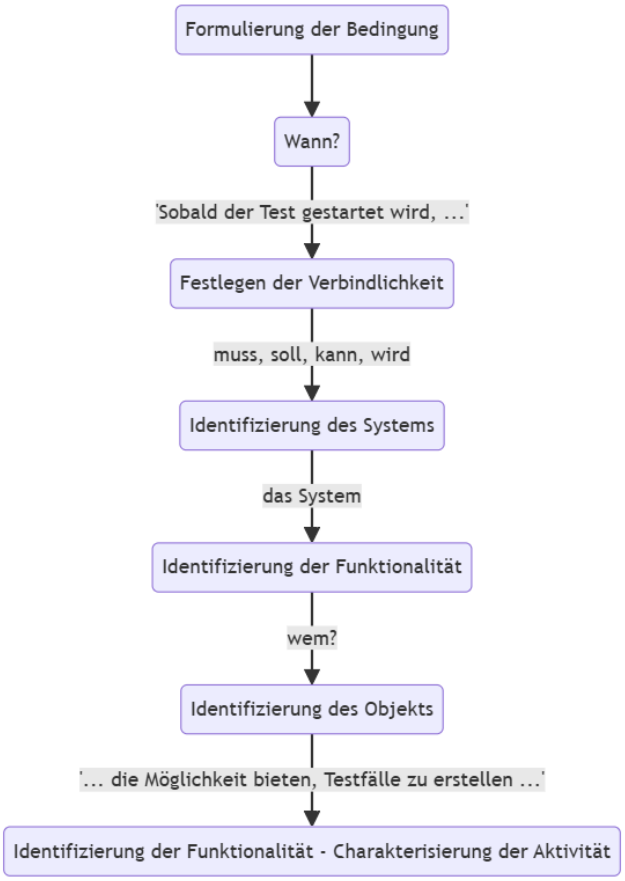


Abbildung 1 Visualisierung der Umsetzung von der Satzschablone des Sophisten-Regelwerkes

5.1 Funktionale Anforderungen

1. Erstellung von Testfällen

- **Beschreibung:** Das Testtool muss die Erstellung und Verwaltung von Testfällen ermöglichen.
- **Satzschablone:** Das System muss ermöglichen, dass Testingenieure Testfälle mit vordefinierten und benutzerdefinierten Parametern erstellen und verwalten können.

Dies umfasst die Möglichkeit, Precondition-, Action- und Post-Action-Blöcke zu definieren, um die Struktur der Testfälle und die Reihenfolge der zu bearbeitenden Schritte festzulegen.

2. Unterstützung verschiedener Testfalltypen

- **Beschreibung:** Das Testtool muss die Ausführung und Verwaltung verschiedener Arten von Testfällen unterstützen.
- **Satzschablone:** Das System muss ermöglichen, dass Testingenieure Testfälle für Diagnoseabfragen, PDU-Manipulationen, Kontrollen sowie Szenarienbasierte Testfälle mit Verbindung zu einer der folgenden Umgebungen :
    - MATLAB Simulink
    - dSPACE (Use Case: Scenario-Based Tests - dSPACE)
    - CANoe (ADAS Feature Set | Vector)

  erstellen und ausführen können.

3. Automatische Testausführung

- **Beschreibung:** Das Testtool muss die automatische Ausführung von Testfällen unterstützen.
- **Satzschablone:** Das System muss in der Lage sein, Testfälle nach der Erstellung und Aktivierung durch den Testingenieur mit nur einem Klick automatisch auszuführen. Dies soll die Testeffizienz erhöhen und menschliche Fehler minimieren.

4. Testberichtserstellung

- **Beschreibung:** Das Tool soll nach Beendigung einer Testserie ohne manuelle Eingriffe Berichte erstellen können. Mit Unterstützung gängige Formate wie PDF und HTML.
- **Satzschablone:** Das Testtool muss in der Lage sein, unmittelbar nach Abschluss der Testreihen automatisch detaillierte Berichte zu generieren. Diese Berichte müssen sowohl grafische Darstellungen der Testergebnisse als auch erweiterte Visualisierungen enthalten. Positive Ergebnisse sind in Grün, negative in Rot und unklar oder nicht ausgeführte Schritte in Gelb zu visualisieren. Zudem soll das Tool spezifische Diagramme für Antwortzeiten und PDU-Datenflüsse bereitstellen. Die Automatisierung muss eine direkte Erstellung von Berichten in standardisierten Formaten wie PDF und HTML ermöglichen, ohne dass nach Testabschluss manuelle Eingriffe notwendig sind.

5.2 Qualitätsanforderungen

1. Skalierbarkeit und Lastverteilung

- **Beschreibung:** Das Testtool muss in der Cloud skalierbar sein und eine effiziente Lastverteilung unterstützen.
- **Satzschablone:** Das System muss in der Lage sein, automatisch zusätzliche Ressourcen bereitzustellen und die Last gleichmäßig über alle verfügbaren Server zu

verteilen, um sicherzustellen, dass die Performance bei steigender Nutzerzahl und höheren Datenvolumen konstant bleibt. Die Skalierbarkeit muss mindestens 10.000 gleichzeitige Testausführungen unterstützen.

2. Fehlertoleranz

- **Beschreibung:** Das Testtool muss robust gegenüber Fehlern und Ausfällen sein.
- **Satzschablone:** Das System muss sicherstellen, dass bei einem Fehler während der Testausführung, wie z.B. Ressourcenausfall, API-Server nicht erreichbar oder Netzwerkfehler, automatisch bis zu drei Wiederholungsversuche durchgeführt werden und der Fehler detailliert in einem Logbuch dokumentiert wird, ohne den Testprozess abzubrechen.

5.3 Randbedingungen

1. Einhaltung der ISO 26262-Standards

- **Beschreibung:** Das Testtool muss die Anforderungen der funktionalen Sicherheit gemäß ISO 26262 erfüllen.
- **Satzschablone:** Das System muss Diagnosefunktionen unterstützen und Compliance-Berichten bereitstellen, die die Sicherheitsanforderungen und Testnachweise gemäß ISO 26262 dokumentieren.

2. Kompatibilität und Integration

- **Beschreibung:** Das Testtool muss mit bestehenden Systemen kompatibel sein und sich nahtlos integrieren lassen.
- **Satzschablone:** Das System muss mit den Entwicklungsumgebungen MATLAB Simulink, dSPACE und CANoe kompatibel sein. Zudem muss es sicherstellen, dass die Integration von APIs für die Kommunikation reibungslos funktioniert, um eine einfache Integration und Interoperabilität mit den Tools und Plattformen zu ermöglichen.

6 Semi-formale Anforderungsmodelle

In diesem Abschnitt werden drei semi-formalen Modellen für das betrachtete System erstellt.

6.1 Use Case Diagramm für die Funktion der Testfallerstellung

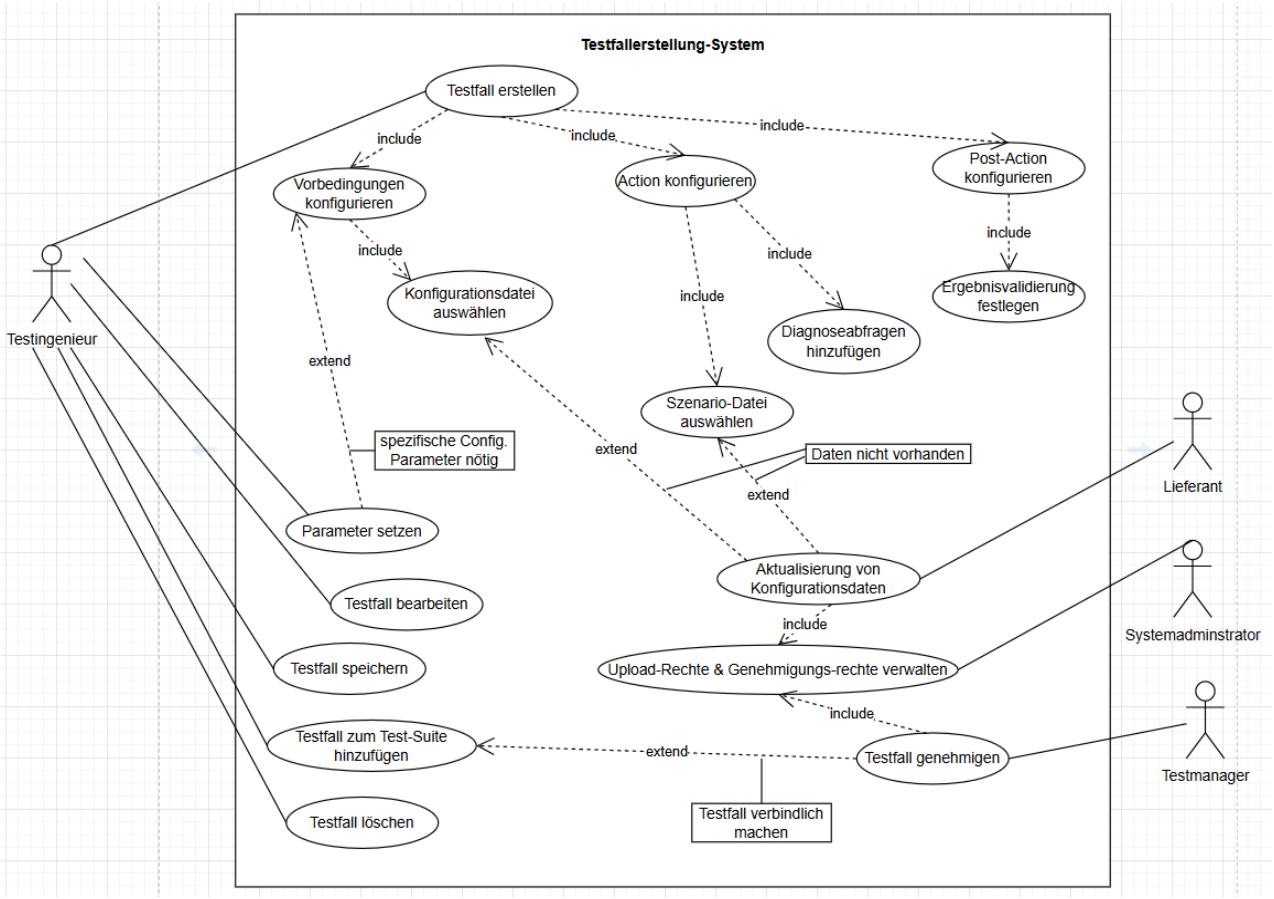


Abbildung 2 Use Case Testfallerstellung

Das Use-Case-Diagramm skizziert die Testfallverwaltungsprozesse innerhalb eines Software-Testsystems und betont die Rollen der verschiedenen Akteure, darunter ein Testingenieur, ein Testmanager, ein Lieferant und ein Systemadministrator. Im Mittelpunkt des Diagramms stehen die primären Anwendungsfälle wie das Erstellen, Bearbeiten, Speichern, Löschen und Genehmigen von Testfällen, die mit „include“-Beziehungen für wichtige Aktionen wie das Konfigurieren von Aktionen und Vorbedingungen ausgearbeitet werden. Zusätzlich verwendet das Diagramm „extend“-Beziehungen, die unter bestimmten Bedingungen optionale Funktionalitäten einführen, wie z. B. die Anpassung von Parametern während der Testfallbearbeitung und die Genehmigung von Testfällen durch den Testmanager wenn ein Testfall verbindlich in dem Testkatalog oder den offiziellen Testsuite hinzugefügt werden soll.

6.2 Aktivitätsdiagramm für die automatische Testausführung

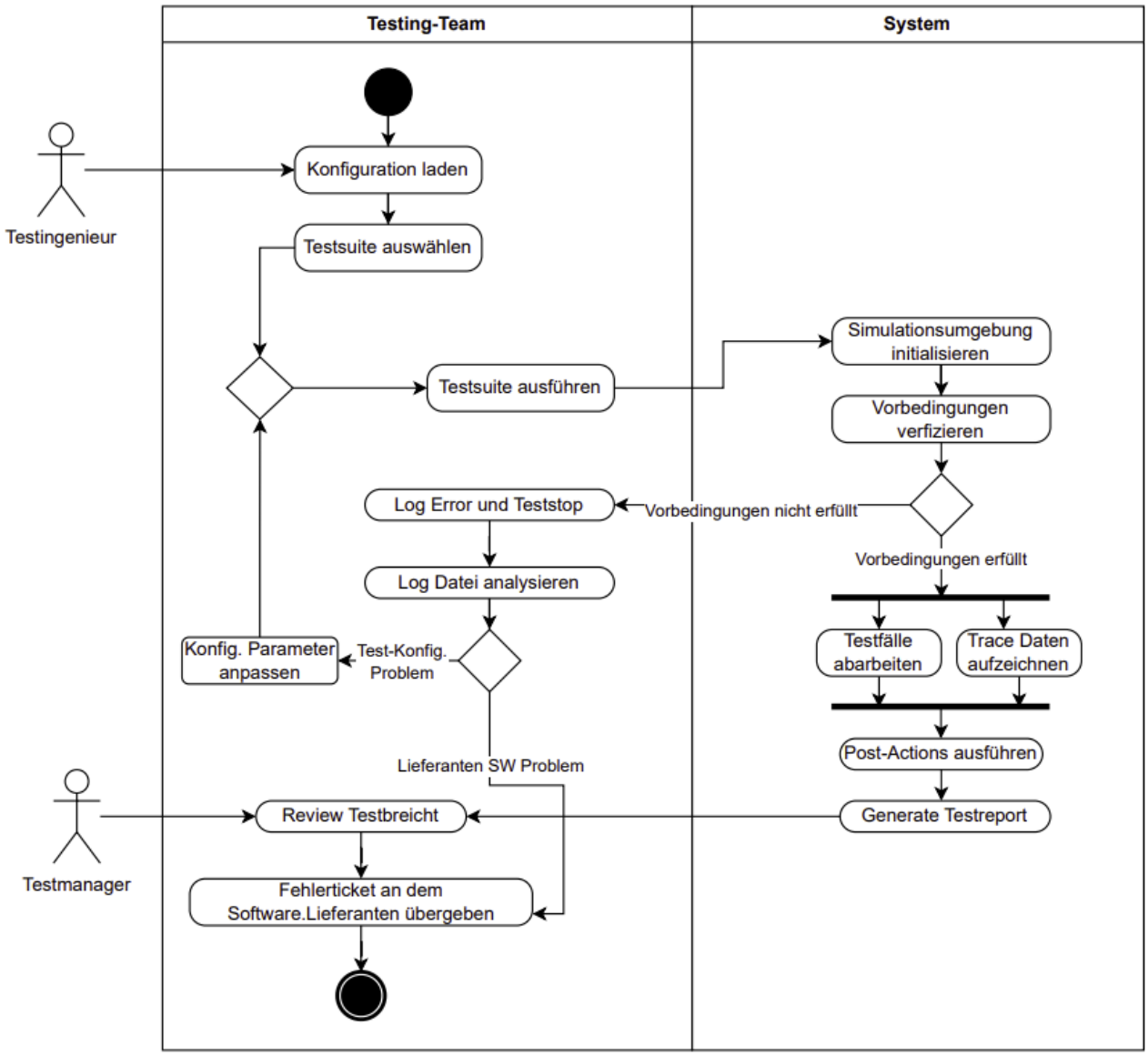
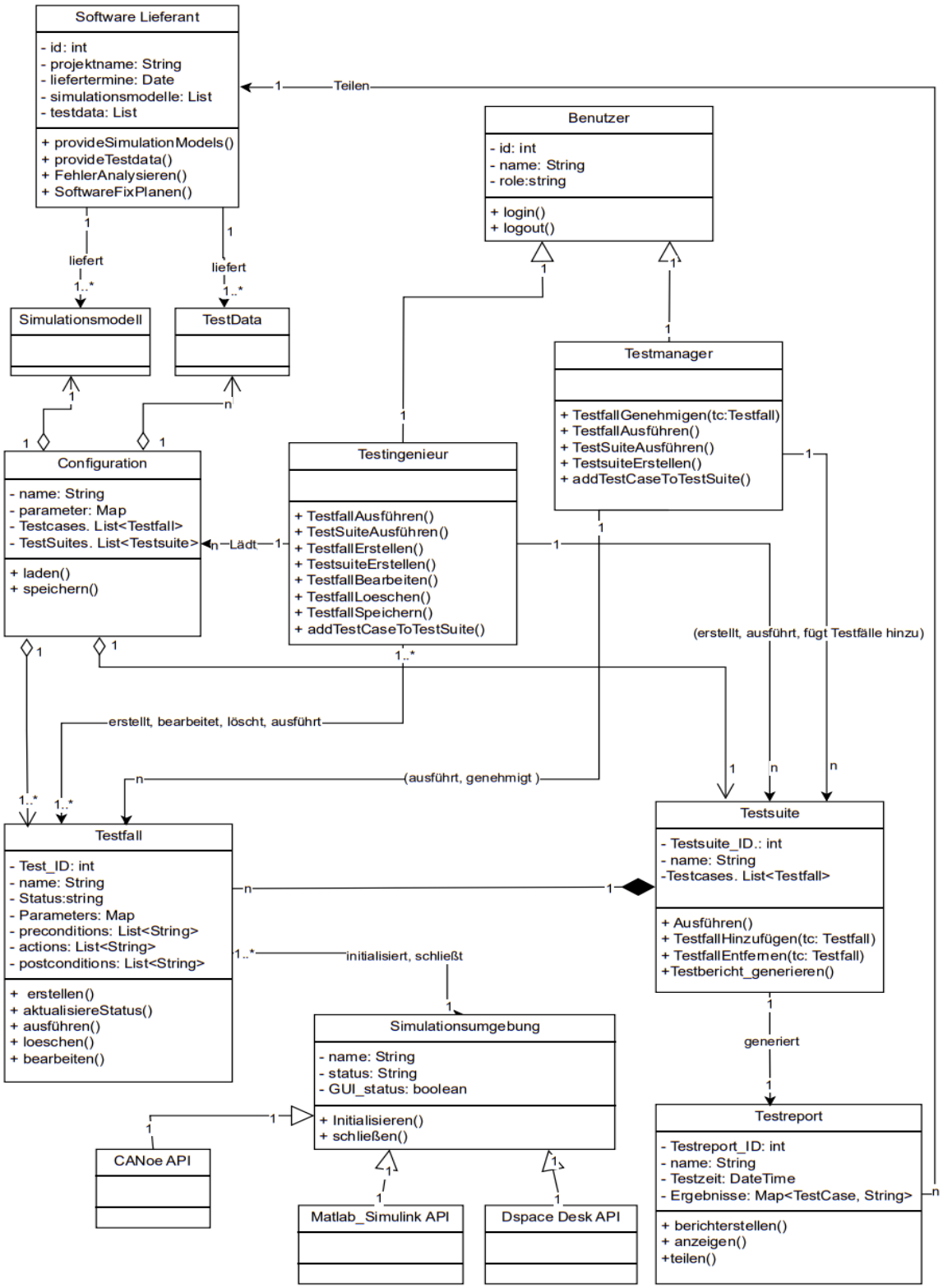


Abbildung 3 Aktivität Automatisierte Testausführung

Das Diagramm skizziert einen automatisierten Testprozess, der mit dem Laden der Konfigurationen beginnt, gefolgt vom Auswahl einer Testsuite. Im weiteren Verlauf des Prozesses werden verschiedene Aufgaben ausgeführt, z. B. die Ausführung der Testsuite, die Analyse der Protokolldateien im Falle eines Fehlers nach dem Entscheidungspunkt, an dem entschieden wird, ob die Vorbedingungen der Testfällen erfüllt sind. Hier sind möglicherweise Anpassungen der Konfigurationen erforderlich. Der Arbeitsablauf setzt sich mit einer möglichen Fehlerprotokollierung und der Einbeziehung des Softwareanbieters fort.

6.3 Klassendiagramm für die Struktur des Systems



Das UML-Klassendiagramm veranschaulicht, wie Testfälle und Testsuiten verwaltet werden, und spiegelt die Architektur des Softwaretestsystems wider. Zentraler Bestandteil des Diagramms ist die Klasse Testfall, die neben Merkmalen wie Test_ID, Name und Status auch Methoden zur Erstellung, Aktualisierung und Ausführung besitzt. Diese Klasse steht in

einer Kompositionsbeziehung zur Klasse der Testsuite, was bedeutet, dass jede Testsuite über zahlreiche Instanzen der Testfälle verfügt, was auf die Zugehörigkeit hinweist. Außerdem ist Testreport über eine Aggregatbeziehung mit der Klasse Testsuite verbunden, was bedeutet, dass Testreports, die die Ergebnisse von Testsuiten aufzeichnen, nicht ausschließlich auf die Existenz der Testsuite beschränkt sind.

Der Testsuite und weiter vom Software Lieferanten gelieferten Testdaten und Simulationsmodell sind ein Aggregat der Konfiguration, die am Ende durch den Testingenieur geladen wird. Eine Konffiguration kann mehrere Testsuites enthalten und der Nutzer kann nach dem Laden der Konfiguration ein Testsuite auswählen und ausführen.

Weitere Verbindungen sind die wiederholte Verwendung von APIs wie Matlab_Simulink API, CANoe API und Dspace Desk API, die mit anderen Systemkomponenten verknüpft sind und auf einen Bedarf an Test- und Simulationsumgebungen hindeuten. Darüber hinaus zeigt das Diagramm Akteure, die für den Testprozess verantwortlich sind, wie den Testmanager und den Testingenieur. Beide sind Benutzer des Systems und rufen Methoden wie testCaseCreate und testCaseApprove (Testmanager) auf.

7 Priorisierung und Validierung der Anforderungen

7.1 Auswahl und Überprüfung der Anforderungen

Die AHP Priorisierung 4 Anforderungen wird sich auf folgende Kriterien anlehnen:

- Bedeutung für die Stakeholder
- Fehlerreduzierung
- Notwendigkeit für die Funktionalität
- Auswirkungen auf die Effizienz

Daher wird bei der Analyse der vier Produktanforderungen die Frage gestellt, inwiefern sind die Produktanforderungen für diese betrachteten Kriterien relevant.

Folgende Produktanforderungen werden für die AHP-Priorisierung ausgewählt:

1. **Automatische Testausführung:** Erhöht die Effizienz und reduziert menschliche Fehler, daher hohe Bedeutung für die Testproduktivität.
2. **Testberichtserstellung:** Wichtig für die Nachvollziehbarkeit und Dokumentation der Testergebnisse, wesentlich für die Qualitätssicherung (Qualitätsanforderung).
3. **Skalierbarkeit und Lastverteilung:** vor allem, wenn das System eine große Anzahl von gleichzeitigen Nutzern oder Tests bewältigen soll. Die Skalierbarkeit stellt sicher, dass das System wachsen und höhere Lasten ohne Beeinträchtigung der Leistung bewältigen kann, was für den langfristigen Erfolg und die Zuverlässigkeit des Systems entscheidend ist.
4. **Fehlertoleranz:** Essenziell für den kontinuierlichen Betrieb und zur Vermeidung von Testunterbrechungen (Basisanforderung).

Die vier ausgewählten Anforderungen sind allgemein für die Projektziele und für die Wünsche der Stakeholder relevant.

7.2 Analytische Priorisierung mit AHP-Verfahren

(1) Vergleich der Anforderungen zur Ermittlung des relativen Nutzens:

Anforderungen	1.Automatische Testausführung	2.Testberichtserstellung	3.Skalierbarkeit	4.Fehlertoleranz
1.Automatische Testausführung	1	1/3	1/2	1/5
2.Testberichtserstellung	3	1	3	1/3
3.Skalierbarkeit	2	1/3	1	1/5
4.Fehlertoleranz	5	3	5	1

Begründung:

- **Fehlertoleranz vs. Automatische Testausführung:** Die automatisierte Testdurchführung wird als weniger wichtig erachtet als die Fehlertoleranz. Die Gewährleistung, dass das System mit Fehlern umgehen und weiterarbeiten kann, hat höhere Priorität als die durch die Automatisierung gewonnene Effizienz.
- **Fehlertoleranz vs. Testberichtserstellung:** Fehlertoleranz wird als dreimal so wichtig erachtet wie die Testberichterstattung. Obwohl die Berichterstattung wichtig ist, stellt die Fähigkeit des Systems, Fehler zu tolerieren und sich von ihnen zu erholen, sicher, dass es weiterhin korrekt funktioniert, was für die Schaffung von Vertrauen und Zuverlässigkeit wesentlich ist.
- **Fehlertoleranz vs. Skalierbarkeit:** Fehlertoleranz wird als fünfmal so wichtig erachtet wie Skalierbarkeit. Die Sicherstellung, dass das System trotz Fehlern oder Ausfällen betriebsbereit bleibt, hat höhere Priorität als reine Skalierung des Systems
- **Skalierbarkeit vs. Automatische Testausführung:** Die Skalierbarkeit wird als doppelt so wichtig wie die automatisierte Ausführung angesehen, da die automatisierte Ausführung zwar die Effizienz verbessert, die Skalierbarkeit aber sicherstellt, dass das System eine höhere Last bewältigen kann, was für die Leistung bei hohen Belastungen entscheidend ist.
- **Skalierbarkeit vs. Testberichtserstellung:** Die Skalierbarkeit wird als weniger wichtig angesehen als die Testberichterstattung, da eine effektive Testberichterstattung entscheidend für das Verständnis der Testergebnisse und die Sicherstellung der Qualität ist. Sie kann auch für die Stakeholder wichtig sein, um die Leistung des Systems zu verstehen.
- **Automatisierte Testdurchführung vs. Testberichtserstellung:** Während die Automatisierung zu einer effizienten Testausführung beiträgt, liefern erstellte Testberichte die notwendigen Erkenntnisse und Dokumentationen.

(2) Prioritäten berechnen in 2 Schritte:

1. Normalisierung der Vergleichsmatrix (Division jeder Zelle durch die Summe der zugehörigen Spalte)
2. Summe der Anteile jeder Anforderung (jede Zeile) und Durschnitt berechnen

Anforderungen	1.Automatische Testausführung (11)	2.Testberichterstellung (4,67)	3.Skalierbarkeit (9,5)	4.Fehlertoleranz (1,73)	Priorität (Durschnitt)
1.Automatische Testausführung	0,091	0,071	0,053	0,058	(0.091+0.071+0.053+0.058)/4=0.068
2.Testberichterstellung	0,273	0,214	0,316	0,192	(0.273+0.214+0.316+0.192)/4=0.249
3.Skalierbarkeit	0,182	0,071	0,105	0,058	(0.182+0.071+0.105+0.058)/4=0.104
4.Fehlertoleranz	0,455	0,643	0,526	0,692	(0.455+0.643+0.526+0.692)/4=0.579

(3) Erkenntnisse:

Fehlertoleranz hat die oberste Priorität, wobei die Notwendigkeit eines robusten Systems betont wird, das in der Lage ist, Ausfällen zu widerstehen und die Funktionalität sicherzustellen. Die Berichterstellung für Tests ist entscheidend für die Gewährleistung der Rückverfolgbarkeit während des gesamten Testing-Prozesses. Skalierbarkeit und Lastverteilung sind für die Bewältigung von erhöhte Anfrage oder Belastung in Software-Lieferung-Phasen und die Sicherstellung einer hohen Leistung bei erhöhter Last. Die automatisierte Testausführung ist zwar wichtig, wird aber in diesem Zusammenhang als weniger kritisch angesehen als Skalierbarkeit und Fehlertoleranz.

A