Fachhochschule der Wirtschaft
FHDW
Bielefeld

Praxisarbeit

Thema:

Vergleich logischer Konzepte zur Testautomation: Capture Replay und Keyword Driven

Prüfer:
Dr. Yvonne Gorniak

Verfasser:
Christopher Martin Müller
Helbingstraße 84
45128 Essen

Bachelor: Wirtschaftsinformatik
Software Engineering
100590

Eingereicht am:
10.03.2022

Inhaltsverzeichnis

Abbildungsverzeichnis

Seite

Tabellenverzeichnis

Seite

Abkürzungsverzeichnis

Rola	Rola Security Solutions GmbH
rsClient	rolasecurityClient (Software)
GUI	graphical user interface
SUT	System under test
et al.	et alii (und andere)
KI	Künstliche Intelligenz
BS Konzept	„Bausteine“ Konzept

1 Einleitung

1.1 Problemstellung

Das Qualitätsmanagement und die dazugehörige Testautomatisierung sind ein wichtiger Grundbaustein jedes Unternehmens. Eine erfolgreiche Testphase einer Software ist von hoher Wichtigkeit und oft der letzte Schritt vor einer Veröffentlichung. Doch das Testen einer Software, ob manuell oder automatisiert bedarf hoher Kosten.

Raffi Margaliot und Micro Focus Senior Vice President und General Manager für Application Delivery Management geben beim World Quality Report 2019 an: „Testing und Automatisierung sind kritische Faktoren für eine zuverlässige und sichere Software Delivery, aber die Teams kämpfen immer noch mit den damit verbundenen Kosten und Komplexitäten.“[1]

Auf dem Markt gibt es einige Test-Softwares und oftmals wissen Unternehmen nicht, welches System passend zum eigenen Bedarf ist. Jedes Konzept und jede Software bringt Vor- und Nachteile mit sich, welche die Kosten und den Aufwand senken oder gegebenenfalls durch eine falsche Verwendung erhöhen können.

Im World Quality Report (WQR) aus dem Jahre 2019 heißt es: „79 Prozent der deutschen Befragten melden einen Anstieg des proportionalen Aufwands und der Ausgaben für QS- und Testaktivitäten in den vergangenen drei Jahren. In den kommenden drei Jahren gehen sie davon aus, dass der Anteil der Tests an ihrem gesamten IT-Budget 31 Prozent betragen wird.“[2]

Der Trend zeigt, dass erfolgreiches Testen und die Testautomatisierung mittlerweile wichtige Bestandteile im Qualitätsmanagement geworden sind. Durch eine generisch reproduzierbare Testautomatisierung mittels Robotik und selbstausführbaren Tests würde der Faktor Zeit, welcher das Testen in Anspruch nimmt, nicht im Widerspruch mit jeglichem Budget stehen.

---

[1] it-Daily.net - Fachmagazin – Ein Drittel der deutschen IT-Ausgaben wird in Testing investiert. URL: https://www.it-daily.net/analysen/22712-ein-drittel-der-deutschen-it-ausgaben-wird-in-testing-investiert - zuletzt aufgerufen am 12.01.2022

[2] It-Daily.net - Fachmagazin – Ein Drittel der deutschen IT-Ausgaben wird in Testing investiert. URL: https://www.it-daily.net/analysen/22712-ein-drittel-der-deutschen-it-ausgaben-wird-in-testing-investiert - zuletzt aufgerufen am 12.01.2022

Durch die Testautomatisierung kann die manuelle Testung ergänzt und sogar komplett abgeschafft werden. „Wenn man 1000 Kombinationen von Testfällen manuell testen will, ist man damit ein paar Wochen beschäftigt. Ein Roboter, der dafür ein Skript abarbeitet, schafft das in wenigen Stunden.“ Durch generisch reproduzierbare Testfälle, welche sich in Testphasen wiederholen, kann deutlich an den Faktoren Zeit und Budget gespart werden.

Die Frage, die sich viele Unternehmen stellen müssen, ist welche Software und welches Konzept sich am besten eignet, um die Effizienz innerhalb der eigenen Qualitätssicherung mit Hinblick auf den Trend und die Zukunft der Softwareentwicklung zu steigern.

1.2 Zielsetzung und Vorgehensweise

Das Ziel der vorliegenden Praxisarbeit ist, eine wissensbasierte und fundierte Gegenüberstellung der Konzepte Capture Replay und Keyword Driven in Hinsicht auf eine erfolgreiche Testautomatisierung.

Vorab wird im ersten Kapitel auf die Definitionen eingegangen, um eine Grundlage aufzubauen und um die Konzepte besser erörtern zu können. Diese Definitionen unterteilen sich in eine klare Definition und dann in eine Grundlage des Konzepts, um im weiteren Vorgehen der Praxisarbeit den Vergleich dieser besser einordnen zu können.

Nachfolgend wird nach einem Grundlagenaufbau der Vergleich der beiden logischen Konzepte zur Testautomatisierung gezogen. Dabei wird vorab eine Rekapitulation beider Konzepte gemacht, um daraufhin einen direkten Vergleich zu ziehen und etwaige Gemeinsamkeiten und oder Differenzen zu finden und zu erörtern. Im weiteren Verlauf wird dann eine eventuelle Nutzungsempfehlung ausgesprochen, welches Konzept in welchem Fall besser passt.

Im vierten Teil, dem Praxisteil dieser Praxisarbeit, wird erstmals das Praxisunternehmen, die Rola Security Solutions GmbH vorgestellt. Des Weiteren wird die Software Tosca als Test-Automation Software vorgestellt. Nach dieser Vorstellung wird eine Ist-Analyse zu der damaligen Testautomatisierung der Qualitätssicherung der Rola Security Solutions GmbH erstellt. Und anschließend wird das Soll-Konzept vorgestellt, welches zurzeit in der Rola Security Solutions GmbH verwendet wird. Hier wird das Soll-Konzept dann nochmal im Detail erklärt und die Möglichkeiten beziehungsweise die Grenzen dieses Konzeptes werden aufgezeigt. Abschließend wird in diesem Teil dann

eine Handlungsempfehlung ausgesprochen, welche das ganze zusammenfassend beendet.

Abschließend wird in der Schlussbetrachtung resümierend ein Ergebnis gezogen und ein eventueller Ausblick gezeigt, welcher die Praxisarbeit abschließt.

2 Grundlagen

2.1 Konzept: Capture Replay

2.1.1 Definition

In der Testautomatisierung werden viele Techniken für eine erfolgreiche Hilfestellung des Testens verwendet. Neben manuellen Testtools werden aber auch Konzepte für die automatisierte Testung verwendet.

Im Graphical user interface (GUI) Testing gibt es neben Keyword Driven auch Modellbasierte und Skriptbasierte Konzepte. Nachfolgend geht es um das Konzept des Capture & Replay.

Dieses Konzept setzt sich zusammen aus dem Englischen „Capture“ für Aufnehmen und „Replay“ für Wiederholung. Mit diesem Konzept ist gemeint, dass der Tester mithilfe eines Tools die Anwendung testet und diese Ausführung aufzeichnet, um diese dann wiederholen zu können.

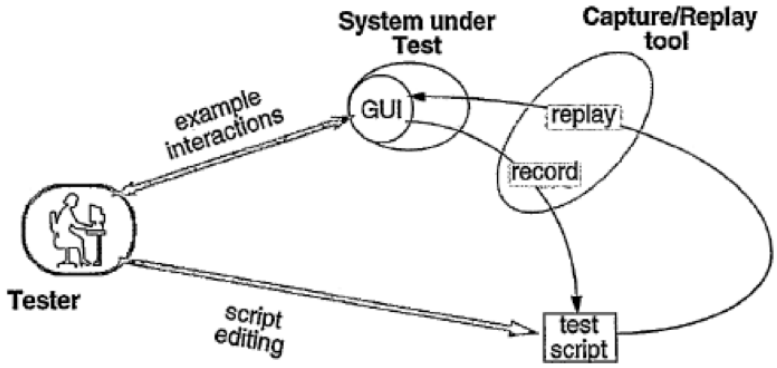
„Dazu sind Testwerkzeuge notwendig, die sowohl die Handlungen des Testers aufzeichnen als auch diese danach an der Schnittstelle des Testobjekts wieder reproduzieren können. Dieser Anspruch an das Testwerkzeug steigt mit der Komplexität des Testobjekts und kann im Extremfall bis zum Einsatz von Testrobotern führen, die z.B. Hardware-Komponenten hinzufügen, ändern oder entfernen.“[3]

2.1.2 Grundlage des Konzepts

Im Konzept „Capture and Replay“ geht es um die Möglichkeit der Aufzeichnung und Protokollierung der einzelnen Schritte des Testers. „Dabei protokolliert das Capture & Replay-Tool alle Ereignisse wie Tastatur- und Mausevents und notiert sie in einem Testskript“[4]

[3] Dr. Stephan Weissleder Richard Seidl (2016) S. 6.
[4] Antonia Kresse (2016) S. 14

Abbildung 1: Konzeptzeichnung Capture & Replay



Quelle 1: Abbildung aus Ostrand (1998)

Dieses Testskript kann der jeweilige Tester anschließend abrufen und es wiederholt ausführen. Dieser Teil des Konzepts nennt sich Replay.

Das Konzept des Capture und Replay zeichnet sich durch einen Hauptakteur aus, das ist der Tester der SUT. Dabei agiert dieser durch wiederholt vorher geplante Interaktionen in einem Test Skript mit der zu testenden GUI im zu testenden SUT. Diese Interaktionen werden durch ein Capture & Replay Tool aufgezeichnet und dann für eine eventuelle Nachprüfung gespeichert, um diese dann wiederholen zu können.

Die Aufzeichnung der Interkationen kann durch mehrere Möglichkeiten getätigt werden. „Zwei Möglichkeiten hierzu sind, die absoluten Koordinaten aller Mausklicks oder die Objekt-IDs (sofern vorhanden) der angeklickten Elemente zu speichern. Letzteres hat den Vorteil, dass die Neuanordnung der Oberflächenelemente nicht direkt das erneute Aufzeichnen der Testfälle bedingt. Einer der bekanntesten Vertreter für „Capture & Replay“-Tests von Web-Anwendungen ist Selenium, dass die Aufzeichnung z.B. mit dem entsprechenden Plug-In für den Browser Firefox ermöglicht und die Wiedergabe dieser Aufzeichnungen auf vielen der am weitesten verbreiteten Browser erlaubt.“[5]

>

[5] Dr. Stephan Weissleder Richard Seidl (2016) S. 6.

2.2 Konzept: Keyword Driven

2.2.1 Definition

Mit dem Konzept und Begriff Keyword Driven gemeint, ist eine schlüsselwortgetriebene Testung gemeint. Es werden Schlüsselwörter oder auch Kommandos genutzt, um durch ein Testtool das zu testende SUT innerhalb einer GUI zu testen. „Beim schlüsselwortgetriebenen Testentwurf werden anstelle konkreter Befehle abstrakte Schlüsselwörter verwendet, die zur Laufzeit je nach Konfiguration in einer Adaptionsschicht durch konkrete Befehle ersetzt werden.“[6]

Doch bevor man das Konzept und seine Grundlage verstehen kann, sollte man den Begriff Keyword-Driven Testing an sich verstehen.

„[...]Die zugrunde liegende Idee von Keyword-Driven Testing (KDT) wird gerne über eine Analogie mit Bausteinen erläutert.“. „[...]So kann ein Tester aus einer begrenzten Zahl an Keywords all die Testfälle zusammensetzen, die benötigt werden.“[7]

2.2.2 Grundlage des Konzepts

Das Handling im Konzept der schlüsselwortgetriebenen Testung ist vorab etwas komplizierter als das Konzept des Capture & Replay. Denn bei dem Konzept müssen vorab Kommandos festgelegt werden, die dann von einem Tester benutzt werden können. Konkret heißt das, dass beim schlüsselwortgesteuerten Testen die Implementierung der Testautomatisierung vom Testfallentwurf getrennt werden. Für jede Aktion im Testfall wird ein Schlüsselwort kreiert. Sobald das Schlüsselwort festgelegt ist, werden keine Programmierkenntnisse mehr benötigt, um die automatisierten Tests zu entwerfen und durchzuführen.[8]

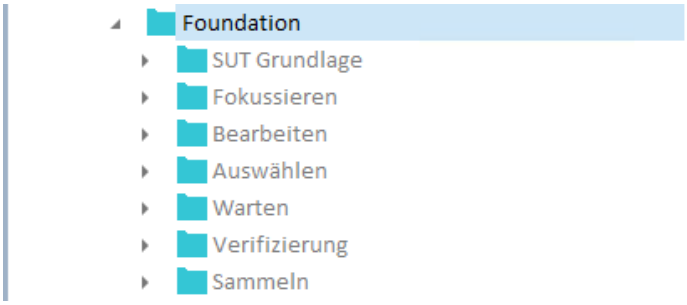
[6] Dr. Stephan Weissleder Richard Seidl (2016) S. 11

[7] Matthias Daigl, Rolf Glunz (2016) S.3

[8] Vgl. Tobias Walter (2016):

Wie in der Definition in 2.2.1 erläutert wird oft mit einer Analogie zu Bausteinen das Konzept des Keyword Driven Testing (KD) erklärt. Der Tester braucht also nicht nur die aus einer begrenzten Anzahl an Keywords festgelegten Befehle, sondern braucht ebenfalls eine Art von Bibliothek in welcher sich die entweder nichtvorgefertigten oder auch fertigen „Bausteine“ / Keywords oder gar ganze Testschritte befinden.

Abbildung 2: Screenshot Tosca - Eine grobe Testschritt Bibliothek



Quelle 2: Eigene Darstellung

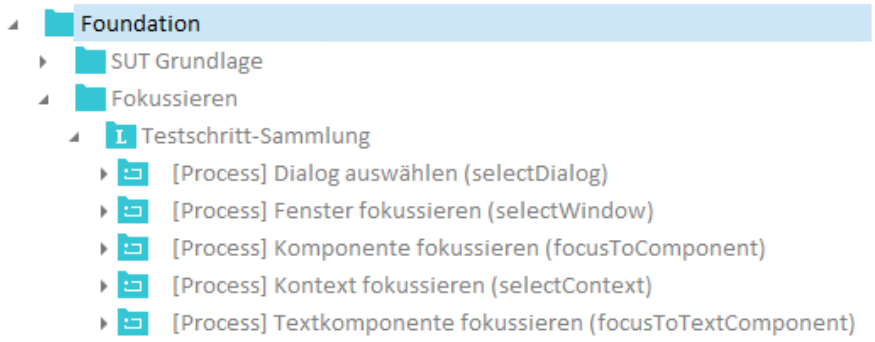
Wie der Abbildung 2 zu entnehmen ist, werden solche Keywords / „Bausteine“ in Testschritt Sammlungen oder auch Testschritt Bibliotheken gesammelt. Diese werden kategorisiert durch etwaige Namen, welche die Schritte ungefähr beschreiben.

Weiter geht es bei dem Konzept des Keyword Driven Testing. Hier geht es nämlich um eine gewisse generische Zusammensetzung der Tests. „Worum es nämlich geht, ist eine Vereinheitlichung (Normierung) der Testfälle, und es wird eine für alle Beteiligten gemeinsame Sprache, maßgeschneidert für genau den jeweiligen Anwendungsfall, geschaffen.“[9]

Abbildung 3 zeigt, wie eine Keyword Sammlung aussehen kann.

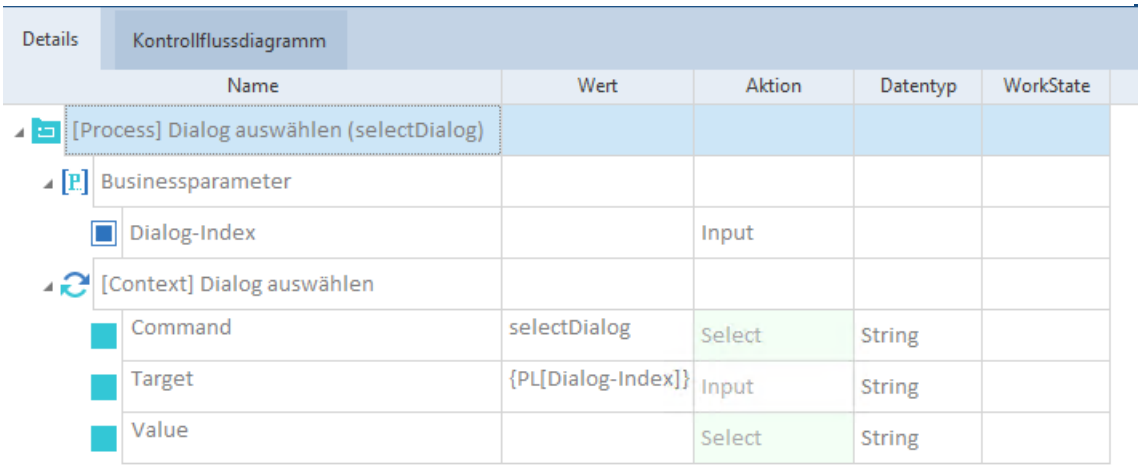
[9] Matthias Daigl, Rolf Glunz (2016) S.3

Abbildung 3: Screenshot Tosca - Eine Testschritt / Keyword Sammlung im Detail



Quelle 3: Eigene Darstellung

Abbildung 4: Screenshot Tosca - Ein Beispiel Keyword im Detail



Quelle 4: Eigene Darstellung

Ebenso zeigt Abbildung 3 die Struktur der Sammlungen der Befehle. Kommandos, welche der Gruppe „fokussieren“ zugeteilt werden können, werden entsprechend gruppiert und können somit schnell wiedergefunden werden.

Im Beispiel des Befehls „Dialog auswählen“ welcher den technischen Namen „selectDialog“ trägt, setzt sich dieser aus dem Kommando selbst und dem Target – dem Dialog Index zusammen. Der Dialog Index wird, damit dieser generisch verändert werden

kann, durch einen Business Parameter ersetzt welcher als Platzhalter dient, um im späteren Verlauf den Index über den Parameter weiterzugeben.

3 Vergleich Capture Replay & Keyword Driven anhand der Tools Ranorex und Selenium

Im nachfolgenden Kapitel wird ein wissensfundierter Vergleich der logischen Konzepte Capture Replay und Keyword Driven Testing gezogen. Kapitel 2 stellte beide Konzepte vor und nachfolgend werden die Gemeinsamkeiten und Differenzen dieser aufgezeigt und analysiert. Dabei werden diese anhand der Tools Ranorex für Capture & Replay und Selenium / Swing Libary für Keyword Driven Testing verglichen.

3.1 Gemeinsamkeiten

Neben dem eigentlichen Sinn des Testens ermöglichen beide logischen Konzepte, Capture Replay und Keyword Driven Testing eine meist vollständige und einfach automatische Testung.

Wie Tabelle 1 zeigt, gibt es einige Hauptaspekte in der Gemeinsamkeit von Ranorex und Selenium also Capture & Replay (C&P) und Keyword Driven Testing (KDT) Software. Bevor es zu einer Automatisierung der Testfälle kommt, ist eine klare Voraussetzung beider Konzepte Teamarbeit und grobe Vorarbeit. Es muss zusammen gearbeitet werden aufgrund der vorab hohen Arbeit von KD und C&P. Beide brauchen eine nötige Einarbeitung der Tester und eine Grundlage auf der getestet werden kann.

Die große Gemeinsamkeit beider Konzepte ist das die Programme auf mehrere Plattformen und Betriebssystemen und oder Browsern laufen. Das bedeutet Software, egal ob eine Java-Grundlagen (Java ist eine Programmiersprache) Software oder eine JavaScript betriebene Software (JavaScript ist eine Programmiersprache) kann die Software durch beide Konzepte getestet und automatisiert werden. Darüber hinaus sind die Kosten der beiden Programme und Konzepte sehr gering, es wird lediglich eine Lizenz benötigt. Diese kann entweder pro Nutzer oder pro Abteilung abgerechnet werden, das hängt vom Programm ab. So kostet Ranorex für ein ganzes Team inklusive jeglicher Funktionen z.B. 5.750, - EUR und ist eine unbefristete Lizenz also ein einmaliger Kauf.[10]

[10] Ranorex.com – Pricing / Preise Ranorex – URL: https://www.ranorex.com/de/preise/ - zuletzt abgerufen am 05.03.2022.

Ein wesentlicher wichtiger Aspekt ist die generische Grundlage der Konzepte und Software. Für die Automatisierung von Testfällen ist für lange Sicht eine generische Struktur wertvoller, darüber verfügen beide. In diesem Punkt ist Selenium mit Keyword Driven Testing weiter als Ranorex, hier sind die Funktionen wesentlicher ausführlicher und komplexer, was einer generischen Automatisierung nicht fehlen darf.

Tabelle 1: Gemeinsamkeiten: Ranorex und Selenium

Ranorex	Selenium
Beide Konzepte und Programme können auf mehreren Plattformen wie Betriebssystemen, Browsern usw. laufen.	
Die Kosten sind ähnlich niedrig. Es ist eine Lizenz notwendig.	
Das Handling ist teilweise ähnlich, die Handhabung ist bei beiden Konzepten relativ trivial.	
Einfache Aufzeichnung bei Tests und Unterstützung der Wiedergabe von Aufzeichnungen.	
Generische Grundlagen sind bei beiden vorhanden. Selenium ist weitaus ausführlicher und komplexer.	

Quelle 5: Eigene Darstellung nach Dheanda Absharina et. al. (2020)

3.2 Differenzen

Die Differenzen der Konzepte und ihrer Tools Ranorex und Selenium sind ungemein groß und werden durch Unternehmen oftmals falsch eingeschätzt.

Die Konzepte für die Testautomatisierung, ob Capture und Replay oder Keyword Driven Testing sind effiziente Konzepte, welche alltäglich in der Qualitätssicherung benutzt werden. Doch leider werden entweder beide Konzepte zusammen genutzt oder falsch eingesetzt. Die Differenzen, obwohl beide sehr häufig gleichzeitig genutzt werden, sind groß und vor allem im Detail zu erkennen.

Tabelle 2: Differenzen: Ranorex und Selenium

Ranorex	Selenium

Sowohl kodierungsfreies Arbeiten als auch Kodierungsmöglichkeiten.	Für die Verwendung von Selenium sind Programmierkenntnisse und ein gewisses Maß an Anwenderwissen erforderlich.
Unterstützt Windows-Plattform.	Unterstützung der Plattformen PC / MAC / UNIX / Windows / LINUX.
Grafische Benutzeroberfläche ist eher minimalistisch. Dafür aber einfach zu verstehen.	Bietet eine erweiterte grafische Benutzeroberfläche zur Aufzeichnung von Benutzeraktionen
unterstützt die Zusammenfassung mit Diagrammen für schnellere und bessere Fehlervergleiche für jeden Prozess.	bietet eine grafische und eine Kommandozeilenausgabe.
Unterstützt die Wiedergabe/Wiederholung von Aufzeichnungen.	Unterstützung der Wiederverwendung von Code.

Quelle 6: Eigene Darstellung nach Dheanda Absharina et. al. (2020)

Der größte Unterschied den Ranorex und Selenium trennt ist, dass Ranorex zwar Kodierungsmöglichkeiten bietet aber trotz dessen weniger, bis keine Programmierkenntnisse braucht. Im Gegenzug dazu benötigt man für die Verwendung von Selenium klare Programmierkenntnisse und vor allem ein gewisses Maß an Anwenderwissen.[11]

Ebenfalls ist eine Unterstützung der Konzepte unter Plattformen leider nicht gleichwertig, so unterstützt Ranorex nur die Testung unter Windows – Selenium aber auch über Mac, Unix und Linux. Eine einfache Handhabung ist je nach Unternehmen dann schwierig, wenn zum Beispiel firmeninterne Rechner nur MacOs als Betriebssystem nutzen. Das spiegelt sich auch in der graphischen Oberfläche beider Programme dar. Wo Selenium eine erweiterte grafische Oberfläche zur Aufzeichnung anbietet, bleibt Ranorex eher minimalistisch. Der Vorteil bei Ranorex liegt auf der Hand, die Tester sind so überaus schneller mit dem Handling bekannt, die Oberfläche bleibt trivial.

[11] Vgl. Dheanda Absharina et. al. (2020) S.178

3.3 Gegenüberstellung der Konzepte

3.3.1 Vor- und Nachteile

Bei der Gegenüberstellung der Konzepte von Keyword Driven Testing und Capture & Replay Testing geht es in der Qualitätssicherung um den wiederkehrenden Kostenfaktor durch nachträgliche Adaption und die initialen Faktoren wie die Einrichtung und die Kosten. Dabei sind die Kosten sehr oft korrelativ mit der Einrichtung und der nachträglichen Adaption.

Mit dem Blick auf das Konzept des Keyword Driven Testing also der Schlüsselgetriebenen Testung ist der Vorteil der „höhere Abstraktionsebene, die das leichte Erstellen und die vereinfachte Wartbarkeit der Testfälle ermöglicht.“[12] Damit ist gemeint, dass die Testfälle, welche durch das Konzept des Keyword Driven Testing erstellt werden, vergleichsweise mit Capture und Replay vereinfacht ebenfalls adaptiert werden können. So können Befehle & Kommandos generisch aufgebaut werden und die Ziele/Targets durch verschiedene Bibliotheken übergreifend zusammengetragen werden.

Das bedeutet konkret, dass „die Testfälle sowieso in der Adaptionsschicht in konkrete Befehle um- gewandelt werden: Etliche Änderungen, die z.B. das konkrete Schnittstellenformat betreffen und in den Testfällen der Adaptionsschicht hinzugefügt werden, können somit vergleichsweise aufwandsarm und einmalig für alle Testfälle in der Adaptions- schicht vorgenommen werden.“[13]

Tabelle 3: Vergleich der Kosten nach Einstufung: niedrig/mittel/hoch.

	Capture & Replay	Schlüsselwortgetrieben (Keyword Driven)
Initiale Kosten	niedrig	mittel
Wiederkehrende Kosten	hoch	niedrig

Quelle: Eigene Darstellung nach Dr. Stephan Weissleder Richard Seidl (2016) S. 16

[12] Dr. Stephan Weissleder Richard Seidl (2016) S. 12
[13] Dr. Stephan Weissleder Richard Seidl (2016) S. 12

Mit einem Blick auf die Kosten, sowohl auf die initialen Kosten als auch die Wiederkehrenden Kosten ist das Konzept des Keyword Driven kostenärmer.[14] Das liegt an der simplen Adaption und durch diese werden beinahe keine notwendigen Änderungen benötigt. Dies geschieht aufgrund der „Foundation“ durch die Befehle, welche vorab durch die mittleren initialen Kosten gedeckt werden.

Im Vergleich dazu: „der große Vorteil von „Capture & Replay“ ist, dass Fachexperten ohne Unterstützung von Test- oder Software-Experten direkt Testfälle erstellen und ausführen können. Das geht ohne große Vorbereitungs- oder Einarbeitungszeit. [...]“[15] Der Tester der Software benötigt für das Verfahren keine große Foundation. Es wird lediglich das Testtool, die Software und die dazugehörigen Testfälle benötigt. Dieses Konzept zeugt von Schnelligkeit und mittlerer Effizienz. Im Gegensatz zum Keyword Driven Testing sind die kosten für eine Adaption hoch.

„Da die Testfallerstellung in der Aufzeichnung manuell ausgeführter Testfälle besteht, muss jeder Testfall also erneut manuell ausgeführt werden. Eine einzelne Änderung ist zwar (in Abhängigkeit von der Komplexität des Systems) mit geringem bis mittleren Aufwand verbunden. Wir gehen aber aufgrund der vergleichsweisen hohen Frequenz von Anpassungen gemessen an Änderungen des Testobjekts von vergleichsweise hohen Testentwurfs- kosten aus.“[16]

[14] Siehe Tabelle 1: Vergleich der Kosten

[15] Dr. Stephan Weissleder Richard Seidl (2016) S. 7

[16] Dr. Stephan Weissleder Richard Seidl (2016) S. 8

4 Praxisteil

4.1 Vorstellung: Rola Security Solutions GmbH

“Softwareentwicklung im Sicherheitsumfeld gibt es seit fast 40 Jahren: vom deutschen Marktführer, der Rola Security Solutions GmbH. Polizeibehörden, Nachrichtendienste, Grenzschutz, Militär, Steuerfahndungen, weitere Sicherheitsorganisationen und Wirtschaftsunternehmen nutzen die Produkte und Lösungen auf dem Gebiet der Inneren und Äußeren Sicherheit. Egal ob Kriminalpolizist, Analyst, Cyber Crime-Experte oder Corporate Security Spezialist – die leistungsstarken Softwaresysteme aus Oberhausen in Nordrhein-Westfalen sind ein Begriff im In- und Ausland. Mit 250 Mitarbeitern entwickelt Rola Software „Made in Germany“ für Aufgaben von Behörden und Organisationen mit Sicherheitsaufgaben und für große Unternehmen.“[17]

Das Praxisunternehmen in dem der praktischen Teil getätigt wird während des Studiums des Autors ist die Rola Security Solutions GmbH. Diese ist ein Software-Tech Unternehmen, welche seit fast 40 Jahren deutscher Marktführer hinsichtlich der Programmentwicklung speziell für Behörden – sowohl national als auch international.

Mit rund 250 Mitarbeitern und am Standort in Oberhausen wirkt die Firma, obwohl sie relativ groß ist, sehr familiär. Aufgeteilt in Qualitätssicherung, Entwicklung und Geschäftsführung gibt es viele Projekte und Kunden, welche von allen bearbeitet werden. Dabei ist es egal ob es sich um einen internationalen englisch sprachigen Kunde handelt oder einen nationalen Kunden wie zum Beispiel die Landespolizei Sektoren oder auch der Geheimdienst.

Die Rola Security Solutions GmbH ist eine Tochtergesellschaft der Deutschen Telekom Security GmbH.

Mit Ihrer „Hauptsoftware“ rsClient beziehungsweise dem Hauptframework „rsClient“ hat jeder Kunde im Prinzip die gleiche Grundlage, welche dann zu den Wünschen und Belieben der Kunden angepasst wird. Egal ob Farbe – Text – Inhalt oder SUT Name. Durch dieses generische Prinzip des Frameworks konnte sich die Rola Security Solutions einen guten Platz in der Security Sicherheitsbranche sichern.

[17] Rola Security Solutions GmbH - https://www.rola.com

Der Autor Christopher Martin Müller wird in der Qualitätssicherung Abteilung Testautomation eingesetzt in welcher er in der Domäne/Abteilung Bundeswehr zugeteilt ist und dieser Abteilung mit seinem Wissen über die Testautomation weiterhilft.

Für die Zukunft plant die Rola Security Solutions GmbH ein großes Update ihres Frameworks – welches dann „rsClient 2.0“ heißen würde und über viele Systeme hinaus genutzt werden kann und vor allem schneller und effizienter läuft.

4.2 Test-Automation Software: Tosca

In der Rola wird als Software für die Testung – also als Testtool die Software Tosca benutzt. Das Tool Tosca ist von der Firma Tricentis und ist perfektioniert für die Testung von systems under test und dessen graphical user interfaces.

Durch Tosca kann das Qualitätsmanagement sowohl manuell als auch automatisiert testen was eine vereinfachte Zusammenarbeit mit sich führt aufgrund dessen das die manuellen Testfälle oftmals eine Vorlage für die Automation an sich sind.

„Erstellen Sie codefreie, stabile, automatisierte Tests durch einen einzigartigen Ansatz, der die technischen Informationen einer Anwendung vom Automatisierungsmodell trennt.“[18]

Tricentis Tosca nachfolgend Tosca genannt, ist in der Rola als Test-Software ein Zwischenmodul zwischen dem eigentlichen automatisierten Testtool und dem Tester selbst. Das bedeutet, dass die Testdaten, welche man für das Ausführen eines Testfalls braucht in Tosca gespeichert werden und hier die Testautomation grundsätzlich vorbereitet wird.

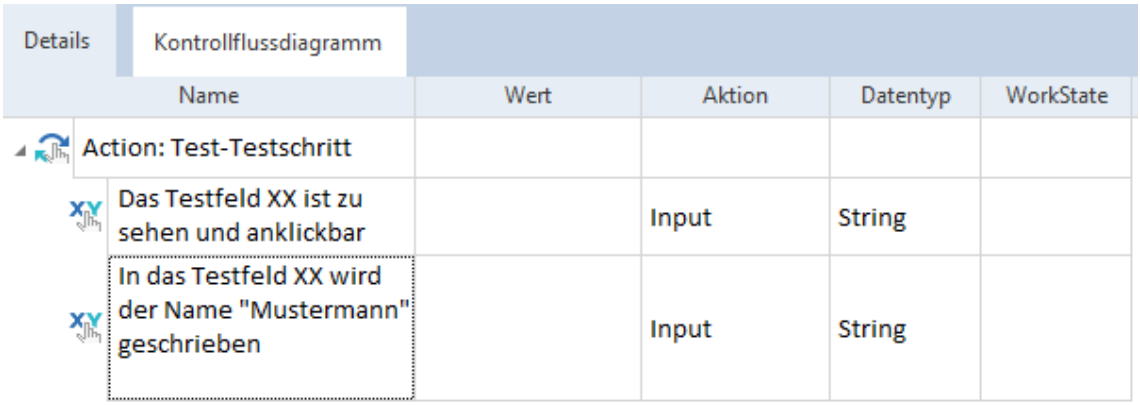
Es können in Tosca neben den automatisierten Testfällen auch manuelle Testfälle geschrieben werden. Diese werden in Ordnern, welche dann Test-Templates beinhalten, gespeichert und abgelegt. Testschritte, welche auch als "Actions" bezeichnet werden. In diesen "Actions" werden die zu beschreibende manuelle Ausführung gespeichert, die durch den Testprozess entstehen und protokolliert werden.

Abbildung 5: Screenshot Tosca - Manuelle Testfälle

[18] Tricentis Tosca – Übersicht - https://www.tricentis.com/de/plattform/automate-continuous-testing-tosca/

Ein Testschritt im manuellen Bereich enthält immer einen Text. Dieser Text beinhaltet den für den Tester relevanten Prozess, der die Ausführung veranlasst. Das kann zum Beispiel das Testfeld "XX" sein. Dieses muss allerdings sichtbar und anklickbar sein. Diesen typischen Testschritt überprüft der Tester.

Abbildung 6: Screenshot Tosca - Ein manueller Testschritt detailliert.



Quelle 7: Eigene Darstellung

Neben „normalen“ Testschritten findet vorab eine „preCondition“, deutsch: “Voraussetzungen“ statt. Diese beinhaltet die Voraussetzung für den jeweiligen Testfall. Das können neben Anmeldedaten auch eine SUT Instanz oder eine andere Vorlage sein.

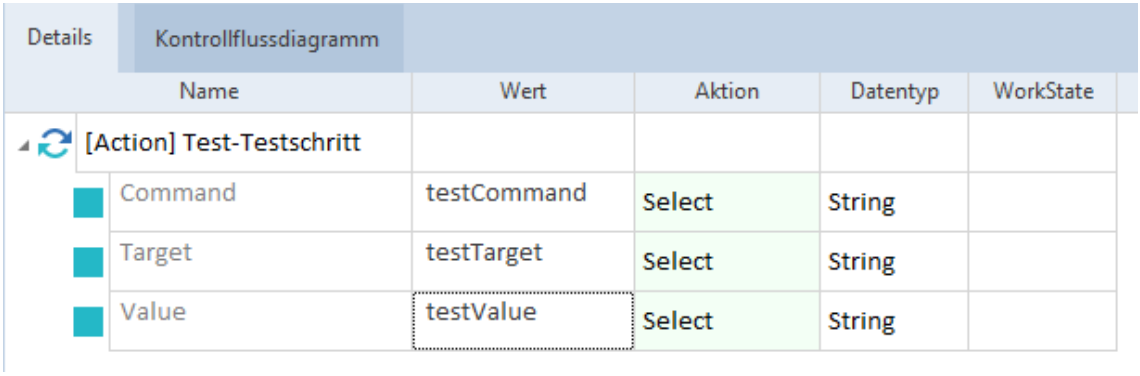
Neben normalen manuellen Testfällen, wo die Informationen in Form eines Texts dargestellt werden, können auch automatisierte Testfälle aufgebaut werden. Diese beinhalten Kommandos und das gegebene Ziel/Target, welches man auf dem GUI/SUT ansprechen möchte.

Abbildung 7: Screenshot Tosca - Automatisierter Testfall

Quelle 8: Eigene Darstellung

Diese Testschritte in den Ordnern – in Tosca auch „Template“ genannt, wie in Abbildung 4 zu sehen ist, beinhalten Command, Target und Value. Diese diktieren neben den Informationen die das Testtool braucht, auch den Befehl, welcher ausgeführt wird, um automatisiert testen zu können.

Abbildung 8: Screenshot Tosca - Automatisierter Testschritt detailliert



Quelle 9: Eigene Darstellung

Wie in Abbildung 5 zu sehen, beinhaltet der „Test-Testschritt“ einen Command, hier kann z.B. ein pushButton oder ein selectWindow benutzt werden. Diese Befehle müssen vorher durch einen Programmierer in Java festgelegt werden damit diese im Testtool selbst dann benutzt werden können. Damit der Befehl ausgeführt werden kann muss ein Ziel gesetzt werden. Dieses Target kann zum Beispiel ein Knopf oder ein Testfeld sein, welches angesteuert wird. Der Wert oder in Tosca „Value“ genannt, ist das Feld, wo dann der dementsprechende Wert, den man übergeben möchte, eingetragen wird. Ein Beispiel für einen fertiges Testschritt wäre z.B. ein „insertIntoTextField“ als Command – ein „TextfeldXX“ als Target und dann im Value/Wert wird dann z.B. „MaxMustermann“ eingetragen.

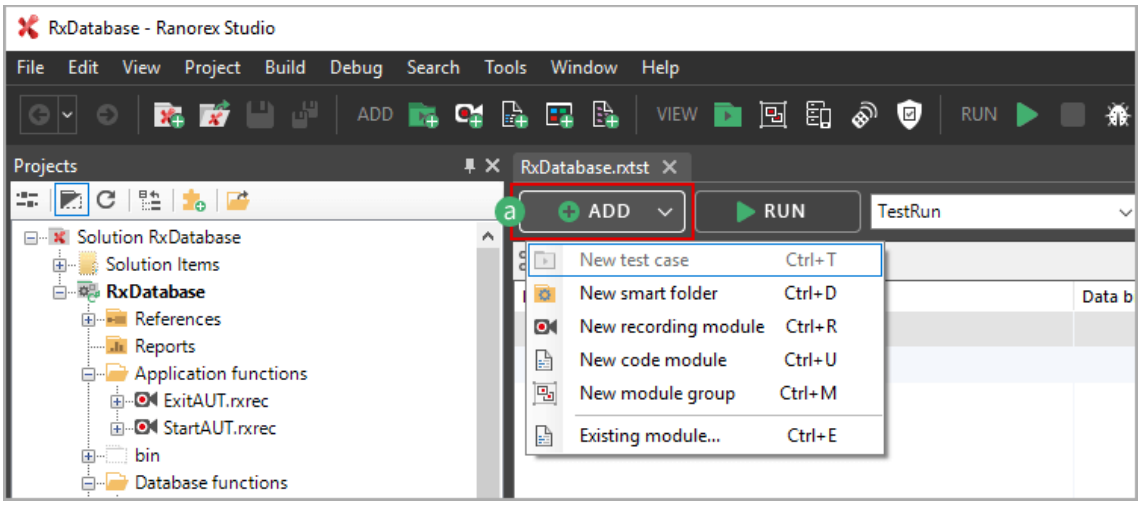
4.3 Ist-Analyse: Capture and Replay Testing mit Ranorex

In der Firma Rola Security Solutions war das Vorgehen stets die Nutzung des Capture und Replay Testing. Es wurde das Testtool Ranorex verwendet.

„Ranorex ist eine Software zur Erstellung automatisierter Tests auf Systemtestebene. Die Anwendungsbereiche umfassen dabei Mobile-, Web- und Desktopanwendungen.

[...] Ranorex bringt dabei eine eigene Entwicklungsumgebung mit, die es ermöglicht Tests auf vielfältige Art zu automatisieren. Entwicklung der Tests über die C# und VB.NET Bibliotheken ist ebenso möglich wie das „Record und Replay“ von Testszenarien mit dem Recorder. Vereinfacht wird dies durch eine automatische Erkennung von GUI-Elementen und der Möglichkeit Testblöcke per Drag-and-Drop einzufügen und anzuordnen. Diese schnelle Implementation von vielfältigen Testszenarien macht Ranorex vor allem im Continous Delivery Workflow interessant.“[19]

Abbildung 9: Screenshot-Ranorex: Details des Test Systems



Quelle 10: Screenshot Ranorex: URL:https://www.ranorex.com/help/latest/ranorex-studio-fundamentals/test-suite/test-suite-structure-elements/ - zuletzt abgerufen: 22.02.2022

Der Vorteil des Konzeptes war zum Zeitpunkt der Nutzung in der Rola vorwiegend das triviale Handling. Der Test-Manager konnte dem Tester eine Liste von Testfällen geben und dieser konnte die Testfälle schnell und effizient abarbeiten. Mit dem Konzept des Capture und Replay konnte der Tester somit den Testfall wie in Abbildung 9 zu sehen in der Datenbank anlegen und dann ein neues Aufzeichnungsmodul anhängen.

Dieses Modul konnte neben der Aufzeichnung der Eingaben des Testers auch den Verlauf des Testfalls skizzieren. Durch diese Aufzeichnung konnte der Testfall im

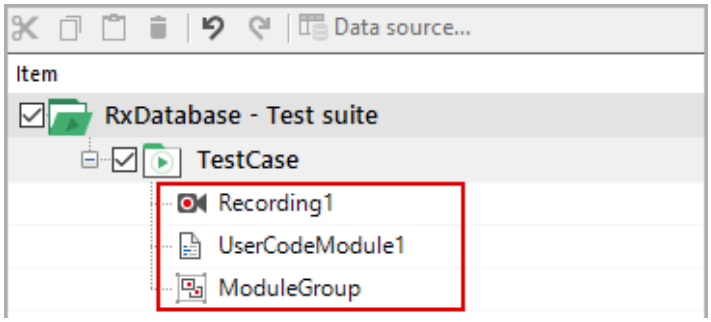
[19] Was ist Ranorex? – Testautomatisierung.org - https://www.testautomatisierung.org/lexikon/ranorex/ zuletzt abgerufen am 14.02.2022

Nachhinein schnell eingesehen werden und vor allem bei einer Auslieferung eines neuen Updates der Software wiederholt werden.

Das Tool Ranorex hatte zum Zeitpunkt der Nutzung auch schon einzelne Ansätze zur generischen Testautomation. So konnten durch Testdaten Provider oder auch Variablen Informationen gespeichert und adaptiert werden, ohne den aufgezeichneten Testfall neu zu erstellen. Die Variablen in Ranorex „[...] sind die Platzhalter für Werte, die Sie in Ihren Test einspeisen wollen, entweder aus Datenquellen oder Parametern“[20].

Abbildung 10 stellt einen detaillierten Testfall dar. Das „Recording Modul“, übersetzt „Aufzeichnungsmodul“ ist in jedem Testfall angehängt und beinhaltet die vollständige Aufzeichnung der Schritte des Testers.

Abbildung 10: Screenshot - Ranorex: TestCase im Detail



Quelle 11: Screenshot Ranorex: URL:https://www.ranorex.com/help/latest/ranorex-studio-fundamentals/test-suite/test-suite-structure-elements/ - zuletzt abgerufen: 22.02.2022

Die bedeutendsten Grenzen zeigten sich in den verschiedenen Auflösungen des Programmes, die angezeigt wurden. So konnte ein Testfall der bei einem Tester mit 1920 mal 1080 Pixel großem Bildschirm läuft oftmals aufgrund der teilweise hard-codierten Werte und Variablen nicht auf einem kleineren Bildschirm laufen, solche hatten im Backend oftmals ein bestimmtes Ziel angehangen, welches je nach Element Bildschirm

[20] Übersetzt aus dem Englischen: Ranorex – Define Test and Variables – URL: https://www.ranorex.com/help/latest/ranorex-studio-advanced/data-driven-testing/preparations-data-driven-testing/ zuletzt abgerufen am: 22.02.2022

Koordinaten beinhaltet oder auch Pixelangaben. So konnte zum Beispiel ein „Button“ (= Element in der SUT) in der Wiederholung nicht geklickt werden, weil sich dieser aufgrund des Bildschirmgrößen Unterschiedes an anderen Koordinaten befand.

Darüber hinaus waren nicht nur die Koordinaten also die Ziele der Elemente eine Grenze, auch die fachlichen und technischen Namen waren bei der Wiederholung der Aufzeichnungen eine Hürde. Hat sich ein Element in der SUT geändert, beispielsweise der technische Name, so hat ihn die Aufzeichnung nicht mehr gefunden und der Testfall ist fehlgeschlagen. Der Tester ist also gezwungen alle Testfälle neu aufzuzeichnen welche Änderungen beinhalten, was eine generische Nutzung und die Effizienz des Konzeptes erheblich einschränkt.

4.4 Soll-Konzept: Keyword Driven Testing mit Swing Agent

In der Rola Security Solutions GmbH ist zum Zeitpunkt der vorliegenden Arbeit das Soll-Konzept im Allgemeinen Sinne der Rola noch das Keyword Driven Testing mit dem Swing Agent.

Das Konzept des Keyword Driven Testing wird hierbei durch das Testsystem „Swing Agent“ unterstützt, welches die Befehle, welche der Tester zunächst konfiguriert, ausführt und auch das Logfile – also die Ergebnisse – des Testfalls festhält. Im Falle der Rola Security Solutions GmbH wird die Swing Libary als Open Source Vorlage verwendet. Das Open Source Projekt, welches auf GitHub (= „Auf hohem Niveau ist GitHub ein webbasierter Dienst, der Entwicklern hilft, ihren Code zu speichern und zu verwalten sowie Änderungen an ihrem Code zu verfolgen und zu kontrollieren“[21]) verfügbar ist, wird als folgendes definiert: „Swing Library ist eine Robot Framework-Bibliothek zum Testen von Java Swing-Anwendungen.“[22]

Das Programm rsClient der Firma Rola ist ein Java basiertes Programm, welches so unter anderem als Grundlage des Testens die Swing Libary in Hinsicht auf das Keyword Driven Testing benutzt. Wichtig zu erwähnen ist, dass die Swing Libary als Komponente nicht als Testsystem gleichgesetzt werden kann. Denn diese ist ein Teil des Testsystems, welches mit einem eigenen Programm der Firma gekoppelt mit dem Namen „OpenTAL“ ist. OpenTal bildet die graphische Ebene ab, welche dem Tester ermöglicht die LogFile (= „Bei einem Logfile, auch als Logdatei oder Protokolldatei bezeichnet, handelt es sich um eine Datei, in der Computerprozesse verschiedene Ereignisse protokollieren. Logfiles sind wichtige Informationsquellen, um die Vorgänge auf einem System nachvollziehbar zu machen. Sie lassen sich beispielsweise für die Problemanalyse oder die Rekonstruktion von verloren gegangenen Daten verwenden“[23]) sofort zu lesen, und herauszufinden, ob der Testschritt erfolgreich war oder missglückt ist und es einen Fehler gab. Einen Einblick in diese graphische Ebene gibt die Abbildung 12.

Der wesentliche technische Teil ist aber im Hintergrund des Programms zu finden, das ist in Fall der Rola die Swing Libary welche neben den Befehlen für die Testautomatisierung auch die Dokumentation und Funktionen zur Automation beinhaltet wie einen

---

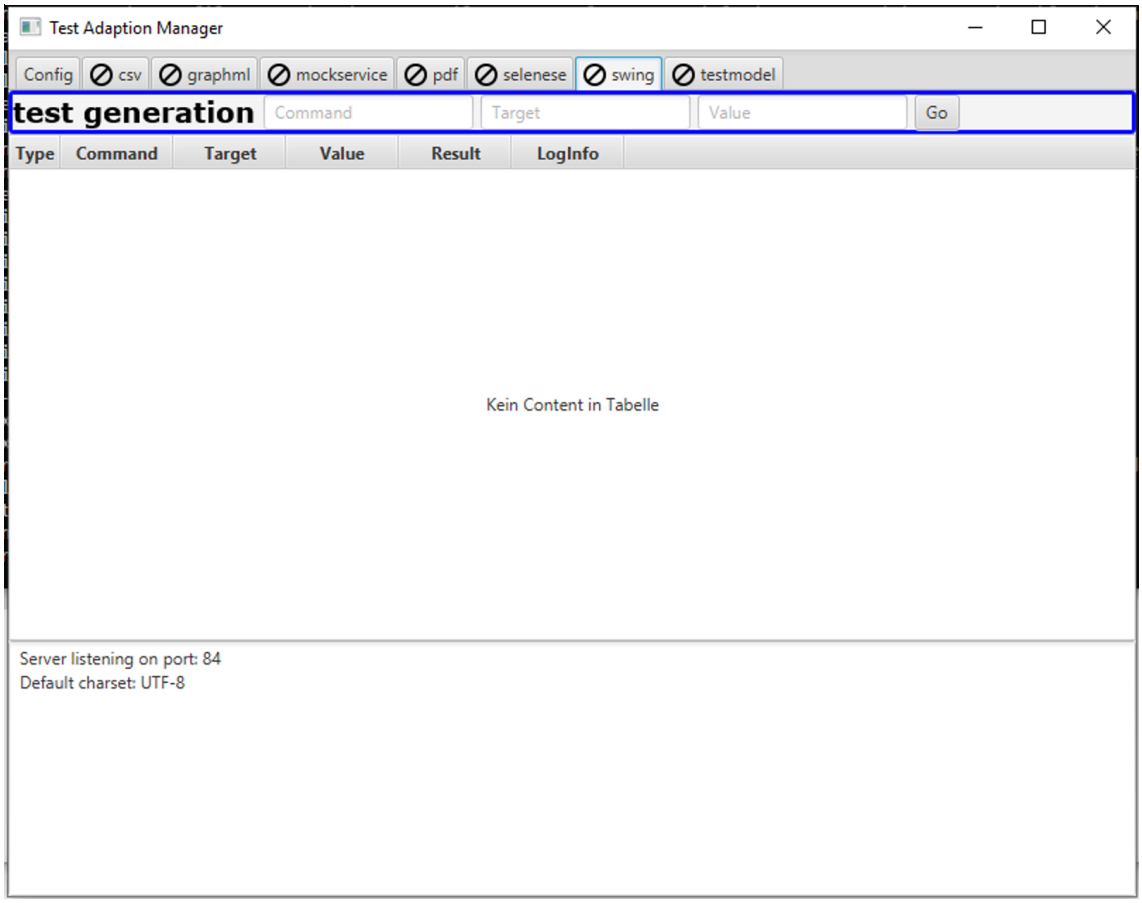
[21] Was ist GitHub? – 4.März 2022 - https://kinsta.com/de/wissensdatenbank/was-ist-github/ - zuletzt abgerufen am 18.02.2022.

[22] Swing Libary Framwork – GitHub - https://github.com/robotframework/SwingLibrary

[23] Was ist ein Logfile? Stefan Luber – 21.03.2018 – URL: https://www.bigdata-insider.de/was-ist-ein-logfile-a-697575/ zuletzt abgerufen am: 18.02.2022

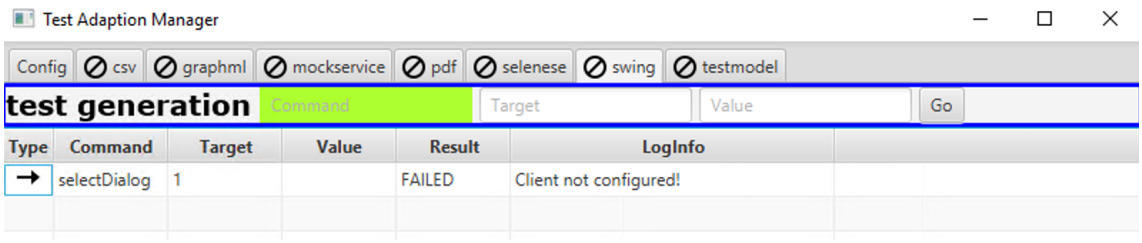
Roboter der einen Tester simuliert und zum Beispiel ein Menü per Mausbewegung öffnen kann.

Abbildung 12: Screenshot: OpenTAL.



Quelle 13: Eigene Darstellung - Screenshot internes Programm.

Abbildung 11: Screenshot: OpenTAL - ein Befehl im Detail



Quelle 12: Eigene Darstellung - Screenshot eines internen Programms

Wie in Abbildung 10 zu sehen ist wird der Befehl detailiert im Programm OpenTAL wiedergegeben. OpenTAL ist ein von Karsten Dröges weiterentwickeltes Programm, welches auf dem Open Source Programm VEBTAL aufbaut.[24]

So kann der ausgeführte Befehl „selectDialog“ als auch das Target „1“, das Value in diesem Falle leer, das Ergebnis „FAILED“ anlegen und die LogInfo, welches das Ergebnis für den Tester einfacher zu verstehen gibt wieso in diesem Falle der Befehl fehlgeschlagen ist. In unserem Beispiel ist der Befehl fehlgeschlagen, weil das Testsystem OpenTAL vorher nicht mit dem zu testenden Programm verbunden wurde – daher kann er keine Befehle ausführen.

Neben den Testsystem Programmen ist ein wesentlicher Teil des Konzepts, welches genutzt wird, das Programm Tosca wie vorher bereits erwähnt. Dieses stellt eine Art Informationsdatenbank dar. In dieser kann der Tester die Testfälle vorbereiten, und dann per Knopfdruck mit den jeweiligen Testsystemen öffnen und ausführen.

Zum Zeitpunkt vor Einführung des neuen Konzeptes, welches in Kapitel 4.4.1 beschrieben wird, wurden in der Testautomatisierung zwar die Testfälle generisch aufgebaut, das bedeutet: Die Testfälle hatten bereits Business Parameter – aber der Tester musste jegliche manuelle Testfälle von Hand durchführen. Zu diesem Zeitpunkt gab es noch nicht die Möglichkeit, dass die Testfälle automatisiert gespeichert und somit wiederverwendet werden können. Das führt zwangsläufig zu einem nicht mehr wiederverwendbaren Testfall, welcher maximal einmalig oder auch zweimalig für eine Ausführung verwendet werden kann. So werden die Vorteile des Keyword Driven Testing nicht ausgeschöpft und die Kosten durch das stetige Erstellen der Testfälle übersteigt die Vorteile des eigentlichen Konzeptes.

[24] VEBTAL – OpenTAL Grundprogramm URL: https://github.com/vebqa/vebtal-pdf

4.4.1 Eine Möglichkeit der Verbesserung des Soll-Konzepts

Damit die Vorteile des Keyword Driven Testing überwiegen muss sichergestellt werden, dass die Testfälle effizient strukturiert werden und es nicht zu einem Mehraufwand durch das stetiges Adaptieren ein oder mehrere Tester kommt.

Die nachfolgende Möglichkeit der Verbesserung des Soll-Konzepts ist eine Idee von Sven Konowski, Fachinformatiker in der Qualitätssicherung der Rola – Projekt Bundeswehr und Christopher Martin Müller – Autor dieses Werkes, dualer Student der Deutschen Telekom – eingesetzt in der Rola Security Solutions GmbH – Projekt Bundeswehr.

In der Rola Security Solutions GmbH ist das nachfolgende Konzept bereits im Projekt Bundeswehr in Einsatz. Es handelt sich dabei um eine grundlegende Verbesserung des Keyword Driven Testing in der Rola – das Konzept der „Bausteine“, nachfolgend „BS Konzept“ genannt.

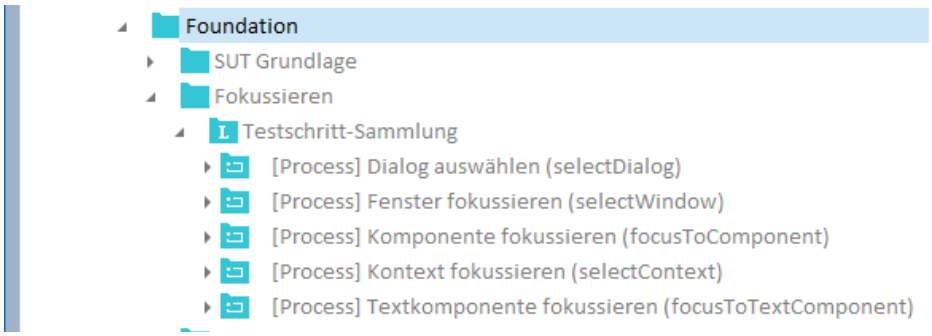
Abbildung 13: Screenshot-Tosca: Testautomatisierung - Foundation Überblick

Quelle 14: Eigene Darstellung – Tosca Intern – Rola Security Solutions GmbH

Das BS Konzept baut auf einer Testautomation Foundation auf. Diese ist neben einer großen Sammlung von Testschritten vor allem die Bibliothek der eingepflegten Kommandos des Keyword Driven Testing. Diese sind der Grundbaustein und daher der erste Schritt des BS Konzept.

Die Kommandos und die dadurch einhergehende wechselseitige Möglichkeiten des Testens der GUI / SUT können aufgeteilt und in verschiedenen Bereichen gesammelt und vorbereitet werden. Die Vorbereitung der Kommandos wird, damit das BS Konzept zum Schluss generisch bleibt, mit Business Parameter verbunden.

Abbildung 14: Screenshot-Tosca: Foundation - TA - Base - Detailiert

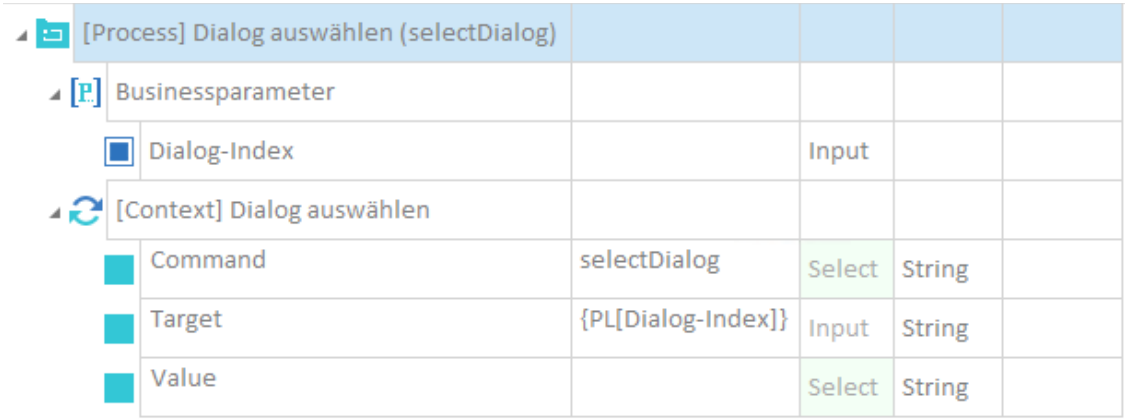


Quelle 15: Eigene Darstellung - intern Rola Security Solutions GmbH

Wie in Abbildung 14 zu sehen ist wird die Foundation in verschiedene Teilgruppen aufgeteilt. Diese beinhalten Kommandos, welche dann generisch aufgebaut sind und dann in Testfällen eingepflegt werden. Der Abbildung kann entnommen werden das neben mehreren Kommandos in der Testschritt Sammlung, wie zum Beispiel „selectDialog“ oder auch „selectWindow“ diese Kommandos zuvor erklärt werden. Im Beispiel ist es dann: „Dialog auswählen (selectDialog“, das führt dazu, dass im späteren Verlauf der Testautomatisierung der „normale“ Tester, welcher sich nicht mit den Kommandos auskennt – diese ohne große Verständnisprobleme für seine Testfälle, die er im jeweiligen Projekt machen muss benutzten und einpflegen kann.

Wie zuvor erklärt werden die Kommandos in der Testschritt Sammlung in der Foundation generisch aufgebaut. Diese generische Arbeitsweise verhilft dem BS Konzept zu seinem trivialen Verständnis für die jeweiligen Nutzer und Tester. So kann der Tester im späteren Verlauf das Kommando, ohne zu verstehen was es im Detail beinhaltet, einfach den Testschritt in seinen Testfall ziehen und die Business Parameter bearbeiten.

Abbildung 15: Screenshot-Tosca: Ein Testschritt detailliert – Business Parameter.



Quelle 16: Eigene Darstellung - intern Rola Security Solutions GmbH

Abbildung 15 präsentiert einen solchen Testschritt detailliert. Zu sehen ist der Testschritt „Dialog auswählen (selectDialog)“, welcher einen Business Parameter „Dialog-Index“ beinhaltet und den eigentlichen Command „Dialog auswählen“ oder auch „selectDialog“ im technischen Namen. Der Teilschritt „Dialog auswählen“ ist dabei nicht für den Tester zu sehen. Dieser sieht nur den Business Parameter und kann auch nur diesen bearbeiten. In diesem Testschritt hat der Befehl, ein Target, also ein „Ziel“.

Dieses Ziel wird durch einen technischen Namen gefüllt und der Befehl wird dann durch das Testsystem ausgeführt. So kann in unserem Beispiel der Befehl „selectDialog“ mit dem Target/Ziel „0“ auf den nullten also ersten Dialog (Fenster) der SUT zugreifen und diesen dann selektieren. Der Tester kann also in den Business Parameter das jeweilige Ziel reinschreiben, weil dieser Parameter im eigentlichen Ziel/Target des Befehls als Platzhalter dient - der Wert wird also über den Business Parameter weitergegeben und der Befehl kann erfolgreich ausgeführt werden.

Das „Bausteine“ Konzept funktioniert so trivial und kann von jedem in der Firma ohne weitere Lernprozesse verstanden werden. Jeder Befehl beziehungsweise jedes Kommando ist so in diesem Konzept aufgebaut und jederzeit mit der Foundation verknüpft. Bei Änderungen von Befehlen oder anderen Anpassungen kann so ein problemfreier Übergang gewährleistet werden. Alle Testfälle, welche mit der Foundation gebaut werden, werden durch die Verbindung ebenfalls geändert.

4.4.2 Derzeitige Grenzen

Da jeder Befehl im BS Konzept einen generischen Business Parameter enthält, muss dieser zwingend gefüllt sein, wenn der Testschritt ausgeführt wird. Wenn der Testschritt keinen Business Parameter enthält, wird dieser nicht richtig ausgeführt und verhindert im weiteren Verlauf das erfolgreiche Beenden aller Testschritte. Durch die Weitergabe der Business Parameter gilt es, das Target / das Ziel stets korrekt zu schreiben. Das ist mitunter eine Grenze, welche man schnell erreicht. Diese Grenze wird durch Fehler seitens der Entwicklung oder des Testers verursacht.

Im Laufe der Testautomatisierung konnte der Autor mehrere Fehlerquellen identifizieren. So sind neben den normalen Standardfehlern, durch eine falsche Eingabe, vor allem die fehlenden Zielnamen durch die Entwicklung eine große Hürde.

Im Programm der Rola wurden viele in der GUI / SUT vorhandene Dialoge und Bedienungselemente nicht wirklich benannt. Normalerweise muss jedes Element in der SUT der Rola einen ToolTipText oder einen Namen der einzigartig ist haben. Durch diesen kann das Ziel identifiziert und anvisiert werden mit dem Befehl, den man benutzen muss. Ist der Name oder der ToolTipText im Bedienungselement der GUI nicht vorhanden kann das Element nicht angesprochen werden und somit nicht automatisiert werden.

Diese Grenze schränkt nicht nur das BS Konzept ein – sondern die Automatisierung im Allgemeinen. Durch die fehlenden Angaben für das Ziel der Automatisierung kann der Roboter (Testsystem) oder auch Agent (im Falle der Rola genannt), den Befehl allgemein nicht ausführen. In diesem Fall ist es egal, ob der Testschritt generisch aufgebaut ist oder „hard-coded“ in das Target eingetragen wurde.

Neben den fehlenden Targets ist die Kommunikation innerhalb der Rola Security Solutions GmbH ein wesentlicher Faktor in Hinsicht auf die Grenzen der Konzepte der Automatisierung. Fehlende Targets oder Fehler im System werden als unwichtig priorisiert oder nach hinten verschoben, wo sie vergessen werden. Durch solche Fehler stockt die Qualitätssicherung sehr und ist im Wesentlichen wieder auf manuelles Testen angewiesen welches dementsprechende Kosten verursacht und in keiner Form nachhaltig für die Effizienz und den Gewinn des Unternehmens ist.

4.5 Handlungsempfehlung

In der Rola Security Solutions GmbH ist dem Autor aufgefallen das Bugs (= „Ein Bug ist ein logischer Fehler im Code der Software, der dazu führt, dass das Programm nicht wie gewünscht oder überhaupt nicht funktioniert. Der Fehler kann nicht durch eine Syntaxprüfung entdeckt werden.“[25]) und Fehler oft nicht richtig behandelt werden oder als unwichtig priorisiert werden. Die Zusammenarbeit der Qualitätssicherung und der Entwicklung ist essenziell für den Erfolg des Unternehmens. Der Autor hat oft zu spüren bekommen das er die letzte Instanz zum Kunden ist und so das System auf Fehler prüft – ob manuell oder automatisch ist unwichtig.

Durch die fehlende Kommunikation der Entwicklung mit der Qualitätssicherung können kleine Fehler oder Bugs nicht sofort oder schnell behoben werden und so stapeln sich solche schnell. Durch die fehlenden Targets können die automatisierten Testfälle nicht weiterbearbeitet werden und müssen gezwungen übersprungen werden. Das führt neben einer Redundanz, zu einer kryptischen Unordnung innerhalb der Testfälle, weil kein Tester mehr weiß welcher funktioniert und welcher nicht.

Wie der Trend und zum Beispiel Umfragen des WQR (=World Quality Reports) zeigen: „[...]88 Prozent der Befragten gaben an, dass der Bereich ihrer Testaktivitäten, der am stärksten wächst, KI-basiert ist. 86 Prozent sehen in KI ein Schlüsselkriterium bei der Auswahl neuer Qualitätssicherungs-Lösungen. [...]“[26] geht es in Richtung von Testautomatisierung mithilfe von Künstlicher Intelligenz (=KI).

Hier wäre eine Handlungsempfehlung seitens des Autors eine mögliche Künstliche Intelligenz oder ein Application Programming Interface (=API), welches die Bereiche der Entwicklung der zu testenden Software mit den Bereichen der Qualitätssicherung – genauer der Testautomatisierung und der wie z.B. im BS Konzept – Foundation verbindet und so eine Adaption bei jeglichen Veränderungen sicherstellt und garantiert.

---

[25] Definition „Software-Fehler“ - Was ist ein Bug? Autor: zeroshope - https://www.dev-insider.de/was-ist-ein-bug-a-941783/ zuletzt abgerufen am 17.02.2022

[26] Entwickler.de Die Zukunft der Qualitätssicherung: Testautomatisierung und KI – 22.03.2021 – URL: https://entwickler.de/machine-learning/die-zukunft-der-qualitatssicherung-testautomatisierung-und-ki/ zuletzt abgerufen am 17. Februar 2022.

5 Schlussbetrachtung

5.1 Ergebnis

Der Vergleich beider Konzepte zeigt, dass beide logischen Konzepte, sowohl Capture and Replay als auch das Keyword Driven Testing zu einem erfolgreichen Unternehmen dazugehören. Beide Konzepte unterstützen in ihrer eigenen Art die verschiedenen Bereiche der Qualitätssicherung und sind oftmals nicht mehr wegzudenken.

Neben schnellerem Testen, kürzeren Wartezeiten, einfacher Wartbarkeit und flexibler Adaption an Situationen überzeugen beide Konzepte mit ihrer Einfachheit und der schnellen Verständlichkeit gegenüber Testern.

Doch diese Konzepte bergen neben den Chancen und Vorteilen leider auch Risiken und Nachteile mit sich, die das Unternehmen abwägen und einschätzen muss. Voraussetzungen wie das Verhindern einer schlechten Kommunikation mit der Entwicklung der Software oder auch eine bessere Konzeption der Adaption des Capture & Replay Konzepts sind essenziell und bei Entscheidung vorher zu bestätigen.

Allgemein lässt sich aus diesem Vergleich der logischen Konzepte aus der Praxisarbeit schließen, dass beide logischen Konzepte je nach Einsatzbereich einen wertvollen Inhalt mitbringen und eine Effizienz in der Testung eigener Software mit sich bringt. Trotz dessen muss ein Unternehmen selbständig entschieden, welchen Weg dieses gehen möchte. Sowohl eine komplett generische Testautomatisierung, wie im BS Konzept erklärt als auch eine eher weniger automatische Capture & Replay Testung kann effizient sein. Diese Entscheidung hängt von Faktoren wie – Kosten, Mitarbeiter, und Zeit ab.

Für welches Unternehmen, welches Konzept nun das „Beste“ ist, kann im Allgemeinen nicht beantwortet werden. Man sollte aber wissen das eine komplett generische Lösung, wie die des BS Konzepts im Nachhinein um weiniges einfacher zu adaptieren und auch wenn es anfangs höhere Kosten verursacht, wird es dem Unternehmen letztendlich sicherlich zu Gewinn verhelfen.

Abschließend kann gesagt werden, dass: „Softwaretests sind für die Überprüfung und Validierung von Software von größter Bedeutung. Sie sind vor allem aus zwei Gründen wichtig: Erstens sichern sie die Softwarequalität, und zweitens werden fast 60 % der

gesamten Softwarekosten für verschiedene Arten von Tests ausgegeben. [...] Wir haben auch festgestellt, dass die Testautomatisierung positive Auswirkungen auf die Softwarequalität hat. Daher können wir behaupten, dass die Testautomatisierung die Gesamteffektivität des Testprozesses erhöht, wenn es sich um sich wiederholende, ähnliche Testaufgaben handelt."[27]

5.2 Ausblick

Um die Testautomatisierung in der Rola Security Solutions GmbH in Zukunft noch effizienter zu gestalten, muss eine gute und wiederholte Kommunikation seitens der Qualitätssicherung und der Entwicklung sichergestellt werden, um so jegliche Fehler und Bugs zeitig zu beseitigen damit die Kosten des BS Konzepts und der Automation niedrig gehalten werden können und externe Mitarbeiter nicht aufgehalten werden im „Automationsflow".

Doch viel wichtiger ist die Zukunft der Qualitätssicherung durch eine wirklich „automatisierte Testautomation" durch den Einsatz von KI (=Künstlicher Intelligenz) zu sichern.

„Neben agilen Methoden und DevOps-Ansätzen findet eine weitere fortschrittliche Praxis langsam Einzug in die Qualitätssicherung: Der Einsatz von KI. Dabei ist dieses kein gänzlich neues Konzept. In den vergangenen Jahren hat sich jedoch die Leistungsfähigkeit von Computern und Cloud-basierten Serviceleistungen stetig verbessert. Außerdem wurde die Anwendung von KI auf bestehende Datensätze einfacher. Diese technologischen Rahmenbedingungen ermöglichen den umfassenden Einzug von KI in die Qualitätssicherung, was die Zahlen des WQRs bestätigen. 88 Prozent der Befragten gaben an, dass der Bereich ihrer Testaktivitäten, der am stärksten wächst, KI-basiert ist. 86 Prozent sehen in KI ein Schlüsselkriterium bei der Auswahl neuer Qualitätssicherungs-Lösungen."[28]

Der Trend zeigt, dass die Testautomatisierung wie wir sie kennen, schon veraltet ist und nicht mehr mit dem Fortschritt der Zeit mithalten kann. Der Einsatz von KI ist eine Möglichkeit die Kommunikation zwischen den Abteilungen zu übergehen und so zum Beispiel automatische Änderungen an einem Ende durch die KI sofortig zu ändern,

[27] Divya Kumar, K.K Mishra S. 14

[28] Entwickler.de Die Zukunft der Qualitätssicherung: Testautomatisierung und KI – 22.03.2021 – URL: https://entwickler.de/machine-learning/die-zukunft-der-qualitatssicherung-testautomatisierung-und-ki/ zuletzt abgerufen am 17. Februar 2022.

sodass es weniger Redundanzen gibt und alle auf die gleiche Information zu greifen, um Bugs und Fehler zu vermeiden.

Quellenverzeichnis

Antonia Kresse (2016): Vergleich der Entwicklungs- und Wartungskosten unterschiedlicher Capture & Replay-Tools beim Testen grafischer Benutzeroberflächen – Masterarbeit: URL: https://www.inf.fu-berlin.de/inst/ag-se/theses/Kresse16-evaluation-werkzeuge-GUI-tests.pdf - zuletzt aufgerufen: 10. Januar. 2022

Christoph Gladisch, et al. (2010): Generating Regression Unit Tests Using a Combination of Verification and Capture & Replay – Buchkapitel: URL: https://www.springerprofessional.de/generating-regression-unit-tests-using-a-combination-of-verifi-ca/3315170?searchResult=1.capture%20replay%20test&searchBackButton=true - zuletzt aufgerufen: 10. Januar. 2022

Dr. Stephan Weissleder Richard Seidl (2016) Meister der Tests – Testautomatisierung 2016.

Divya Kumar, K.K Mishra (2016) The Impacts of Test Automation on Software's Cost, Quality and Time to Market – URL: https://www.sciencedirect.com/science/article/pii/S1877050916001277 - zuletzt abgerufen am: 20.02.2022

Dheanda Absharina et. al. (2020) SURVEY PAPER: SOFTWARE AUTOMATED TESTING TOOL USING SYSTEMATIC LITERATURE REVIEW METHOD – URL: http://ejournal.nusamandiri.ac.id/index.php/pilar/article/view/1456 - zuletzt abgerufen am 20.02.2022

Raffi Margaliot (2021) Entwickler.de – Die Zukunft der Qualitätssicherung: Testautomatisierung und KI – URL: https://entwickler.de/machine-learning/die-zukunft-der-qualitatssicherung-testautomatisierung-und-ki/ zuletzt abgerufen: 18.02.2022.

Filippo Ricca et. al. (2021) KI-basierte Testautomatisierung: Eine Analyse der grauen Literatur – Buch - URL: https://ieeexplore.ieee.org/abstract/document/9440153 zuletzt abgerufen am 21.02.2022

Frank Witte (2019): Testautomatisierung - Buchkapitel: URL: https://www.springerprofessional.de/testautomatisierung/16772786?searchResult=1.Testautomatisierung&searchBackButton=true&fulltextView=true zuletzt aufgerufen: 10. Januar. 2022

Jussi Malinen et. al. (2021) SwingLibrary – GITHUB – URL: https://github.com/robotframework/SwingLibrary - zuletzt abgerufen am 05.03.2022

Matthias Daigl, Rolf Glunz (2016) ISO 29119 - Die Softwaretest-Normen verstehen und anwenden – Buch

Michel Nass et. al. (2021) Warum viele Herausforderungen bei der GUI-Testautomatisierung bestehen bleiben (werden) – Buch – URL: https://www.sciencedirect.com/science/article/abs/pii/S0950584921000963 - zuletzt abgerufen am 19.02.2022

Ranorex et. al. (2022) Ranorex Help Desk – Data Driven Testing – URL: https://www.ranorex.com/help/latest/ranorex-studio-advanced/data-driven-testing/ - zuletzt abgerufen am 1.03.2022

Ricardo B. Pereira,

et al. (2020): Architecture Based on Keyword Driven Testing with Domain Specific Language for a Testing System – Buchkapitel
URL: https://www.springerprofessional.de/architecture-based-on-keyword-driven-testing-with-domain-specifi/18647316?searchResult=1.Keyword%20driven%20test&searchBackButton=true&fulltextView=true zuletzt aufgerufen: 10. Januar. 2022

Sebastian Pöpsel (2017): Automatisierung von Regressionstests an der grafischen Benutzerschnittstelle einer agil entwickelten Enterprise-Anwendung - Bachelorarbeit: URL: https://cs.uni-pader-born.de/fileadmin/informatik/fg/mci/Bachelorarbeiten/2017/Poepsel__Sebastian.pdf zuletzt aufgerufen: 10. Januar. 2022

Tobias Walter (2016): Keyword-Driven Test Automation with Ranorex: URL: https://www.ranorex.com/blog/keyword-driven-test-automation-framework/ zuletzt aufgerufen: 10. Januar. 2022

Thomas Osterand et.al. (1998): A visual test development environment for GUI systems. In: ACM SIGSOFT Software Engineering Notes, Bd. 23 ACM, S. 82– 92

Ehrenwörtliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Praxisarbeit selbständig angefertigt habe. Es wurden nur die in der Arbeit ausdrücklich benannten Quellen und Hilfsmittel benutzt. Wörtlich oder sinngemäß übernommenes Gedankengut habe ich als solches kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form ganz oder teilweise noch keiner Prüfungsbehörde vorgelegen.

Essen, 10.03.2022

Ort, Datum

Unterschrift