MASTERARBEIT

TESTAUTOMATISIERUNG UND DESSEN MÖGLICHKEITEN SOWIE TECHNOLOGIEN ZUR SCHNELLEN UND KORREKTEN UI-OBJEKTERKENNUNG IM KONTEXT MODERNER WEBANWENDUNGEN

ausgeführt am

Studiengang

Informationstechnologien und Wirtschaftsinformatik

Von: Elena Krois

Personenkennzeichen: 1710320013

Graz, am 12. Juli 2019 .....

Unterschrift

EHRENWÖRTLICHE ERKLÄRUNG

Ich erkläre ehrenwörtlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benützt und die benutzten Quellen wörtlich zitiert sowie inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

.....

Unterschrift

KURZFASSUNG

Eine Vielzahl an Softwareunternehmen hat sich in den letzten Jahren dem Thema Testautomatisierung gewidmet. Software-Tests auf Basis des User Interfaces bauen auf unterschiedliche Objekterkennungs-Technologien auf, beispielsweise die Identifikation über ID's der HTML-Elemente oder der grafischen Bilderkennung. Besonders beim Testen moderner Webanwendungen ist Automatisierung naheliegend, denkt man an die Darstellung von Responsive Designs auf unterschiedlichen Bildschirmgrößen und Browsern.

Um festzustellen, wie sich die unterschiedlichen Technologien der UI-Objekterkennung auf die Stabilität automatisierter Tests auswirken, wurde anhand praktischer Beispiele die Identifikation unterschiedlicher Elemente näher betrachtet. Aufbauend auf eine Literatur- und Online-Recherche wurden UI-Objekterkennungs-Technologien vorgestellt und diskutiert sowie Faktoren zur Beurteilung der Stabilität aufgeschlüsselt. Unterschiedliche Vorgehensweisen zur Aufzeichnung der UI-Objekte wurden beschrieben und gegenübergestellt.

Die Evaluierung verschiedener Technologien der UI-Objekterkennung verdeutlichte die Unterschiede in der Art der Aufzeichnung unterschiedlicher Elemente. Während die Identifikation der UI-Objekte über XPath durch eine Gewichtung der Pfad-Komponenten gesteuert werden kann, ist besonders der Einsatz von Image Recognition initial mit viel manuellem Aufwand bei der Aufzeichnung der Elemente verbunden. Jedoch sind Technologien wie OCR oder Image Recognition vermehrt für Spezialfälle einzusetzen, wenn etwa durch technische Einschränkungen nicht über ID oder XPath auf UI-Objekte zugegriffen werden kann.

Zwar erwähnen immer mehr Hersteller von Testautomatisierungs-Tools Begriff Aritificial Intelligence im Kontext der UI-Objekterkennung, jedoch ist auch heute noch die Durchführung manueller Schritte für die Wartung der automatisierten Tests notwendig. Daher kann die Verantwortung über die Erstellung einfach zu wartender Tests zum aktuellen Zeitpunkt noch nicht an die jeweiligen Tools abgegeben werden, sondern verlangt noch immer Fachwissen und die strukturierte Vorgehensweise erfahrener Anwender. Dennoch bietet die Aussicht auf hybride Lösungen, die sich künstlicher Intelligenz bedienen, eine vielversprechende Zukunft im Bereich der Software-Testautomatisierung.

ABSTRACT

Many software companies have been focusing on software test automation for the last few years. Software tests based on user interfaces use different technologies for object identification, like identification by ID or identification by visual information. Using automation seems obvious especially when it comes to testing modern web applications. The reason for this is the use of responsive design in web development, as well as the representation of one web application on a broad range of different browsers and display sizes.

To understand how different technologies for object identification affect the stability of automated tests, various test automation tools were evaluated. Based on literature and online research the technologies for object identification were introduced and discussed. Multiple factors for assessing the stability of object identification were defined. The different ways for object identification were then compared against each other.

This evaluation did illustrate the differences between the various options for identifying user interface objects. While the automated creation of XPath values offers path weights for more reliable path creation, image recognition is very time consuming concerning the initial recording of elements. But in comparison to XPath or ID's, image recognition or OCR are more likely to be used for special cases, where technology limitation prevents the use of other technologies.

While a lot of software companies that focus on test automation have started introducing artificial intelligence lately, a lot of manual tasks are still necessary in test creation an maintenance. Hence it is still up to users to take care of those aspects of test automation, while no tool is able to take care of those tasks. Expert knowledge and experience in automation or software development are still necessary at the present moment. Nonetheless the prospect of hybrid solutions, combining test automation with artificial intelligence, is still very promising for the future of software testing.

INHALTSVERZEICHNIS

1 EINLEITUNG 1

1.1 Ausgangssituation 1
1.2 Forschungsfrage 1
1.3 Hypothesen 2
1.3.1 Hypothese 1 2
1.3.2 Hypothese 2 2
1.3.3 Hypothese 3 2

1.4 Zielsetzung 3
1.5 Motivation 3
1.6 Methode 4

2 EINFÜHRUNG IN DIE TESTAUTOMATISIERUNG 5

2.1 Was ist Testautomatisierung? 5
2.2 Einsatzgebiete der Testautomatisierung 5

3 TESTAUTOMATISIERUNG IM KONTEXT MODERNER WEBANWENDUNGEN 7

3.1 Komponenten moderner Webanwendungen 7
3.1.1 HTML5 7
3.1.2 CSS3 8
3.1.3 JavaScript 8
3.1.4 Dynamische Websites 8
3.1.5 Responsive Webdesign 9

3.2 Testautomatisierung bei Webanwendungen 9

4 TECHNOLOGIEN DER UI-OBJEKTERKENNUNG 11

4.1 Koordinaten 11
4.2 ID's 12
4.3 XPath und Document Object Model 12
4.4 Image Recognition 14

4.5 Optical Character Recognition 15

5 STABILITÄT AUTOMATISIERTER TESTS 16

5.1 Eigenschaften stabiler automatisierter Tests 16
5.2 Faktoren zur Beurteilung der Stabilität automatisierter Test 17
5.2.1 Eindeutige Identifikation 17
5.2.2 Identifikation bei Responsive Designs 17
5.2.3 Identifikation bei Cross-Browser Testausführung 18
5.2.4 Geschwindigkeit 18
5.2.5 Wiederherstellung nach unerwarteten Ereignissen 19
5.2.6 Identifikation nicht sichtbarer UI-Objekte 19
5.2.7 Verschachtelung von UI-Objekten 19

5.3 Aufstellen einer Metrik zur Beurteilung der Stabilität 20

6 TESTAUTOMATISIERUNGS-TOOLS UND DEREN MÖGLICHKEITEN ZUR UI-OBJEKTERKENNUNG 21

6.1 Auswahl der Testautomatisierungs-Tools für die Evaluierung der Stabilität 21
6.2 Auswahl der Webanwendung für die Evaluierung der Stabilität 23
6.2.1 Evaluierung der Stabilität anhand des Entwicklungstools Jira 23
6.2.2 Definition der Testdaten 25
6.2.3 Vorgehen der Evaluierung 25

7 UI-OBJEKTERKENNUNG ÜBER KOORDINATEN MIT RANOREX SPY UND TOSCA XSCAN 27

7.1 Ranorex Spy für die UI-Objekterkennung über Koordinaten 27
7.2 Evaluierung der Stabilität mit Ranorex Spy 29
7.2.1 Identifikation unter Voraussetzungen wie bei Aufzeichnung 29
7.2.2 Identifikation bei Testausführung mit anderem Browser 29
7.2.3 Identifikation unter Einsatz von Responsive Design 30
7.2.4 Geschwindigkeit der Identifikation und Zeitüberschreitungsfehler 30
7.2.5 Identifikation eines nicht sichtbaren Elements 30
7.2.6 Zusammenfassung in Beurteilungsmetrik 30

7.3 IBM Rational Functional Tester für die UI-Objekterkennung über Koordinaten 31
7.4 Vergleich zwischen Ranorex und IBM Rational Functional Tester 33

8 UI-OBJEKTERKENNUNG ÜBER ID'S MIT TOSCA XSCAN UND SELENIUM WEBDRIVER ... 34

8.1 Tosca XScan für die UI-Objekterkennung über ID's ... 34
8.2 Evaluierung der Stabilität mit Tosca XScan ... 35
8.2.1 Identifikation unter Voraussetzungen wie bei Aufzeichnung ... 36
8.2.2 Identifikation bei Testausführung mit anderem Browser ... 36
8.2.3 Identifikation unter Einsatz von Responsive Design ... 36
8.2.4 Geschwindigkeit der Identifikation und Zeitüberschreitungsfehler ... 37
8.2.5 Identifikation eines nicht sichtbaren Elements ... 37
8.2.6 Zusammenfassung in Beurteilungsmetrik ... 37

8.3 Selenium WebDriver für die UI-Objekterkennung über ID's ... 38
8.4 Vergleich zwischen Tosca XScan und Selenium WebDriver ... 39

9 UI-OBJEKTERKENNUNG ÜBER XPATH MIT RANOREX SPY UND TOSCA XSCAN ... 41

9.1 Ranorex Spy für die UI-Objekterkennung über XPath ... 41
9.2 Evaluierung der Stabilität mit Ranorex Spy ... 42
9.2.1 Identifikation unter Voraussetzungen wie bei Aufzeichnung ... 43
9.2.2 Identifikation bei Testausführung mit anderem Browser ... 43
9.2.3 Identifikation unter Einsatz von Responsive Design ... 43
9.2.4 Geschwindigkeit der Identifikation und Zeitüberschreitungsfehler ... 43
9.2.5 Identifikation eines nicht sichtbaren Elements ... 44
9.2.6 Zusammenfassung in Beurteilungsmetrik ... 44

9.3 Tosca XScan für die UI-Objekterkennung über Pfad ... 45
9.4 Vergleich zwischen Ranorex und Tosca ... 46

10 UI-OBJEKTERKENNUNG ÜBER IMAGE RECOGNITION MIT TESTCOMPLETE UND EGGPLANT FUNCTIONAL ... 47

10.1 TestComplete Image Set Editor für die UI-Objekterkennung über Image Recognition ... 47
10.2 Evaluierung der Stabilität mit TestComplete ... 49
10.2.1 Identifikation unter Voraussetzungen wie bei Aufzeichnung ... 50
10.2.2 Identifikation bei Testausführung mit anderem Browser ... 50
10.2.3 Identifikation unter Einsatz von Responsive Design ... 51
10.2.4 Geschwindigkeit der Identifikation und Zeitüberschreitungsfehler ... 51
10.2.5 Identifikation eines nicht sichtbaren Elements ... 51
10.2.6 Zusammenfassung in Beurteilungsmetrik ... 52

10.3 Eggplant Functional Image Viewer für die UI-Objekterkennung über Image Recognition 53
10.4 Vergleich zwischen TestComplete und Eggplant Functional 54

11 UI-OBJEKTERKENNUNG ÜBER OCR MIT TESTCOMPLETE UND EGGPLANT FUNCTIONAL 55

11.1 TestComplete Image Set Editor für die UI-Objekterkennung über OCR 55
11.2 Evaluierung der Stabilität mit TestComplete 56
11.2.1 Identifikation unter Voraussetzungen wie bei Aufzeichnung 56
11.2.2 Identifikation bei Testausführung mit anderem Browser 57
11.2.3 Identifikation unter Einsatz von Responsive Design 57
11.2.4 Geschwindigkeit der Identifikation und Zeitüberschreitungsfehler 57
11.2.5 Identifikation eines nicht sichtbaren Elements 58
11.2.6 Zusammenfassung in Beurteilungsmetrik 58

11.3 Eggplant Functional Image Viewer für die UI-Objekterkennung über OCR 59
11.4 Vergleich zwischen TestComplete und Eggplant Functional 61

12 AUSWIRKUNGEN DER UI-OBJEKTERKENNUNG AUF DIE STABILITÄT AUTOMATISIERTER TESTS 62

12.1 Einsatzgebiete unterschiedlicher UI-Objekterkennungs-Technologien auf Grundlage der Evaluierung 62
12.2 Wahl der Technologie zur UI-Objekterkennung 63

13 AUFWAND DER TESTERSTELLUNG UND WARTUNG 65

13.1 Kosten der Automatisierung als Entscheidungshilfe 65
13.2 Aufwand der Testerstellung unter Berücksichtigung der Technologien zur UI-Objekterkennung 66
13.3 Auswirkung der Stabilität auf den Wartungsaufwand automatisierter Test 67

14 AUSBLICK AUF INTELLIGENTE UI-OBJEKTERKENNUNG 68

14.1 Artificial Intelligence in der Software Testautomatisierung 68

15 SCHLUSSFOLGERUNG 70

ABKÜRZUNGSVERZEICHNIS 72

ABBILDUNGSVERZEICHNIS **73**

TABELLENVERZEICHNIS **74**

LITERATURVERZEICHNIS **75**

1 EINLEITUNG

Diese Arbeit befasst sich mit den verschiedenen Technologien der UI-Objekterkennung, welche bei automatisierten Software-Tests genutzt werden. In diesem Kapitel wird näher auf die Fragestellung, Motivation sowie Methodik eingegangen. Die der Arbeit zugrunde liegenden Hypothesen werden angeführt.

1.1 Ausgangssituation

Das Testen von Software stellt einen wesentlichen Aspekt der Software Entwicklung dar. Durch das Testen soll sichergestellt werden, dass der Kunde ein qualitativ hochwertiges Produkt erhält. Dabei kann das Testen viele Formen annehmen. Über einzelne Funktionen bis hin zu umfangreichen End-to-End Prozessen, wie auch Performance oder Usability, können diverse Komponenten der Software getestet werden, um ein paar Beispiele zu nennen.

Da es sich beim Testen von Software um einen kontinuierlichen und umfangreichen Prozess handelt, ist dieser mit entsprechend viel Aufwand verbunden. Seit einigen Jahren versucht man, diese Herausforderung durch Automatisierung zu meistern. Immer mehr Unternehmen setzen daher auf Testautomatisierungs-Tools. Eine Vielzahl von Anbietern stellt diese Tools für die unterschiedlichsten Anforderungen in Bezug auf die zu testende Software zur Verfügung. Die Schwerpunkte liegen dabei beispielsweise auf der zu testenden Plattform, handelt es sich etwa um mobile Apps oder Desktop Anwendungen, oder ob es sich bei den Software-Testern um Personen mit oder ohne Programmier-Kenntnissen handelt. Dabei gibt es sowohl kommerzielle, wie auch Open-Source Tools.

Da es eine Vielzahl an unterschiedlichen Software-Produkten gibt, die mittels Automatisierung getestet werden können, fokussiert diese Arbeit das Gebiet der Testautomatisierung im Kontext moderner Webanwendungen. Damit wird ein Teilbereich betrachtet, welcher allein bereits ein breites Spektrum an möglichen Ausprägungen abdeckt. In weiterer Folge werden Herausforderungen der Testautomatisierung im Bereich der Webanwendungen einer eingehenden Betrachtung unterzogen, welche sich zum Teil auch aufgrund der unterschiedlichen dahinterliegenden Technologien von Desktopanwendungen oder mobilen Apps unterscheiden.

1.2 Forschungsfrage

Diese Arbeit widmet sich der Frage, welche Auswirkungen der Einsatz unterschiedlicher Technologien zu UI-Objekterkennung auf die Stabilität der automatisierten Tests hat. Der Fokus liegt dabei auf modernen Webanwendungen, die neben mobilen Applikationen und Desktopanwendungen einen umfassenden Teilbereich der Testautomatisierung darstellen.

Um die Frage nach den Auswirkungen der UI-Objekterkennung zu beantworten, werden im ersten Schritt verschiedene Technologien für die UI-Objekterkennung vorgestellt und im weiteren Verlauf der Arbeit anhand praktischer Beispiele näher betrachtet und analysiert. Dabei werden die Vor- und Nachteile dieser Technologien anhand im Vorhinein definierter Kriterien betrachtet, welche der Beurteilung der Stabilität der Testfälle dienlich sein sollen.

1.3 Hypothesen

Für diese Arbeit wurden die nachfolgend angeführten Hypothesen formuliert. Es soll überprüft werden, ob die Alternativhypothese angenommen und folglich die Nullhypothese verworfen werden kann.

1.3.1 Hypothese 1

Alternativhypothese: Zwischen dem Einsatz unterschiedlicher Testautomatisierungs-Tools bzw. den ihnen zugrundeliegenden Technologien zur UI-Objekterkennung und der Stabilität bzw. dem Wartungsaufwand der Testfälle besteht ein Zusammenhang.

Nullhypothese: Die Auswahl des Testautomatisierungs-Tools und deren Vorgehensweise bzw. Technologien zur UI-Objekterkennung hat keine Auswirkung auf die Stabilität bzw. den Wartungsaufwand der Testfälle.

1.3.2 Hypothese 2

Alternativhypothese: Wenn mit UI-Elementen moderner Webanwendungen durch Testautomatisierungs-Tools interagiert werden soll, ist dies ohne Anpassungen des dahinterliegenden Codes möglich.

Nullhypothese: UI-Elemente moderner Webanwendungen können nicht ohne Anpassungen des dahinterliegenden Codes identifiziert werden.

1.3.3 Hypothese 3

Alternativhypothese: Der Nutzen, welcher aus der Einführung von Testautomatisierung im Testprozess bzw. im Hinblick auf die Qualitätssicherung erzielt werden kann, übersteigt den Aufwand, der für den initialen Aufbau und die Wartung anfällt.

Nullhypothese: Die Gegenüberstellung von Aufwand und Nutzen für die Einführung von Testautomatisierung, in Hinblick auf Optimierung des Testprozesses bzw. die Qualitätssicherung, zeigt keinen überwiegenden Nutzen.

1.4 Zielsetzung

Diese Arbeit soll aufzeigen, welche Technologien es zur UI-Objekterkennung bei der Software Testautomatisierung gibt. Die für die UI-Objekterkennung gewählte Technologie stellt einen wesentlichen Aspekt für die Erstellung, Wartbarkeit und Stabilität der automatisierten Testfälle dar. Es werden unterschiedlichste Technologien, wie etwa die grafische Bilderkennung oder die Identifikation der Elemente über dahinterliegende statische ID's, genutzt. Dabei gilt es, die Identifikation der Elemente der grafischen Benutzeroberfläche möglichst stabil zu gestalten, denn in der Regel werden diese Oberflächen immer wieder Änderungen unterzogen und dabei ändern sich die Elemente z.B. in ihrer Position, Optik oder Reihenfolge, was zu einem hohen Wartungsaufwand der automatisierten Testfälle führen würde. Daher sollte man sich dieser Herausforderung im Vorfeld bewusst sein und bei der Wahl des Testautomatisierungs-Tools bzw. der Technologie und Vorgehensweise zur Identifikation der Elemente der Benutzeroberfläche berücksichtigen.

Nachdem in dieser Arbeit eine Übersicht über die verschiedenen Technologien zur UI-Objekterkennung in der Testautomatisierung gegeben wurde, wird das Thema Stabilität von automatisierten Tests diskutiert. Verschiedene Faktoren werden herausgehoben, anhand deren die Stabilität solcher Tests bewertet werden soll. Diese Faktoren beschreiben beispielsweise, wie sich die UI-Objekterkennung verhält, wenn Änderungen an der grafischen Benutzeroberfläche durchgeführt wurden. Durch diese Metrik soll letztendlich ein Vergleich der verschiedenen Technologien zur UI-Objekterkennung erleichtert werden. Das Bewusstsein für die Herausforderungen der Testautomatisierung soll geschärft und eine Hilfestellung für die Wahl des passenden Tools, in Bezug auf die Technologie der UI-Objekterkennung, gegeben werden. Es soll aufgezeigt werden, wie wichtig es ist, sich bereits vor der Einführung eines Testautomatisierungs-Tools mit den Anforderungen bzw. dahinterliegenden Technologien der zu testenden Software auseinanderzusetzen und dies in die Entscheidung über die Einführung von Testautomatisierung sowie der Wahl des passenden Tools miteinzubeziehen.

1.5 Motivation

Automatisierte Tests möglichst resistent gegen Änderungen der grafischen Benutzeroberfläche zu gestalten ist eines von vielen Zielen der Automatisierung, denn der Wartungsaufwand kann sonst schnell sehr viel Zeit in Anspruch nehmen, die bei anderen Aufgaben sinnvoller eingesetzt wäre. Man möchte erreichen, dass automatisierte Tests auch nach Änderungen am Code bzw. der grafischen Benutzeroberfläche noch funktionieren und nicht daran scheitern, dass Elemente, wie beispielsweise Buttons oder Textfelder, plötzlich nicht mehr identifiziert werden können.

In dieser Arbeit soll ein tieferer Einblick in die Testautomatisierung gegeben werden, indem konkrete Technologien, welche diese Tools für die Identifikation der Elemente der grafischen Benutzeroberfläche nutzen, vorgestellt und analysiert werden.

1.6 Methode

Diese Arbeit verwendet einen Methodenmix aus quantitativer und qualitativer Methode. Die Grundlage bildet die Analyse von Fachliteratur und Online-Recherche rund um aktuelle Technologien und Komponenten des Software-Testprozesses. Daraus werden Kriterien zur Beurteilung der Stabilität von automatisierten Testfällen gebildet. Anhand dieser Kriterien werden verschiedene Technologien der UI-Objekterkennung evaluiert und gegenübergestellt. Was im Rahmen dieser Arbeit unter stabilen automatisierten Tests verstanden wird, wird im Vorhinein definiert. Die verschiedenen Technologien zur UI-Objekterkennung werden dabei im ersten Schritt in ihrer Funktionsweise beschrieben und im zweiten Schritt anhand konkreter Beispiele, welche mit verschiedenen Testautomatisierungs-Tools umgesetzt werden, näher betrachtet.

2 EINFÜHRUNG IN DIE TESTAUTOMATISIERUNG

Bevor näher auf die konkreten Möglichkeiten zur UI-Objekterkennung eingegangen wird, soll der Begriff Testautomatisierung kurz erläutert werden. Weiters wird im Folgenden angeführt, welche Bereiche des Testens von Software mittels Automatisierung umgesetzt werden können. Dazu zählen beispielsweise Regressionstests, also jene Tests, welche unabhängig der Änderungen der Software immer zu testen sind.

2.1 Was ist Testautomatisierung?

Testautomatisierung ist eine Vorgehensweise zur Ausführung von Software Tests. Gerade wenn es um agile Softwareentwicklung geht, ist Testautomatisierung kaum mehr wegzudenken. In einem solchen iterativen Entwicklungsprozess ist die regelmäßige Ausführung von Tests unablässig. Bei der agilen Softwareentwicklung werden Änderungen in zeitlichen Abständen von etwa einer Woche bis einem Monat eingespielt. Sobald dies passiert, muss getestet werden. Erst wenn alles wie gewünscht funktioniert, werden die nächsten Änderungen hinzugefügt. (Crispin & Gregory: 12)

Man kann die Automatisierung als Teilbereich des gesamten Software-Testprozesses betrachten, daher verfolgt sie in der Regel auch eigene Ziele. Beispielsweise soll durch die automatisierte Durchführung von Regressionstests die korrekte Funktionsweise der Software nach Einspielung von Änderungen am Code überprüft werden. In diesem Fall wäre nicht das Finden möglichst vieler Bugs das Ziel, in anderen Fällen wiederum kann genau dies das Ziel der Automatisierung sein. (Graham & Graham 2012: 2 ff.)

2.2 Einsatzgebiete der Testautomatisierung

Wie bereits in Punkt 2.1 erwähnt ist Testautomatisierung beispielsweise bei der Durchführung von Regressionstests ein naheliegendes und daher oft eingesetztes Werkzeug. Die Automatisierung ist aber auch in anderen Bereichen des Testens von Vorteil.

Auch für Performance Tests eignet sich die Automatisierung. Performance kann generell auf verschiedenste Arten getestet werden, wobei es dabei abhängig von der zu testenden Software verschiedene Performance Indikatoren gibt, welche die Leistung messbar machen, wie etwa Response Time oder wie viel Speicherplatz ein Installer benötigt. Um zu bestimmen, ob ein Performance Test erfolgreich war, muss der kleinste, noch akzeptable Schwellenwert definiert werden, ab welchem das Performance Ziel als erreicht eingestuft werden kann. Bei Indikatoren, die als hoch relevant betrachtet werden, spricht man von Key Performance Indikatoren. Bei einem Computerspiel wäre die Response Time beispielsweise ein wichtiger Faktor, die CPU Auslastung womöglich weniger interessant. (Laboon 2016: 169 - 171)

Ein weiteres mögliches Einsatzgebiet für Automatisierung stellen Security Tests dar. DevSecOps ist beispielsweise ein relativ junger Begriff, welcher sich von DevOps ableitet, allerdings den Security Aspekt noch miteinbezieht. Security rückt generell immer weiter in den Vordergrund, man möchte beispielsweise Schaden durch Sicherheitslücken vermeiden und auch Aspekte wie Datenschutz bestärken dieses Vorhaben. Bei DevSecOps wird Security bei dem Entwicklungsprozess bewusst miteinbezogen. Security Praktiken sind dabei fixer Bestandteil des Continuous Integration Vorgehens. (Hsu 2018: 7)

Während man bei anderen Arten von Tests überprüft, ob das vom Benutzer erwartete Ergebnis eintritt, möchte man durch Security Tests sicherstellen, dass die Anwendung nicht durch schädliche Eingaben auf unerwünschte Weise manipuliert wird. Ein Beispiel hierfür wäre der Download sensibler Daten durch einen nicht eingeloggten User. Man testet in diesem Sinne eine Auswahl möglicher Eingaben die nicht dem eigentlichen Zweck entsprechen und zu unerwünschtem Verhalten im Kontext der Sicherheitsvorgaben führen können. Ebenso wie beim Testen der Funktionalität gibt es hier einen definierten Input und ein erwartetes Ergebnis. Allerdings ist das Security Testing ein fortlaufender und stetig wachsender Prozess, da die Definition der erwarteten Ergebnisse weniger transparent und vorhersehbar ist als bei funktionalen Tests, welche sich nach den klar definierten Requirements der Kunden richten. (Hope & Walther 2008: 1 ff.)

Allerdings gibt es auch Bereiche des Testens, in welchen der Einsatz von Testautomatisierung nicht sinnvoll ist. Hierbei geht es beispielsweise um Komponenten, welche subjektiv beurteilt werden, wie etwa Usability.

Auch wäre es widersprüchlich, exploratives Testen automatisiert umzusetzen. Wie der Name schon verrät, geht es dabei um das Ausprobieren, um die Betrachtung von Komponenten der Software, die noch nicht durch andere Tests abgedeckt wurden. Bugs oder Verbesserungspotential soll aufgezeigt werden, an Stellen, an welchen noch kein Testfall definiert wurde. Denn für alle Sachverhalte einen Testfall zu definieren wäre ein beinahe endloser Prozess, der kaum Zeit ließe, noch Tests auszuführen. (Hendrickson 2013: 4 f.)

3 TESTAUTOMATISIERUNG IM KONTEXT MODERNER WEBANWENDUNGEN

Da die Einsatzgebiete der Testautomatisierung vielfältig sind, liegt der Fokus in dieser Arbeit auf Webapplikationen. Neben Desktopanwendungen und mobilen Applikationen ist dieser ein weiterer umfangreicher Teilbereich der Testautomatisierung.

Die Betrachtung von Webanwendungen schränkt dabei die Auswahl an verschiedenen Technologien zur UI-Objekterkennung für die Betrachtung in nachfolgenden Kapiteln der Arbeit nicht ein. Grundsätzlich sind diese Technologien für alle Plattformen, also Webanwendungen, Desktopanwendungen und mobile Applikationen, verfügbar. Letztendlich entscheidet, welche konkrete Technologie hinter der zu testenden Software liegt, denn auch hier gibt es die unterschiedlichsten Varianten und Kombinationen. Auf diese Technologien soll in diesem Kapitel näher eingegangen werden.

3.1 Komponenten moderner Webanwendungen

Moderne Webanwendungen stehen heutzutage vor vielen Herausforderungen, etwa die Darstellung der Inhalte auf verschiedensten Geräten. Daher greift man auf Technologien zurück, welche sich auf diese Herausforderungen eingestellt haben. Zu den Bausteinen moderner Webanwendungen zählen daher beispielsweise HTML5, CSS3 oder JavaScript. Diese Technologien werden in diesem Kapitel kurz vorgestellt. Allerdings handelt es sich hierbei nur um einen kleinen Auszug an Komponenten moderner Webanwendungen, tatsächlich gibt es natürlich ein breites Spektrum an Technologien die genutzt werden. Jedoch soll anhand dieser Beispiele repräsentativ auf die Anforderungen an moderne Webanwendungen und deren Aufbau eingegangen werden.

3.1.1 HTML5

Hypertext Markup Language wird zur Erstellung der Inhalte von Websites genutzt, diese textbasierte Sprache kann dabei von jedem Webbrowser interpretiert werden. HTML übernimmt die Aufgabe, Texte, Hyperlinks und Grafiken strukturiert darzustellen. Die aktuelle Version HTML5 umfasst aufgrund laufender Erweiterungen mittlerweile ein breites Spektrum an Technologien und kann nicht mehr als reine Auszeichnungssprache betrachtet werden. Daher wird HTML5 mittlerweile als Oberbegriff für HTML, JavaScript, CSS und weitere neue APIs betrachtet. HTML5 umfasst außerdem neue Features, wie die Wiedergabe von Video oder Audio, aber auch 2D- und 3D-Grafiken, die bisher noch nicht unterstützt wurden. (Wolf 2016: 39 f.)

3.1.2 CSS3

Während HTML den Inhalt einer Website strukturiert darstellt, wird CSS für das Design der Website herangezogen. Durch die Trennung von Inhalt und Design können beispielsweise unterschiedliche Designs für Desktop und mobile Geräte erstellt werden, oder das Design gespeichert und auf andere HTML5 Seiten angewandt werden. Außerdem können Webdesigner dadurch unabhängig von Programmierern an der Website arbeiten. In der aktuellen Version CSS3 können, ähnlich wie im Printbereich, Komponenten wie Schriften, Abstände, Farben, Ränder und Linien angepasst werden, um ein paar Beispiele zu nennen. (Bühler, Schlaich & Sinner 2017: 45)

3.1.3 JavaScript

JavaScript wird genutzt, um die Interaktionen des Benutzers mit Elementen der Website zu programmieren, die allein durch HTML nicht gegeben sind. Dadurch können komplexe Webanwendungen erstellt werden. Durch das Einbinden von Formularen, welche JavaScript zusätzlich auf gültige Eingaben überprüfen kann, können beispielsweise Daten an den Webserver übertragen werden. (Theis 2018: 17 f.)

Im Gegensatz zu Programmiersprachen wie C++ oder Java benötigt JavaScript keinen Compiler. Stattdessen wird der Code direkt über den Interpreter des jeweiligen Browsers ausgeführt. Code geschrieben in JavaScript wird innerhalb der HTML Dokumente geschrieben. Beispielsweise nutzt man dazu <script> Tags innerhalb des <head> Tags. Alternativ kann der JavaScript Code auch in eine separate Datei mit der Endung .js geschrieben werden, das <script> Tag verweist nur mehr auf diese Datei. Die schnellste Option ist aber, diesen Verweis an das Ende des Dokuments, bevor das <body> Tag wieder geschlossen wird, zu setzen. Auf diese Weise kann die Website schneller geladen werden. (Keith 2010: 7 ff.)

Es gibt auch mehrere Alternativen zu JavaScript. Ein Beispiel hierfür ist CoffeeScript, eine leicht lesbare Scriptsprache mit einfacher Syntax, welche zu JavaScript kompiliert wird. (CoffeeScript 2019a)

3.1.4 Dynamische Websites

Durch serverseitige Skriptsprachen werden Websites dynamisch erzeugt. Eine dynamische Website unterscheidet sich zur statischen Website dadurch, dass Inhalte wie Texte und Bilder getrennt werden vom dahinterliegenden Layout und Skripten. Erst beim Aufruf der Website werden die Inhalte aus der Datenbank gelesen und die Website dynamisch zusammengestellt. Das hat den Vorteil, dass Personen ohne Programmierkenntnisse die Inhalte der Website pflegen können und das Design an zentraler Stelle verwaltet werden kann. (Wolf 2016: 37, 43)

3.1.5 Responsive Webdesign

Eine der größten Herausforderungen ist immer noch die Darstellung der Webinhalte auf unterschiedlichen Geräten bzw. Displaygrößen. Ein wesentlicher Teil der Interaktion mit Webinhalten findet heutzutage auf mehreren Geräten statt, dennoch gibt es immer noch eine Vielzahl an Websites, deren Darstellung auf mobilen Geräten nicht optimal ausfällt. In diesen Fällen fehlt der Einsatz von Responsive Webdesign. Dabei bietet Responsive Webdesign viele Vorteile. Durch die nahtlose Anpassung der Webinhalte bei der Darstellung auf mobilen Geräten bekommt der Benutzer einen professionellen und vertrauenswürdigen Eindruck des Website Betreibers vermittelt. Die Inhalte und Größenverhältnisse der Elemente der grafischen Benutzeroberfläche werden dabei dynamisch an die Displaygröße angepasst und so optimal dargestellt. Responsive Webdesign ist allerdings keine einfache Aufgabe aus Sicht der Entwickler, denn diese sind dafür verantwortlich die Übergänge zwischen den verschiedenen Displaygrößen nahtlos erscheinen zu lassen. (Kinsbruner 2018: 135 f.)

3.2 Testautomatisierung bei Webanwendungen

Testautomatisierungs-Tools gibt es für Desktopanwendungen, mobile Applikationen und Webanwendungen. Viele solcher Tools unterstützen dabei das Testen auf allen drei dieser Optionen. Andere Tools wiederum sind auf eine bestimmte Plattform ausgerichtet, wie etwa das Open-Source Tool Selenium, welches zur Automatisierung von Webbrowsern verwendet wird.

Während man beim Testen von Desktopanwendungen und mobilen Applikationen die Anwendung im ersten Schritt erst installieren muss, reicht für das Aufrufen der Webanwendung die URL. Auch die Betrachtung der UI-Objekte fällt bei Webanwendungen in der Regel einfacher aus, denn hier reicht es, auf die Entwicklertools des Browsers zurückzugreifen, die einen sofort einen Einblick in den Aufbau der grafischen Benutzeroberfläche geben. Bei Desktopanwendungen hingegen hat man diese Option üblicherweise nicht. Zwar bieten Testautomatisierungs-Tool die Möglichkeit die UI-Objekte zu identifizieren, hier ist aber meist weniger Transparenz gegeben. Es gibt allerdings Tools, welche die Betrachtung der UI-Objekte von Desktopanwendungen ermöglichen, wie etwa das Windows-basierte Tool Inspect. (Windows 2019a)

Obwohl das Testen von Webanwendungen verglichen zu Desktopanwendungen und mobilen Applikationen in manchen Punkten weniger umständlich erscheint, gibt es auch hier diverse Herausforderungen zu bewältigen, denkt man beispielsweise an die breite Fülle an Geräten, über welche Webinhalte mittlerweile abgerufen werden können.

Beim Testen von Responsive Websites gibt es verschiedene Aspekte, die betrachtet werden können. Dazu zählen die Überprüfung des Designs, also ob die Darstellung auf unterschiedlichen Displaygrößen korrekt umgesetzt wird, sowie Kontrolle der Darstellung auf verschiedenen Browsern und Geräten. Performance Tests hinsichtlich der Zeit, welche die Website bei Laden der Elemente und Anpassen auf die Displaygröße benötigt, können durchgeführt werden. Auch Tests hinsichtlich störender Elemente, wie etwa Popups oder eingehende Anrufe beim Aufrufen

der Website über das Smartphone, können durchgeführt werden. Ein weiterer wichtiger Aspekt der beim Testen von Responsive Websites berücksichtig werden soll ist die Navigation, aber auch das Verhalten der Website bei Nutzung alternativer Eingabemöglichkeiten, wie etwa Spracherkennung, können überprüft werden. (Kinsbruner 2018: 137)

Die Vorteile der Testautomatisierung zeigen sich beispielsweise dann, wenn es darum geht die Funktionalität der Website auf unterschiedlichen Browsern zu testen. Das sogenannte Cross-Browser Testing kommt dabei zum Einsatz, welches von diversen Testautomatisierungs-Tools unterstützt wird. Dabei werden die Elemente der grafischen Benutzeroberfläche über einen Browser aufgezeichnet und der automatisierte Test anschließend auf verschiedenen Browsern ausgeführt.

SmartBear, der Hersteller des Testautomatisierungs-Tools TestComplete, ist beispielsweise ein Anbieter eines eigenen Cross-Browser Tools. Dabei handelt es sich um eine Cloud Lösung, die das Testen auf vielen unterschiedlichen Browsern zusätzlichen erleichtern soll. Dabei wird unter anderem das zuvor erwähnte visuelle Testen von Responsive Designs berücksichtigt, indem Screenshots beim Aufruf der Webinhalte über verschiedene Browser erstellt werden. Diese können anschließend einfach verglichen und auf unerwünschte Abweichungen, wie etwa im Layout, kontrolliert werden. Auch die Aufzeichnung von Tests und automatisierte Ausführung dieser, parallel auf mehreren Browsern, wird dabei unterstützt. (SmartBear 2019a)

4 TECHNOLOGIEN DER UI-OBJEKTERKENNUNG

Testautomatisierung befasst sich oft mit dem Testen der grafischen Benutzeroberfläche, denn diese ist sofort verfügbar und gibt überdies hinaus auch Aufschluss über die User Experience. Denkt man beispielsweise an das Testen auf Datenbank Ebene, so gewinnt man verglichen wenig Information über die Handhabung der Anwendung aus Sicht der Endanwender. Daher bietet ein Großteil moderner Testautomatisierungs-Tools über einen sogenannten „Recoder“ die Möglichkeit, UI-Objekte zu identifizieren sowie mit einer Interaktion zu verknüpfen und auf diese Weise Tests zu erstellen. Darüber hinaus schafft dies den Vorteil, dass für die UI-Objektaufzeichnung und Testerstellung nicht zwingend Programmierkenntnisse gefordert sind. (SmartBear 2019d)

In diesem Kontext werden die unterschiedlichen Möglichkeiten der UI-Objekterkennung vorgestellt. Dabei wird auf die grundlegende Funktionsweise der einzelnen Technologien eingegangen und es werden mögliche Einsatzgebiete, sowie Vorteile und Nachteile aufgezeigt. Dieses Kapitel dient als Grundlage für die spätere Evaluierung der UI-Objekterkennung, in welcher anhand konkreter Beispiele die Funktionsweise einzelner Technologien zur UI-Objekterkennung aufgezeigt wird.

4.1 Koordinaten

Die wohl simpelste Form der Objekterkennung ist die Verwendung der X und Y Koordinaten des Bildschirms. Dabei wird die Aktion des automatisierten Tests, beispielsweise ein Mouse Klick, an den vorgegebenen Koordinaten ausgeführt. Besonders für das Testen älterer Software, welche zwar gewartet aber grundsätzlich an ihrer grafischen Benutzeroberfläche nicht mehr verändert wird, ist diese Technologie interessant, wenn auch nicht immer anwendbar. (SmartBear 2019d)

Diese Aktion kann von dem Testautomatisierungs-Tool relativ schnell ausgeführt werden, da streng genommen keine Identifikation eines konkreten UI-Objekts im Hintergrund stattfindet, sondern lediglich die Position am Bildschirm vorgegeben wird. Dennoch soll diese Alternative hier angeführt werden, da sie, wie die nachfolgend vorgestellten Technologien das Ziel verfolgt, die Elemente der grafischen Benutzeroberfläche zu lokalisieren und damit zu interagieren.

Jedoch stößt diese Technologie schnell an ihre Grenzen. Selbst wenn sich die Position eines Elements nicht durch Code-Anpassungen verändert, reicht schon eine veränderte Bildschirmauflösung aus um das Element unauffindbar zu machen. Besonders wenn man an Responsive Design denkt, scheint die Identifikation über Koordinaten wenig sinnvoll.

Responsive Design beschäftigt sich mit der Frage, wie im Web abrufbare Inhalte auf unterschiedlichen Browsern oder Bildschirmen dargestellt werden. Dazu zählt auch die Darstellung auf mobilen Geräten, welche in den letzten Jahren stark an Bedeutung gewonnen hat. Bei eigens entwickelten Mobilversionen muss sich der Nutzer oft komplett neu orientieren und findet das Gesuchte womöglich trotzdem nicht, weil dies nur in der Desktopversion auffindbar ist. Statt duzender Eigenentwicklungen die auf verschiedenste Geräte optimiert wurden, wird aber

mit Responsive Design ein dynamischer Layoutraster genutzt, durch welchen sich der Inhalt automatisch der Displaygröße anpasst. Während Mobilversionen von Websites üblicherweise eine inhaltlich und im Funktionsumfang reduzierte Version der Website darstellen, passt sich beim Responsive Design der originale Inhalt der Website auf die mobile Displaygröße an. (Ertel & Laborenz 2017: 21, 25)

4.2 ID's

Die Identifikation der Elemente der Benutzeroberfläche über ID's ist naheliegend. Meist verfügen die Elemente der grafischen Benutzeroberfläche über eine eindeutige Kennung, beispielsweise über ein ID-Attribut, über welche die Interaktion des automatisierten Tests durchgeführt werden kann.

Über JavaScript kann nach bestimmten HTML Elementen gesucht werden. Die Methode *getElementById()* ist eine der Optionen, alternativ kann man beispielsweise auch *getElementByClassName()* oder *getElementByTagName()* verwenden. Über die ID zu suchen ist der einfachste, wie auch schnellste Weg ein Element zu finden. (Wolf 2016: 797)

Allerdings gibt es diverse Frameworks, welche standardmäßig mit dynamischen ID's arbeiten und diese Art der UI-Objekterkennung somit erschweren oder gar unmöglich machen, wenn die Vergabe der ID's vollkommen willkürlich stattfindet.

Ein Beispiel für die Verwendung dynamischer ID's ist die Yahoo User Interface Library, kurz YUI Library genannt. Dabei handelt es sich um eine Open Source JavaScript und CSS Library. Wird ein Element über dessen ID identifiziert und die Website dann erneut geladen, hat das Element eine andere ID und kann nicht mehr über das Testautomatisierungs-Tool gefunden werden. (Ranorex 2019a)

4.3 XPath und Document Object Model

Eine weitere Möglichkeit zur Identifikation der UI-Objekte kann über den Aufbau der Website stattfinden. Dafür werden die Elemente der grafischen Benutzeroberfläche in ihrer Struktur betrachtet.

Das Document Object Model, kurz DOM genannt, beschreibt den Aufbau der Website, die beim Aufruf durch den Benutzer in den Browser geladen wird. Dieser Aufbau wird als Tree, also in Baumstruktur, dargestellt. Zwischen den einzelnen Elementen bestehen folglich Beziehungen wie parent, child oder sibling. Durch Verschachtelung können komplizierte Beziehungen abgebildet werden. (Keith 2010: 31 f.) In nachfolgender Abbildung sieht man beispielhaft den vereinfachten Aufbau einer Website in Baumstruktur.

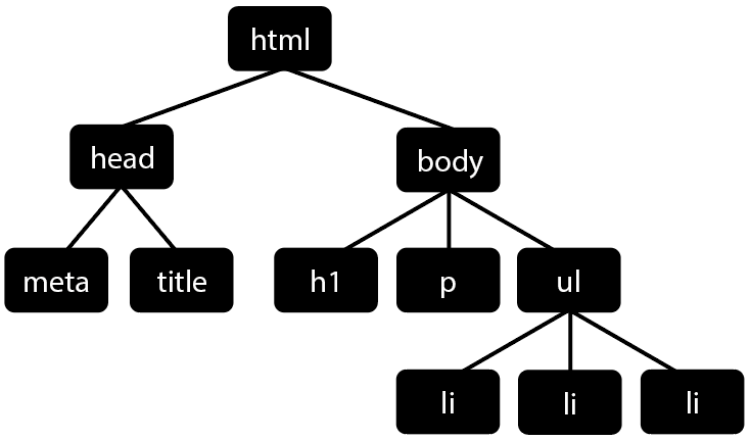

Abbildung 1: Vereinfachtes DOM Beispiel (Keith 2010: 33)

In Abbildung 1 erkennt man schnell die Beziehungen zwischen den einzelnen Elementen, so handelt es sich beispielsweise bei *head* und *body* um silbings und bei *h1*, *p* und *ul* um children von *body*.

Das DOM kann also zur Beschreibung des Aufbaus einer Website genutzt werden. Um zu den einzelnen Elementen zu navigieren, die sich in dieser Baumstruktur befinden, werden Pfade definiert.

Manche Testautomatisierungs-Tools wie etwa Selenium bieten die UI-Objekterkennung sowohl über DOM als auch über XPath an. Jedoch wird die Identifikation über DOM kaum mehr genutzt, da xPath alle Funktionen eines DOM abdeckt sowie darüber hinaus noch mehr Optionen für die Objekterkennung anbietet. Die Identifikation über DOM kann aber bei sogenanntem Legacy Code, also in der Regel älterem Code von Software die nicht mehr gewartet wird, eingesetzt werden. (Selenium 2019c)

Über XPath können die einzelnen Elemente einer XML oder HTML Struktur angesprochen werden. Dabei werden verschiedene Faktoren bei der sogenannten XPath Expression berücksichtigt. Das Ergebnis einer XPath Expression, welche meistens aus einem Pfad besteht, ist entweder ein Set aus Nodes, ein String, ein boolscher oder ein numerischer Wert. Die XPath Expression kann neben dem Pfad aber auch noch andere Funktionen ausführen. Dazu zählen unter anderem Vergleichsoperatoren, wie etwa „=“, „>“ oder „!=“. Auch arithmetische Operatoren wie „+“ oder „-“ können genutzt werden, ebenso wie logische Ausrücke wie „and“ oder „or“. Auch die Beziehungen zwischen den Elementen sind Bestandteil von XPath. So kann zwischen den verschiedenen Achsen der Baumstruktur mit „::“ navigiert werden, beispielsweise werden mit „child::node“ die darunterliegenden Elemente des aktuellen Nodes gewählt. (Melton & Buxton 2006: 217 - 226)

Das World Wide Web Consortium, auch als W3C bekannt, beschäftigt sich mit der Standardisierung diverser Komponenten des World Wide Web. Darunter fällt auch die Empfehlung für die Definition von XPath, welche zum Zeitpunkt der Fertigstellung dieser Arbeit in der Version XPath 3.1 vorhanden ist. W3C beschreibt XPath als ein Subset von XQuery, auch XML Query Language genannt, welche eine Sprache zur Abfrage von XML Datenbanken ist. (W3C 2017)

Es wird empfohlen, den kürzest möglichen Pfad zu wählen. Die Geschwindigkeit soll dabei optimiert werden, während gleichzeitig die bestmögliche Stabilität erzielt wird. Dabei soll darauf geachtet werden, dass jenes Element in der Baumstruktur als Anhaltspunkt zur UI-Objekterkennung angesprochen wird, welches nahe beim Zielobjekt liegt, beispielsweise das Parent oder Grandparent Element. (Ranorex 2018)

Die Vorteile von xPath gegenüber der UI-Objekterkennung über ID's oder Attribute des Elements zeigen sich dann, wenn das UI-Objekt abhängig von einem anderen Objekt ist. Beispielsweise kann sich die Suche dann auch einen bestimmten Seitenabschnitt beziehen, wie etwa die Validierung eines Logos im Seiten-Header. (Selenium 2019c)

Denkt man beispielsweise auch an die in Kapitel 4.3 angesprochenen dynamischen ID's, kann man die Identifikation über den Pfad als Alternative zu Identifikation über ID's in Betracht ziehen. Denn auch wenn die ID nicht eindeutig ist, da sie sich beim erneuten Laden der Website ändert, kann es dennoch einen eindeutigen Pfad geben, da sich andere Attribute des zu identifizierenden Elements nicht ändern.

4.4 Image Recognition

Bei der Image Recognition, also der grafischen Bilderkennung, werden die UI-Objekte anhand ihrer optischen Merkmale identifiziert. Beispielsweise wird von einem Button ein Screenshot erstellt. Wird der automatisierte Test ausgeführt, gleicht das Testautomatisierungs-Tool Elemente der grafischen Benutzeroberfläche Pixel für Pixel mit dem Screenshot ab.

Eine mögliche Technologie hinter der Bilderkennung ist Grey-Level Segmentation, auch Schwellenwertverfahren oder „thresholding" genannt. Diese Technologie kann den Prozess der Bilderkennung erheblich erleichtern. Dabei wird die Bildinformation Pixel für Pixel auf Graustufen reduziert. Bei angrenzenden Pixel mit ähnlichem Grauwert kann man auf zusammenhängenden Flächen schließen. Man kann diese Information weiter reduzieren und so, je nach Schwellenwert, die Pixel entweder in Schwarz oder Weiß umwanden. Der Schwellenwert kann beispielsweise auf dem Mittelwert der Graustufen des Bildes oder dem Verhältnis zwischen hellen und dunklen Pixel beruhen. (Parker 2010: 137 f.)

Bei der Aufzeichnung solcher Elemente der GUI muss beispielsweise auch darauf geachtet werden, welcher Bildausschnitt gewählt wird, oder im welchem Status das Element aufgezeichnet wird. Manche Elemente verändernd ihr Aussehen bei Interaktion, wie etwa ein Button der geklickt und danach farblich herausgehoben wurde.

In bestimmten Situationen kann diese Art der UI-Objekterkennung sinnvoll sein, beispielsweise, wenn einzelne Elemente nicht über ID's oder ihren Inhalt eindeutig identifizierbar sind. Jedoch sollte man hinterfragen, ob es nicht doch noch eine andere Alternative gibt, welche fehlertoleranter ist. Denn meist gibt es zwar einen kleinen Toleranzbereich bezüglich Abweichungen der Bildinformationen, aber dieser ist meist so gering, dass das Element bei kleinen optischen Veränderungen nicht mehr identifiziert werden kann.

Ebenso wie beim Verfahren der UI-Objekterkennung über Koordinaten, kann auch Image Recognition sehr fehleranfällig bei dynamischen Layouts reagieren. Durch die Verschiebung der Elemente im Layout können sich bestimmte UI-Objekte beispielsweise neben ihrer Position auch in ihrer Skalierung ändern, was das Erscheinungsbild und folglich die Grundlage zur optischen Bilderkennung verändert und das Objekt unauffindbar macht. (Ranorex 2018)

Auch beim Rendern von Schriften auf verschiedenen Browsern können diese Textpassagen unterschiedliche Optik aufweisen. In solchen Situationen bietet es sich beispielsweise an, stattdessen über die Baumstruktur der Webseite auf den Text zuzugreifen statt Image Recognition zu nutzen. Dasselbe muss auch bei der Darstellung von Buttons etc. auf unterschiedlichen Browsern beachtet werden. (Testplant 2019b)

4.5 Optical Character Recognition

Optical Character Recognition (OCR) ist eine Technologie, welche ihre Anfänge bereits in den 1930er Jahren hat. Ziel von OCR ist es, mit der Hand oder Schreibmaschine geschriebene, wie auch gedruckte Buchstaben maschinell einzulesen. Diese Buchstaben sollen dabei erkannt werden. Da in den meisten Sprachen Zeichen verwendet werden, welche aus Linien und Kurven bestehen, konnte mit OCR eine Möglichkeit geschaffen werden, diese Zeichen maschinell zu erfassen. Zu diesen Sprachen zählen unter anderem Englisch, Latein und Deutsch. Heutzutage wird diese Technologie beispielsweise dazu genutzt, Dokumente wie Reisepässe, Rechnungen oder Briefe einzulesen. OCR kann als Subkategorie der Bilderkennung verstanden werden, denn auch hier geht es darum, bekannte Muster zu erkennen. Eine der größten Herausforderungen von OCR ist dennoch die Genauigkeit, mit welcher die Zeichen erkannt werden, so kann es dennoch immer wieder zu Fehlern kommen. (Chaudhuri et. al. 2018: 1)

Neben den eben genannten Einsatzgebieten ist OCR auch Bestandteil diverser Testautomatisierungs-Tools. TestComplete (SmartBear 2019b) oder Eggplant (Testplant 2019a) zählen beispielsweise zu den Testautomatisierungs-Tools, welche diese Technologie unterstützen. In Kapitel 6 wird näher auf den praktischen Einsatz von OCR zur UI-Objekterkennung eingegangen.

5 STABILITÄT AUTOMATISIERTER TESTS

Da in dieser Arbeit die verschiedenen Technologien zur UI-Objekterkennung anhand ihrer Eigenschaften zur Gestaltung stabiler automatisierter Tests beurteilt werden, wird in diesem Kapitel näher auf den Begriff Stabilität eingegangen. Es wird definiert, welche Eigenschaften stabile UI-Objekterkennung im Kontext dieser Arbeit ausmachen. Anhand dieser Kriterien werden in weiterer Folge verschieden Technologien und Testautomatisierungs-Tools im Zusammenspiel mit modernen Webanwendungen betrachtet und überprüft, welche Auswirkungen Änderungen der UI-Objekte bei der Ausführung der automatisierten Tests mit sich bringen.

Man will eine stabile Objekterkennung erzielen, die auf Änderungen der UI-Objekte reagieren kann, um so den Wartungsaufwand der Tests möglichst gering zu halten. Denn einer der wichtigsten Vorteile der Automatisierung ist die Wiederholbarkeit der Tests, denkt man beispielsweise an Regressionstests.

An dieser Stelle soll erwähnt werden, dass man zukünftig das Thema Artificial Intelligence, kurz AI genannt, im Kontext der UI-Objekterkennung vermehrt hören wird. In Kapitel 14 soll ein Ausblick in das Thema intelligente UI-Objekterkennung gegeben werden.

5.1 Eigenschaften stabiler automatisierter Tests

Bei der Stabilität automatisierter Test ist der Wartungsaufwand ein grundlegender Faktor. Da die Automatisierung dann sinnvoll ist, wenn es darum geht, Tests in großer Anzahl durchzuführen, möchte man den Aufwand für die Wartung der Tests minimieren. Bei modernen, agilen Vorgehensweisen in der Software Entwicklung, wie Continuous Integration, ist Testautomatisierung ein fester Bestandteil des Entwicklungsprozesses. Regressionstests, welche bei solchen Vorgehensweisen in sehr kurzen Zeitabständen durchgeführt werden, stehen vor der Herausforderung, auf die laufenden Veränderungen mit einer gewissen Anpassungsfähigkeit reagieren zu können. Letztendlich scheitert Testautomatisierung aber oft an den hohen Kosten für den Wartungsaufwand. Dies kann beispielsweise dann passieren, wenn Personen mit wenig bis keinen Programmierkenntnissen an der Erstellung der Tests maßgeblich beteiligt sind und kleine Änderungen in Folge dessen viele Anpassungen fordern. (Graham & Graham 2012: 4, 8)

Mehrere Faktoren können dabei ausschlaggebend für notwendige Änderungen zur Instandhaltung der automatisierten Tests sein. Besonders beim Testen moderner Webanwendungen kann beispielsweise der Umgang mit Responsive Designs eine Herausforderung darstellen. Dieses und weitere Beispiele werden im folgenden Kapitel näher betrachtet.

5.2 Faktoren zur Beurteilung der Stabilität automatisierter Test

Da in dieser Arbeit verschiedene Technologien zur UI-Objekterkennung im Kontext moderner Webanwendungen betrachtet und verglichen werden, ist nun eine Auflistung verschiedener Kriterien zur Beurteilung dieser Technologien angeführt. Dabei ist ein besonderer Fokus auf Faktoren gelegt, welche spezifisch für das Testen moderner Webanwendungen gelten, wie etwa der UI-Objekterkennung beim Testen einer Anwendung auf unterschiedlichen Browsern, auch „Cross-Browser“ genannt.

Die eindeutige Identifikation von Elementen der grafischen Benutzeroberfläche kann, auch wenn die Aufzeichnung über einen Recorder vergleichsweise simpel ist, schnell Fehler hervorrufen. Verschiebt sich beispielsweise das UI-Objekt, wurde das UI-Objekt noch nicht geladen oder wird die Technologie der Benutzeroberfläche verändert, muss der Testfall oft mit viel Aufwand nachgebessert werden. (SmartBear 2019d)

5.2.1 Eindeutige Identifikation

Als erster und grundlegender Faktor zur Beurteilung der Stabilität wird festgelegt, ob die Identifikation eines bestimmten Elements mit der betrachteten UI-Objekterkennung-Technologie grundsätzlich funktioniert. Dabei wird davon ausgegangen, dass weder bei der zu testenden Webanwendung noch über das Testautomatisierungs-Tool, Eingriffe in den Code notwendig sind. Es wird überprüft, ob das UI-Objekt bei Testausführung unter denselben Umständen wie bei Aufzeichnung des Objekts, wie beispielsweise der Wahl des Browsers oder der Bildschirmauflösung, gefunden wird.

5.2.2 Identifikation bei Responsive Designs

Kommt die UI-Objekterkennung bei Responsive Designs zum Einsatz, gilt es, mit verschiedenen Änderungen dieser UI-Objekte umgehen zu können. Durch das dynamische Layout können sich Faktoren wie Position oder Größe dieser Objekte ändern. Es gibt Tools, welche sich allein dem Simulieren der Darstellung von Websites auf verschiedenen Geräten bzw. Bildschirmgrößen widmen, wie etwa Google DevTools des Chrome Browsers (Google Developers 2019) oder der Material Design Resizer. (Material Design 2019)

Die simpelste Form der UI-Objekterkennung ist die Definition der Position bzw. der Pixelkoordinaten. Doch genau bei solchen Vorgehensweisen stößt man mit dynamischen Layoutrastern, wie sie auch bei Responsive Designs genutzt werden, schnell auf Probleme. (SmartBear 2019d)

Daher soll in Bezug auf dieses Kapitel überprüft werden, wie sich die verschiedenen Technologien der UI-Objekterkennung verhalten, wenn das Layout der zu testenden Webanwendung auf unterschiedliche Bildschirmgrößten angepasst wird.

5.2.3 Identifikation bei Cross-Browser Testausführung

Auch beim Testen auf verschiedenen Browsern kann es zu Abweichungen in der Darstellung der UI-Objekte kommen. Besonders soll die Unterstützung der Cross-Browsers Funktionalität der Testautomatisierungs-Tools, also der Möglichkeit des Testens auf allen gängigen Browsern, betrachtet und überprüft werden, welche Auswirkungen dies auf die Stabilität der UI-Objekterkennung hat.

Der Großteil moderner Testautomatisierungs-Tools unterstützt das Testen auf gängigen Browsern. Dazu zählen Chrome, Firefox, Internet Explorer und Safari, aber in weiterem Sinne auch das Testen auf mobilen Endgeräten. Berücksichtigt man die verschiedenen Versionen der einzelnen Browser sowie die Plattformen, auf welchen die Webanwendungen aufgerufen werden, kommt man aber schnell auf hunderte Kombinationen. Diese Komplexität ist für Entwickler, wie auch Tester, eine Herausforderung, bei welcher Automatisierung sinnvoll eingesetzt werden kann. Tatsächlich stellt das manuelle Testen aufgrund dieser Komplexität kaum mehr eine Alternative zum automatisierten Testen dar. (Micro Fokus 2016: 2 f.)

Um angemessen auf die Darstellungsabweichungen beim Aufrufen von Webanwendungen auf unterschiedlichen Browser zu reagieren, bietet es sich an, auf die Inhalte einzelner UI-Objekte zuzugreifen, beispielsweise über ID’s sowie Namens-Attribute oder XPath, statt Image Recognition anzuwenden. Generell wird in einem Cross-Browser Szenario nach Möglichkeit von Image Recognition abgeraten. Sollte dies jedoch nicht vermeidbar sein, bieten viele Testautomatisierungs-Tools die Option eine Fehlertoleranz bei der Bilderkennung zu definieren. Darüber hinaus sollte man im Vorhinein eine Namenskonvention festlegen, um stets eine Übersicht über möglicherweise Browser-spezifische die UI-Objekte zu bewahren. (Testplant 2019b)

Um festzustellen, wie sich die Cross-Browser Funktionalität auf die Stabilität der Objekterkennung auswirkt, wird die zu testende Webanwendung mit unterschiedlichen Browsern aufgerufen.

5.2.4 Geschwindigkeit

Besonders bei Webanwendungen kommt der Faktor Geschwindigkeit zu tragen. Die Möglichkeit einer Zeitüberschreitung, wenn ein UI-Objekt nicht schnell genug gefunden werden kann, muss betrachtet werden. Auch kann es passieren, dass das Testautomatisierungs-Tool das gesuchte UI-Objekt als gefunden betrachtet, jedoch die grafische Oberfläche noch nicht vollständig geladen und somit fälschlicherweise die Interaktion mit dem Objekt als erfolgreich verzeichnet wurde.

Konkret kann es sich dabei um sogenannte „false negative“ Fehler handeln, die durch eine Wait-Funktionalität des Testautomatisierung Tools, also dem Warten auf ein UI-Objekt, hervorgerufen werden. Dies kann ausgelöst werden durch eine langsame Netzwerkverbindung oder einem zeitaufwändigen Datenbankaufruf. Auch kann es passieren, dass UI-Objekte zwar in den Browser geladen, aber noch nicht dargestellt wurden. Man kann dem beispielsweise entgegenwirken, indem man auf das vollständige Laden einer Website wartet, statt mit willkürlichen hardcodierten und folglich wartungsaufwändigen Waits zu arbeiten. (Katalon Studio 2017)

5.2.5 Wiederherstellung nach unerwarteten Ereignissen

Unvorhersehbare Ereignisse erschweren die Automatisierung. Immer wieder kann es vorkommen, dass Pop-Ups, die nicht Teil des automatisierten Tests sind, erscheinen und das Testergebnis negativ ausfällt, obwohl korrekte Funktionalität gegeben ist.

Die Tosca Test Suite des Herstellers Tricentis bietet hierfür beispielsweise die „Recovery“ Funktionalität. Dabei werden Szenarien festgelegt, welche ausgeführt werden, wenn der automatisierte Test an bestimmten Stellen auf Fehler stößt. Durch das Definieren dieser Szenarien auf bestimmten Ebenen des Testfalls wird vorgegeben, welches Recovery Szenario ausgeführt wird. Ist etwa auf Ebene des Testschrittes, welcher den Fehler ausgelöst hat, kein Recovery Szenario definiert, wird gegebenenfalls auf der darüber liegenden Ebene nach einem solchen Szenario gesucht. Es können mehrere Recovery Szenarien definiert werden welche nacheinander ausgeführt werden, bis eines davon zu einem erfolgreichen Testergebnis geführt hat. (Tricentis Tosca 2019b)

Kommt es im Rahmen der im späteren Kapitel angeführten Evaluierung zu einem solchen Szenario, wird überprüft, ob und wie das Testautomatisierungs-Tool diese Störungen umgeht und wie viel Aufwand hinter der Definiton eines solchen Recovery Szenarios liegt.

5.2.6 Identifikation nicht sichtbarer UI-Objekte

Manche UI-Objekte, wie beispielsweise Drop-Down Elemente oder sogenannte Combo Boxen, welche neben der Drop-Down Funktionalität auch eigene Eingaben eines Users annehmen, sind klassische Beispiele für nicht sichtbare UI-Objekte. Auch Pop-Up Fenster können in diese Kategorie fallen. Die hier betroffenen UI-Objekte kommen üblicherweise dann zum Vorschein, wenn der User im Vorhinein eine bestimmte Aktion ausführt, wie etwa das Anklicken des Drop-Down Elements. Daher muss diese Aktion bei der Testerstellung berücksichtigt werden. Darüber hinaus muss darauf geachtet werden, gegebenenfalls von der Wait-Funktionalität des jeweiligen Testautomatisierungs-Tools Gebrauch zu machen, um auf die Sichtbarkeit des Elements zu warten. (Ranorex 2018)

Daher wird überprüft, ob nicht sichtbare UI-Objekte gefunden werden und ob es ohne Definition eines Waits zu Zeitüberschreitungsproblemen kommt. Aufgrund des Aufbaus der für die Evaluierung genutzten Webanwendung werden zur Überpüfung nicht sichtbarer Elemente jene Elemente genutzt, welche in einem Bereich der Darstellung liegen, die erst über Scrollen sichtbar wird.

5.2.7 Verschachtelung von UI-Objekten

Auch die Verschachtelung von UI-Objekten kann das automatisierte Testen erschweren. Beispielsweise können durch die Verwendung sogenannter inline frames oder „iframes“, also dem Inhalt einer fremden Quelle eingebettet in ein HTML Dokument, Probleme bei der Identifizierung herbeigeführt werden. Denn dadurch sind einzelne UI-Objekte innerhalb dieses

Bereichs schwer identifizierbar bzw. deren Text, obwohl für den Menschen lesbar, für das Testautomatisierungs-Tool nicht mehr zugreifbar. Tritt dieses Problem auf, kann man es umgehen, indem man dem Tool vorgibt, auf welcher Ebene auf die Inhalte des iframes zugegriffen werden kann. (Katalon Studio 2017)

5.3 Aufstellen einer Metrik zur Beurteilung der Stabilität

Anhand der in Kapitel 5.2 betrachteten Faktoren zur Beurteilung der Stabilität von automatisierten Tests, wird in diesem Kapitel eine Metrik erstellt. Diese soll im weiteren Verlauf der Arbeit dazu dienen, Testautomatisierungs-Tools und ihre UI-Objekterkennungs-Technologien anhand konkreter Beispiele einzustufen.

Beurteilungskriterium	**Ergebnis**
Identifikation unter Voraussetzungen wie bei Aufzeichnung (Browser, Plattform und Layout unverändert)	Erfolgreich / Nicht erfolgreich
Identifikation bei Testausführung mit anderem Browser	Erfolgreich / Nicht erfolgreich mit x Browsern Unterstützung aller gängiger Browser Ja / Nein
Identifikation bei Anpassung des Layouts / Responsive Design	Erfolgreich / Nicht erfolgreich
Geschwindigkeit der Identifikation / Zeitüberschreitungsfehler	Dauer Zeitüberschreitungsfehler / Kein Zeitüberschreitungsfehler
Identifikation eines nicht sichtbaren Elements (gegebenenfalls Definition eines Wait Parameters)	Erfolgreich / Nicht erfolgreich
Sonstiges (UI-Objekt verändert sich, iframe Inhalte, Pop-Ups, etc.)	Kommentar

Tabelle 1: Beurteilungsmetrik zur Beurteilung der Stabilität verschiedener UI-Objekterkennungs-Technologien

6 TESTAUTOMATISIERUNGS-TOOLS UND DEREN MÖGLICHKEITEN ZUR UI-OBJEKTERKENNUNG

In diesem Kapitel werden die in Kapitel 4 vorgestellten Technologien zur UI-Objekterkennung näher betrachtet. Es wird anhand konkreter Beispiele auf deren Funktionsweise eingegangen. Dafür werden verschiedene Testautomatisierungs-Tools ausgewählt, welche die jeweiligen Technologien unterstützen. Getestet wird anhand von Webanwendungen, um auf die in Kapitel 3 erläuterten Besonderheiten, wie auch Herausforderungen dieser Anwendungen einzugehen.

Dies dient als Grundlage zur Beurteilung der Stabilität der verschiedenen UI-Objekterkennungstechnologien. Es wird näher darauf eingegangen, was passiert, wenn sich verschiedene Komponenten des zu identifizierenden Elements der grafischen Benutzeroberfläche ändern, wie etwa die Position oder ID.

6.1 Auswahl der Testautomatisierungs-Tools für die Evaluierung der Stabilität

Um einen Überblick über die verschiedenen Testautomatisierungs-Tools zu erhalten, wird an dieser Stelle der aktuelle Gartner Magic Quadrant der Testautomatisierung betrachtet. Die Tools zur Betrachtung der verschiedenen Technologien zu UI-Objekterkennung werden primär aus dieser Übersicht gewählt. Ergänzend werden Open-Source Tools in Betracht gezogen.

Abbildung 2: Gartner Magic Quadrant der Testautomatisierung (Gartner 2018)

Der Magic Quadrant gibt einen schnellen Überblick über die Positionierung der verschiedenen Hersteller am Markt. Dieser teilt die verschiedenen Testautomatisierungs-Tools in die Kategorien Leaders, Challengers, Niche Players und Visionaries ein. Ein Leader zu sein bedeutet dabei nicht zwingend, dass es sich dabei um das beste Produkt für den Kunden handelt, ein Niche Player kann bestimmte Anforderungen möglicherweise sogar besser erfüllen. Der Magic Quadrant beurteilt die Hersteller anhand ihrer Fähigkeit, den Markt zu verstehen und zukunftsweisende Ideen einzubringen, sowie deren Möglichkeiten, diese mit den zur Verfügung stehenden Ressourcen umzusetzen. (Gartner 2019)

Eine Open-Source Alternative zu den im Gartner Quadranten genannten kommerziellen Tools ist beispielsweise Selenium, bzw. der Selenium Web Driver. Die Selenium Web Browser Automatisierung ermöglicht die Erstellung und Ausführung zum Testen diverser Webanwendungen. Dabei werden die gängigen Browser, wie beispielsweise Chrome, Firefox oder auch Safari, unterstützt. Die Verwendung von Selenium zur Testautomatisierung setzt

allerdings Programmierkenntnisse voraus. So wird beispielsweise Java als Programmiersprache zur Erstellung der Test empfohlen, aber auch viele andere Optionen angeboten. (Selenium 2019b) Als Möglichkeiten zur UI-Objekterkennung bietet Selenium verschiedene Strategien. Über ID's oder Namens-Attribute, Text in einem Link wie auch über XPath oder DOM kann das UI-Objekt gefunden werden. Selenium empfiehlt dabei die Verwendung von ID's und, wenn diese aussagekräftig sind, Namens-Attributen in Hinblick auf Performance und einfachere Lesbarkeit der Test Skripte. Die Identifikation über XPath hingegen benötigt in der Regel mehr Zeit für die Ausführung, ist aber in manchen Situationen sinnvoller als XPath oder DOM, wenn beispielsweise das UI-Objekt in einem bestimmten Abschnitt oder in Relation zu anderen Objekten auftritt. (Selenium 2019c)

6.2 Auswahl der Webanwendung für die Evaluierung der Stabilität

Um aufzuzeigen, wie unterschiedliche Testautomatisierungs-Tools die UI-Objekterkennung von Webanwendungen umsetzen, wird dieses Vorgehen anhand eines praktischen Beispiels aufgezeigt. Es wird eine Webanwendung gewählt, welche dem Vorbild moderner Webanwendungen entspricht bzw. auch Responsive Design nutzt.

Zur Evaluierung der Stabilität wird das Entwicklungstool Jira gewählt, welches im Bereich der Softwareentwicklung bereits von vielen Firmen genutzt wird. Dieses Tool dient unter anderem der Erstellung von Kanban- oder SCRUM-Boards zur Stützung des Entwicklungsprozesses. Die Anwendung ist in Java geschrieben und nutzt die in Kapitel 3 vorgestellten Technologien moderner Webanwendungen. (Atlassian 2019)

6.2.1 Evaluierung der Stabilität anhand des Entwicklungstools Jira

Für die Evaluierung wird ein Kanban Board in Jira erstellt und mit Testdaten befüllt. Dieses Board bzw. diese Testdaten werden anschließend manipuliert um festzustellen, wie sich die UI-Objekterkennung verhält, wenn beispielsweise Texte oder Positionen der Elemente verändert werden. Die grafische Benutzeroberfläche des Jira Boards für die Evaluierung ist wie in Abbildung 3 abgebildet aufgebaut.

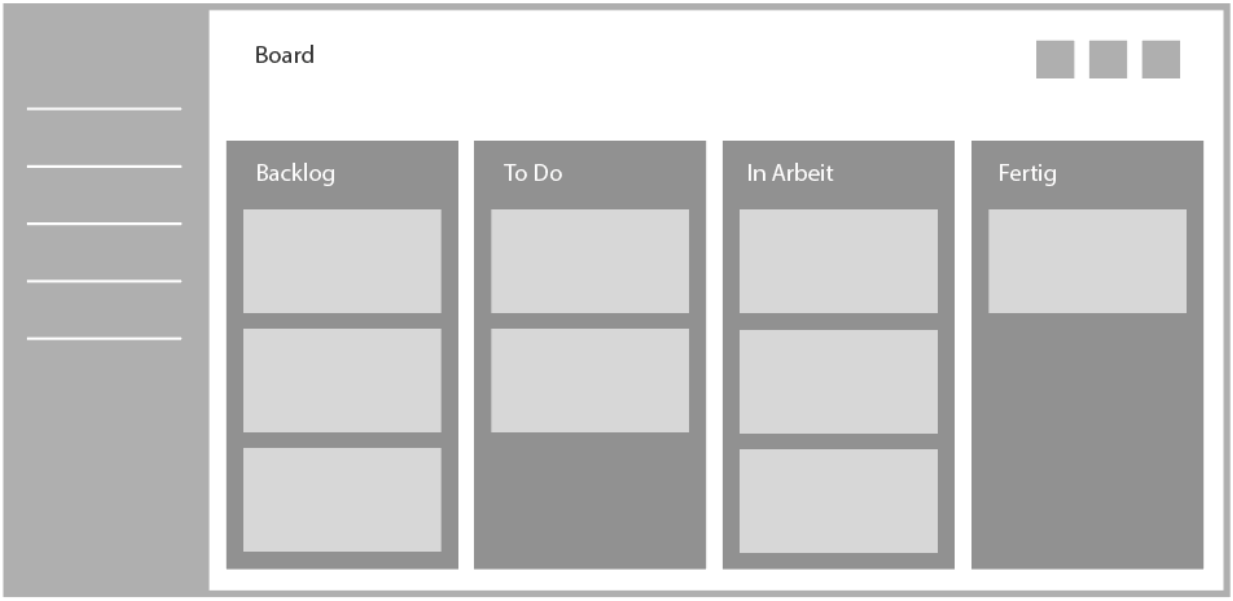

Abbildung 3: Aufbau eines Jira Boards (Atlassian 2019)

Das in Abbildung 3 abgebildete Board besteht aus einer konfigurierbaren Auswahl an Status-Bereichen, die wie in diesem Beispiel mit „Backlog“ oder „In Arbeit“ bezeichnet werden können. Innerhalb dieser Bereiche befinden sich einzelne Vorgänge, deren Aufbau Abbildung 4 zu entnehmen ist. Diese Vorgänge kann man je nach Status mittels Drag and Drop zwischen den Bereichen verschieben, beispielsweise von „In Arbeit“ nach „Fertig“. Für diese Bereiche kann man zusätzlich Limits setzen, so dass sich beispielsweise nur fünf Vorgänge im Bereich „In Arbeit“ befinden dürfen. Wird dieses Limit überschritten, wird dieser Bereich rot eingefärbt. Des Weiteren können zusätzliche Informationen zur Übersicht eines Vorgangs angezeigt werden, wenn man den Mouse Cursor über dessen einzelne Elemente platziert.

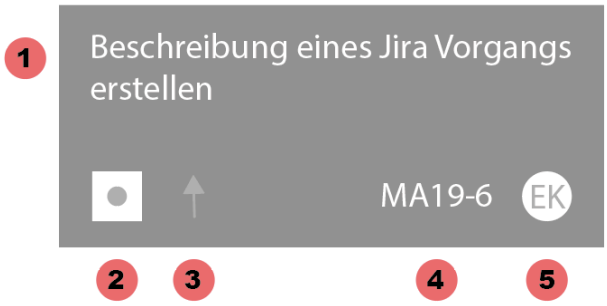

Abbildung 4: Aufbau eines Jira Vorgangs (Atlassian 2019)

1. An oberster Stelle steht der Titel des Vorgangs. Nimmt dieser Titel mehr als drei Zeilen in dieser Ansicht ein, wird der Text abgeschnitten. Mit einer Mouse Hover Aktion, also der Platzierung des Mouse Cursors auf dem Titel, wird der gesamte Text über dem Vorgang eingeblendet.

2. Es gibt verschiedene Arten von Vorgängen. Dieses Icon zeigt an, ob es sich beispielsweise um einen Bug, Task oder eine Story handelt.
3. Die Farbe des Pfeils zeigt dem Anwender, welche Priorität der Vorgang hat.
4. Diese alphanumerische Kombination ist die Nummer des Vorgangs. Der vordere Teil, in diesem Beispiel „MA19“, gibt Aufschluss über das Projekt, welchem die Vorgänge zugehören und wird aufgrund der Eindeutigkeit auch Projektschlüssel genannt. Der hintere Teil, also „6“, beschreibt die Nummer des Vorgangs.
5. Dieser Bereich verweist auf den Verfasser des Vorgangs. Hier kann sich ein Profilbild der Person, oder wie im Beispiel die Initialen des Verfassers befinden, sofern kein Profilbild gewählt wurde.

Klickt man in der Board Ansicht auf einen der Vorgänge, werden Detailinformationen, wie etwa die genauere Beschreibung des Vorgangs, angezeigt. Auch weitere Aktivitäten, wie etwa Kommentare oder die Historie der Statusänderungen, können hier eingesehen werden. (Atlassian 2019)

6.2.2 Definition der Testdaten

Um Beispieldaten für die Evaluierung zu generieren, wird in Jira ein Projekt „UITA“, kurz für User Interface Test Automation, sowie ein zugehöriges Kanban Board erstellt. Dieses enthält wie in Abbildung 3 dargestellt die vier Spalten „Backlog“, „To Do“, „In Arbeit“ und „Fertig“. Für die Spalte „To Do“ wird ein Minimum von zwei Vorgängen definiert, für die Spalte „In Arbeit“ ein Maximum von drei Vorgängen. Diese beiden Werte werden unter- bzw. überschritte, sodass die Spalten farblich hervorgehoben werden und sich visuell unterscheiden und von den restlichen Status-Bereichen abheben.

Es werden acht Vorgänge vom Typ „Task“ und vier Vorgänge vom Typ „Bug“ erstellt. Die Benennung wurde aus Gründen der leichten Nachvollziehbarkeit nach dem Schema „Task 3“ oder „Bug 5“ durchgeführt und wird gegebenenfalls für die Evaluierung einzelner UI-Objekterkennungs-Technologien angepasst. Etwaige notwendige Änderungen der Testdaten werden in den einzelnen Kapiteln erläutert.

Die Vorgänge erhalten automatisch generierte ID’s von „UITA-1“ bis „UITA-12“. Außerdem werden verschiedene Prioritäten vergeben, sodass jede Priorität zumindest einmal vorkommt. Die Prioritäten umfassen fünf Ausprägungen von „Highest“ bis „Lowest“.

6.2.3 Vorgehen der Evaluierung

Anhand der festgelegten Kriterien zur Beurteilung der Stabilität sowie der auf Basis des Gartner Quadranten der Testautomatisierung werden anschließend die UI-Objekterkennung-Technologien evaluiert. Es wird beschrieben, wie die UI-Objekterkennung mit unterschiedlichen Testautomatisierungs-Tools durchgeführt wird. Einige dieser Tools bieten Werkzeuge an, die sich

rein auf die UI-Objekterkennung fokussieren. Der Aufbau dieser Tools zur Objekterkennung sowie die konkrete Aufzeichnung einzelner UI-Objekte wird nachfolgend näher beschrieben.

Pro UI-Objekterkennungs-Technologie werden zwei Testautomatisierungs-Tool beispielhaft vorgestellt. Im jeweils ersten dieser Beispiele wird näher auf die thematisierte Technologie zur UI-Objekterkennung eingegangen, bevor die Evaluierung der Stabilität anhand konkreter Beispiele durchgeführt wird. Für diese Evaluierung wird auf verfügbare Testversionen dieser Tools zurückgegriffen, pro Technologie wird eines der beiden vorgestellten Tools angewendet. Das zweite vorgestellte Tool dient zum Aufzeigen der Unterschiede bei der Vorgehensweise der UI-Objekterkennung.

7 UI-OBJEKTERKENNUNG ÜBER KOORDINATEN MIT RANOREX SPY UND TOSCA XSCAN

Da die UI-Objekterkennung bzw. Interaktion mit UI-Objekten auf Basis von Pixelkoordinaten die wohl simpelste Form der Testautomatisierung beschreibt, ist davon auszugehen, dass Faktoren hinsichtlich Performance wie etwa Geschwindigkeit positiv ausfallen, die Stabilität jedoch an einigen Stellen Einbußen aufzeigen wird. Es wird betrachtet, wie die Testautomatisierungs-Tools Ranorex und die Tosca Testsuite die Identifikation in diesem Kontext umsetzen und wie sich die Interaktion verhält, wenn sich UI-Objekte aufgrund von Responsive Design oder dem Aufrufen der Webanwendung auf einem anderen Browser verändern.

7.1 Ranorex Spy für die UI-Objekterkennung über Koordinaten

Zur UI-Objekterkennung über das Testautomatisierungs-Tool Ranorex wird das sogenannte Ranorex Spy Tool genutzt, welches eigens dafür zur Verfügung gestellt und auch als Stand-Alone zur Aufzeichnung der UI-Objekte genutzt werden kann. Dieses erkennt Elemente von Desktop- und Webanwendungen wie auch Mobile Applications. Die identifizierten UI-Objekte werden dann in dem Ranorex Repository gespeichert, von wo aus sie als Basis für die Testerstellung dienen. In der nachfolgenden Abbildung 5 sieht man den Aufbau des Tools. (Ranorex 2019b)

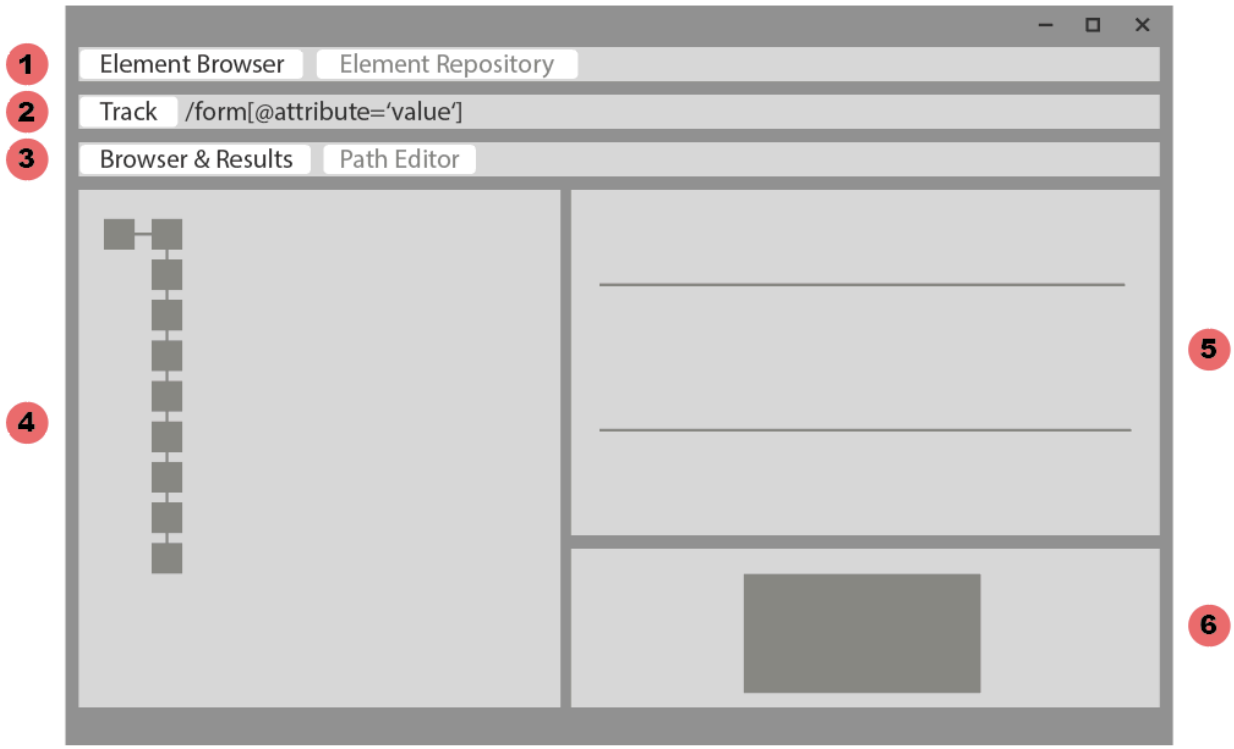

Abbildung 5: Ranorex Spy Aufbau (Ranorex 2019b)

1. Man kann vom Element Browser für die UI-Objekterkennung zum Element Repository wechseln, welches für die Ansicht des gesamten Repositories der UI-Objekte dient.

2. Über den Track Button werden einzelne UI-Objekte ausgewählt. In derselbe Zeile wird auch der zugehörige, identifizierte XPath bzw. RanoreXPath angezeigt.
3. Standardmäßig ist die Ansicht der aufgezeichneten UI-Objekte aktiviert. Es kann in den sogenannten Path Editor gewechselt werden, in welchem der aufgezeichnete XPath, bei Ranorex auch RanoreXPath genannt, bearbeitet werden kann.
4. Die UI-Objekte, welche bei der ausgewählten Anwendung identifiziert wurden, werden in diesem Bereich in Baumstruktur gelistet. Hierarchisch an oberster Stelle steht das Root Element. Alle weiteren aufgezeichneten UI-Objekte müssen diesem Root-Element untergeordnet sein.
5. Dieser Bereich umfasst eine Auflistung von Detailinformationen, wie beispielsweise grafische Informationen, zum ausgewählten UI-Objekt. Hier ist unter anderem auch die Größe und Position des UI-Objekts abzulesen. Des Weiteren kann von der „Overview“ zu einer „Advanced“ Ansicht gewechselt werden. Dort werden weitere Detailinformationen zu dem UI-Objekt gelistet.
6. Eine grafische Abbildung des ausgewählten UI-Objekts wird als Vorschau in diesem Fenster angezeigt.

Die Aktionen im Test über Pixelkoordinaten zu steuern kann beispielsweise dann von Vorteil sein, wenn der Mauscursor relativ zur letzten Position bewegt werden muss. Beispiele hierfür sind die Interaktion mit Kontextmenüs oder Drag & Drop Aktionen.

Um an der gewünschten Position eine Interaktion, in diesem Fall einen einfachen Mouse Klick, auszuführen, kann beispielsweise eigener User Code mit der entsprechenden Mouse.Click Methode angelegt werden. Als Programmiersprache kann bei Ranorex zwischen C# und VB.NET gewählt werden. Der Ranorex Namespace bietet unter anderem nachfolgend genannte Funktionen zur Interaktion mit UI-Objekten über Koordinaten an. (Ranorex 2019d)

- *Location()* kann eine neue Position am Bildschirm festlegen. Als Parameter können beispielsweise zwei Integer Werte übergeben werden, welche als Pixelkoordinaten dienen. Wird stattdessen ein Parameter von Typ PointF aus dem .NET System.Drawing Namespace übergeben, kann beispielsweise auch eine Position proportional zur Größe des UI-Objekts vorgegeben werden.
- *CenterRight* oder *LowerCenter* sind Beispiele für Positionen, an denen eine Interaktion mit einem UI-Objekt durchgeführt wird.
- *GetElementLocation()* liefert als Ergebnis die Koordinaten des als Parameter übergebenen UI-Objekts. Über einen Boolean Wert kann zusätzlich definiert werden, ob die Koordinaten relativ zum Element oder absolut zur Bildschirmgröße zurückgeliefert werden.

7.2 Evaluierung der Stabilität mit Ranorex Spy

Auf Basis der in Kapitel 5 definierten Kriterien für stabile UI-Objekterkennung wird Ranorex Spy anhand von Testdaten näher betrachtet. Da der Fokus dieser Betrachtung auf der UI-Objekterkennung über Koordinaten liegt, ist davon auszugehen, dass die Resultate dieser Evaluierung bei Cross-Browser Testing und besonders bei Responsive Designs zu Teilen nicht erfolgreich ausfallen. Für die Evaluierung werden zwei unterschiedliche UI-Objekte der Webanwendung aufgezeichnet. Es werden die absolute Position am Bildschirm sowie die relative Position eines UI-Objekts getestet. Es werden ein Status-Bereich mit der absoluten Position, sowie ein Vorgang innerhalb des Status-Bereichs mit der relativen Position betrachtet.

Für die Aufzeichnung wird über Ranorex Studio der Ranorex Spy gestartet. Über die Track Funktion wird ein Jira Vorgang ausgewählt. In der Ranorex Spy Ansicht sieht man dessen Properties. Dort sind auch die Location bzw. die relativen Koordinaten, wie auch die ScreenLocation, also die absoluten Koordinaten, zu finden. Im ScreenRectangle findet man die Koordinaten sowie die Größe des UI-Objekts und kennt somit dessen gesamten Bereich am Bildschirm.

Für die Evaluierung werden Koordinaten sowie Größe der getesteten UI-Objekte berücksichtigt. Die Objekterkennung gilt als erfolgreich, wenn sich das Element im aufgezeichneten Bereich befindet.

7.2.1 Identifikation unter Voraussetzungen wie bei Aufzeichnung

Die aufgezeichneten UI-Objekte verändern beim erneuten Aufrufen der Website ihre Position nicht, daher werden die Elemente erfolgreich identifiziert bzw. lokalisiert. Da für die Testdaten ein Kanban Board genutzt wurde welches sich in der Regel in kurzen Abständen ändert ist aber davon auszugehen, dass dies in der Praxis nicht sinnvoll ist.

7.2.2 Identifikation bei Testausführung mit anderem Browser

Ranorex unterstützt die Browser Firefox, Chrome, Edge, Internet Explorer sowie Safari und bietet somit das Testen einiger der gängigsten Browser. Zur Überprüfung dieses Beurteilungskriteriums werden die Browser Firefox, über welche die UI-Objekte aufgezeichnet wurden, sowie Chrome und Internet Explorer zur Überprüfung genutzt.

Es sind kleine Abweichungen bei den Positionen der Elemente zwischen den Browsern zu bemerken. Der Status-Bereich ist bei Chrome drei Pixel weiter links und der Vorgang drei Pixel höher, ansonsten sind die Werte zu Größe und Position ident. Bei Chrome befinden sich der Status-Bereich sowie der Vorgang drei Pixel weiter als bei Firefox. Die relative Position des Vorgangs in Abhängigkeit zum Bereich bleibt identisch.

Die Abweichungen der absoluten Position sind minimal sowie bei der relativen Position nicht vorhanden, daher wird dieses Beurteilungskriterium als erfolgreich betrachtet.

7.2.3 Identifikation unter Einsatz von Responsive Design

Um zu testen, wie die UI-Objekterkennung bei Responsive Design funktioniert, wird die Größe des Browserfensters angepasst. Dabei wird darauf geachtet, dass sich die Elemente auch durch den dynamischen Layoutraster verändern, sodass ein Unterschied in der Darstellung ersichtlich ist. Das Browserfenster wird auf zwei Drittel der Breite des Vollbild-Modus verkleinert.

In der anschließenden Identifikation der Elemente können diese nicht mehr korrekt lokalisiert werden, da sich die Position maßgeblich verändert hat. Das UI-Objekt des Jira Vorgangs hat sich ebenso wie der Status-Bereich durch die Anwendung des Responsive Design auf etwa 60 Prozent der Breite im Vergleich zum Vollbild-Modus reduziert.

7.2.4 Geschwindigkeit der Identifikation und Zeitüberschreitungsfehler

Da Koordinaten zur UI-Objekterkennung genutzt werden und kein Zugriff auf DOM bzw. die HTML Elemente erfolgt, gibt es keinen Prozess der von XSpy geladen werden muss. Daher wird dieses Beurteilungskriterium von der Evaluierung ausgeschlossen.

7.2.5 Identifikation eines nicht sichtbaren Elements

Die Koordinaten umfassen nur den sichtbaren Bereich. Daher kann über Koordinaten nicht mit UI-Objekten interagiert werden, welche außerhalb des dargestellten Bereichs im Browserfenster liegen. Zwar ist über Testautomatisierungs-Tools das Scrollen innerhalb dieser Bereiche möglich, jedoch ist zu hinterfragen wie sinnvoll hier die Interaktion über Koordinaten wäre.

7.2.6 Zusammenfassung in Beurteilungsmetrik

Beurteilungskriterium	**Ergebnis**
Identifikation unter Voraussetzungen wie bei Aufzeichnung (Browser, Plattform und Layout unverändert)	Erfolgreich
Identifikation bei Testausführung mit anderem Browser	Erfolgreich getestet mit Firefox, Chrome, Internet Explorer Unterstützung der Browser Browser Firefox, Chrome, Edge, Internet Explorer, Safari
Identifikation bei Anpassung des Layouts / Responsive Design	Nicht erfolgreich
Geschwindigkeit der Identifikation / Zeitüberschreitungsfehler	Nicht gemessen da kein Zugriff auf DOM/HTML

	Keine Fehler in Ranorex angezeigt
Identifikation eines nicht sichtbaren Elements (ggf. mit Wait)	Nicht gemessen da kein Zugriff auf nicht sichtbaren Bereich
Sonstiges (UI-Objekt verändert sich, iframe Inhalte, Pop-Ups, etc.)	Nicht aufgetreten

Tabelle 2: Evaluierung der UI-Objekterkennung über Koordinaten mit Ranorex Spy

In manchen Situationen bietet sich die UI-Objekterkennung über Koordinaten an, wenn etwa keine Änderungen des UI zu erwarten sind oder relative Positionen genutzt werden. In Fällen von Kontextmenüs oder Mouse-Hover Aktionen kann der Einsatz dieser Technologie zur UI-Objekterkennung sinnvoll sein. Dennoch ist diese UI-Objekterkennungs-Technologie stark fehleranfällig, besonders wenn man einen Vergleich mit den alternativen Technologien überlegt.

Die Aufzeichnung der UI-Objekte gestaltet sich unkompliziert, da Ranorex über Ranorex Spy ein eigenständiges Tool zur Aufzeichnung und Verwaltung dieser Elemente zur Verfügung stellt. Über den Track Button können einzelne UI-Objekte auf der Webanwendung selektiert werden, hier ist darauf zu achten auch den richtigen Bereich auszuwählen. Entsprechend der HTML-Komponenten werden einzelne Bestandteile der UI-Objekte während der Identifikation farblich herausgehoben, welche Aufschluss über den identifizierten Bereich geben. Im Ranorex Spy ist schließlich der XPath des identifizierten Elements zu finden, wie auch verschiedene andere Properties, darunter die relative und absolute Position am Bildschirm.

7.3 IBM Rational Functional Tester für die UI-Objekterkennung über Koordinaten

Auch IBM bietet diverse Tools, die den Prozess der Software-Entwicklung unterstützen. Eines davon ist das Testautomatisierungs-Tool Rational Functional Tester (RFT), welches auch das UI-Testen von Webanwendungen umfasst. Neben der Erstellung von Testscripts ist auch die Aufzeichnung der UI-Objekte über einen Recorder möglich. Die Browser Firefox, Internet Explorer, Chrome und Safari werden von dem Testautomatisierungs-Tool unterstützt.

Der Rational Functional Tester umfasst zwei sogenannte Perspektiven: Die Functional Test Perspektive wird grundsätzlich zum Testen von nativen Anwendungen genutzt, während die Web UI Test Perspektive für die Aufzeichnung von Web-Tests wie etwa auf HTML5, CSS3 oder JavaScript Basis verwendet wird. Für die Web Perspektive gibt es eine Eclipse Integration des Rational Functional Testers, es kann Java für das Scripting oder ein Recorder für die Aufzeichnung der UI-Objekte genutzt werden. Das Tool umfasst dabei die Aufzeichnung der UI-Objekte sowohl auf Desktop wie auch auf mobilen Geräten. Die Tests können parallel auf mehreren Browsern sowie mobilen Geräten ausgeführt werden, dies kann auch über die

zugehörige Cloud-Lösung durchgeführt werden. Des Weiteren zählt auch der IBM Ration Functional Tester zu jenen Tools, welche Selenium unterstützen. (IBM 2019a)

Zur Verwaltung der aufgezeichneten UI-Objekte wird die sogenannte Test Object Map genutzt. Diese Ansicht umfasst, ähnlich der anderen in Kapitel 7 vorgestellten Tools, eine hierarchische Übersicht der aufgezeichneten UI-Objekte in Baumstruktur und kann dabei auch die Elemente mehrerer Anwendungen umfassen. Wird ein Element dieser Liste gewählt, werden weitere Informationen, wie Index, ID oder Text des Elements angezeigt. Man hat in dieser Ansicht außerdem die Möglichkeit, die Informationen der UI-Objekte anzupassen. Werden Test Scripte erstellt, wird auf die in der Object Map verwalteten Elemente zugegriffen. (IBM 2019b)

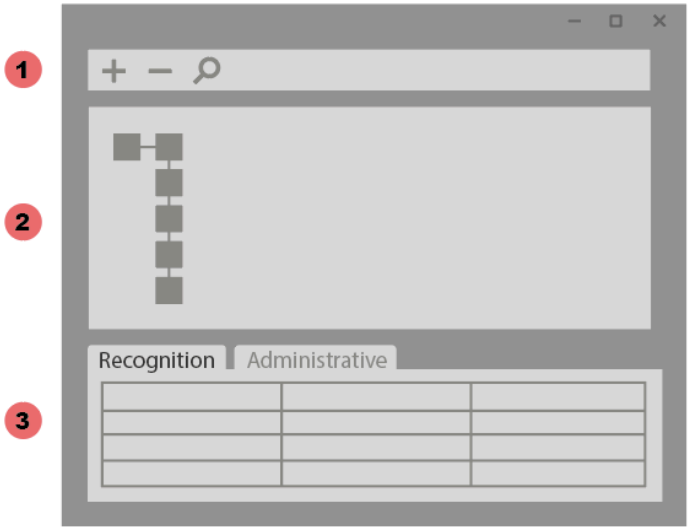

Abbildung 6: Aufbau RFT Test Object Map (IBM 2019b)

1. Das Test Object Menü umfasst unter anderem die Optionen, UI-Objekte hinzuzufügen, bereits aufgezeichnete UI-Objekte in der Anwendung zu markieren, oder vorgenommene Anpassungen der UI-Objekterkennung zu übernehmen.
2. In diesem Bereich findet man eine hierarchische Auflistung der Identifizierten Elemente.
3. Dieser Bereich ist aufgegliedert in Recognition Parameters und Administrative Parameters.
    a. Im Abschnitt „Recognition“ sind die Informationen der einzelnen UI-Objekte gelistet, welche RFT beim Recording erkennt. Dazu zählen beispielsweise ID, Name, Typ oder Index. Jeder dieser Parameter hat abhängig von der Relevanz bei der Objekterkennung ein Gewicht bzw. „Weight“ definiert.
    b. Im Bereich „Administrative“ sind weitere Informationen gelistet, welche aber nicht direkt zur Objekterkennung bei der Testausführung genutzt werden. Dort findet man einen aussagekräftigen Namen des Elements oder die Map ID. Diese können beim Schreiben der Test Scripte zum Finden der Elemente genutzt werden.

Um UI-Objekte über Code abzubilden, bietet der Rational Functional Tester die Klasse GuiTestObject an. Um das UI-Objekt über Koordinaten zu finden gibt es nachfolgend gelistete Optionen. (IBM 2019c)

- *getScreenPoint()* gibt als Ergebnis die Position des Elements relativ zum Bildschirm zurück. Dabei wird für die zurückgelieferten Pixelkoordinaten ein Punkt innerhalb des UI-Objekts gewählt.
- *getScreenRectangle()* liefert im Gegensatz zu *getScreenPoint()* nicht nur einen Punkt innerhalt des UI-Objekts, sondern den rechteckigen Bereich, welcher das Element umfasst, zurück.
- *getPointInObject()* ermöglicht, Koordinaten für einen bestimmten Punkt als Parameter zu übergeben um anschließend zu überprüfen, ob sich diese im betroffenen UI-Objekt befinden.
- *nClickDragToScreenPoint()* führt einen Mouse-Klick auf einem vorgegebenen Element durch und zieht dieses dann per Drag and Drop zu den als Parameter übergebenen Pixelkoordinaten. Diese Koordinaten werden bei dieser Methode relativ zum UI-Objekt verstanden.

7.4 Vergleich zwischen Ranorex und IBM Rational Functional Tester

Vergleicht man diese Methoden mit den in Kapitel 7.1 genannten Optionen für Ranorex, sieht man gewisse Ähnlichkeiten. Mit *getPointInObject()* und *nClickDragToScreenPoint()* werden außerdem anderem praktische Optionen geboten. Man kann nicht nur Positionen am Bildschirm auslesen und über Koordinaten definieren, sondern auch Benutzeraktionen wie beispielsweise Drag and Drop mit vorgefertigten Methoden simulieren. Wird eine solche Funktion von einem Testautomatisierungs-Tool nicht unterstützt, müsste jeder Schritt, wie anhand des Beispiels Drag an Drop jeder Mouse-Klick und -Release sowie jede Mouse-Bewegung, einzeln definiert werden. Gibt es in einer Anwendung viele solcher Benutzeraktionen zu simulieren, sollten auch solche Aspekte bei der Auswahl des Tools betrachtet werden.

8 UI-OBJEKTERKENNUNG ÜBER ID’S MIT TOSCA XSCAN UND SELENIUM WEBDRIVER

Als mögliche Testautomatisierungs-Tools für die UI-Objekterkennung mittels ID’s werden Tosca XScan und der Selenium WebDriver vorgestellt. Viele kommerzielle Testautomatisierungs-Tools, welche auch in Kapitel 7 vorgestellt werden, nutzen Selenium für Web-Tests. Dennoch soll in Kapitel 8.2 aufgezeigt werden, wie der Selenium WebDriver funktioniert und welche Unterschiede es zu kommerziellen Tools gibt.

Grundsätzlich ist die Wahl von ID’s zur Identifikation der Elemente naheliegend, sofern die grafische Benutzeroberfläche nicht auf dynamische ID’s aufbaut. In der Evaluierung der Stabilität wird unter anderem fokussiert, wie aufwändig die initiale Aufzeichnung der UI-Objekte ist und wie viel Zeit die Erkennung bereits aufgezeichneter Elemente in Anspruch nimmt.

8.1 Tosca XScan für die UI-Objekterkennung über ID’s

Das Testautomatsierungs-Tool Tosca, auch Tosca Test Suite gennant, des Herstellers Tricentis umfasst unter anderem die Komponenten XBrowser Engine zum Interagieren mit Webanwendungen, sowie XScan zum Bearbeiten und Selektieren der identifizierten UI-Objekte.

Mit der XBrowser Engine wird automatisch die im gewählten Browser geöffnete Webanwendung auf dessen UI-Objekte gescannt, dies stellt somit die Basis zur Erstellung von Web-Tests dar. Die Browser Firefox, Chrome und Edge werden durch Installation der zugehörigen Extension unterstützt. Beim Scannen der UI-Objekte von Webanwendungen werden anschließend die zugehörigen XBrowser Modules erstellt. Diese Modules enthalten alle notwendigen technischen Informationen, um mit den UI-Objekten interagieren zu können. Neben dem Scannen von UI-Objekten über die Option „Desktop Scan“ ist außerdem die Option „Remote Web Scan“ verfügbar. Über diese Option können auch Elemente von Browsers auf remote Maschinen gescannt werden. (Tricentis Tosca 2019c)

In der Tosca Test Suite werden Elemente der grafischen Benutzeroberfläche über das eigens dafür zur Verfügung gestellte XScan Fenster verwaltet. Über Tosca wird die zu testende Anwendung gewählt und anschließend jene Elemente, welche für die Ausführung der automatisierten Tests benötigt werden, über das XScan Fenster selektiert. Dabei werden bei der Auswahl eines solchen Elements verschiedene Optionen zur Art der Identifikation angeboten, wie etwa „Identify by Image“ oder „Identify by Properties“. Mit „Identify by SmartID“ beispielsweise können zwei unterschiedliche Arten von ID’s für die Identifikation festgelegt werden. Um das Element zu finden muss nur eine dieser ID’s zutreffen. Stimmt beispielsweise nur die erste ID, wird die zweite automatisch auf den aktuellen Stand angepasst. Mit „Identify by Property“ kann eine Auswahl an identifizierten Properties für die eindeutige Objekterkennung getroffen werden, wie etwa der Name oder der Typ des Elements, beispielsweise ein Element vom Typ Button. (Tricentis Tosca 2019a)

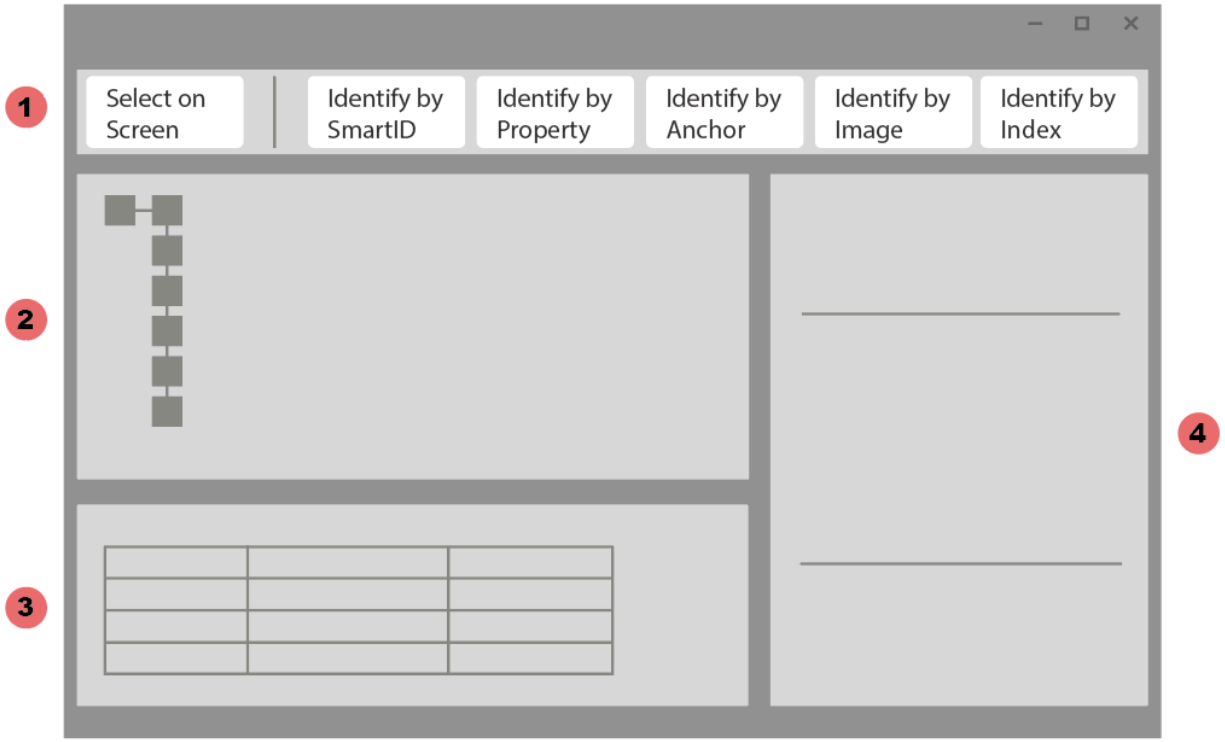

Abbildung 7: Tosca XScan Aufbau (Tricentis Tosca 2019a)

1. In dieser Menüleiste werden diverse Auswahlmöglichkeiten für die UI-Objekterkennung angezeigt. Die Art der Identifikation, wie etwa durch Image Recognition oder durch Properties des Objekts, kann hier ausgewählt werden. Des Weiteren kann über dieses Menü mit der Option „Select on Screen“ die Auswahl des UI-Objekts direkt auf der Oberfläche getriggert werden.
2. In dieser Ansicht werden die identifizierten UI-Objekte der ausgewählten Anwendung hierarchisch in Baumstruktur gelistet. Jene Objekte, welche für die Testerstellung benötigt werden, werden hier mittels Checkbox ausgewählt und somit für die weitere Verarbeitung bzw. Testerstellung in der Tosca Test Suite verfügbar gemacht.
3. Im sogenannten Content View Bereich wird der Inhalt des ausgewählten UI-Objekts angezeigt. Beispielsweise wird der Inhalt einer ausgewählten Tabelle hier gelistet. In dieser Ansicht kann der User beispielsweise definieren, welche Zeilen und Spalten als Überschriftszeile bzw. –spalte gehandhabt werden soll.
4. Je nach Art der Identifikation, welche im Bereich 1. von Abbildung 7 erwähnt wird, gibt es in diesem Abschnitt die Möglichkeiten zur konkreten Definition, welche Kriterien für die Objekterkennung des jeweiligen UI-Objekts herangezogen werden.

8.2 Evaluierung der Stabilität mit Tosca XScan

Die Tosca Testsuite wird genutzt, um die Stabilität der UI-Objekterkennung über ID’s zu evaluieren. Dafür wird betrachtet, auf welche Properties einzelner UI-Objekte über XScan zugegriffen werden kann und über welche verschiedenen Arten von ID’s die Testdaten identifiziert

werden können. Da die Identifikation über ID’s mitunter als eine der stabilsten Möglichkeiten zur UI-Objekterkennung gehandhabt wird, ist davon auszugehen, dass die Ergebnisse der Evaluierung in einigen Punkten positiv ausfällt.

8.2.1 Identifikation unter Voraussetzungen wie bei Aufzeichnung

Es werden zwei unterschiedliche UI-Objekte für die Evaluierung aufgezeichnet. Diese werden auch bei den nachfolgenden Überprüfungen betrachtet. Es wird ein bestimmter Vorgang identifiziert sowie ein gesamter Status-Bereich.

Die Vorgänge werden in dieser Evaluierung nicht über eine ID der HTML-Elemente identifiziert, sondern anhand der von Jira vergebenen ID der Vorgänge. Diese sind im Property „Inner Text“ hinterlegt und werden von XScan korrekt als eindeutig klassifiziert. Status-Bereiche werden als List-Objekte erkannt und haben eine ID im Property „attributes_data-id“ hinterlegt. Die UI-Objekterkennung wird über diese beiden ID’s überprüft.

Nach der Aufzeichnung der UI-Objekte wurde die Webanwendung neu geladen. Beide Elemente konnten erfolgreich gefunden werden.

8.2.2 Identifikation bei Testausführung mit anderem Browser

Mit der Tosca Test Suite können Tests auf den Browsern Internet Explorer, Chrome, Firefox sowie Edge ausgeführt werden. Zur Überprüfung des Cross-Browser Beurteilungskriteriums werden ein konkreter Vorgang sowie ein Status-Bereich herangezogen, welche bereits in Kapitel 8.1.1 aufgezeichnet wurden. Für die initiale Aufzeichnung wurde der Browser Firefox genutzt, die Objekterkennung anschließend über Chrome und Internet Explorer überprüft.

Die Elemente konnten ohne Anpassungen der zuvor definierten Properties für die UI-Objekterkennung erkannt werden.

Für die Aufzeichnung werden die zugehörigen Browser AddOns installiert. Über die Modules Ansicht in Tosca wird das Scannen einer Anwendung über XScan initiiert. Dabei wird die Struktur des UI analysiert und anschließend werden zwei der Testdatensätze bzw. Vorgänge im Kanban Board direkt im Browserfenster gewählt.

Anschließend werden die zugehörigen Properties je Element gelistet. XScan wählt hier selbständig die Identifikation per Tag aus, welche jedoch in beiden Fällen „SECTION“ als Wert enthält. Der Hinweis, dass es sich um zwei nicht eindeutige Elemente handelt, wird entsprechend angezeigt. Das Propertie „Id“ ist leer, Jira Vorgänge verfügen jedoch über ein eigenes ID Feld im UI.

8.2.3 Identifikation unter Einsatz von Responsive Design

Das Browserfenster, welches zuvor auf den gesamten Bildschirm maximiert war, wird in der Breite auf zwei Drittel der vorherigen Größe verkleinert. Dadurch werden die Status-Bereiche

entsprechend der neuen Dimensionen schmäler sowie die Felder der Vorgänge kleiner. Die ID’s der Felder werden nun unter statt neben der Priorität angezeigt.

Für die Evaluierung werden die zuvor über Firefox aufgezeichneten Elemente herangezogen. Eine Veränderung in der Darstellung wirkt sich bei der Identifikation über ID’s nicht auf die Objekterkennung aus. Die Elemente werden weiterhin identifiziert.

8.2.4 Geschwindigkeit der Identifikation und Zeitüberschreitungsfehler

Um zu testen, wie sich Tosca verhält, wenn Elemente sehr langsam in den Browser geladen werden, wird in dieser Evaluierung Chrome genutzt und über die Entwicklertools eine schwache Internetverbindung simuliert.

Über die Settings der Test Suite können die sogenannten Special Engines Parameter angepasst werden. Darunter sind auch die Browser Engines zu finden. Hier können für jeden von Tosca unterstützten Browser unter anderem die Wait Parameter angepasst werden. Für dieses Evaluierungskriterum wird das RetryInterval, welches angibt, wieviel Zeit zwischen zwei Versuchen ein Element zu identifizieren verstreicht, auf 20000 Millisekunden erhöht. Ebenso werden für diesen Testzweck der WaitForControl Parameter, welcher definiert wie lange nach einem UI-Objekt gesucht wird, sowie der WaitForWindow Parameter mit der Definition der Wartezeit für Browserfenster, auf 20 Sekunden festgelegt. Zusätzlich kann über TBox Werte ein Wait Zeitfenster als Testschritt festgelegt werden. Zehn Sekunden werden definiert.

Nach dem zehn Sekunden Zeitfenster werden die zwei UI-Objekte nicht gefunden. Die Fehlermeldung, dass ein Element nicht gefunden wurde, erscheint. Der TBox Timer wird auf 25 Sekunden erhöht, nun werden die Elemente erfolgreich identifiziert.

8.2.5 Identifikation eines nicht sichtbaren Elements

Es wird überprüft, wie sich die UI-Objekterkennung verhält, wenn ein Element im nicht sichtbaren Bereich ist. Für diese Überprüfung werden mehrere Vorgänge in eine Spalte verschoben, sodass diese erst durch hinunterscrollen im Browser sichtbar werden.

Das zu identifizierende Element bzw. der Vorgang, welcher sich nun außerhalb des sichtbaren Bereichs befindet, kann erfolgreich gefunden werden.

8.2.6 Zusammenfassung in Beurteilungsmetrik

Beurteilungskriterium	**Ergebnis**
Identifikation unter Voraussetzungen wie bei Aufzeichnung (Browser, Plattform und Layout unverändert)	Erfolgreich
Identifikation bei Testausführung mit anderem Browser	Erfolgreich getestet mit Firefox, Chrome, Edge

	Unterstützung der Browser Firefox, Chrome, Edge, Internet Explorer
Identifikation bei Anpassung des Layouts / Responsive Design	Erfolgreich
Geschwindigkeit der Identifikation / Zeitüberschreitungsfehler	Erfolgreich bei Wait von 25000 Millisekunden Fehler wenn kein Wait bei Simulation langsamer Netzwerkverbindung
Identifikation eines nicht sichtbaren Elements	Erfolgreich
Sonstiges (UI-Objekt verändert sich, iframe Inhalte, Pop-Ups, etc.)	Nicht aufgetreten

Tabelle 3: Evaluierung der UI-Objekterkennung über ID's mit Tosca XScan

Nicht immer verfügen die UI-Objekte über eigenen HTML ID-Attribute. Stattdessen kann sich der ID’s der Webanwendung bedient werden, wie etwa der Vorgangs-Nummern des Jira Boards. Über Tosca kann auf verschiedene Properties eines Elements zugegriffen werden, darunter auch jene Vorgangs-Nummer bzw. ID.

Grundsätzlich sind ID’s ein zuverlässiges Mittel für die UI-Objekterkennung. Das Testautomatisierungs-Tool weiß aber nicht immer welche Komponente eines Elements die ID beinhaltet, wie es sich auch im Rahmen dieser Evaluierung gezeigt hat. Daher ist ein grundlegendes Verständnis der zu testenden Anwendung Voraussetzung dafür, dem Tool beizubringen welche Properties für die Erstellung stabiler Tests herangezogen werden sollen.

8.3 Selenium WebDriver für die UI-Objekterkennung über ID’s

Verfügt der Tester über Programmierkenntnisse, bietet es sich an, für die Automatisierung das Open Source Tool Selenium bzw. den Selenium WebDriver zu nutzen. Lizenz- und Support-Kosten können somit eingespart werden. Die Testerstellung unterscheidet sich jedoch in einigen Punkten von jener der kommerziellen Testautomatisierungs-Tools. Viele dieser Tools bauen zwar unter anderem auf Selenium auf, bieten aber zusätzlich eine benutzerfreundliche grafische Oberfläche zur einfacheren Testerstellung. Der Selenium WebDriver hingegen fokussiert das Testen über die API, also der Programmierschnittstelle.

Für die Erstellung der Testscripts kann unter anderem zwischen den Programmiersprachen Java, C# oder Python gewählt werden. Der WebDriver greif dabei auf die verschiedenen nativen Driver der einzelnen Browser zu, welche Automatisierung unterstützen. Wenn man Tests auf mehreren Maschinen bzw. virtuellen Maschinen durchführen möchtet, bietet Selenium außerdem den Selenium-Server an. Entscheidet man sich für die Automatisierung mittels Selenium WebDriver, weicht die Umsetzung von den in dieser Arbeit vorgestellten kommerziellen Tools ab. Die

Entwicklungsumgebung kann frei gewählt werden und beispielsweise mit Maven ein Selenium 2.0 Java Projekt erstellt werden, welches Selenium und alle zugehörigen Dependencies umfasst. (Selenium 2019d)

Der Selenium WebDriver bietet wie nachfolgend gelistet zwei Möglichkeiten, auf der aktuellen Webseite nach Elementen zu suchen. Ebenso werden beispielhaft Optionen zur Auswahl einzelner Elemente genannt. (Navneesh 2014: 155 f.)

- *findElement()* liefert als Ergebnis das erste Element, welches auf das im Parameter übergebene Suchkriterium zutrifft.
- *findElements()* beinhaltet alle Elemente, die auf die Suche zutreffen.

Zusätzlich gibt es einige Methoden der Klasse „Select", welche die Auswahl einzelner Elemente vereinfacht. Diese Methoden sind beispielsweise bei Elementen vom Typ Liste sehr hilfreich.

- *selectByIndex()* und *deselectByIndex()* wählt UI-Objekte anhand ihres Indexes, bzw. hebt die Auswahl auf.
- *selectByValue()* wählt jenes Element, welches mit dem übergebenen String übereinstimmt. Auch hier gibt es eine *deselect*-Variante der Methode.
- *getFirstSelectedOption()* liefert das erste Element der zuvor durchgeführten Auswahl zurück.

Der laut Selenium effizienteste Weg ein UI-Objekt zu finden ist die Suche über ID. Voraussetzung hierfür ist natürlich, dass keine automatisch generierten ID's, sondern stattdessen eigens vergebense Klassen verwendet wurden. Zur Identifikation über ID's wird die zuvor vorgestellte *findElement()* Methode verwendet. Existiert beispielsweise ein Element * ... *, lautet der zugehörige Selenium Code zur Identifikation WebElement element = driver.findElement(By.id(„*customerFavorites*")); Würde man stattdessen beispielsweise nach dem Klassen Namen suchen, würde der Parameter *(By.className(„..."))* lauten. (Selenium 2019d)

8.4 Vergleich zwischen Tosca XScan und Selenium WebDriver

Vergleicht man die Aufzeichnung der UI-Objekte über Selenium mit dem in Kapitel 7.2.1 vorgestellten Tool Tosca XScan, fallen einige Unterschiede auf. Während Selenium auf die Erstellung von Test Scripts über Programmiersprachen setzt, bietet die Tosca Test Suite, wie der Großteil der kommerziellen Testautomatisierungs-Tools, eine grafische Benutzeroberfläche zur UI-Objektaufzeichnung und Testerstellung an. Testfälle können einfach über Drag and Drop zusammengestellt werden. Zusätzlich bietet die Tosca Test Suite die Testerstellung über Code an und unterstützt unter anderem die Sprachen .NET und Java Swing.

Die Unterschiede zwischen diesen beiden Tools reichen letztendlich aber weit über die Vorgehensweise zur Aufzeichnung der UI-Objekte hinaus. Während sich die Open Source Lösung Selenium auf die Automatisierung von Web-Oberflächen spezialisiert, ist das Testen von Webanwendungen nur ein Teilbereich der Tosca Test Suite, welche sich auch mit Testcase-

Design und Testmanagement befasst. Liegt der Fokus auf Geschäftsprozessen und funktionalen Tests statt Performance-Tests, kann die Tosca Test Suite eine interessante Lösung darstellen. Soll hingegen auf technischer Ebene getestet werden bzw. handelt es sich bei den Testern um Entwickler, kann Selenium in Betracht gezogen werden.

MicroFocus bietet mit Silk Test ein weiteres Testautomatisierungs-Tool, welches unter anderem das Testen von Webanwendungen aufbauend auf den Selenium WebDriver unterstützt. Das Tool erleichtert über den sogenannten Silk WebDriver die Erstellung von Selenium-Testscripts durch eine grafische Benutzeroberfläche und automatische Script-Erstellung. Somit wird die Erstellung automatisierter Tests über Selenium für Anwender mit weniger Programmierkenntnissen unterstützt. (Micro Focus 2019)

9 UI-OBJEKTERKENNUNG ÜBER XPATH MIT RANOREX SPY UND TOSCA XSCAN

Die Testautomatisierungs-Tool Ranoerx und Tosca XScan werden herangezogen, um die UI-Objekterkennung mittels Pfad bzw. XPath zu beschreiben. Eine Übersicht über Ranorex Spy, welcher zur UI-Objekterkennung genutzt wird, ist bereits in Kapitel 7.1 im Kontext UI-Objekterkennung über Koordinaten angeführt. Eine Beschreibung des Tools Tosca XScan, welches im Kontext der UI-Objekterkennung über ID's betrachtet wurde, ist in Kapitel 8.1 zu finden. Die XPath Syntax wird nachfolgend näher betrachtet und konkrete Beispiele angeführt.

9.1 Ranorex Spy für die UI-Objekterkennung über XPath

Wird ein UI-Objekt erfolgreich mit Ranorex Spy aufgezeichnet, wird wie in Abbildung 5 unter Punkt 2 der XPath bzw. RanoreXPath angezeigt. Klickt man in diese Zeile, kann man den Pfad bearbeiten. Unter Punkt 3 der eben genannten Abbildung kann man zusätzlich in den Path Editor wechseln. Dieser listet den Pfad in Baumstruktur und ermöglicht Anpassungen des Pfades, die jedoch nicht das Parent Element und Elemente darüber miteinbeziehen, da Änderungen an dieser Stelle auch andere aufgezeichnete UI-Objekte betreffen würden. Wird ein editierbares Element in dieser Liste gewählt, erscheinen auf der rechten Seite im Path Editor diverse Informationen, beispielsweise der zugehörige Controlname oder der Typ des Elements. Diese Informationen können einzeln über eine Checkbox gewählt werden, um Ranorex vorzugeben, nach welchen Informationen das Tool nach dem UI-Objekt suchen soll. Wird hier eine Anpassung getroffen, wird dies automatisch im RanoreXPath übernommen. (Ranorex 2019b)

RanoreXPath basiert auf XPath, erweitert diese aber um gewisse Funktionen. Ranorex definiert das Prinzip von RanoreXPath mit der Aussage, der dieser in seiner Komplexität zwischen dem notwendigen Detaillierungsgrad und ausreichender Flexibilität ausbalanciert werden soll. Also nur mit der Genauigkeit, die notwendig ist, um den Pfad eindeutig zu halten. So soll erreicht werden, dass gleichermaßen Robustheit und schnelle Ausführung optimiert werden. Grundsätzlich ist der RanoreXPath wie folgt aufgebaut:

/role[@attribute='value']

Mit dieser Struktur wäre eine Ebene in der Baumstruktur beschrieben, welche hinter der grafischen Benutzeroberfläche der Anwendung liegt. Mit dieser Syntax kann beginnend mit einem weiteren „/“ auf die darunterliegende Ebene dieser Struktur navigiert werden. Auch andere Operatoren können genutzt werden, beispielsweise „//“ um eine beliebige Anzahl an Ebenen zu überspringen. Verschiedene Komponenten des Pfads können auch durch Variablen substituiert werden, die es beispielsweise ermöglichen, die Wahl eines Radiobuttons dynamisch durch Testdaten zu definieren. (Ranorex 2019c)

Um die Suche noch weiter zu verfeinern können auch Regular Expressions, auch regex genannt, genutzt werden. Dabei handelt es sich um ein auf Zeichen basierendes Vorgehen, durch welche

eine Suche definiert bzw. gefiltert werden kann. Beispielsweise können solche Muster für typische Find/Replace Mechanismen genutzt werden, oder auch um mit einer Suche mehrere Varianten eines Wortes abzubilden, wie etwa die Wörter „gray“ und „grey“. In diesem Fall lautet der zugehörige regex Ausdruck *gr[ae]y*. Wenn hingegen ein optionaler Buchstabe vorhanden ist, wie beispielsweise „color“ und „colour“, kann der zugehörige regex Begriff *colou?r* lauten. Das Fragezeichen definiert, dass das davor kommende Zeichen optional ist. Alternativ kann auch *(color|colour)* genutzt werden, um mehrere Varianten zu akzeptieren. Wenn nach keinem bestimmten Begriff, aber nach zulässigen Zeichen gesucht werden soll, kann [A-Z] für alle groß geschriebenen Buchstaben von A bis Z genutzt werden, oder auch [1-9] für alle Zahlen von 1 bis 9. Zusätzlich definieren {min,max} wie oft diese Zeichen vorkommen dürfen. So bedeutet beispielsweise [A-Z]{2}, dass es sich um zwei Großbuchstaben handeln soll. Verschiedene solcher regex Filter können kombiniert werden. (Rudd 2018: 7 - 11)

Um auch kompliziertere Testszenarien abzubilden, bietet Ranorex neben der Testerstellung über Drag and Drop auch diverse Funktionen an, die den Bereich RanoreXPath einschließen. Diese sind im Ranorex.Core Namespace zu finden. Pfade werden in Ranorex mit dem Datentyp RxPath repräsentiert. Nachfolgend sind beispielhaft ein paar dieser Funktionen genannt. (Ranorex 2019d)

- *CompareTo()* vergleicht den aktuellen Pfad mit jenem RxPath, welcher als Parameter übergeben wurde. Alternativ kann auch *Equality()* oder *Inequality()* zum Vergleich zweier als Parameter übergebenen Pfade genutzt werden.
- *IsSimilar()* vergleich ebenfalls zwei Pfade, kontrolliert aber nicht, ob diese komplett identisch sind, sondern ob sich die Pfade ähneln. Dabei wird ein sogenannter WarnCount zurückgeliefert, also die Anzahl an nicht kritischen Abweichungen der Pfade.
- *GetParentPath()* liefert als Ergebnis den Pfad jenes Parent-Elements zurück, welcher als Parameter übergeben wurde.
- *Concat()* verbindet die zwei übergebenen Pfade und liefert als Ergebnis einen zusammengeführten Pfad zurück.

9.2 Evaluierung der Stabilität mit Ranorex Spy

Ranorex Spy wurde bereits bei der Evaluierung der UI-Objekterkennung über Koordinaten näher betrachtet. Bei der Identifikation über Koordinaten wird nicht direkt auf den technischen Aufbau der Webanwendung zugegriffen, sondern lediglich die Position am Bildschirm oder in Abhängigkeit zu einem anderen UI-Objekt angegeben. Daher ist besonders die Betrachtung der UI-Objekterkennung über XPath interessant, da Ranorex Spy hier auf die Struktur der Anwendung zugreift. Des Weiteren ist zu erwähnen, dass Ranorex durch dessen eigene Version von XPath, dem RanoreXPath, dieser Art der UI-Objekterkennung besondere Aufmerksamkeit geschenkt hat.

9.2.1 Identifikation unter Voraussetzungen wie bei Aufzeichnung

Wieder werden bei den Testdaten des Jira Boards ein Vorgang sowie ein gesamter Status-Bereich für die Evaluierung aufgezeichnet. Die Webanwendung wird für die initiale Aufzeichnung der UI-Objekte über Firefox geöffnet.

Ranorex Spy erkennt über die Track Funktion folgenden Pfad für den gewählten Jira Vorgang:

/dom[@domain='ma19ui.atlassian.net']//div[#'ghx-pool']/div[2]/ul/li[3]/div/div[4]

Das Div-Objekt „div[#'ghx-pool']/div[2]/ul" beschreibt jenen Bereich, in welchem sich die Elemente des Boards, also die Status-Bereiche und Vorgänge, befinden. Das List-Objekt „li[3]" Beschreibt den dritten Status-Bereich, der über den Index „3" gewählt wird. Der Vorgang wird über das Objekt „div[4]" selektiert.

Dies hat natürlich zur folge, dass über diesen RanoreXPath immer der vierte Vorgang des dritten Status-Bereichs identifiziert wird. Bleiben die Komponenten der Webanwendung unverändert, ist die Identifikation erfolgreich.

9.2.2 Identifikation bei Testausführung mit anderem Browser

Die initial mit Firefox aufgezeichneten UI-Objekte werden für die Evaluierung der Cross-Browser Funktionalität anschließend über Chrome und Internet Explorer gesucht. Um dies zu beurteilen werden wieder die Properties Inner Text des Vorgangs und des Status-Bereichs herangezogen. Da diese Werte unabhängig des Browsers gleich sind, können die UI-Objekte erfolgreich auf den unterschiedlichen Browsern identifiziert werden.

9.2.3 Identifikation unter Einsatz von Responsive Design

Wie bei der Evaluierung der UI-Objekternennung über Koordinaten und ID's wird auch bei der Betrachtung von XPath das Browserfenster auf zwei Drittel der Breite des Vollbildmodus reduziert. Die zwei UI-Objekte des Testdatensatzes können anschließend erfolgreich identifiziert werden. Die Skalierung der Elemente hat keine Auswirkung auf den RanoreXPath.

9.2.4 Geschwindigkeit der Identifikation und Zeitüberschreitungsfehler

Um ein realitätsnahes Szenario für die Evaluierung der Geschwindigkeit nachzustellen, wird der zuvor aufgezeichnete RanoreXPath des Vorgangs sowie des Status-Bereichs angepasst. Der Vorgang wird nun über dessen eindeutige Vorgangsnummer identifiziert, welche sich in einem Span-Objekt als Inner Text befindet. Der Status-Bereich wird über dessen Header identifiziert, welcher ebenfalls im Inner Text Property einzusehen ist.

Die Identifikation des Vorgangs sowie des Status-Bereichs läuft innerhalb weniger Millisekunden ab. Es kommt zu keinem Zeitüberschreitungsfehler.

Man sollte jedoch beachten, dass lange Pfade bzw. umfangreiche HTML-Strukturen zu Einbußen in der Geschwindigkeit der Identifikation der UI-Objekte führen können. Selbst wenn einzeln betrachtet nur eine kaum merkliche Verzögerung bei der Identifikation eines Elements auftritt, kann dies in Summe große Auswirkungen auf ein Set an Testfällen haben.

9.2.5 Identifikation eines nicht sichtbaren Elements

Es wird überprüft, wie die Identifikation über RanoreXPath bei Elementen außerhalb des sichtbaren Bereichs im Browserfenster reagiert. Mehrere Vorgänge werden in einen Status-Bereich verschoben, sodass das gesuchte Element beim Aufrufen der Webanwendung im ersten Schritt nicht sichtbar ist.

Anschließend werden die zuvor im Ranorex Spy aufgezeichneten UI-Objekte neu geladen. Das gesuchte Element wird erneut gefunden, ist nun allerdings ausgegraut. Dies ist als Indikator für ein UI-Objekt innerhalb eines nicht sichtbaren Bereichs zu interpretieren. Tritt das Element in den sichtbaren Bereich und werden die aufgezeichneten Elemente nochmals neu geladen, ist das UI-Objekt nicht weiter ausgegraut.

9.2.6 Zusammenfassung in Beurteilungsmetrik

Beurteilungskriterium	**Ergebnis**
Identifikation unter Voraussetzungen wie bei Aufzeichnung (Browser, Plattform und Layout unverändert)	Erfolgreich
Identifikation bei Testausführung mit anderem Browser	Erfolgreich getestet mit Firefox, Chrome, Internet Explorer Unterstützung der Browser Browser Firefox, Chrome, Edge, Internet Explorer, Safari
Identifikation bei Anpassung des Layouts / Responsive Design	Erfolgreich
Geschwindigkeit der Identifikation / Zeitüberschreitungsfehler	Kein Zeitüberschreitungsfehler
Identifikation eines nicht sichtbaren Elements (ggf. mit Wait)	Erfolgreich
Sonstiges (UI-Objekt verändert sich, iframe Inhalte, Pop-Ups, etc.)	Nicht aufgetreten

Tabelle 4: Evaluierung der UI-Objekterkennung über XPath mit Ranorex Spy

Standardmäßig versucht Ranorex Spy alle UI-Objekte über dessen XPath bzw. RanoreXPath zu identifizieren. Für die initiale Identifikation der beiden Jira Elemente, dem Vorgang sowie dem Status-Bereich, war keine nachträgliche Anpassung des Pfades notwendig. Für die Erstellung stabiler Tests im Rahmen der Evaluierung wurden die Pfade angepasst, sodass sie nicht anhand ihrer Position, sondern ihrer Inhalte erkannt werden. Daher ist diese Technologie zur UI-Objekternennung auf den ersten Blick eine interessante Wahl für das automatisierte Testen der Webanwendung.

Für die Aufzeichnung der UI-Objekte über Ranorex Spy wurden keine Path Weights festgelegt. Dadurch kann es passieren, dass der Pfad standardmäßig nicht fehlertolerant bzw. stabil gegenüber Änderungen angelegt wird. Der in dieser Evaluierung aufgezeichnete Jira Vorgang wurde beispielsweise initial über den Index des List-Objekts identifiziert. Da sich aber Vorgänge eines SCRUM oder Kanban Boards typischerweise in kurzen Zeitabständen verschieben, ist die Identifizierung über einen Index nicht zweckmäßig. Daher ist, sofern das gewählte Testautomatisierungs-Tool diese Funktion anbietet, zu empfehlen, Path Weights zu nutzen. Damit soll vordefiniert werden, welche Komponenten des UI-Objekts einer stabilen Objekterkennung nützlich sind und somit Aufwand bei der Erstellung und Wartung der automatisierten Tests eingespart werden. Jedoch kann es natürlich auch Teil des Testfalls sein, beispielsweise immer das erste Element der ersten Spalte zu wählen. In diesem Fall kann der Index zur Identifikation herangezogen werden.

Des Weiteren können über Variablen, welche als Platzhalter für Testdaten innerhalb der RanoreXPath dienen, Tests dynamisch erstellt werden. Diese Funktion ist bei der Automatisierung von Tests kaum mehr wegzudenken.

9.3 Tosca XScan für die UI-Objekterkennung über Pfad

Auch das Testautomatisierungs-Tool Tosca des Herstellers Tricentis unterstützt XPath zur Identifikation der UI-Objekte. Die Aufgezeichneten Elemente werden bei Tosca im zugehörigen Tool XScan angeführt. In dieser Ansicht werden die UI-Objekte der getesteten Anwendung hierarchisch gelistet und können zur weiteren Verwendung in der Tosca Test Suite selektiert werden. In Kapitel 8.1, Abbildung 7, findet man die grafische Darstellung des Aufbaus von Tosca XScan. Dort ist außerdem eine detaillierte Beschreibung der Komponenten des Tools angeführt.

Über die XBrowser Engine der Tosca Test Suite werden UI-Objekte gescannt. HTML Elemente können dabei über den xPath identifiziert werden. In dem Bereich „Identify by Properties“ in der XScan Ansicht findet man den Pfad des betrachteten UI-Objekts, hier kann die Option “XPath“ zur Identifikation ausgewählt werden. Tosca zeigt standardmäßig den absoluten XPath beim Scannen der Anwendung, also der vollständige Pfad ausgehend vom Root Element. Kommt es aber zu Änderungen in der HTML Struktur, was üblicherweise immer wieder eintritt, kann das gesuchte Element nicht mehr über diesen XPath gefunden werden. Wenn man sich für die Identifikation über XPath entscheidet, sollte stattdessen der relative Pfad genutzt werden. Dieser nimmt ein anderes Element als Anhaltspunkt bzw. Referenzwert und navigiert von dort über XPath zum gesuchten Element. (Tricentis Tosca 2019c)

9.4 Vergleich zwischen Ranorex und Tosca

Während Ranorex eine eigene XPath Version „RanoreXPath“ anbietet, verweist Tosca auf die W3C Recommendation für XPath. Diese definiert die einzelnen Komponenten eines XPath als Expressions und beschreibt ausführlich die Syntax von XPath. (W3C 2017) Da RanoreXPath aber auch auf ein Subset von XPath beruht, ist die Verwendungen von Pfaden zur Identifikation der UI-Objekte zwischen den beiden vorgestellten Tools letztendlich sehr ähnlich. Ranorex bietet allerdings einen eigenen Path Editor an, welcher die Bestandteile eines XPath in Baustruktur auflistet. Da solche Pfade in manchen Fällen auch lang und unübersichtlich werden können, erleichtert der Ranorex Spy durch diese hierarchische Listung der Elemente die Nachvollziehbarkeit des XPath und bietet auch eine einfache Möglichkeit zum Editieren des Pfades.

10 UI-OBJEKTERKENNUNG ÜBER IMAGE RECOGNITION MIT TESTCOMPLETE UND EGGPLANT FUNCTIONAL

Während die Identifikation der UI-Objekte über Pixelkoordinaten und ID's wie in Kapitel 7 und 8 beschrieben in der Umsetzung relativ simpel sein kann, zeigt Image Recognition neue Herausforderungen auf. Es wird betrachtet, welche Unterschiede es in der Art und Weise der Aufzeichnung der UI-Objekte über Image Recognition im Vergleich zu den in den vorhergehenden Kapiteln vorgestellten Technologien gibt. Für die Betrachtung der Technologie und Evaluierung der Stabilität werden die Testautomatisierungs-Tools TestComplete und Eggplant Functional vorgestellt, welche in Kapitel 11 nochmals für die Betrachtung von OCR herangezogen werden. Eine detaillierte Beschreibung der grafischen Benutzeroberfläche des Eggplant Functional Image Viewers ist in Kapitel 11 sowie Abbildung 10 angeführt.

10.1 TestComplete Image Set Editor für die UI-Objekterkennung über Image Recognition

TestComplete des Herstellers SmartBear ist eines von vielen Testautomatisierungs-Tools, welches Image Recognition unterstützt. Für die visuelle UI-Objekterkennung setzt TestComplete darüber hinaus Artificial Intelligence ein, um somit den in manchen Fällen aufwändigen Prozess der UI-Objekterkennung zu unterstützen und potentielle manuellen Schritte für die Testerstellung zu reduzieren. (SmartBear 2019e) Das Thema AI für die UI-Objekterkennung wird in Kapitel 14 näher betrachtet.

Immer wieder kann es vorkommen, dass UI-Objekte aufgrund technischer Einschränkungen der zu testenden Anwendung nicht im Rahmen des automatisierten Testens erkannt werden können, beispielsweise bei benutzerdefinierten Elementen. Daher stellt Image Recognition eine hilfreiche Ergänzung zu den bereits vorgestellten Technologien dar. Pixel für Pixel wird beim TestComplete auf der Benutzeroberfläche nach dem zuvor aufgezeichneten Element gesucht und anschließend bei den gefundenen Koordinaten die gewünschte Userinteraktion simuliert. Über verschiedene Parameter kann dieser vergleichende Algorithmus modifiziert werden, beispielsweise über die Pixeltoleranz oder die Farbtoleranz. Diese Parameter definieren, wie viele Pixel etwa in ihrer Farbe vom gesuchten Element abweichen dürfen. Man muss allerdings darauf achten, dass die Bilder grundsätzlich identisch sind, also unter denselben Bedingungen aufgezeichnet wurden. (SmartBear 2019c)

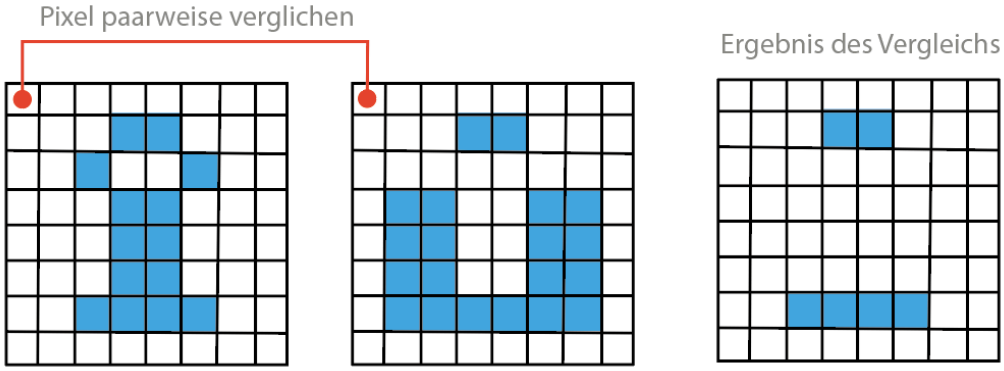

Abbildung 8: Image Comparison mit TestComplete (SmartBear 2019c)

Die aufgezeichneten Bilder werden bei TestComplete anschließend im sogenannten Image Repository gespeichert. Da die visuelle Darstellung einzelner UI-Objekte, beispielsweise je nach Bildschirmauflösung oder Browser, abweichen kann, können bei diesem Tool zu jedem UI-Objekt mehrere Screenshots aufgezeichnet werden. Diese sind im sogenannten Image Strip einzusehen. Die Aufzeichnung und Bearbeitung der erkannten UI-Objekte wird bei dem vorgestellten Testautomatisierungs-Tool über den TestComplete Image Set Editor durchgeführt. In Abbildung 9 wird der Aufbau des Tools dargestellt. (SmartBear 2019c)

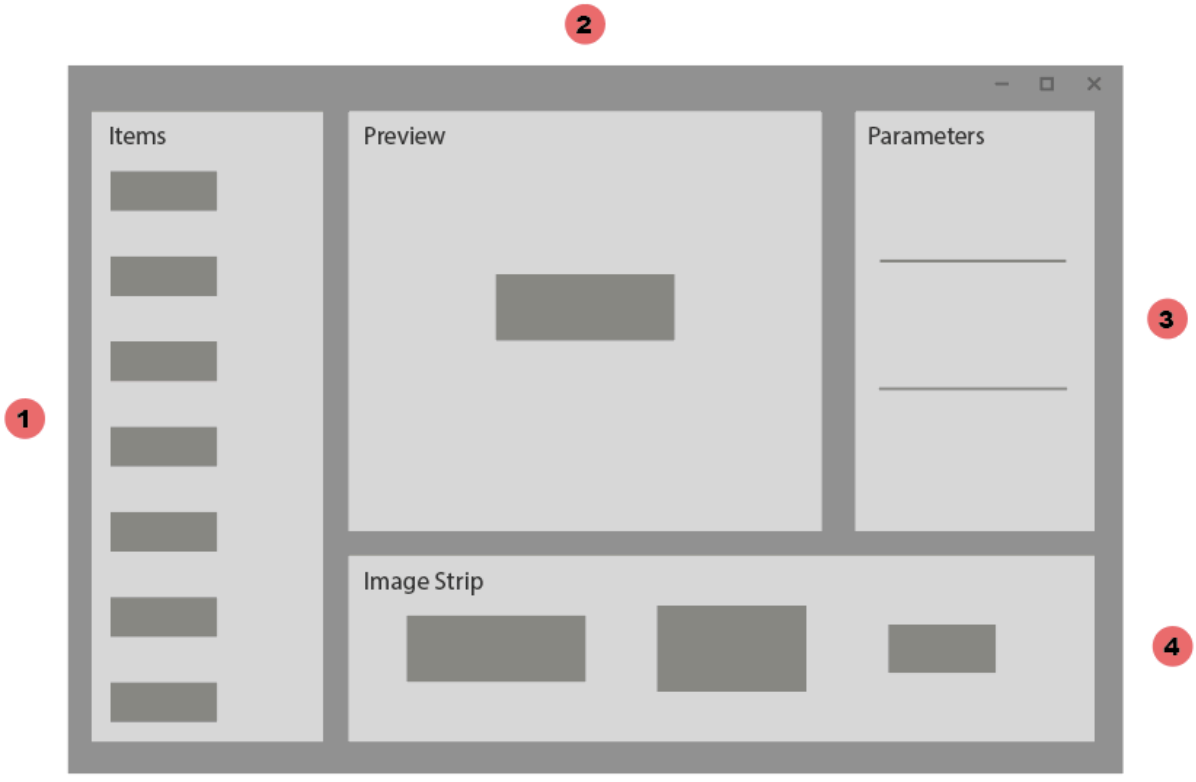

Abbildung 9: Test Complete Image Set Editor (SmartBear 2019c)

1. In der Items Liste werden die aufgezeichneten Elemente, also die Screenshots, des gewählten Image Sets gelistet. Für die Erstellung der automatisierten Tests werden diese

Elemente über jene Namen identifiziert, welche in dieser Liste den Elementen vergeben werden.

2. Die Preview-Ansicht zeigt jenes Element an, welches in der Item Liste ausgewählt ist. Dabei wird ein größerer Ausschnitt der Anwendung, Desktop oder mobile, in diesem Fenster angezeigt sowie jener Bereich markiert, in welchem über Image Recognition nach dem zugehörigen UI-Objekt gesucht wird. Dieser Bereich kann nachträglich angepasst werden, beispielsweise kann der Ausschnitt verkleinert werden, wenn etwa ein Mouse-Cursor am Rand mit aufgezeichnet wurde.
3. Die Item Properties umfassen Item Parameters und Image Parameters. Zusätzlich ist in diesem Abschnitt auch der Bereich Recognition Parameters vorzufinden.
    a. Unter Item Parameters kann man unter anderem den Namen des gewählten UI-Objekts einsehen und ändern, mit welchem das Element zur Testerstellung angesprochen wird.
    b. Die Image Parameters beinhalten eine Beschreibung des gewählten Elements, welche optional vom Anwender hinzugefügt werden kann, sowie Informationen zur Bildschirmauflösung, welche bei der Aufzeichnung gegeben war.
    c. Die Recognition Parameters beeinflussen die Genauigkeit, mit welcher TestComplete bei der Image Recognition vorgeht. Man kann hier die sogenannte Color Tolerance und die Pixel Tolerance definieren. Die Color Tolerance gibt vor, welche farbliche Abweichung ein Pixel von der originalen Aufnahme haben darf, um noch als identisch zu gelten. Unter Pixel Tolerance kann vorgegeben werden, wie viele Pixel in Summe oder prozentuell von der originalen Aufnahme abweichen dürfen.
4. Im Image Strip werden verschiedene Varianten des UI-Objekts aufgezeigt, da die grafische Darstellung von mehreren Faktoren abhängt. Beispielsweise können die Bildschirmauflösung, das Farbschema oder verschiedene UI-Styles die Visualisierung des Elements beeinflussen. Auch für das Cross-Browser Testen kann dies sehr hilfreich sein, da die Darstellung einer Webanwendung abhängig von den Browsern unterschiedlich ausfallen kann. Daher bietet TestComplete die Möglichkeit, bei der Image Recognition einem UI-Objekt mehrere Darstellungsvarianten zuzuweisen.

10.2 Evaluierung der Stabilität mit TestComplete

Da sich TestComplete unter anderem den Technologien Image Recognition und OCR zur Identifikation der UI-Objekte verschrieben hat, wird dieses Tool für die Evaluierung herangezogen. Beide dieser Technologien werden über TestComplete evaluiert, besonders um

aufzuzeigen, dass die Aufzeichnung dieser Elemente, obwohl über ein und dasselbe Tool durchgeführt, dennoch in der Vorgehensweise variieren kann.

10.2.1 Identifikation unter Voraussetzungen wie bei Aufzeichnung

Das Jira Board mit den Testdaten für die Evaluierung wird über den Browser Chrome geöffnet. Nach Aktivierung der TestComplete Erweiterung für Chrome werden zwei UI-Objekte, ein Vorgang sowie ein Status-Bereich, aufgezeichnet. Dafür wird dem zuvor erstellten Projekt ein Image Repository hinzugefügt, in welchem ein Image Set mit dem Namen „Jira“ hinzugefügt wird. Dieses Image Set beinhaltet beispielsweise alle Elemente einer Webanwendung.

Über „Add Item“ werden nun einzelne UI-Objekte hinzugefügt. Wird diese Funktion aufgerufen, erscheint die gewählte Anwendung als Screenshot in einem Fenster. Dort wird die Abbildung des gesamten Browserfensters, ähnlich einem Bildbearbeitungs-Programm, auf das gesuchte Element zugeschnitten. Ein kompletter Vorgang sowie die Überschrift eines Status-Bereichs werden auf diese Weise aufgezeichnet. Es wird darauf geachtet, dass die Außenränder dieser UI-Objekte nicht bei der Bildaufzeichnung beinhaltet sind. Nach Aufzeichnung der beiden Elemente sieht man in der Übersicht, dass pro UI-Objekt ein Bild aufgezeichnet wurde.

Über „Highlight on Desktop“ wird überprüft, ob die aufgezeichneten UI-Objekte identifiziert werden können. Diese Überprüfung kann erfolgreich durchgeführt werden.

10.2.2 Identifikation bei Testausführung mit anderem Browser

TestComplete unterstützt die Browser Internet Explorer, Firefox, Chrome und Edge. Jene UI-Objekte, welche im vorhergehenden Kapitel über Chrome aufgezeichnet wurden, werden nun über Firefox und Internet Explorer zu identifizieren versucht. Dafür wird vorerst nicht auf die Option zugegriffen, zu einem UI-Objekt mehrere Aufnahmen zu erstellen. Für die Farbtoleranz und Pixeltoleranz werden keine Toleranzbereiche festgelegt.

Sowohl bei Internet Explorer wie auch Firefox können die mit Chrome aufgezeichneten Elemente nicht identifiziert werden. Dies macht deutlich, dass selbst minimale Änderungen in der Darstellung der UI-Objekte die Image Recognition beeinflussen.

Im ersten Schritt wird versucht, die Farbtoleranz von dem Default Wert Null auf 50 zu erhöhen. Der Maximalwert 255 würde bedeuten, dass die Farbe keine Rolle mehr spielt und immer als ident betrachtet wird. Die Anpassung der Farbtoleranz wird durchgeführt, weil die Überschrift des Status-Bereichs mit einer Hintergrundfarbe hinterlegt ist. Dies stellt sich als nicht erfolgreich heraus.

Im nächsten Schritt wird nun auch die Pixeltoleranz erhöht. Durch diese Anpassung können beide UI-Objekte, der Vorgang wie auch der Status-Bereich anhand dessen Überschrift, auf Firefox und Internet Explorer gefunden werden. Für den Vorgang mit weißem Hintergrund und wenigen bunten Komponenten ist keine Farbtoleranz notwendig, für die Status-Bereich Überschrift muss eine Farbtoleranz vergeben werden.

Wird für jeden Browser das UI-Objekt nochmals aufgezeichnet, funktioniert die Identifikation ohne Definition einer Farbtoleranz oder Pixeltoleranz. Dafür wird lediglich ein neues Item dem Image Strip hinzugefügt.

10.2.3 Identifikation unter Einsatz von Responsive Design

Da sich durch den Einsatz von Responsive Design auch die Skalierung einzelner UI-Objekte ändert, können diese nun nicht mehr über Image Recognition gefunden werden. Selbst die Überschrift des Status-Bereichs, welche nur aus einem relativ kleinen Bereich des Browserfensters und wenigen Buchstaben besteht, ist damit nicht mehr identifizierbar.

Die Aufzeichnung mehrerer Screenshots zu einem UI-Objekt gestaltet sich in diesem Kontext ebenso als nicht sinnvoll, da mit Responsive Design eine Vielzahl an Darstellungsvarianten diverser UI-Objekte anfallen.

10.2.4 Geschwindigkeit der Identifikation und Zeitüberschreitungsfehler

Bei der Identifikation der UI-Objekte kam es zu keinen Zeitüberschreitungsfehlern. Für die Image Recognition wird nicht auf das DOM bzw. die HTML-Struktur zugegriffen, welche bei langsamer Ladezeit zu solchen Fehlern führen kann. Wurden Elemente nach der vordefinierten Zeitspanne nicht identifiziert, wurde die Fehlermeldung ausgegeben, dass das Element nicht gefunden werden konnte.

Wie in Kapitel 10.1.2 dargestellt wurde, können UI-Objekte, welche Browser-übergreifend identifiziert und nur einmal aufgezeichnet werden, über die Erhöhung der Pixeltoleranz identifiziert werden. Hier war aber eindeutig bemerkbar, dass dich durch die erhöhte Pixeltoleranz auch die Zeitspanne für die Identifikation deutlich erhöht hat. Des Weiteren benötigt die Identifikation des Vorgangs, welcher mit der elffachen Größe einen deutlich größeren Bereich am Bildschirm einnimmt als die Überschrift des Status-Bereichs, eine im Vergleich merkbar kleinere Zeitspanne für die Identifikation. Dies lässt darauf schließen, dass große Bereiche bei Verwendung einer Pixeltoleranz dennoch schneller gefunden werden können als Elemente in vergleichbar kleinen Bereichen.

10.2.5 Identifikation eines nicht sichtbaren Elements

Da Image Recognition nur für sichtbare UI-Objekte eingesetzt werden kann, kann dieses Beurteilungskriterium nicht evaluiert werden. Die Identifikation nicht sichtbarer Elemente wird als nicht erfolgreich gewertet.

10.2.6 Zusammenfassung in Beurteilungsmetrik

Beurteilungskriterium	**Ergebnis**
Identifikation unter Voraussetzungen wie bei Aufzeichnung (Browser, Plattform und Layout unverändert)	Erfolgreich
Identifikation bei Testausführung mit anderem Browser	Nicht erfolgreich ohne weitere Anpassungen Erfolgreich nach Definition der Pixeltoleranz mit Chrome, Firefox und Internet Explorer Unterstützung der Browser Internet Explorer, Firefox, Chrome und Edge
Identifikation bei Anpassung des Layouts / Responsive Design	Nicht erfolgreich
Geschwindigkeit der Identifikation / Zeitüberschreitungsfehler	Geschwindigkeit abhängig von Größe des gesuchten UI-Objekts/Bereichs am Bildschirm sowie Faktoren wie Pixeltoleranz, Farbtoleranz Kein Zeitüberschreitungsfehler
Identifikation eines nicht sichtbaren Elements (ggf. mit Wait)	Nicht gemessen da kein Zugriff auf nicht sichtbaren Bereich
Sonstiges (UI-Objekt verändert sich, iframe Inhalte, Pop-Ups, etc.)	Nicht aufgetreten

Tabelle 5: : Evaluierung der UI-Objekterkennung über Image Recognition mit TestComplete

Image Recognition kann in speziellen Situationen sinnvoll eingesetzt werden, wenn etwa die korrekte Darstellung eines Logos auf einer Website überprüft werden soll. Jedoch hat sich in dieser Evaluierung wie erwartet gezeigt, dass diese Form der UI-Objekterkennung durchaus aufwändig in der Erstellung und Wartung sein kann und auch ein gewisses Fehlerpotential mit sich bringt. Durch Faktoren wie Farbtoleranz und Pixeltoleranz kann die Identifikation der Elemente grundsätzlich fehlertoleranter gestaltet werden. Jedoch scheint es sinnvoller, besonders im Kontext von Cross-Browser Testing, die Möglichkeit von TestComplete zur Aufzeichnung mehrerer Screenshots zu einem UI-Objekt zu nutzen. Dies bringt wiederum mehr

Aufwand bei der initialen Identifikation der Elemente mit sich, wenn pro Element für jeden zu testenden Browser ein Screenshot aufgezeichnet werden muss.

Bei der initialen Aufzeichnung der UI-Objekte über Image Recognition muss darüber hinaus darauf geachtet werden, dass der gewählte Bereich korrekt gewählt wurde. Ränder sollten beispielsweise nicht mitaufgezeichnet werden, sondern nur der für die Identifikation relevante Inhalt eines Elements. Auch verändern diverse UI-Objekte ihr Erscheinungsbild, etwa wenn Buttons gewählt wurden und als Indikator für diese Selektion beispielsweise ihre Farbe ändern. Daher ist darauf zu achten, dass vor der Aufzeichnung der UI-Objekte keine Interaktionen auf der grafischen Benutzeroberfläche durchgeführt wurden.

10.3 Eggplant Functional Image Viewer für die UI-Objekterkennung über Image Recognition

Ein weiteres Testautomatisierungs-Tool, welches außerdem einen besonderen Fokus auf Image Recognition hat, ist Eggplant Functional des Herstellers Testplant. Eggplant Functional setzt gezielt auf Tests, welche die Ansicht eines Users bzw. die sogenannte User Experience nachahmen, also mit der grafischen Benutzeroberfläche interagieren. Wie einige andere Testautomatisierungs-Tools integriert auch Eggplant Functional das Testen von Webanwendungen mittels Selenium WebDriver. Im Falle von Eggplant Functional ist hier zusätzlicher Aufwand für die Installation und Konfiguration zu berücksichtigen. Der Selenium Server inklusive zugehörigem Treiber für den bzw. die Browser muss vom User installiert werden. (Testplant 2019c)

Die Aufzeichnung der UI-Objekte wird im Eggplant Viewer Window durchgeführt. In diesem Fenster wird die zu testende Anwendung angezeigt und über den Capture Mode können die benötigten UI-Objekte aufgezeichnet werden. Die aufgezeichneten UI-Objekte findet man anschließend im Image Viewer von Eggplant. (Testplant 2019e)

Um die Herausforderungen der Image Recognition, hervorgerufen durch das Cross-Browser-Testing, zu meistern, bietet das Eggplant Functional die sogenannte „Smooth“-Funktion an. Diese ermöglicht, UI-Objekte trotz leichter Abweichungen in der grafischen Darstellung zu erkennen. Soll jedoch der Text in einem UI-Objekt erkannt werden, ist es sinnvoller, OCR zu verwenden. (Testplant 2019b)

Während viele Testautomatisierungs-Tools die Testerstellung mittels Drag-and-Drop der aufgezeichneten UI-Objekte anbieten, ist bei Eggplant Functional Scripting notwendig. In Bezug auf Image Recognition sind hier unter anderem folgende Befehle verfügbar (Testplant 2019d):

- *EveryImageLocation()* sucht nach allen Vorkommnissen des UI-Objekts innerhalb der getesteten Anwendung. Der Name des Objekts wird als Parameter an die Funktion übergeben, es kann auch nach mehreren Objekten gleichzeitig gesucht werden. Als Ergebnis erhält man die Koordinaten der gefundenen Objekte. *ImageLocation()* hingegen liefert als Ergebnis nur das erste gefundene Element.

- *ImageFound()* sucht im Viewer Window nach dem vorgegebenen UI-Objekt. Als Ergebnis wird True oder False zurückgeliefert.
- *WaitFor()* stoppt das Testscript, bis das gesuchte UI-Objekt gefunden wurde.

10.4 Vergleich zwischen TestComplete und Eggplant Functional

Über TestComplete werden dem Anwender mehrere Möglichkeiten geboten, die Identifikation über Image Recognition genauer zu definieren. So kann etwa über die Pixel Tolerance bzw. Color Tolerance die akzeptable Abweichung des identifizierten UI-Objekts mit dem Refernzbild festgelegt werden. Außerdem können für ein UI-Objekt mehrere Screenshots aufgezeichnet werden, um verschiedene Darstellungsvarianten berücksichtigen zu können. Über Eggplant hingegen hat man über die vordefinierte Option „Smooth“. Des Weiteren ist jedenfalls zu beachten, dass TestComplete die Testerstellung über Drag and Drop erlaubt und somit nicht gezwungenermaßen von Entwicklern genutzt werden muss, während Eggplant wiederum Scripting für die Testerstellung fordert.

11 UI-OBJEKTERKENNUNG ÜBER OCR MIT TESTCOMPLETE UND EGGPLANT FUNCTIONAL

Ähnlich wie bei Image Recognition fokussiert auch Optical Character Recognition die visuelle Erkennung von Elementen. Wenn nach bestimmtem Textinhalt gesucht wird und das Testautomatisierungs-Tool nicht direkt darauf zugreifen kann, sollte die Wahl jedoch auf OCR fallen. Für die Betrachtung von OCR für die UI-Objekterkennung werden die Testautomatisierungs-Tools TestComplete und Eggplant Functional herangezogen. Eggplant Functional wird in Kapitel 10.3 bereits im Kontext Image Recognition vorgestellt. Die Aufzeichnung von UI-Objekten über Image Recognition ähnelt bei Eggplant Functional stark der Aufzeichnung über OCR. Eine Beschreibung von TestComplete sowie eine Evaluierung von dessen UI-Objekterkennung, ebenfalls im Kontext Image Recognition, ist bereits in Kapitel 10.1 angeführt. Für die Evaluierung von OCR wird TestComplete betrachet.

Der Einsatz von OCR zeigt beispielsweise besonders dann seine Vorteile, wenn sich erst in der Ausführung des Tests der gesuchte Text dynamisch ergibt, oder durch die verschiedenen Darstellungsvarianten unterschiedlicher Browser das Textelement in der visuellen Darstellung variiert. Manchmal können sich auch durch andere Gründe die Farbe, Schriftart oder etwa Größe des Elements ändern, wodurch der Zugriff auf Textinhalte sinnvoller ist als der Einsatz von Image Recognition. Andererseits kann auch einfach das Auslesen und Speichern von Text Teil des Testprozesses sein. (Testplant 2019a) Image Recognition und OCR sollten jedoch nur gewählt werden, wenn andere Technologien zur UI-Objekterkennung nicht mehr anwendbar sind bzw. ein spezieller Testfall dieses Vorgehen zur Identifikation eines Elements fordert.

11.1 TestComplete Image Set Editor für die UI-Objekterkennung über OCR

Das Testautomatisierungs-Tool TestComplete des Herstellers SmartBear nutzt die Google Vision API zur UI-Objekterkennung über OCR. Werden UI-Objekte über den Recorder aufgezeichnet, kommuniziert TestComplete mit dem Webservice der API, welche den ausgelesenen String zurückliefert. Gehört der Text zu einem bestimmten UI-Objekt, können diesem anschließend verschiedene Aktionen zugeordnet werden, wie beispielsweise ein Mouse-Klick. Um die Objekterkennung auf einfache Art über OCR durchzuführen, kann die Option „Record unsupported controls using OCR“ aktiviert werden, welche automatisch unbekannte UI-Objekte über OCR zu identifizieren versucht statt sich dessen Koordinaten zu speichern. Alternativ kann auch manuell ein sogenannter OCR Checkpoint über den OCR Wizard von TestComplete erstellt werden. Wird ein solcher Checkpoint angelegt, wird der Text eines vorgegebenen Objekts ausgelesen und kontrolliert, ob dieser mit dem gewünschten Ergebnis übereinstimmt. Verschiedene Befehle können für die Texterkennung genutzt werden, es folgen ein paar Beispiele. Führen diese Aufrufe zur Identifikation einzelner UI-Objekte, können im selben Befehl

auch Aktionen wie beispielsweise *HoverMouse* oder *ClickNextTo* hinzugefühgt werden. (SmartBear 2019f)

- *OCR.Recognize.FullText()* liefert den gesamten gefundenen Text zurück.
- *OCR.Recognize.CheckText()* kontrolliert, ob der erwartete Text, welcher dieser Funktion als Parameter übergeben wird, im Fenster der getesteten Anwendung gefunden wird.
- *OCR.Recognize.Block()* bzw. *OCR.Recognize.BlockByText()* ermöglicht den Zugriff auf UI-Objekte, die den als Parameter übergebenen String enthalten. Mit zusätzlichen Parametern wie spLargest oder spBottomMost kann auf einzelne Elemente zugegriffen und mit ihnen interagiert werden.

11.2 Evaluierung der Stabilität mit TestComplete

TestComplete wurde im Rahmen der Evaluierung bereits im Kontext UI-Objekterkennung über Image Recognition betrachtet. Hier zeigten sich die Stärken dieses Testautomatisierungs-Tools beispielsweise darin, dass auf die unterschiedlichen Darstellungsvarianten einzelner UI-Objekte durch die Aufzeichnung mehrerer Screenshots zu einem Element reagiert werden kann. Dies ist besonders für Webanwendungen, also Responsive Designs, hilfreich. Daher wird nun betrachtet, wie TestComplete auf diese Herausforderungen reagiert, wenn OCR zum Einsatz kommt.

11.2.1 Identifikation unter Voraussetzungen wie bei Aufzeichnung

Für die Evaluierung wird ein UI-Objekt des Jira Boards mittels OCR identifiziert. Dafür wird das Logo der Webanwendung, der Schriftzug „Atlassian“, welcher als SVG File in die Anwendung eingebunden ist, aufgezeichnet. Für die erste Aufzeichnung des UI-Objekts in dieser Evaluierung wird der Browser Chrome verwendet.

OCR kann über Code, aber auch über den im vorigen Kapitel vorgestellten Checkpoint Wizard, welcher in Folge verwendet wird, eingesetzt werden. Um OCR über Code zu verwenden, muss das zugehörige tcOCR Plugin installiert werden. Wird dieses Plugin genutzt, wird die Testerstellung nicht wie in Kapitel 10.1 über „Keywords“, also ohne den Einsatz von Programmiersprache, durchgeführt. Stattdessen müssen nun Testscripte erstellt werden. Um OCR zu nutzen wird das Objekt „OCR“ bei der Scripterstellung genutzt. Über folgenden Befehl kann beispielsweise der Text der aktiven Anwendung erfragt werden:

```
Log.Message(OCR.CreateObject(Sys.Desktop.ActiveWindow()).GetText());
```

Da der Fokus dieser Evaluierung aber auf der Aufzeichnung der UI-Objekte über einen Recorder oder ähnliches gerichtet ist, wird OCR über den Checkpoint Wizard aufgerufen. Dieser Überprüft beispielsweise, ob sich in einem Element der erwartete Text enthält. Dafür ist über das „Install Extensions“ Menü in TestComplete ein eigenes OCR Plugin zu installieren.

Dafür wird ein Keyword Test über den Recorder aufgezeichnet. Über das Menü des aktiven Recorders wird „Add Checkpoint“ gewählt und anschließend die Option „Object property“ gewählt.

Dort kann die Option „OCR“ selektiert werden, welche im Rahmen dieser Evaluierung sofort den Text des UI-Objekts korrekt ausliest.

Für die Überprüfung des Beurteilungskriteriums wird das erstellte Recording, welches auch das erneute Aufrufen der Webanwendung umfasst, ausgeführt. Der Text des UI-Objekts kann korrekt über OCR ausgelesen werden.

11.2.2 Identifikation bei Testausführung mit anderem Browser

Die UI-Objekterkennung mittels OCR, welche zuvor über Chrome initial erstellt wurde, wird für die Evaluierung der Cross-Browser Funktionalität zusätzlich über Firefox und Internet Explorer durchgeführt. Dafür wird im zuvor aufgezeichneten Recording lediglich der Browser in einem Auswahlfenster geändert.

Das UI-Objekt kann erfolgreich identifiziert werden. Das Validieren des enthaltenen Textes über OCR ist auf beiden Browsern erfolgreich.

11.2.3 Identifikation unter Einsatz von Responsive Design

Die getestete Webanwendung wird in ihrer Breite auf ein Drittel der Vollbild Größe reduziert, sodass sich das Layout stark verändert und einzelne Elemente neu skaliert werden. Wie im ersten Test wird wieder der Browser Chome verwendet.

Trotz veränderter Position am Bildschirm und kleinerer Darstellung kann das UI-Objekt identifiziert und der Textinhalt ausgelesen und validiert werden.

11.2.4 Geschwindigkeit der Identifikation und Zeitüberschreitungsfehler

Da TestComplete mit der Google Vision API zur UI-Objekterkennung mittels OCR arbeitete, ist zu beobachten, wie viel Zeit solche Aufrufe in Anspruch nehmen. Die aufgezeichneten UI-Objekte werden an das zugehörige Webservice gesendet, dort ausgewertet und letztendlich das Ergebnis als String an TestComplete zurückgeliefert.

In Bezug auf die in dieser Evaluierung durchgeführten Versuche waren keine auffällig zeitaufwändigen Testausführungen zu verzeichnen. Für den Testschritt des Identifizierens des UI-Objekts sowie der Validierung des enthaltenen Textes waren durchschnittlich etwa 3,5 Sekunden zu verzeichnen.

Da der Einsatz von OCR in der Testautomatisierung grundsätzlich nur für spezielle Szenarien gedacht ist, in welchen andere Technologien zur UI-Objekterkennung nicht alternativ verwendet werden können, kann diese Zeitspanne als akzeptabel betrachtet werden. Wäre bei einem Testszenario aber ein großer Teil der Testkomponenten mittels OCR durchzuführen, sollte die möglicherweise hohe Gesamtdauer berücksichtigt werden.

Im Verlauf dieser Evaluierung im Kontext von OCR kam es darüber hinaus zu keinen Zeitüberschreitungsfehlern. Mögliche Einstellungen hinsichtlich der Wartezeiten wurden nicht manipuliert, es wurde mit den Default Einstellungen von TestComplete gearbeitet.

11.2.5 Identifikation eines nicht sichtbaren Elements

Da für das Auslesen der Textwerte einzelner UI-Objekte die Darstellung der Elemente im Browserfenster Voraussetzung ist, kann dieses Beurteilungskriterium nicht evaluiert werden. Die Identifikation über OCR von nicht sichtbaren Elementen ist nicht möglich.

11.2.6 Zusammenfassung in Beurteilungsmetrik

Beurteilungskriterium	**Ergebnis**
Identifikation unter Voraussetzungen wie bei Aufzeichnung (Browser, Plattform und Layout unverändert)	Erfolgreich
Identifikation bei Testausführung mit anderem Browser	Erfolgreich getestet auf Firefox, Chrome, Internet Explorer Unterstützung der Browser Internet Explorer, Firefox, Chrome und Edge
Identifikation bei Anpassung des Layouts / Responsive Design	Erfolgreich
Geschwindigkeit der Identifikation / Zeitüberschreitungsfehler	Etwa 3,5 Sekunden pro Identifikation des UI-Objekts inklusive OCR Validierung Keine Zeitüberschreitungsfehler
Identifikation eines nicht sichtbaren Elements (ggf. mit Wait)	Nicht gemessen da kein Zugriff auf nicht sichtbaren Bereich
Sonstiges (UI-Objekt verändert sich, iframe Inhalte, Pop-Ups, etc.)	Nicht aufgetreten

Tabelle 6: : Evaluierung der UI-Objekterkennung über OCR mit TestComplete

Ähnlich wie bei der UI-Objekterkennung über Image Recognition zeigen sich spezielle Vorteile von OCR, wie aber auch einige Testszenarien, in welchen diese Technologie zur UI-Objekterkennung weniger sinnvoll oder aus technischen Gründen nicht einsetzbar ist. Zwar beschränkt sich OCR auf sichtbare Elemente bzw. den sichtbaren Bereich der getesteten Anwendung, kann aber auch beim Testen von Responsive Designs eingesetzt werden, wenn sich

die Elemente in ihrer Position oder Größe verändern. Auch bei Cross-Browser Tests kann diese Technologie verwendet werden, denn während der Großteil der UI-Objekte von den unterschiedlichen Browsern gerendert werden, wird bei Bildern meist auf dieselbe Datei zugegriffen.

Jedoch ergeben sich auch beim automatisierten Testen immer wieder Szenarien, in welchen gerade OCR die optimale Lösung ist. Denkt man beispielsweise an UI-Objekte, welche Text in Bildern enthalten, die nicht über HTML ausgelesen werden können. Auch als Alternative, wenn durch technische Einschränkungen nicht auch bestimmte UI-Objekte zugegriffen werden kann, bietet sich OCR an. Hier sollte man jedoch beachten, dass bei einer größeren Anzahl an Elementen die Dauer des Testfalls in die Länge gezogen werden kann.

Der Vorteil in der Aufzeichnung solcher Elemente über TestComplete im Vergleich zu Image Recognition ist darüber hinaus, dass die UI-Objekte nicht über einen Screenshot aufgezeichnet werden müssen. Diese Aufzeichnung über Screenshots bringt einigen manuellen Aufwand mit sich, da jedes Element für sich zugeschnitten und gegebenenfalls mehrfach aufgezeichnet werden muss, denkt man etwa an Cross-Browser Testing. Vergleichen dazu erkennt TestComplete im Kontext von OCR Checkpoints die UI-Objekte über den Recorder, es muss lediglich die OCR Validierung über den Checkpoint Wizard definiert werden.

11.3 Eggplant Functional Image Viewer für die UI-Objekterkennung über OCR

Eggplant Functional des Herstellers Testplant zählt zu jenen Testautomatisierungs-Tools, welche sich stärker auf das Testen von Prozessen und weniger auf die technische Ebene bzw. Code fokussieren. Daher steht auch bei diesem Tool besonders die grafische Benutzeroberfläche im Mittelpunkt.

Zur Aufzeichnung und Verwaltung der UI-Objekte nutzt Eggplant die Komponenten Viewer Window und Image Viewer. Wenn eine Verbindung zur getesteten Anwendung aufgebaut wird, wird das Viewer Window geöffnet, in welchem die Anwendung angezeigt wird. Über dieses Fenster kann der Capture Mode gestartet werden, welcher zur visuellen Aufzeichnung der UI-Objekte genutzt wird. Dafür muss lediglich das gewünschte UI-Objekt angeklickt werden und Eggplant Functional erkennt automatisch den Bereich, in welchem sich das Element befindet. Der Bereich kann zusätzlich auch manuell angepasst werden. Neben der Aufzeichnung der Elemente können hier die Aktionen zur Simulation einer User-Interaktion erstellt werden. Editiert werden können die aufgezeichneten UI-Objekte im Image Viewer. Dieser ist wie in Abbildung 10 dargestellt aufgebaut. (Testplant 2019e)

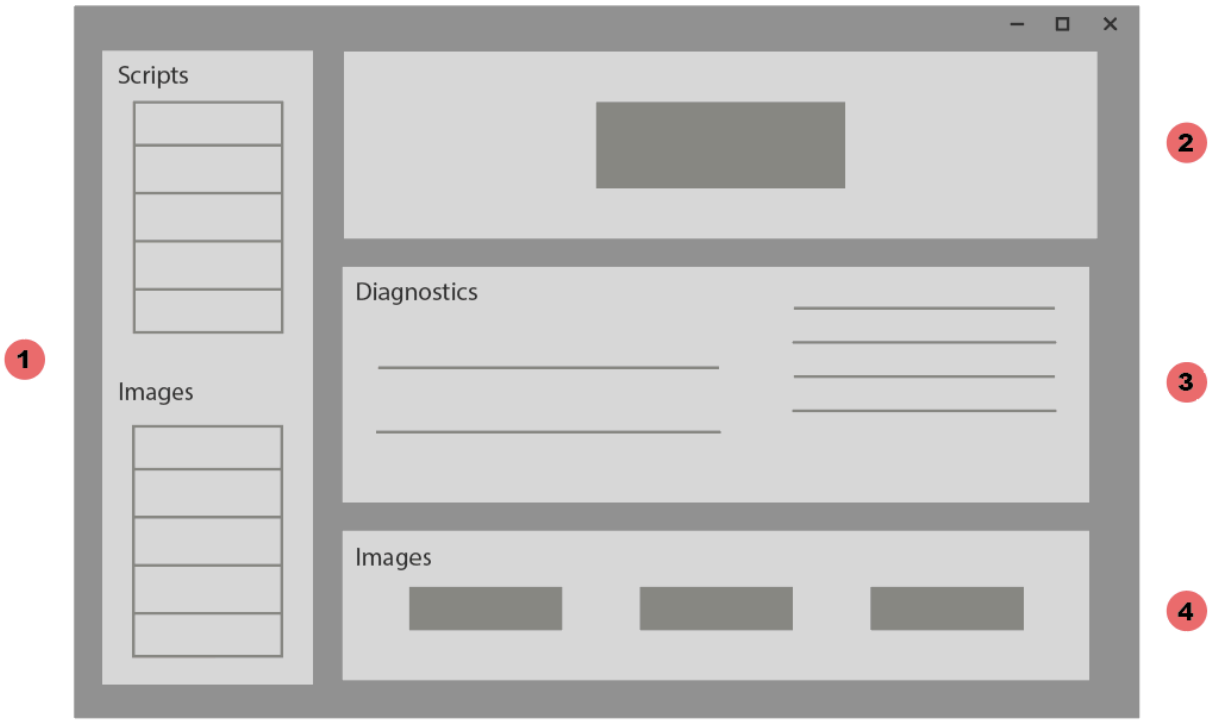

Abbildung 10: Eggplant Functional Image Viewer (Testplant 2019e)

1. In diesem Bereich findet sich eine Listung der erstellten Testscripts und aufgezeichneten UI-Objekte.
2. Wird im Images Bereich ein Element ausgewählt, wird das UI-Objekt hier angezeigt. Außerdem sieht man in dieser Ansicht anhand einer editierbaren Markierung, an welcher Position des Elements die Aktionen, wie etwa ein Mouse-Klick, ausgeführt werden.
3. Im Bereich Image Diagnostics können die Suchkriterien für jene UI-Objekte angepasst werden, welche nicht über das Viewer Window gefunden wurden. Wenn eine Verbindung zur getesteten Anwendung besteht, kann das gewünschte UI-Objekt in einer Liste ausgewählt und eine Suche über Image Diagnostics gestartet werden. Wird ein Element bei dieser Suche beispielsweise an einer anderen Position gefunden, kann diese Information über diesen Bereich aktualisiert werden.
4. Über den Image Browser werden Vorschaubilder der aufgezeichneten UI-Objekte und Order der Test Suite angezeigt.

Im Capture Mode sieht man bei der Aufzeichnung der Elemente deren Capture Area, also eine Vorschau jenes Bereichs, welche das Tool automatisch als ein UI-Objekt erkennt. Diese Capture Areas verfügen über ein Kontextmenü, welches unter anderem die Option OCR Tuner beinhaltet. Um über Eggplant Functional weitere Einstellungen zu OCR zu treffen bzw. die OCR Properties festzulegen, wird dieser OCR Tuner verwendet. Es kann beispielsweise der Bereich festgelegt werden, in welchem Eggplant nach einem bestimmten Element sucht, statt das Tool den ganzen Bildschirm absuchen zu lassen. Ein anderes Beispiel ist das Ausschließen bestimmter Zeichen, um mögliche Missinterpretationen auszuschließen, wie beispielsweise den Buchstaben „O“ mit der Ziffer „0“ zu verwechseln. (Testplant 2019e)

Eggplant Functional arbeitet mit einer Text Engine, welche standardmäßig die Erkennung verschiedener Schriftarten unterstützt. Die Text Engine funktioniert dabei so, dass zuallererst alle Elemente der Benutzeroberfläche nach Text abgesucht und deren Inhalte ausgelesen werden. Die Darstellung des Textes ist dabei nicht von Relevanz, es wird nur der Inhalt analysiert um den gesuchten String zu finden. (Testplant 2019a) Es lassen sich einige der Befehle von TestComplete zur Image Recognition ebenso auf OCR anwenden. So kann beispielsweise über *EveryImageLocation()* auch eine OCR Textreferenz als Parameter übergeben und nach allen Vorkommnissen des Elements in der getesteten Anwendung gesucht werden. Auch die *ImageLocation()* Funktion lässt sich auf diese Weise mit OCR kombinieren, um noch ein Beispiel zu nennen. *Click(text:“Home“)* würde beispielsweise das erste Element klicken, welches den String „Home“ beinhaltet. (Testplant 2019d)

11.4 Vergleich zwischen TestComplete und Eggplant Functional

Während Eggplant Functional für die UI-Objekterkennung mit OCR eine eigene Text Engine nutzt, welche standardmäßig den ganzen Bildschirm nach Text absucht, arbeitet TestComplete mit einem Webservice, welches den Text einzelner Elemente ausliest. Jedoch ermöglicht Eggplant Functional über dessen OCR Tuner eine Einschränkung jenes Bereichs, in welchem mittels OCR nach Textinhalten gesucht wird. Aus Performance Sicht sollte der Aspekt, wie groß der mit OCR gescannte Bereich ist, jedenfalls beachtet und der Bereich nach Möglichkeit klein gehalten werden.

12 AUSWIRKUNGEN DER UI-OBJEKTERKENNUNG AUF DIE STABILITÄT AUTOMATISIERTER TESTS

Die verschiedenen UI-Objekterkennungs-Technologien unterscheiden sich zum Teil stark in ihren Vorteilen und Nachteilen. Während Technologien wie XPath die schnelle und zuverlässige Testautomatisierung vieler moderner Anwendungen ermöglichen, helfen Image Recognition oder OCR wiederum, bei schwierig zu identifizierenden UI-Objekten dennoch auf die Inhalte dieser Elemente zugreifen zu können.

Basierend auf den in Kapitel 5.2 definierten Kriterien zur Beurteilung stabiler automatisierter Tests, sowie den aus der Evaluierung gewonnenen Erkenntnissen, werden die verschiedenen Technologien nachfolgend zusammenfassend gegenübergestellt.

12.1 Einsatzgebiete unterschiedlicher UI-Objekterkennungs-Technologien auf Grundlage der Evaluierung

Wie die Evaluierung zeigt, stellt die Identifikation der UI-Objekte im ersten Schritt für keine der Technologien eine Herausforderung dar, wenn die Wahl des Browsers sowie die Bildschirmgröße unverändert blieben. Erste Anpassungen der aufgezeichneten Elemente müssen aber bereits getroffen werden, wenn sich beispielsweise der Browser, welcher für die Testausführung herangezogen wird, verändert.

Schon bei der Interaktion mit UI-Objekten anhand von Koordinaten zeigt sich, dass diese Form der Objekterkennung in den meisten Fällen nicht sinnvoll ist. Abgesehen von Spezialfällen, wie etwa Drag and Drop Funktionen oder ähnlichen Mouse Bewegungen, können die UI-Objekte nur bei unveränderter Darstellung, verglichen mit der initialen Aufzeichnung des Elements, korrekt lokalisiert werden. Es kann aber davon ausgegangen werden, dass eine solche Situation, in welcher die Anwendung in ihrer grafischen Darstellung dauerhaft unverändert bleibt, nur in wenigen Fällen auftritt.

Bei der Überprüfung der Technologien unter Anwendung von Cross-Browser, also der Identifikation eines Elements auf unterschiedlichen Browsern, zeigen sich weitere Schwachpunkte mancher Technologien. Um UI-Objekte etwa über Image Recognition mittels Cross-Browser zu identifizieren, müssen Anpassungen der Toleranzbereiche, wie etwa eine größere Pixeltoleranz, getroffen werden. Erst durch die erlaubte Abweichung einer gewissen Anzahl an Pixel können manche UI-Objekte browserübergreifend identifiziert werden. Man kann davon ausgehen, dass ein Großteil der Webanwendungen auf mehreren Browsern aufgerufen werden. Daher sollte vor UI-Objektaufzeichnung und Testerstellung eingehend betrachtet werden, wie die UI-Objekte ohne viel zusätzlichen Aufwand auf den relevanten Browsern vom Testautomatisierungs-Tool gefunden werden können.

Des Weiteren ist zu beachten, dass nicht alle Testautomatisierungs-Tools dieselben Browser unterstützen. Beispielsweise werden die Browser Safari oder Opera nicht von allen in der Arbeit

evaluierten Tools unterstützt. Auch ist für die Interaktion mit den einzelnen Browsern zusätzlich jeweils ein eigenes Plugin pro Browser und Testautomatisierungs-Tool notwendig, da sonst nicht oder nur mit schlechter Performance auf die UI-Objekte zugegriffen werden kann.

Die Identifikation anhand von ID's wie auch XPath stellt sich grundsätzlich als sehr zuverlässig heraus. Erlaubt die Technologie der getesteten Anwendung eine Identifikation der UI-Objekte durch diese Vorgehensweisen, kann man von relativ stabiler UI-Objekternennung ausgehen, welche auch bei Responsive Designs und Cross-Browser Tests die Elemente korrekt erkennt. Jedoch ist bei langsamer Netzwerkverbindung, wie auch bei anderen Technologien zur UI-Objekterkennung, darauf zu achten, Waits zu definieren. Diese sollen garantieren, dass auf das Laden der Anwendung in den Browser gewartet wird. Viele Tools bieten daher bereits vordefinierte Funktionen an, die beim Testen von Webanwendungen auf das Laden der Darstellung im Browser warten, bevor versucht wird, einzelne Elemente zu identifizieren.

Manche Technologien stoßen außerdem in Situationen auf ihre Grenzen, wenn es um die Identifikation von UI-Objekten geht, welche außerhalb des sichtbaren Bereichs liegen. Dazu zählen die Technologien Image Recognition und OCR, welche auf die grafische Darstellung der Elemente aufbauen. In diesen Fällen zeigen sich wieder die Vorteile von XPath und ID's. Durch den Zugriff auf den Aufbau der Darstellung, welcher bei diesen beiden Vorgehensweisen stattfindet, ist die UI-Objekterkennung nicht nur auf den sichtbaren Bereich eingeschränkt, sondern verfügt über alle Elemente, die der aktuellen Ansicht angehören.

12.2 Wahl der Technologie zur UI-Objekterkennung

Naheliegend ist, dass vor der Entscheidung über die Wahl der Technologie zur UI-Objekterkennung bekannt sein muss, wie der Aufbau der zu testenden Anwendung ist. Dies betrifft nicht nur die Technologie hinter der Anwendung, sondern auch die Prozesse, welche durch automatisierte Tests abgebildet werden sollen. Manche Tools setzten auch bewusst den Fokus auf das Testen ganzer Geschäftsprozesse. Damit ermöglichen sie neben der automatisierten Durchführung komplizierter Prozesse auch das technologiegestützte Dokumentieren dieser Verfahren. Diese Tools machen sich die automatisierte Erstellung von Screenshots zu Nutze, um Dokumentationen am aktuellen Stand zu halten und die zeitaufwändige, manuelle Wartung dieser Dokumente zu beschleunigen. Ein Beispiel für ein Testautomatisierungs-Tool mit Business-Fokus ist Worksoft, welches auch im Gartner Quadranten für Testautomatisierung genannt ist, siehe Abbildung 2.

Die in der Evaluierung getestete Anwendung, das Jira Kanban Board, beinhaltete beispielsweise eine Vielzahl an Vorgängen, welche sich üblicherweise in der Praxis in kurzen Zeitabständen ändern. Hier wäre eine Identifikation über Index nicht zielführend. Daher ist unter anderem zu betrachten, ob sich die grafische Benutzeroberfläche oft verändert, welche Userinteraktionen mit den einzelnen UI-Objekten über die Testautomatisierung simuliert und welche Browser oder Geräte bei den Tests berücksichtigt werden sollen.

Außerdem ist auch vorab abzuklären, auf welchen Browsern das Cross-Browser Testing stattfinden soll, da nicht alle Tools alle Browser unterstützen. Auch das Testen von Responsive

Designs, besonders mit Fokus auf mobile Geräte, darf nicht außer Acht gelassen werden. Viele Testautomatisierungs-Tools ermöglichen bereits das automatisierte Testen auf mobilen Endgeräten, wie auch auf Emulatoren. Für die Abdeckung möglichst vieler Testszenarien stellt diese Option einen interessanten Aspekt dar.

Auf technischer Ebene ist zu betrachten, ob die einzelnen UI-Objekte dynamische ID’s hinterlegt haben, um somit gegebenenfalls den Einsatz von ID’s zur Objekterkennung auszuschließen. Werden wiederum Anwendungen getestet, die zum Teil auf Technologien basieren welche vom Testautomatisierungs-Tool nicht überstützt werden, kann für diese Situationen auf Image Recognition oder OCR ausgewichen werden.

13 AUFWAND DER TESTERSTELLUNG UND WARTUNG

Wie durch die Evaluierung der unterschiedlichen UI-Objekterkennungs-Technologien ersichtlich, sind die Vorgehensweisen zur Aufzeichnung der Elemente durchaus unterschiedlich. Diese Unterschiede belaufen sich nicht nur auf die verschiedenen Testautomatisierungs-Tools hinaus. Beispielsweise ist auch durch die Betrachtung von TestComplete und Ranorex, welche für jeweils zwei Technologien zur UI-Objekterkennung näher betrachtet wurden, aufgefallen, dass die Aufzeichnung der Elemente abhängig von der Technologie auch innerhalb eines Tools unterschiedlich ausfallen kann. Besonders bei dem Vergleich von Image Recognition und OCR war der Unterschied deutlich, auch da es sich bei diesen beiden Technologien der UI-Objekterkennung um spezielle Formen der Identifikation handelt, welche eher in Sonderfällen als Alternative dienen.

Zusätzlich hat sich bei der Betrachtung der Testautomatisierungs-Tools verdeutlicht, dass die Aufzeichnung der UI-Objeket meist nachträglichen Aufwand fordert, um die Identifikation dieser Elemente im weiteren Verlauf der Testerstellung stabil zu gestalten und den Wartungsaufwand zu reduzieren. Bei der Aufzeichnung der Elemente über Pfad bzw. XPath zeigte sich des Weiteren die Notwendigkeit von Path Weights. Durch diese Gewichtung wird dem Testautomatisierungs-Tool mitgeteilt, welche Komponenten der UI-Objekte der eindeutigen Identifikation dienlich sind. Werden diese Informationen nicht genutzt, kann es passieren, dass sich der Pfad beispielsweise wie in der Evaluierung aus Index-Werten zusammensetzt, welcher die Elemente nicht bzw. falsch identifiziert.

Die Stabilität der automatisierten Tests wirkt sich folglich auf den Wartungsaufwand der Tests aus. Daher kann die Wahl der passenden Technologie zur UI-Objekterkennung sowie dessen strukturierte Umsetzung beeinflussen, wie viel Zeit später in die Instandhaltung der Tests fließen wird. Da dies auch eine Frage der Kosten ist, kann dies letzten Endes auch ein Faktor für die Entscheidung zwischen manuellen und automatisierten Tests sein.

13.1 Kosten der Automatisierung als Entscheidungshilfe

Die Einführung von Testautomatisierung bedeutet nicht, alle manuellen Tests zukünftig automatisiert durchführen zu müssen. In vielen Situationen erleichtert die Automatisierung den Entwicklungsprozess maßgeblich, in anderen Situationen wiederum steht der Aufwand für die Testerstellung und Wartung in keinem Verhältnis zum gewonnenen Nutzen. Besonders beim Testen komplexer Spezialfälle ist die manuelle Ausführung von Testern mit Fachwissen in der Regel sinnvoller. Es gibt aber auch Entscheidungshilfen, die diese Optionen zur Umsetzung eines Tests abwägen.

Für die Entscheidung, ob ein bestimmter Test manuell oder automatisiert umgesetzt werden soll, muss der Aufwand für die Automatisierung betrachtet werden. Als Entscheidungshilfe kann eine Rechnung zur Aufstellung der Kosten für die Automatisierung herangezogen werden, die mit dem

Aufwand sowie den Kosten für die manuelle Ausführung verglichen wird. Navneesh stellt hierfür folgende Rechnung auf:

Kosten für Automatisierung =
Kosten für Tool + *Arbeitskosten für Testerstellung* + *Arbeitskosten für Testinstandhaltung*

Wenn beispielsweise ein Testdurchlauf über zwei Jahre hinweg wöchentlich ausgeführt werden soll, ergibt dies 104 Testausführungen. Die Kosten für die Automatisierung auf Basis der eben angeführten Rechnung sollen geringer sein als jene für die manuelle Testausführung, wenn man Automatisierung einführen möchte. (Navneesh 2014: 17 f.)

13.2 Aufwand der Testerstellung unter Berücksichtigung der Technologien zur UI-Objekterkennung

Die initiale Identifikation unterschiedlicher UI-Objekte mittels verschiedener UI-Objekterkennung-Technologien variiert in Betracht des Aufwands, was bei einer großen Anzahl an UI-Objekten deutlich werden kann. Wie bei der Evaluierung beschrieben ist die Vorgehensweise der Identifikation von Technologie zu Technologie unterschiedlich, die Wahl und Strategie für die Umsetzung soll gut überlegt sein.

Während bei der UI-Objekterkennung über XPath Aufwand eingespart werden kann, indem mit Path Weights gearbeitet wird, verlangen andere Technologien weit mehr Aufwand für die initiale Aufzeichnung der Elemente. Image Recognition beispielsweise stellt dem Anwender Screenshots der zu testenden Anwendung zur Verfügung, die nachträglich manuell bearbeitet werden müssen. Selbst wenn gewisse Tools eine Vorauswahl dieses Bereichs auf der grafischen Benutzeroberfläche treffen, ist dieser meist nachzubearbeiten oder zumindest zu kontrollieren. Besonders wenn, wie bei TestComplete, die Möglichkeit gegeben ist, mehrere Varianten eines Elements über Image Recognition aufzuzeichnen, ist im ersten Schritt einiges an Zeit für die Aufzeichnung der UI-Objekte zu kalkulieren. Aus wirtschaftlicher Sicht wirkt sich dies natürlich auch steigende Personalkosten der Mitarbeiter aus, müssen diese mehr Zeit in die Testerstellung investieren. Zusätzlich ist zu beachten, dass neben der Auswahl des Bereichs auch weitere Optionen, wie etwa die Position der Interaktion bzw. des Mouse Cursors auf dem Element, getroffen werden müssen.

Aber auch bei vermeintlich einfachen Technologien wie XPath gibt es gewisse manuelle Schritte bei der Aufzeichnung der Elemente zu tätigen. Während Path Weights bei der Auswahl bzw. Gewichtung der passenden Properties für die eindeutige Identifikation der UI-Objekte hilfreich sind, muss auch bei der Abbildung des Pfades auf die passende Granularität geachtet werden. Bei komplexen Anwendungen oder langen Pfaden kann dies Einbußen in der Performance der Testausführung bedeuten. Daher gilt hier zu beachten, den Pfad nur so genau wie notwendig zu definieren, dass die Identifikation eindeutig durchgeführt werden kann, ohne zu viele Schritte vorzugeben.

Werden UI-Objekte über einen Recorder aufgezeichnet, so wie es ein Großteil der kommerziellen Testautomatisierung-Tools anbietet, werden zu einem selektierten Element üblicherweise eine

Vielzahl an Properties gefunden, über die das Element in der Theorie identifiziert werden kann. Ist der Aufbau der getesteten Anwendung unbekannt, kann es ein zeitaufwendiger Prozess werden, mittels Ausprobieren die optimale Strategie für die stabile UI-Objekterkennung zu finden.

13.3 Auswirkung der Stabilität auf den Wartungsaufwand automatisierter Test

Im Verlauf der Evaluierung der Stabilität verschiedener UI-Objekterkennungs-Technologien traten diverse Fehler bei der Identifikation bereits aufgezeichneter Elemente auf. Die aufgetretenen Fehler waren aufgrund der verschiedenen Technologien unterschiedlichen Ursprungs. Auffällig war, dass keine Vorschläge zur Behebung der Fehler oder Einschränkungen des Problems vom Tool angegeben wurden. Auch Informationen zu fehlen Plugins, etwa bei Identifikation über OCR, wurden nicht angezeigt. In diesem Kontext wäre Artificial Intelligence durchaus hilfreich, um eine schnellere Fehlerbehebung zu ermöglichen. Im nachfolgenden Kapitel wird dieser Aspekt näher betrachtet.

Testautomatisierung ist dann einsatzfähig, wenn die zu testende Software zwar in Entwicklung ist, zumindest aber in ihrem Aufbau und ihrer Darstellung nahe am Endprodukt liegt. Ist die grafische Oberfläche noch zu weiten Teilen unfertig, wäre der Wartungsaufwand automatisierter Tests aufgrund der ständigen Änderungen an der Benutzeroberfläche unverhältnismäßig hoch. Doch selbst bei relativ stabilen Oberflächen kommt es immer wieder zu kleinen Änderungen, da man laufen sein Verbesserungspotential ausschöpfen möchte.

Diese Änderungen führen auch zu notwendigen Anpassungen der automatisierten Tests. Werden zusätzliche Felder der Oberfläche hinzugefügt, kann dies als Konsequenz beispielsweise eine Anpassung der ID’s der aufgezeichneten UI-Objekte mit sich bringen. Ändert sich die Struktur der Anwendung, müssen mit großer Wahrscheinlichkeit diverse XPath Werte angepasst werden. Viel Wartungsaufwand kann sich auch ergeben, wenn schwer zu identifizierende Elemente nochmals mittels Image Recognition aufgezeichnet werden müssen.

14 AUSBLICK AUF INTELLIGENTE UI-OBJEKTERKENNUNG

Die Digitalisierung hat in den letzten Jahren einen Stand erreicht, in welchem Unternehmen gefordert sind, immer schneller auf neue Anforderungen reagieren zu können und sich ständig zu verbessern. Während der Begriff „Agilität“ in der Software Entwicklung nicht mehr wegzudenken ist, müssen auch Unternehmen immer schneller auf Kundenwünsche reagieren können. Software Produkte wachsen in ihrer Komplexität und baut auf Automatisierung möglichst vieler Komponenten, um mit dieser Geschwindigkeit mithalten zu können. Aber auch Automatisierung stößt in manchen Situationen an ihre Grenzen. Daher bauen bereits manche Hersteller von Testautomatisierungs-Tools auf Hybride Lösungen, welche die Automatisierung durch den Einsatz künstlicher Intelligenz optimieren sollen.

Künstliche Intelligenz bzw. Artificial Intelligence (AI) wird durch verschiedene Definitionen beschrieben. Von der Möglichkeit, dass Maschinen selbständig lernen und rationale Entscheidungen treffen, bis hin zur Entwicklung eines Verstandes. Die Entwicklung von AI hat letztendlich das Ziel, Vorgänge und Entscheidungen, die von Menschen durchgeführt werden, durch lernfähige Maschinen zu automatisieren. (Russell & Norvig 2016: 2)

Auch im Kontext der Testautomatisierung wird Artificial Intelligence bald eine Rolle spielen. Dies würde neue Möglichkeiten zur noch effizienteren Automatisierung bringen, die den Software Testprozess zusätzlich beschleunigen kann. Besonders die Instandhaltung automatisierter Tests könnte von dem AI-gestützten Identifizieren der UI-Objekte profitieren.

Mithilfe von AI können UI-Objekte beispielsweise automatisiert erkannt und Tests durch Natural Language Processing einfacher erstellt werden. Auch in der Ausführung der automatisierten Tests könnte man bessere Performance erreichen. Sogar das Entdecken fehlerhafter Tests könnte intelligent gestaltet werden, indem das Testautomatisierungs-Tool den Ursprung des Fehlers identifiziert und die Behebung durch weitere Informationen unterstützt. Jedoch sollte man bedenken, dass es nie eine vollständige Automatisierung im Rahmen des Testens von Software geben wird, da zumindest in gewissen Aspekten auch manueller Aufwand miteinfließen wird müssen. (Cser 2018: 16 f.)

14.1 Artificial Intelligence in der Software Testautomatisierung

TestComplete des Herstellers SmartBear ist ein Beispiel für ein Testautomatisierungs-Tool, in welchem bereits in Richtung intelligente UI-Objekterkennung mithilfe von Artificial Intelligence entwickelt wurde. AI unterstützt dabei die Identifizierung von UI-Objekten, welche schwer zu identifizieren sind. So wird beispielsweise die visuelle Objekterkennung über Text unterstützt, sodass die Aufzeichnung des UI-Objekts ohne weitere manuelle Schritte durchgeführt werden kann. Besonders wenn die Technologie der Testumgebung das Testautomatisierungs-Tool einschränkt, beispielsweise die Benutzeroberfläche nur als Bild erkannt wird und nicht auf technischer Ebene auf einzelne UI-Objekte zugegriffen werden kann, ist diese hybride AI-Lösung hilfreich. Auch beim Manipulieren von Aktionen wie beispielsweise Mouse Klicks oder Mouse

Hover Funktionen, also dem Positionieren des Mouse Cursors über einem Element, soll AI besonders Tester ohne Programmierkenntnisse unterstützten. (SmartBear 2019e)

Auch das Testautomatisierungs-Tool Ranorex hat speziell in Bezug auf Web-Testing zuletzt Schritte in Richtung AI-gestützte UI-Objekterkennung getätigt. Dies soll unter anderem bei der Herausforderung helfen, Elemente auch mit dynamischen ID's zu identifizieren. Dynamische ID's sind besonders beim Testen von Webanwendungen ein häufiges Problem. Während einige Tools eine Gewichtung bzw. „Weights" der Properties zur Objekterkennung nutzen, wie beispielsweise das in Kapitel 7.3 vorgestellte Tool IBM Rational Functional Tester, erkennt Ranorex durch Machine Learning eigenständig solche ID's. In Folge dessen werden andere Properties zur Identifikation verwendet. Dies soll die Testerstellung erleichtern und die Tests stabiler machen. Zusätzlich soll AI bei der Verbesserung der Performance unterstützen. (Ranorex 2019e)

Ein weiteres Beispiel für den Einsatz von AI in der Testautomatisierung ist Eggplant AI. Der Einsatz von künstlicher Intelligenz fokussiert sich bei diesem Produkt auf die Erstellung und Durchführung automatisierter Tests. Dazu zählt etwa die Planung eines Testdurchlaufs, in welchem Eggplant AI die optimale Reihenfolge der Durchführung einzelner Test bestimmt. Zusätzlich wird bei dieser Lösung versucht, durch die Lernfähigkeit der AI aus Fehlern der automatisierten Tests zu lernen und somit den Anwender bei der Fehlerbehebung unterstützen zu können. Auch sollen auf diese Weise Fehlerpotentiale aufgedeckt werden, welche die Qualität der getesteten Anwendung negativ beeinflussen könnten. (Testplant 2019f)

Automatisierung wird aber auch im Testprozess nie vollständig den manuellen Aufwand ersetzen können. Die Aufzeichnung der UI-Objekte oder die Wartung der Testfälle kann zwar AI-gestützt sein, dennoch verfügen Personen über das Wissen komplexer Testprozesse. Auch ist es nicht sinnvoll, alles Tests zu automatisieren, und Explorative Tests oder Tests mit Usability Fokus obliegen auch zukünftig den manuellen Testern.

15 SCHLUSSFOLGERUNG

Webanwendungen zu entwickeln bringt in der heutigen Zeit viele Herausforderungen mit sich. Zusätzlich ist der Webauftritt das Aushängeschild eines Unternehmens und wird oftmals über eine Vielzahl von Geräten abgerufen. Durch die vielen unterschiedlichen Bildschirmgrößen kann die dynamische Darstellung der Inhalte schnell kompliziert werden. Dies führt zu einem umfangreichen Pool an Aspekten, die bei der Qualitätssicherung von Webanwendungen betrachtet werden müssen. Hinzu kommt die Auswahl an unterschiedlichen Browsern, wodurch die Darstellungsvarianten einzelner Elemente der grafischen Benutzeroberfläche erneut zu Abweichungen untereinander führen können.

Manche der im Gartner Quadranten für Testautomatisierung 2018 genannten Tools widmen sich gezielt der grafischen Benutzeroberfläche und dem Thema UI-Objekterkennung, wie etwa Eggplant (Testplant 2019d) oder TestComplete (SmartBear 2019e), welche sogar AI-gestützte UI-Objekterkennung umfassen. Andere Tools oder auch Open Source Lösungen wie Selenium (Selenium 2019a) setzten wiederum auf API-Tests und verlangen entsprechende Programmierkenntnisse von den Testern. Daher gilt es unbedingt vorab zu klären, welche Art von Tests durchgeführt werden sollen und welche Fähigkeiten die Tester mitbringen. Der Wartungsaufwand von automatisierten Tests, welche von Testern mit wenig Programmierkenntnissen definiert wurden, kann durch fehlende Erfahrung bei der Erstellung von Code schnell unerwünschte Ausmaße annehmen. Selbst über Recorder aufgezeichnete UI-Tests verlangen oft technisches Know-How, da für die Testerstellung auch immer manueller Aufwand getätigt werden muss, denkt man beispielsweise an die in der Evaluierung beschriebenen Vorgehensweisen zur Aufzeichnung unterschiedlicher UI-Objekte. Zusätzlich ergab sich im Laufe der Recherche rund um die verschiedenen Testautomatisierungs-Tools und ihre Vorgehensweisen zur UI-Objekternennung die Erkenntnis, dass die Dokumentationen der Hersteller im Umfang, wie auch qualitativ stark voneinander abweichen. Da eine umfangreiche, aktuelle und gut strukturierte Dokumentation die Einführung eines Testautomatisierungs-Tools im Unternehmen erleichtern kann, sollte auch dieser Aspekt betrachtet werden.

Um Stabilität bei der Automatisierung von Software Tests zu erreichen, muss die passende Technologie zur UI-Objekterkennung abhängig von dem technischen Aufbau der zu testenden Anwendung gewählt werden. Dabei kann durchaus eine Kombination verschiedener Technologien genutzt werden, wenn unterschiedliche Typen von UI-Objekten in der Anwendung vorhanden sind. Jedenfalls sollte aber für eine Art von UI-Objekt ein einheitliches Vorgehen für die Identifikation genutzt werden, um potentiellen Wartungsaufwand so unkompliziert wie möglich zu gestalten und die Nachvollziehbarkeit zu gewähren. Auch ein Schema für die Benennung der aufgezeichneten UI-Objekte erleichtern die Testerstellung und Wartung. XPath Werte sollten beispielsweise einheitlich in ihrer Granularität und Aufzeichnungen für die Identifikation über Image Recognition vollständig sein, denkt man an unterschiedliche Darstellungsvarianten der Browser.

Stabile automatisierte Tests zu erstellen bedeutet, die Technologie hinter der zu testenden Anwendung zu kennen und auf Basis dieses Wissens die Wahl für die UI-Objekterkennung zu

treffen. Wählt man beispielsweise Image Recognition zur UI-Objekterkennung, während ebenso statische ID’s zur Identifikation herangezogen werden können, nimmt man unnötigen Mehraufwand für die Erstellung und Wartung in Kauf. Ebenso sollte strukturiert bei der Aufzeichnung der UI-Objekte vorgegangen werden. Viele Testautomatisierungstool, wie etwa Ranorex (Ranorex 2019b) oder Eggplant (Testplant 2019e), verfügen über eine Art Repository, welches zur Verwaltung der aufgezeichneten UI-Objekte dient. Dort werden die UI-Objekte in einer Hierarchie abgelegt, analog zur Baumstruktur auf welche die grafische Oberfläche der getesteten Anwendung aufbaut.

Festzuhalten ist, dass während der Evaluierung der verschiedenen Technologien der UI-Objekterkennung kein Zugriff auf den Code der getesteten Webanwendung stattfand. Die Identifikation der UI-Objekte fand ausschließlich über die grafische Benutzeroberfläche sowie den HTML-Bestandteilen der Anwendung statt, welche allen Anwendern sichtbar sind.

Manche Technologien stellen Testautomatisierungs-Tool jedoch immer noch auf die Probe. Dynamsiche ID’s etwa erschweren die Identifikation der Elemente. Für solche Situationen soll zukünftig Aritficial Intelligence Abhilfe schaffen. Viele Hersteller von Testautomatisierungs-Tools werben bereits mit hybriden Lösungen, welche künstliche Intelligenz in unterschiedliche Bereiche der Testautomatisierung einfließen lassen. Neben der Planung der Abfolge von Testdurchläufen und Prognosen von Qualitätseinbußen ist es besonders die UI-Objekterkennung, welche zukünftig durch AI gestützt werden soll.

ABKÜRZUNGSVERZEICHNIS

API	Application Programming Interface
AI	Artificial Intelligence
DOM	Document Object Model
OCR	Optical Character Recognition
UI	User Interface
URL	Uniform Resource Locator
QA	Quality Assurance

ABBILDUNGSVERZEICHNIS

Abbildung 1: Vereinfachtes DOM Beispiel (Keith 2010: 33) 13
Abbildung 2: Gartner Magic Quadrant der Testautomatisierung (Gartner 2018) 22
Abbildung 3: Aufbau eines Jira Boards (Atlassian 2019) 24
Abbildung 4: Aufbau eines Jira Vorgangs (Atlassian 2019) 24
Abbildung 5: Ranorex Spy Aufbau (Ranorex 2019b) 27
Abbildung 6: Aufbau RFT Test Object Map (IBM 2019b) 32
Abbildung 7: Tosca XScan Aufbau (Tricentis Tosca 2019a) 35
Abbildung 8: Image Comparison mit TestComplete (SmartBear 2019c) 48
Abbildung 9: Test Complete Image Set Editor (SmartBear 2019c) 48
Abbildung 10: Eggplant Functional Image Viewer (Testplant 2019e) 60

TABELLENVERZEICHNIS

Tabelle 1: Beurteilungsmetrik zur Beurteilung der Stabilität verschiedener UI-Objekterkennungs-Technologien 20
Tabelle 2: Evaluierung der UI-Objekterkennung über Koordinaten mit Ranorex Spy 31
Tabelle 3: Evaluierung der UI-Objekterkennung über ID's mit Tosca XScan 38
Tabelle 4: Evaluierung der UI-Objekterkennung über XPath mit Ranorex Spy 44
Tabelle 5: : Evaluierung der UI-Objekterkennung über Image Recognition mit TestComplete 52
Tabelle 6: : Evaluierung der UI-Objekterkennung über OCR mit TestComplete 58

LITERATURVERZEICHNIS

Baumgartner, M., et. al. (2017). *Agile Testing. Der agile Weg zur Qualität.* Carl Hanser Verlag.

Bucsics, T., et. al. (2015). *Basiswissen Testautomatisierung. Konzepte, Methoden und Techniken.* dpunkt.verlag.

Bühler, P., Schlaich, P. & Sinner, D. (2017). *HTML5 und CSS3. Semantik - Design - Responsive Layouts*. Springer Vieweg.

Chaudhuri et. al. (2018). *Optical Character Recognition Systems for Different Languages with Soft Computing*. Springer.

Crispin, L. & Gregory J. (2008). *Agile Testing - A Practical Guide for Testers and Teams*. Addison-Wesley.

Cser, T. (2018). *AI in Test Automation.* In DZone Automated Testing Volume 2.

Ertel, A. & Laborenz K. (2017). *Responsive Webdesign. Konzepte, Techniken, Praxisbeispiele.* Rheinwerk Computing.

Garg, N. (2014). *Test Automation using Selenium WebDriver with Java*. AdactIn Group Pty Ltd.

Graham, M. & Graham, D. (2012). *Experiences of Test Automation - Case Studies of Software test Automation*. Addison-Wesley.

Hendrickson, E. (2013). *Explore it! Reduce Risk an Increase Confidence with Exploratory Testing*. O'reilly UK Ltd.

Hope, P. & Walther, B. (2008). *Web Security Testing Cookbook. Systematic Techniques to Find Problems Fast*. O'Reilly and Associates.

Hsu, T. (2018). *Hands-On Security in DevOps: Ensure continuous security, deployment, and delivery with DevSecOps*. Packt Publishing.

Humble, J. & Farley, D. (2010). *Continuous Delivery. Reliable Software Releases through Build, Test, and Deployment Automations*. Addison.Wesley.

Jorgensen, P. (2013). *Software Testing. A Craftsman's Approach*. Taylor & Francis Ltd.

Keith, J. (2010). *DOM Scripting. Web Design with JavaScript and the Document Object Model*. friendsofED.

Kinsbruner, E. (2018). *Continuous Testing for DevOps Professionals. A Practical Guide From Industry Experts.* CreateSpace Independent Publishing Platform.

Melton, J. & Buxton, S. (2006). *Querying XML. xQuery, xPath, and SQL/XML in context.* Morgan Kaufmann.

Molina, D. (2018). *Selenium Fundamentals. Speed up your internal testing by automating user interaction with browsers and web applications.* Packt Publishing.

Myers, G., Sandler C. & Badgett, T. (2011). *The Art of Software Testing.* Wiley.

Navneesh, G. (2014). *Test Automation using Selenium WebDriver with Java. Step by Step Guide.* AdactIn Group Pty Ltd.

Parker, J. R. (2010): *Algorithms for Image Processing and Computer Vision*. Wiley.

Rudd, A. (2018): *Practica Usage of Regular Expressions. An Introduction to regexes for Translators.* Create Space Independent Publishing Platform.

Russell, S. & Norvig, P. (2016). *Artificial Intelligence. A Modern Approach.* Pearson.

Suden, G. (2016). *Automated Web Testing. Step by Step Automation Guide.* CreateSpace Independent Publishing Platform.

Theis, T. (2018). *Einstieg in JavaScript. Dynamische Webseiten erstellen. Inkl. Ajax, jQuery, Onsen UI u.v.m.* Rheinwerk Computing.

Wolf, J. (2016): *HTML5 und CSS3. Das umfassende Handbuch zum Lernen und Nachschlagen*. Rheinwerk Computing.

Atlassian (2019). *Jira Software.* Abgerufen am 20.06.2019 von https://de.atlassian.com/software/jira

CoffeeScript (2019a). *Introduction to CoffeeScript*. Abgerufen am 25.01.2019 von https://coffeescript.org

Gartner (2018). *Gartner Magic Quadrant der Testautomatisierung.* Abgerufen am 24.01.2019 von https://www.tricentis.com/resource-assets/gartner-magic-quadrant-software-test-automation/

Gartner (2019). *Magic Quadrant Research Methodology.* Abgerufen am 27.01.2019 von https://www.gartner.com/en/research/methodologies/magic-quadrants-research

Google Developers (2019). *Simulate Mobile Devices with Device Mode in Chrome DevTools*. Abgerufen am 13.04.2019 von https://developers.google.com/web/tools/chrome-devtools/device-mode/

IBM (2019a). *IBM Knowledge Center. Rational Functional Tester Overview.* Abgerufen am 16.06.2019 von https://www.ibm.com/support/knowledgecenter/SSBLQQ_9.5.0/com.ibm.rational.test.ft.ovrvw.doc/topics/c_rft_ovw.html

IBM (2019b). *IBM Knowledge Center. Test Object Maps.* Abgerufen am 17.06.2019 von https://www.ibm.com/support/knowledgecenter/SSJMXE_9.5.0/com.ibm.rational.test.ft.doc/topics/OmObjectMap.html

IBM (2019c). *IBM Knowledge Center. GuiTestObject.* Abgerufen am 18.06.2019 von https://www.ibm.com/support/knowledgecenter/en/SSBLQQ_9.1.0/com.rational.test.ft.api.help/ApiReference/com/rational/test/ft/object/interfaces/GuiTestObject.html

ISO (2019a). *ISO/IEC 25010*. Abgerufen am 27.01.2019 von https://iso25000.com/index.php/en/iso-25000-standards/iso-25010

Katalon Studio (2017). *How to Solve the Common Web UI Test Automation Problems using the Katalon Studio Free Toolset.* Abgerufen am 14.04.2019 von https://medium.com/katalon-studio/how-to-solve-the-common-web-ui-test-automation-problems-using-the-katalon-studio-free-toolset-c36794f52554

Material Design (2019). *Material Design Resizer.* Abgerufen am 13.04.2019 von https://material.io/tools/resizer/

Micro Focus (2016). *The Cross-Browser Configuration Conundrum. Ensuring Websites and Apps Function Correctly.* White Paper. Abgerufen am 22.04.2019 von https://www.microfocus.com/media/white-paper/WP-The-cross-browser-configuration.pdf

Micro Focus (2019). *Silk WebDriver. Fast, efficient Selenium script creation and execution*. Abgerufen am 10.06.2019 von https://www.microfocus.com/de-de/products/silk-webdriver/overview

Ranorex (2018). *10 Best Practices in Test Automation #4. Use Reliable Locators.* Abgerufen am 26.04.2019 von https://www.ranorex.com/blog/best-practices-4-use-stable-locators/

Ranorex (2019a). *Automated Testing and Dynamic IDs*. Abgerufen am 22.01.2019 von https://www.ranorex.com/blog/automated-testing-and-dynamic-ids/

Ranorex (2019b). *Ranorex Documentation. Ranorex Spy.* Abgerufen am 10.05.2019 von https://www.ranorex.com/help/latest/ranorex-studio-advanced/ranorex-spy/introduction/

Ranorex (2019c). *Ranorex Documentation. RanoreXPath.* Abgerufen am 14.05.2019 von https://www.ranorex.com/help/latest/ranorex-studio-advanced/ranorexpath/introduction/

Ranorex (2019d). *Ranorex Documentation. Ranorex Namespace.* Abgerufen am 01.06.2019 von https://www.ranorex.com/Documentation/Ranorex/

Ranorex (2019e). *Ranorex Studio Test Automation. Ranorex Studio 9.1.* Abgerufen am 20.06.2019 von https://www.ranorex.com/what-is-new/

Selenium (2019a). *Selenium. Web Browser Automation*. Abgerufen am 24.01.2019 von https://www.seleniumhq.org

Selenium (2019b). *Selenium. Selenium WebDriver.* Abgerufen am 07.05.2019 von https://www.seleniumhq.org/projects/webdriver/

Selenium (2019c). *Selenium Documentation. Location Strategies.* Abgerufen am 09.05.2019 von https://www.seleniumhq.org/docs/06_test_design_considerations.jsp#location-strategies

Selenium (2019d). *Selenium Documentation. Selenium WebDriver.* Abgerufen am 08.06.2019 von https://www.seleniumhq.org/docs/03_webdriver.jsp

SmartBear (2019a). *Cross Browser Testing Tool/ 1500+ Real Browsers & Devices.* Abgerufen am 25.01.2019 von https://crossbrowsertesting.com

SmartBear (2019b). *Speed Up Testing & Improve Software Quality with Optical Character Recognition.* Abgerufen am 27.01.2019 von https://smartbear.de/SmartBearBrand/media/Images/resources/ebooks/TC_WP_OCR.pdf

SmartBear (2019c). *TestComplete Documentation. Image Based Testing.* Abgerufen am 29.03.2019 von https://support.smartbear.com/testcomplete/docs/testing-with/object-identification/image-based/index.html

SmartBear (2019d). *TestComplete Documentation. Introduction to Object Driven Testing.* Abgerufen am 20.04.2019 von https://smartbear.com/learn/automated-testing/intro-to-object-recognition/

SmartBear (2019e). *TestComplete Documentation. GUI Object Recognition.* Abgerufen am 18.05.2019 von https://smartbear.com/product/testcomplete/features/gui-object-recognition/

SmartBear (2019f). *TestComplete Documentation. Optical Character Recognition*. Abgerufen am 27.05.2019 von https://support.smartbear.com/testcomplete/docs/testing-with/object-identification/ocr/index.html

Testplant (2019a). *Eggplant Documentation. Working with Optical Character Recognition.* Abgerufen am 27.01.2019 von http://docs.testplant.com/ePF/using/epf-working-with-optical-character-recognition-ocr.htm

Testplant (2019b). *Eggplant Documentation. Cross-Browser and Cross-Platform Scripting.* Abgerufen am 09.05.2019 von http://docs.testplant.com/ePF/using/epf-cross-browser-scripting.htm

Testplant (2019c). *Eggplant Documentation. Selenium WebDriver Teting with Eggplant Functional.* Abgerufen am 16.05.2019 von http://docs.testplant.com/ePF/using/epf-selenium-webdriver-testing.htm

Testplant (2019d). *Eggplant Documentation. Image and OCR Searches.* Abgerufen am 26.05.2019 von http://docs.testplant.com/ePF/SenseTalk/stk-image-ocr-searches.htm

Testplant (2019e). *Eggplant Documentation. Capturing Images.* Abgerufen am 26.05.2019 von http://docs.testplant.com/ePF/using/epf-capturing-images.htm

Testplant (2019f). *Eggplant AI. The brain of the Digital Automation Intelligence Suite.* White Paper. Abgerufen am 05.07.2019 von https://eggplant.io/products/dai/eggplant-ai

Tricentis Tosca (2019a). *Scanning XModules with Tosca XScan.* Abgerufen am 27.03.2019 von https://documentation.tricentis.com/en/1010/content/tchb/create_modules_xscan.htm#IdentifyingBySmartID

Tricentis Tosca (2019b). *Tricentis Tosca Documentation. Recovery.* Abgerufen am 19.04.2019 von https://documentation.tricentis.com/de/1000/content/tbox/recovery.htm

Tricentis Tosca (2019c). *Tricentis Tosca Documentation. Steering XBrowser controls.* Abgerufen am 20.05.2019 von https://documentation.tricentis.com/tosca/1210/en/content/engines_3.0/xbrowser/xbrowser_controls.htm#Id_XPath

Windows (2019a). *Inspect - Windows applications.* Abgerufen am 25.01.2019 von https://docs.microsoft.com/de-at/windows/desktop/WinAuto/inspect-objects

W3C (2017). *XML Path Language (XPath) 3.1. W3C Recommendation.* Angerufen am 20.05.2019 von https://www.w3.org/TR/xpath-31/