iSYS Software GmbH
Individualsoftware und IT-Beratung

Konzeption einer Methodik zur systematischen Auswahl eines Testautomatisierungstools für ein Softwareentwicklungsprojekt am Beispiel der Immobilienverwaltung

Conception of a methodology for the systematic selection of a test automation tool for a software development project, using the example of real estate management

Bachelorarbeit

zur Erlangung des akademischen Grades Bachelor of Science B.Sc.

Vorgelegt an der
Hochschule München
Fakultät für Informatik und Mathematik

Eingereicht von:
Erzsébet Nicole Harmat
Studiengang: Wirtschaftsinformatik
Matrikelnummer: 19318417

Erstprüfer: Prof. Dr. Peter Mandl
Betreuer: Josef Neuwirth, Ingo Richter
Unternehmen: iSYS Software GmbH

Abgabetermin: 02.08.2021

Erklärung gemäß § 16 Abs. 10 APO i.V.m. § 35 Abs. 7 RaPO

Hiermit erkläre ich, Erzsébet Nicole Harmat, dass ich die vorliegende Bachelorarbeit selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Ort, Datum Unterschrift

München, 02. August 2021

Inhaltsverzeichnis

Abkürzungsverzeichnis

A-Nr.	Anforderungsnummer
API	Programmierschnittstelle (Application Programming Interface)
bspw.	beispielsweise
bzw.	beziehungsweise
CI/CD	Continuous Integration/Continuous Deployment
DevExpress	Developer Express Inc.
DevOps	Development and IT Operations
d. h.	das heißt
ggf.	gegebenenfalls
GUI	grafische Benutzeroberfläche (Graphical User Interface)
ID	Identifikationsnummer
IDE	integrierte Entwicklungsumgebung (Integrated Development Environment)
IoT	Internet of Things
ISO	internationale Organisation für Normung (International Organization for Standardisation)
iSYS	iSYS Software GmbH
KEGP	Kriteriumerfüllungsgradpunktzahl
K-Nr.	Kriterien-Nummer
max KEGP	maximale Kriteriumerfüllungsgradpunktzahl
MLR	multivokale Literaturrecherche
OS	Betriebssystem (Operating System)
QFS	Quality First Software GmbH

QS	Qualitätssicherung
SLR	systematische Literaturrecherche
SUT	System unter Test
Ta	Testautomatisierung
TaT	Testautomatisierungstool
usw.	und so weiter
VCS	Versionsverwaltungssystem (Version Control System)
VS3	VerwalterService 3
VS Code	Visual Studio Code
z. B.	zum Beispiel

1 Einleitung

Dieses Kapitel stellt die Motivation und Ziele der Bachelorarbeit vor. Die Bachelorarbeit entstand im Rahmen eines Softwareentwicklungsprojekts der iSYS Software GmbH. Das Unternehmen aus München ist seit über 25 Jahren am Markt und ein stabiler, verlässlicher Partner in den Bereichen individuelle Softwareentwicklung, Solutions und IT-Consulting [24].

1.1 Motivation und Forschungsfrage

Das Testen von Software gilt als eines der wichtigsten Elemente in der Softwareentwicklung [19][20]. Mit dem Softwaretesten wird die Qualität der Software und damit die Kundenzufriedenheit gesichert. Die Notwendigkeit, Testautomatisierung (Ta) in einem Projekt einzuführen, ist in den letzten Jahren gestiegen. Durch die vermehrte Präsenz von Testautomatisierung geriet die Auswahl eines Testautomatisierungstools (TaT) zunehmend in den wissenschaftlichen Fokus [6] [59].

Die iSYS Software GmbH (iSYS) entwickelte eine Individualsoftware, die von der Hausbank München eG (Hausbank) beauftragt wurde. Im Rahmen eines Projekts wurde die Software VerwalterService 3 (VS3) implementiert. VS3 ist eine Client-Server-Software im Bereich der Wohnungswirtschaft zur Immobilienverwaltung. Mit dieser Software wird die Immobilienfinanzbuchhaltung für alle Einnahmen und Ausgaben durchgeführt, die ein Wohnobjekt betreffen können. Die Hausbank stellt ihren KundInnen, den Immobilienverwaltern, die die Wohnobjekte verwalten, die Software zur Verfügung.

Die VS3 Client-Server-Anwendung wurde im Jahr 2003 in Betrieb genommen. Die Software wird seitdem fortlaufend erweitert und gewartet. Auf Basis von Kundenwünschen werden weitere neue Features implementiert. In der Wartung werden die Technologien aktualisiert und Fehler behoben.

Die Hausbank München eG plant die Benutzeroberfläche der Anwendung auf eine Weboberfläche zu migrieren. Damit soll ein modernes und internetbasiertes Softwaresystem für die KundInnen der Hausbank realisiert werden. Als erster Ansatz dafür ist 2017 ein Webclient-Prototyp in ReactJS implementiert worden. Das Projekt ruht seitdem, wird aber zukünftig wieder im Rahmen eines neuen Projekts aufgegriffen.

Das neue Projekt wird als agiles Software-Entwicklungsprojekt umgesetzt – nach der in iSYS gelebten DevOps Philosophie [51]. Dabei werden die Softwareversionen in kurzen Iterationen geliefert. Um die hohe Qualität der Versionen zu sichern, bestehen der Be-

darf und die Notwendigkeit, eine Testautomatisierung und damit auch ein Testautomatisierungstool einzuführen. Die Automation als Bauelement von DevOps wird in der Abbildung 1 dargestellt.

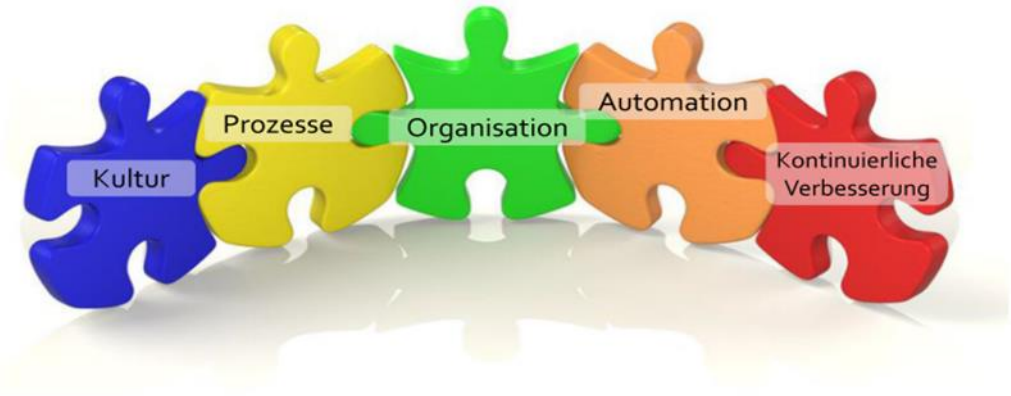

Abbildung 1: Die fünf Handlungsfelder von DevOps (Quelle: [51])

Für die Erarbeitung der Bachelorarbeit wurde eine grundlegende wissenschaftliche Literaturrecherche durchgeführt. Teil der Forschungsmethode ist auch die Analyse der Erfahrungsberichte über industrielle Projekte und Kontexte, die von ExpertInnen online geteilt werden. Diese Quellen werden als „graue Literatur“ bezeichnet. Die Bedeutung und Notwendigkeit der Analyse und Einbeziehung der grauen Literatur wird immer mehr erkannt und in der Forschung berücksichtigt [4][15][16][42][44].

Eine Vielzahl wissenschaftlicher Arbeiten hebt die Notwendigkeit des Einführens der Testautomatisierung und damit von Testautomatisierungstools hervor, um den Marktentwicklungen und den Kundenbedürfnissen gerecht zu werden. Dabei weisen mehrere AutorInnen auf das Auswahlproblem von passenden Testautomatisierungstools für Projekte hin [15][42]. Die führenden Marktforschungs- und -analyseunternehmen in der Informationstechnologie, Capgemini und Forrester, berichten in ihren letzten Jahresreports über die wachsende Bedeutung und Notwendigkeit der Automatisierung und wenden sich den dazu führenden Gründen zu [6][59]. Es konnte auch festgestellt werden, dass die Art und Weise, wie ein passendes projektbezogenes Testautomatisierungstool gefunden werden kann, noch nicht ausführlich genug untersucht wurde[42].

Die Bachelorarbeit behandelt die Herausforderung der Auswahl eines passenden Test für ein Projekt und setzt sich insbesondere mit der Erarbeitung einer Methodik zur Auswahl eines Testautomatisierungstools auseinander. Die grundsätzliche Frage dabei lautet, ob es möglich ist, eine Methodik auszuarbeiten, die es ermöglichen könnte, projektbezogen mit hoher Wahrscheinlichkeit das passendste Testautomatisierungstool auszuwählen.

1.2 Ziel der Bachelorarbeit

Ziel der Bachelorarbeit ist

1. den Stand der Forschung im Bereich der Auswahl eines Testautomatisierungstools aufgrund einer wissenschaftlichen Literaturrecherche mit Einbeziehung der grauen Literatur darzustellen,

2. eine Methodik zur Auswahl eines Testautomatisierungstools zu konzipieren und

3. die in Punkt 2 entwickelte Methodik durch Auswahl eines Testautomatisierungstools für den Webclient-Prototyp des VS3-Web-Anwendungprojekts zu evaluieren.

1.3 Abgrenzung der Themenstellung

Die Bachelorarbeit befasst sich nicht mit unterschiedlichen methodischen Ansätzen zur Testfall- und Testdatenspezifikation wie auch nicht mit verschiedenen Arten der Durchführung von automatisierten Tests und möglichen Fehlerquellen.

Der Schwerpunkt liegt nicht auf dem Testprozess als solchem. Es wird auch nicht im Detail auf die Planung, Überwachung und Steuerung von Testfällen eingegangen.

1.4 Aufbau der Bachelorarbeit

Hier folgt ein Überblick über den Aufbau der Bachelorarbeit:

Kapitel 2 – Stand der Technik – stellt die theoretischen Grundlagen für die anschließenden Kapitel vor und weist die Erkenntnisse aus der Literaturrecherche auf. Es liefert einen Überblick über die Testautomatisierung und über die Testautomatisierungstools. Es werden die verschiedenen Techniken der Testautomatisierung vorgestellt.

Kapitel 3 – Methodik zur Auswahl eines Testautomatisierungstools – beschreibt die vier Phasen der erstellten Methodik: (1) die Grundsatzentscheidung, (2) die Anforderungen und Kriterien, (3) die Bewertung und Evaluierung und (4) die Auswertung und Auswahl.

Kapitel 4 – Auswahl eines Testautomatisierungstools für das VS3-Web-Anwendungsprojekt – präsentiert die Umsetzung der in Kapitel 3 erstellten Methodik. Hierbei wird detaillierter auf die vier Testautomatisierungstools, die aus der Sicht des iSYS Unternehmens für das VS3-Web-Anwendungsprojekt infrage kommen, eingegangen.

Kapitel 5 – Diskussion – beinhaltet die Analyse der Erkenntnisse und zeichnet die Bereiche für mögliche weiterführende Untersuchungen auf.

Kapitel 6 – Zusammenfassung und Ausblick – fasst die Erkenntnisse und die Implikationen der Ergebnisse zusammen.

2 Stand der Technik

In diesem Kapitel werden die Grundlagen und Voraussetzungen für diese Bachelorarbeit beschrieben. Um die Aussagen zu belegen, wurde eine grundlegende Literaturrecherche unternommen. Die Literaturrecherche beinhaltete folgende Stichwörter zum Thema der Bachelorarbeit:

- Testautomatisierung (Englisch: Software Test Automation)
- Testautomatisierungstools (Englisch: Software Test Automation Tools)
- Testen von Webanwendungen (Englisch: Web Application Testing)

Die folgenden Suchmaschinen bzw. Portale wurden für die Literaturrecherche benutzt:

- ACM
- IEEE Xplore
- OPAC der Hochschule München
- Google Scholar

Die Suche wurde mit den Schlüsselbegriffen in verschiedenen Kombinationen mit AND oder OR Operatoren vorgenommen. Ziel war insbesondere, den Forschungsstand der letzten fünf Jahre zu recherchieren. Deshalb war der Zeitraum der Suchergebnisse auf „seit 2016“ begrenzt. Die zitierte Literatur aus den recherchierten Werken ist in dieser Bachelorarbeit auch als Forschungsquelle miteinbezogen.

In dieser Bachelorarbeit ist der State of the Practice, der von ExpertInnen selbst erfasst, dokumentiert und online geteilt wurde, ebenfalls berücksichtigt worden, insbesondere beim Evaluieren der Testautomatisierungstools. Diese Art von Literatur wird als „graue Literatur“ bezeichnet. Die Recherche nach grauer Literatur wurde mit Google durchgeführt. Zur grauen Literatur gehören unter anderem die Expertenmeinungen, Berichte, Blogs und White Papers [4][42][44].

2.1 Testen der Software

Die Begriffsdefinitionen, die in Beziehung mit dem Softwaretesten stehen und in dieser Arbeit erwähnt werden, sind dem Glossar des ISTQB® entnommen [22]. „Das ISTQB® (International Software Testing Qualifications Board) hat das Schema ‚ISTQB® Certified Tester‘ definiert, das weltweit führend in der Zertifizierung von Kompetenzen im Softwaretesten geworden ist“ [21].

Das **Testen** wird nach ISTQB® wie folgt definiert:

„Der Prozess, der aus allen statischen und dynamischen Lebenszyklusaktivitäten besteht, die sich mit der Planung, Vorbereitung und Bewertung einer Komponente oder eines Systems und zugehörigen Arbeitsergebnissen befassen, um festzustellen, ob sie festgelegte Anforderungen erfüllen, für den Zweck geeignet sind sowie um etwaige Fehlerzustände zu finden“ [22].

Es gibt zwei Testarten für Softwaresysteme, mit welchen festgestellt werden kann, ob die Funktionalitäten der festgelegten Anforderungen an das Softwareprodukt entsprechen:

- manuelles Testen und
- automatisiertes Testen (Testautomatisierung).

Das manuelle Testen ist der Ausgangspunkt des funktionalen Testens, um die Funktionalitäten des Systems zu testen. In klassischen Software-Entwicklungsprojekten steht das manuelle Testen im Vordergrund [2]. Der **funktionale Test** wird nach ISTQB® wie folgt definiert:

„Testen, welches durchgeführt wird, um die Erfüllung der funktionalen Anforderungen durch eine Komponente oder ein System zu bewerten“ [22].

Wenn aber eine langjährige Produktweiterentwicklung geplant ist, ist zu empfehlen, Testautomatisierung einzusetzen [2]. Die Testautomatisierung ist eine werkzeugorientierte Domäne, die im nächsten Unterabschnitt erörtert wird.

2.2 Testautomatisierung

Die **Testautomatisierung** wird nach ISTQB® wie folgt definiert:

„Der Einsatz von Software zur Durchführung oder Unterstützung von Testaktivitäten“ [22].

Darüber hinaus ist im ISTQB®-Glossar die Definition von „**Automatisierung der Testdurchführung**“ auffindbar. Diese Definition beschreibt die Testautomatisierung noch genauer:

„Die Verwendung einer Software, z. B. eines Capture/Replay-Werkzeugs[1], um die Ausführung von Tests zu steuern, tatsächliche mit erwarteten Ergebnissen zu vergleichen, die definierten Vorbedingungen herzustellen sowie weitere Testüberwachungs- und Berichtsfunktionen durchzuführen“ [22].

Die Notwendigkeit des Vorhandenseins einer Testautomatisierung im Lebenszyklus der Softwareentwicklung ist von Unternehmen erkannt [6]. Die Softwareentwicklung erfolgt als agiler Prozess, um Kundenwünsche zu erfüllen [3]. Die Lieferung von Inkrementen, die als ein funktionsfähiges Zwischenprodukt in der agilen Softwareentwicklung gelten, erfolgt in kurzen Iterationen, d. h. oft auch mehrmals pro Woche. Dabei sind in der jeweils neuen Version zwei Arten von Funktionalitäten vorhanden:

1. die neu implementierten oder geänderten Funktionalitäten und
2. die in allen vorherigen Versionen implementierten und aktuell nicht geänderten Funktionalitäten.

Die neuen Funktionalitäten werden auf Korrektheit manuell getestet. Um die Qualität des schon vorhandenen Softwaresystems sicherzustellen, ist es nötig, die bestehenden Funktionalitäten auch zu testen. Diese Testprozesse sind die sogenannten **Regressionstests**. Der Regressionstest wird nach ISTQB® wie folgt definiert:

„Eine Art änderungsbezogenes Testen, um festzustellen, ob in unveränderten Bereichen der Software Fehlerzustände eingebaut oder freigelegt wurden“ [22].

Die Regressionstests sollten mit jeder neuen Version der Software durchgeführt werden [32]. Durch manuelles Testen würde das Ausführen von Regressionstests sehr lange dauern. „So sind im agilen Umfeld die automatisierten Regressionstests ein ‚must have‘, um die Testaufgaben in den kurzen Intervallen der Entwicklungszyklen zeitgerecht und in guter Qualität abwickeln zu können“ [3]. In manchen komplexen Softwaresystemen ist die manuelle Durchführung der Regressionstests nicht machbar. In solchen Projekten ist die Einführung der Testautomatisierung unentbehrlich.

Das manuelle Ausführen von Regressionstestprozessen ist zudem sehr zeit- und kostenintensiv. Die Automatisierung dieser Prozesse kann zur Optimierung von Kosten, Zeit- und Personalressourcen führen. Aber auch die Software-Testautomatisierung erfordert Investitionen in Zeit, Kosten und Aufwand, selbst wenn bei der Testautomatisierung Open-Source-Tools eingesetzt werden [42]. Diese fallen jedoch in Relation zum manuellen Testen ingesamt deutlich niedriger aus.

[1] Die Definition von Capture/Replay-Werkzeug befindet sich in Abschnitt 2.3

Mit den automatisierten Regressionstests können alle wichtigen Anforderungen an das Softwaresystem abgedeckt und getestet werden. Für die Automatisierung von Softwaretests gilt die Faustregel, dass ein Testfall sich insbesondere dann für die Umstellung auf Automatisierung eignet, wenn er das Potenzial besitzt, besonders häufig ausgeführt zu werden [2]. Mit den Regressionstests wird die Qualität des Softwaresystems sichergestellt. Es müssen nicht alle Qualitätsmerkmale geprüft werden, sondern die aus der Sicht der Anforderungen relevanten Qualitätsmerkmale. Qualitätsmerkmale des Softwaresystems nach ISO 25010 sind der folgenden Abbildung 2 zu entnehmen.

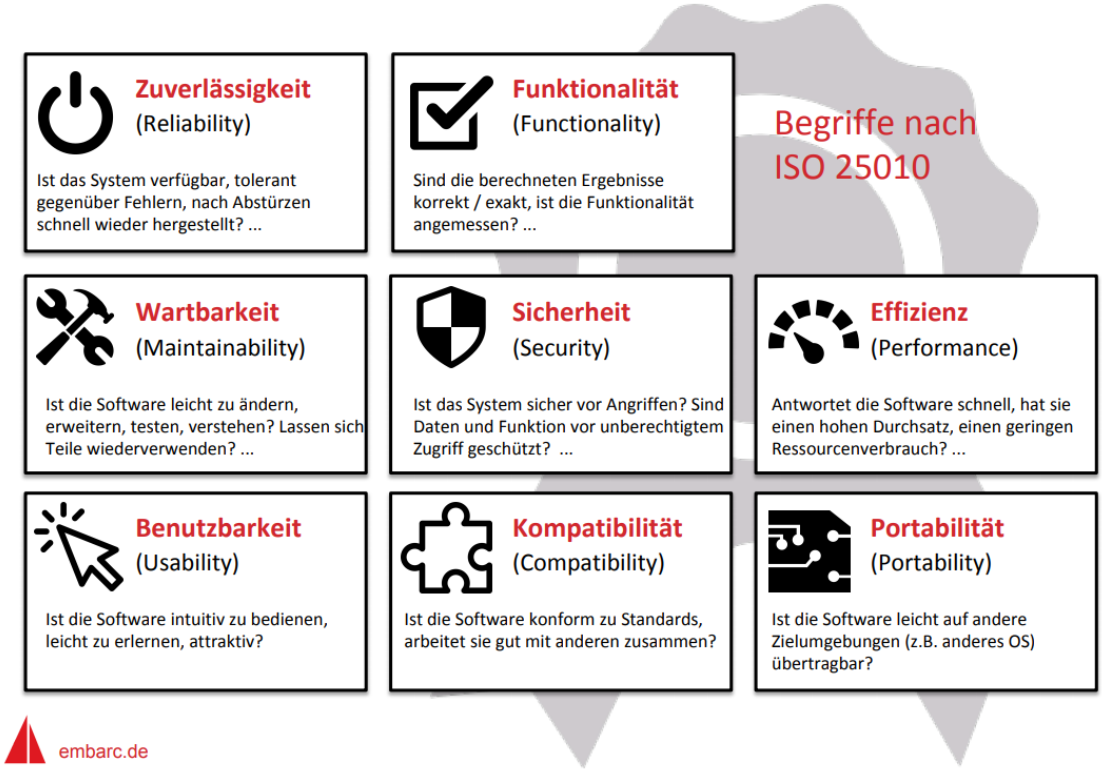

Abbildung 2: Die Software-Qualitätsmerkmale nach ISO 25010
(Quelle: (c) embarc GmbH, mit freundlicher Genehmigung [13])

Die Notwendigkeit der Testautomatisierung zeigt auch die Tatsache, dass zahlreiche Unternehmen in den letzten Jahren die Testautomatisierung vorantreiben und nach ExpertInnen suchen. Cerioli et al. sind zu der Erkenntnis gekommen, dass automatisiertes Testen die Szene der Stellenanzeigen dominiert und bei Weitem mehr gefragt ist als manuelles Testen. Vergleicht man Stellenanzeigen, die nur eine der beiden Kategorien erwähnen, ist das Verhältnis zwischen beiden 1:20 (eine Anzeige zum manuellen Testen auf 20 zum automatisierten Testen) [7].

Eine sinnvolle Regressionstest-Durchführung ist auf die Unterstützung leistungsfähiger Testautomatisierungstools angewiesen [32], die im nächsten Abschnitt beleuchtet werden.

2.3 Testautomatisierungstools

Auf der Suche nach der Definition für den Begriff **Testautomatisierungstool** war es interessant zu sehen, dass nach ISTQB® keine direkte Beschreibung existiert. Die Begriffe *Testautomatisierungstool* und *Testautomatisierungswerkzeug* werden in dieser Bachelorarbeit als Synonyme für nachstehend genannte Begriffe gesehen.

Dem folgend werden an dieser Stelle zwei Definitionen aus dem ISTQB-Glossar genannt. Als erstes die Definition des **Testwerkzeugs**:

„Software oder Hardware, die eine oder mehrere Testaktivitäten unterstützt“ [22].

Zweitens die Definition vom **Capture/Replay-Werkzeug** (Synonym: **Record/Playback-Testautomatisierungstool)**, das in der Definition der Testautomatisierung als Beispiel beinhaltet ist:

„Ein Werkzeug zur Unterstützung der Testausführung. Eingaben der Benutzer werden während der manuellen Testdurchführung zum Erzeugen von ausführ- und wiederholbarer Testskripten aufgezeichnet und verwendet. Solche Testwerkzeuge werden häufig zur Unterstützung automatisierter Regressionstests genutzt“ [22].

Ein Testautomatisierungstool interagiert mit der Benutzeroberfläche der Webanwendung. Dabei führt es Aktionen auf den einzelnen Webelementen aus. Das Testautomatisierungstool ermöglicht es, Benutzeraktionen zu simulieren und diese zu automatisieren [52]. Dabei werden aktuell zwei Techniken angewendet, die im folgenden Unterabschnitt erläutert werden.

2.3.1 Technologien der Browser-Testautomatisierung

Bei den automatisierten Tests für Webanwendungen wird der Webbrowser durch die Testautomatisierungssoftware bedient. Es gibt zwei Technologien, die das ermöglichen: die Proxy-Server-Technologie und die WebDriver-Technologie.

Bei der **Proxy-Server-Technologie** werden die Daten vom Webserver über den Proxy geleitet und manipuliert. An den ursprünglichen Daten werden Änderungen und Erweiterungen vorgenommen. Der entstandene Code wird an den Browser weitergeleitet und die Interaktionen in der GUI der Webanwendung ausgeführt. Im Vergleich zur WebDri-

ver-Technologie interagiert die Proxy-Technologie nicht mit dem Driver des Webbrowsers. Deshalb ist eine Installation der unterschiedlichen browserspezifischen Driver nicht nötig. Damit wurde in der Proxy-Technologie die Browserabhängigkeit umgangen. Lediglich die Webbrowser müssen für diesen Ansatz HTML5-kompatibel sein. Eine schematische Darstellung des Proxy-Ansatzes ist in der Abbildung 3 zu sehen.

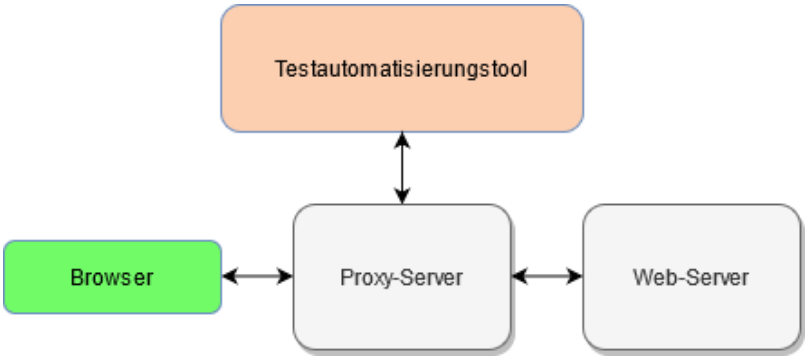

Abbildung 3: Die Proxy-Server-Technologie der Testautomatisierung
(Quelle: eigene Darstellung, angelehnt an [9])

Bei der **WebDriver-Technologie** wird der Driver eines Browsers angesprochen. Das geschieht mit dem WebDriver, der von der Firma Selenium entwickelt wurde. Der WebDriver wurde in das W3C (World Wide Web Consortium) aufgenommen, das für Standardisierungen von Webtechnologien verantwortlich ist. Mit dem WebDriver wird sichergestellt, dass über ein einheitliches API die verschiedenen Browser bedienbar sind. Dem API entsprechend wurden in vielen Fällen die browserspezifischen Driver von Browserherstellern entwickelt, um mit WebDriver interagieren zu können. Die WebDriver-Engine wird durch Skripte, die mit einem Testautomatisierungstool erstellt sind, gesteuert [49]. Eine schematische Darstellung ist in der Abbildung 4 zu sehen.

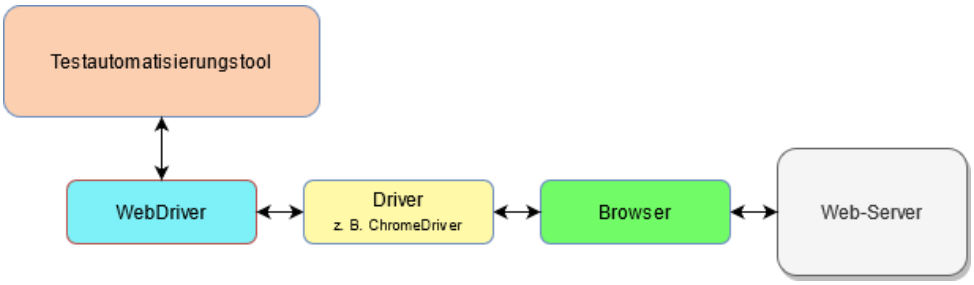

Abbildung 4: Die WebDriver-Technologie der Testautomatisierung
(Quelle: eigene Darstellung, angelehnt an [49])

Die automatischen Tests können auf zwei Arten erstellt werden. Die beiden Erstellungsmethoden von automatisierten Tests werden in folgendem Unterabschnitt erläutert.

2.3.2 Erstellungsmethoden der automatisierten Tests

Eine Methode ist die Implementierung der Testfälle. Dies erfordert Wissen über die Grundlagen von Webanwendungen. Für die Implementierung der Testfälle sollten HTML-, CSS-, Programmier- und Testautomatisierungskenntnisse vorhanden sein, um unter anderem Elemente zu identifizieren und Selektoren zu erstellen. Gute Programmierkenntnisse ermöglichen es, die Testfälle wartbarer zu implementieren. Der Code, der bei der Erstellung der Testfälle implementiert wird, sollte die Regeln von Clean Code befolgen, wie beispielsweise eindeutige Namen für Variablen, Funktionen und Klassen. Der Quellcode sollte einfach zu lesen und zu ändern sein [52]. Die Implementierung des Codes könnte sich nachteilig sowohl auf den finanziellen Aufwand als auch auf die Geschwindigkeit der Veröffentlichung auswirken.

Die zweite Methode ist die Aufnahme der Testfälle. Hierbei werden die Testfälle schneller erstellt. Bei der Aufzeichnung der Testfälle werden Interaktionen mit der GUI einer Webanwendung getätigt, aufgenommen und gespeichert. Die Aufzeichnungen mit TaTs erfordern stellenweise eine Erweiterung in Form von Skripten, die implementiert werden müssen. Manche Testfall-Aufnahmen müssen in eine entsprechende Programmiersprache konvertiert und erweitert bzw. geändert werden.

Es gibt zahlreiche Testautomatisierungstools auf dem Markt. Manche werden als Open-Source-Produkte kostenlos angeboten, andere als kommerzielle Produkte, die den Kauf einer Lizenz benötigen. Bei Open-Source-Testautomatisierungstools werden die Testfälle vor allem programmiert. Bei kommerziellen Testautomatisierungstools werden die Testfälle vorwiegend mit dem Einsatz der Record/Playback-Funktion realisiert. Die Auswahl eines Testautomatisierungstools ist eine komplexe Aufgabe, die im nächsten Abschnitt erörtert wird.

2.4 Auswahl eines Testautomatisierungstools

Das Fehlen der richtigen Testautomatisierungstools für Testaktivitäten in einem Projekt und der Mangel an qualifiziertem Personal wurden als Hindernisse für die Testautomatisierung identifiziert [6]. Es stellte sich heraus, dass die Suche nach einem passenden Software-Testautomatisierungstool für ein bestimmtes Projekt keine triviale Aufgabe ist [41]. Es besteht Bedarf an praktischen und effizienten Methoden zur Durchführung von Tool-Evaluationen, die zuverlässige Evidenz liefern [43].

In der Literatur wurde die Testautomatisierung und die Testautomatisationswerkzeug- und Testautomatisierungsberatung als der Hauptbereich für Verbesserungsmöglichkeiten erkannt. Die Beratung zu Testautomatisierungstools wurde als die am häufigsten nachgefragte Dienstleistung von externen BeraterInnen identifiziert [23].

Passos et al. stellten in ihrer Untersuchung bezüglich der Auswahl des Testautomatisierungstools fest, dass die Entscheidungen großteils auf Meinungen von ExpertInnen basieren [35].

Es wird behauptet, dass ein *Confirmation Bias*[2] bei der Auswahl und dem Benutzen des Werkzeugs zu falschen Lösungen führen kann. „Wenn Ihr einziges Werkzeug ein Hammer ist, dann ist es verlockend, alles wie einen Nagel zu behandeln“ (Maslowscher Hammer[3]) [33] [42].

Die Ergebnisse von Raulamo-Jurvanen et al. deuten darauf hin, dass mehr als nur drei befragte ExpertInnen erforderlich sind, um zuverlässige Erkenntnisse für die Bewertung von Testautomatisierungstools zu gewinnen. Sie kommen zu der Schlussfolgerung, dass im Durchschnitt die Meinungen von sieben ExpertInnen ein angemessenes Genauigkeitsniveau liefern könnten. Wesentlich mehr ExpertInnen könnten ein exzellentes Niveau der Zuverlässigkeit geben. Das Vertrauen in die Meinung von nur einem oder wenigen ExpertInnen ist fragwürdig und könnte zu falschen Entscheidungen führen [43].

Es wurden Untersuchungen durchgeführt, die sich auf einige Werkzeuge konzentrierten und sie beschrieben und/oder verglichen, wie z. B. in [19][27][60]. Auf der anderen Seite finden sich in der grauen Literatur auch zahlreiche Quellen, die sich auf das Vergleichen, die Auswahl und den Einsatz von Testautomatisierungstools beziehen [15][16][42]. Das Internet ermöglicht es den Menschen, zusammenzuarbeiten und Informationen, Erfahrungen und Wissen über die damit verbundenen Vorteile, Herausforderungen und Einschränkungen auszutauschen [42].

Garousi und Mäntylä erforschten in ihrer Arbeit, *wann* und *was* automatisiert werden soll. Dafür haben sie eine *multivokale Literaturrecherche (MLR)*[4] vorgenommen. Als Schlussfolgerung empfehlen sie die Entwicklung systematischer, empirisch validierter Entscheidungsunterstützungsansätze, weil sie die vorhandenen Ratschläge in der grauen Literatur oft als unsystematisch und auf schwachen empirischen Evidenzen basierend beurteilten [15].

[2] **Confirmation Bias** ist ein Begriff der Kognitionspsychologie, der die Neigung bezeichnet, Informationen so auszuwählen, zu ermitteln und zu interpretieren, dass diese die eigenen Erwartungen erfüllen (bestätigen). [https://de.wikipedia.org/wiki/Bestätigungsfehler].

[3] **Maslowscher Hammer** bezeichnet die Beobachtung, dass Menschen, die mit einem Werkzeug (oder einer Vorgehensweise) gut vertraut sind, dazu neigen, dieses Werkzeug auch dann zu benutzen, wenn ein anderes Werkzeug besser geeignet wäre. [https://de.wikipedia.org/wiki/Law_of_the_Instrument].

[4] **Multivokale Literaturrecherche (MLR)** ist eine Form der SLR (systematische Literaturrecherche), die neben der wissenschaftlichen Literatur die graue Literatur auch einbezieht [15].

3 Methodik zur Auswahl eines Testautomatisierungstools

Die Auswahl eines Testautomatisierungstools für ein Projekt ist keine *ad hoc* Aktion, sondern ein Prozess. Ein Prozess besteht grundsätzlich aus mehreren Phasen. So kann auch ein Testautomatisierungstool-Auswahlprozess in mehrere Phasen aufgeteilt werden. Nach Baumgartner et al. besteht dieser Prozess aus vier Phasen [2]:

- Grundsatzentscheidung
- Anforderungen und Kriterien
- Evaluierung
- Auswertung und Auswahl

Alle Phasen können der Abbildung 5 entnommen werden [2].

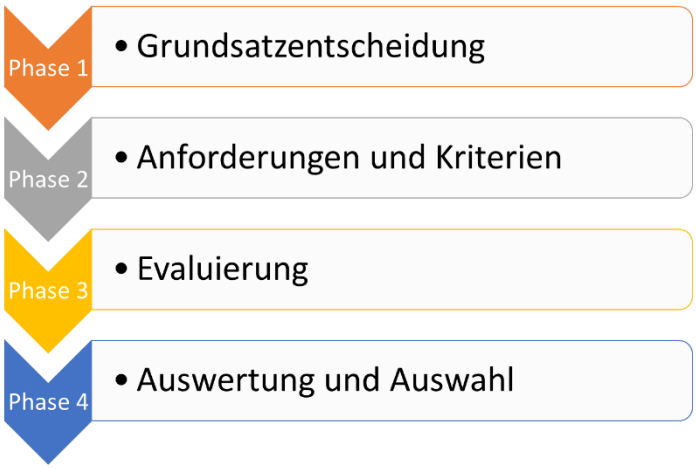

Abbildung 5: Auswahlprozess eines Automatisierungswerkzeuges
(Quelle: eigene Darstellung, angelehnt an [2])

Die oben aufgelisteten vier Phasen sollen als Basis für die angestrebte Methodik zur Auswahl eines Testautomatisierungstools dienen. In den folgenden Unterabschnitten werden die vier Phasen der Methodik im Detail beschrieben. Die Beschreibungen sind für eine transparente Darstellung des Auswahlprozesses notwendig.

3.1 Phase 1: Grundsatzentscheidung

Die Grundsatzentscheidung trägt ihr wichtigstes Merkmal schon in ihrem Namen. Sie ist ein wesentlicher Schritt für die Strukturen bzw. Organisationen, bei welchen ein Testautomatisierungstool eingesetzt werden soll. Dabei soll die Entscheidung getroffen werden, ob die Testautomatisierung für das Projekt notwendig ist und ob ein Testautomatisierungstool eingeführt wird. Unter Strukturen bzw. Organisationen wird in dieser Bachelorarbeit ein Projekt verstanden.

Die Beantwortung der Fragen, wie, wann und warum eine Grundsatzentscheidung getroffen werden kann und/oder soll, ist bereits ein sehr breiter und tiefer Prozess. Diese Phase wird an dieser Stelle nicht weiter analysiert bzw. bearbeitet. Es wird für die Methodik angenommen, dass eine positive Entscheidung zur Einführung eines Testautomatisierungstools getroffen wurde.

3.2 Phase 2: Anforderungen und Kriterien

Die zweite Phase befasst sich mit der Identifikation von Anforderungen, nach denen Kriterien abgeleitet werden, und mit der Bewertung, der Anzahl und der Gewichtung dieser abgeleiteten Kriterien.

3.2.1 Identifikation der Anforderungen

Vor der Auswahl des Testautomatisierungstools müssen zuerst die Anforderungen identifiziert werden, die von einem Projekt an das zukünftige Testautomatisierungstool gestellt werden. Hierbei dienen die Projektdokumentationen als Quelle. Die Dokumentationen werden analysiert und aufgrund von verwendeten Technologien, verfügbaren Ressourcen, Qualifikation der Software-TesterInnen usw. werden die für das Projekt wichtigen Anforderungen identifiziert. Darauffolgend werden die identifizierten Anforderungen mit dem/der ProjektleiterIn und/oder mit erfahrenen, in dem betroffenen Projekt involvierten QS-MitarbeiterInnen besprochen und ergänzt.

3.2.2 Der individuelle Kriterienkatalog

Der individuelle Kriterienkatalog ist eine Liste oder ein Satz von Kriterien. Diese Kriterien werden von den identifizierten Anforderungen abgeleitet. Diese Ableitung wird im nächsten Unterabschnitt näher beschrieben. Die Erstellung des individuellen Kriterienkataloges ist, neben den Gewichtungen der Kriterien dieser Liste, der entscheidende Teil des Auswahlprozesses. Die Gewichtungen werden im Unterabschnitt 3.2.5 detailliert erörtert.

3.2.3 Ableitung der Kriterien

Beim Erstellen des individuellen Kriterienkataloges müssen die Kriterien von den identifizierten Anforderungen abgeleitet werden. Von einer identifizierten Anforderung können ein oder mehrere Kriterien abgeleitet werden. Diese Ableitung der Kriterien kann über verschiedene Ansätze erfolgen.

1. Eine Möglichkeit ist die Anwendung von in der Organisation vorhandenem Expertenwissen. Dabei wird ein individueller Kriterienkatalog von QS-MitarbeiterInnen erstellt, die schon Erfahrung in der Testautomatisierung in einem dem Projekt ähnlichen Kontext haben.

2. Eine andere Möglichkeit ist, den individuellen Kriterienkatalog mithilfe von „Fremdexpertise“ zu erstellen. In diesem Fall wird ein detaillierter Standardkriterienkatalog als Basis genommen. Aus diesem Standardkriterienkatalog können die für das Projekt zutreffenden Kriterien identifiziert und in den individuellen Kriterienkatalog übernommen werden.

In dieser Bachelorarbeit wird die Kombination aus den beiden Möglichkeiten (Expertenwissen und Standardkriterienkatalog) empfohlen:

Organisationseigenes Expertenwissen. Das erste Element bei der Ableitung der Kriterien ist die unternehmenseigene Expertise. Dabei werden vom/von der ProjektleiterIn und/oder von erfahrenen QS-MitarbeiterInnen aus den identifizierten Anforderungen die Kriterien abgeleitet und in den individuellen Kriterienkatalog erfasst. Erfahrung und Wissen sind dabei nötig, um die Kriterien projekt- und kontextbezogen genau zu definieren.

Standardkriterienkatalog. Als zweites Element wird der „Kriterienkatalog zur Testwerkzeugauswahl“ der Autoren Baumgartner et al. als Standardkriterienkatalog genommen, der sich im Anhang des Buches der Autoren befindet [2]. Das Buch ist entlang der Struktur des Lehrplans „ISTQB® Certified Tester Advanced Level Testautomatisierungsentwickler“ entstanden. Die Autoren erläutern in ihrem Kriterienkatalog zahlreiche Kriterien, die bei der Auswahl eines Testautomatisierungstools in Betracht gezogen werden können. Mit dem Kriterienkatalog von Baumgartner et al. wird sichergestellt, dass die wichtigsten Kriterien bewusst bearbeitet werden. Unter *bewusst bearbeitet* ist zu verstehen, dass die Kriterien ganz bewusst aufgenommen oder ganz bewusst abgewählt werden, nicht aber übersehen oder vergessen werden.

Der „Kriterienkatalog zur Testwerkzeugauswahl“ ist in den Werken der Autoren Baumgartner et al. und Bucsics et al., wie in dieser Arbeit auch, im Anhang zu finden [2] [5].

Die Kriterien sind einem einheitlichen Schema folgend aufgebaut. Sie sind im Kriterienkatalog in der Spalte *Kriterium* aufgelistet. Zu jedem Kriterium gehört eine *Frage* und oft auch eine *Erläuterung* in den gleichnamigen Spalten, die deutlich machen sollen, worauf sich das Kriterium bezieht. Ein Beispiel dieses Schemas ist in Abbildung 6 zu sehen. In dieser Arbeit wird auch der individuelle, auf das bereits vorgestellte Projekt bezogene Kriterienkatalog nach diesem Schema aufgebaut und im Verlauf der Methodik mit zusätzlichen Spalten erweitert.

Kriterium	Frage	Erläuterung
Allgemeine Kriterien		
Support/Wartung		
Qualität der Supportplattform	Gibt es Onlineformulare, Foren, Knowledge Bases? Wie lange ist die Reaktionszeit bei schweren/ leichten Problemen?	Knowledge Bases und Userforen bieten oft rasche Lösungen bei einfacheren Problemen. Bei schwereren ist dann guter Support gefragt.
Updatefrequenz und Aufwand	Wie oft kommen Updates heraus? Wie aufwendig ist es, auf eine neue Version umzusteigen?	Zu häufige Updates erzeugen Overhead oder werden ignoriert, zu seltene Updates bedeuten oft persistente Probleme.
Consultingteam/ Support/ Community	Wie viele Mitarbeiter unterstützen den Kunden? Vor Ort?	Viele größere Hersteller haben keine regionalen Niederlassungen, während lokale Hersteller ihr gesamtes Team in greifbarer Nähe haben.

Abbildung 6: Auszug aus dem „Kriterienkatalog zur Testwerkzeugauswahl“ (Quelle: Baumgartner et al. [2])

Im Standardkriterienkatalog sind die Kriterien in drei Bereiche unterteilt:

- Allgemeine Kriterien
- Kriterien Testautomatisierungswerkzeuge
- Finanzielle Kriterien

Die Gruppierung von Kriterien in Bereiche ermöglicht eine schnelle Orientierung bei der Erstellung des individuellen Kriterienkataloges. Eine Änderung oder Erweiterung der Gruppierung für den individuellen Kriterienkatalog ist je nach Bedarf möglich.

3.2.4 Bewertung der Kriterien

Die Kriterien aus dem individuellen Kriterienkatalog werden für jedes zu bewertende Testautomatisierungstool ausgewertet. Nicht jedes Testautomatisierungstool wird ein

Kriterium zum gleichen Grad erfüllen. Diese unterschiedliche Erfüllung von Kriterien ist die Grundlage für das Endergebnis des Auswahlprozesses. Deshalb wird zur Bewertung von identifizierten Kriterien in dieser Arbeit der **Kriteriumerfüllungsgrad** definiert. Der Kriteriumerfüllungsgrad macht eine Aussage darüber, wie weit ein Testautomatisierungstool ein Kriterium, das vom Projekt an das Testautomatisierungstool gestellt wird, erfüllt. Den Kriteriumerfüllungsgraden werden Punktzahlen zugeordnet, die für die Auswertung der Auswahlmethodik nötig sind. Diese werden im Abschnitt 3.4 erörtert.

Der Kriteriumerfüllungsgrad kann mit Modellen, die verschiedene Stufenzahlen haben, abgebildet werden:

- 2-stufiges Kriteriumerfüllungsgrad-Modell -> das Kriterium wurde erfüllt oder nicht erfüllt,
- 3-stufiges Kriteriumerfüllungsgrad-Modell -> das Kriterium wurde erfüllt, nicht erfüllt oder teilweise erfüllt,
- 4-stufiges Kriteriumerfüllungsgrad-Modell -> das Kriterium wurde erfüllt, fast erfüllt, kaum erfüllt oder nicht erfüllt,
- x-stufiges Kriteriumerfüllungsgrad-Modell -> es gibt beliebig weitere Möglichkeiten, nach obigem Schema fortsetzend.

Die Formulierungen von zu den Kriterien gehörenden Fragen bzw. zu den Kriterien gehörenden Erläuterungen müssen den Kriteriumerfüllungsgrad-Modellansatz unterstützen. Deshalb sollten die Fragen und/oder die Erläuterungen genau quantifiziert, d. h. mit Grenzwerten versehen werden. Damit wäre eine transparente und messbare Auswertung ermöglicht, um eine objektive Antwort auf die allgemeine Frage „Erfüllt das zu bewertende Testautomatisierungstool das betrachtete Kriterium?“ zu bekommen.

Beispiel: Aus der Sicht eines Projekts wurden die Lizenzkosten des Testautomatisierungstools als relevantes Kriterium identifiziert. In der Abbildung 7 ist ein Abschnitt des Kataloges „Kriterienkatalog zur Testwerkzeugauswahl“ der Autoren Baumgartner et al. mit dem Kriterium „Lizenztypen“ zu sehen [2]. In der Spalte *Frage* steht „Gibt es Floating-Lizenzen? Per-Seat-Lizenzen? Preise?“. Eine Erläuterung ist nicht vorhanden. Hier wird das Kriterium aus dem Aspekt der Lizenzkosten, d. h. aus dem Aspekt der Teilfrage „Preise?“ detaillierter analysiert.

Wie sollte eine Bewertung für dieses Kriterium erfolgen, wenn drei Testautomatisierungstools zu vergleichen wären: ein Open-Source-Tool mit keinen, ein Tool mit 1.000 EUR und ein Tool mit 2.500 EUR Lizenzkosten?

Kriterium	Frage	Erläuterung
Finanzielle Kriterien		
Kosten und Lizenz (Fortsetzung)		
Lizenztypen	Gibt es Floating-Lizenzen? Per-Seat-Lizenzen? Preise?	

Abbildung 7: Auszug aus dem „Kriterienkatalog zur Testwerkzeugauswahl“ – Abschnitt mit Lizenztypen (Quelle: Baumgartner et al. [2])

Um eine Bewertung für diesen Fall zu ermöglichen, wäre es nötig, eine genau formulierte Erläuterung zuzufügen, wie z. B. „Die maximalen Lizenzkosten sollten 2.000 EUR betragen“. Diese Formulierung würde dann eine eindeutige und vor allem objektive Tool-Bewertung ermöglichen: Die ersten zwei Testautomatisierungstools *erfüllen* und das dritte Testautomatisierungstool *erfüllt nicht* das Kriterium. Damit wäre ein 2-stufiges-Modell mit 2.000 EUR Grenzwert dargestellt.

Ein 3-stufiges-Modell bräuchte zwei Grenzwerte und könnte wie folgt aussehen: „Angestrebt sind möglichst minimale Lizenzkosten und sie dürfen 2.000 EUR nicht übersteigen“. Das Open-Source-Testautomatisierungstool hat das Kriterium *vollständig*, das Testautomatisierungstool für 1.000 EUR *teilweise* und das Testautomatisierungstool für 2.500 EUR *nicht erfüllt*.

Die Anzahl der Stufen von Kriteriumerfüllungsgrad-Modellen ist als eine relevante Kennzahl erkannt. Es stellt sich die Frage, welche Stufenzahl bei dem Kriteriumerfüllungsgrad-Modell zur Auswahl des Testautomatisierungstools verwendet werden sollte.

Die Autoren Döring et al. haben neben der Analyse auch eine quantitative Empfehlung ausgesprochen [12]. Für die Sozial- und Humanwissenschaften wurde eine Stufenzahl zwischen fünf und sieben empfohlen. Die Skala mit drei Stufen wurde als eine Skala mit sehr wenigen Stufen eingeordnet. In den Analysen wurde unter anderem darauf hingewiesen, dass eine optimale Stufenzahl von Faktoren wie Datenerhebungsmethodik, Differenzierungsfähigkeit der Befragungspersonen als auch von der Menge und Qualität des vorhandenen Wissens und/oder der Informationen abhängig ist [12].

In der Entscheidungstheorie wurden die obige Frage und viele weiterführende Fragen ebenfalls analysiert. Laut der Autoren Laux et al. sollte die Anzahl von Stufen, d. h. die mögliche Anzahl von Antworten auf ein Kriterium, begrenzt sein [30]. Sie nehmen in ihrem Werk über Entscheidungstheorie aber keine konkrete Quantifizierung vor. Es wird

von den Autoren keine Stufenanzahl favorisiert. Nach ihrer Theorie sollte die Stufenzahl vielmehr aufgrund der jeweiligen Problemstellung gewählt werden [30].

Aufgrund obiger Ergebnisse der Forschung wird in dieser Arbeit für die Auswahlmethodik ein 3-stufiges Modell und für die Kriterien, bei welchen eine Entscheidungsfrage zu beantworten ist, ein 2-stufiges Modell empfohlen.

Aufgrund folgender Überlegungen und obiger Forschungsergebnisse wird auch empfohlen, beim Kriteriumerfüllungsgrad-Modell die fünf Stufen nicht zu übersteigen:

1. In der Praxis sind viele Kriterien mit einem *ja* oder *nein* zu beantworten bzw. zu „messen“.

2. Bei vielen Kriterien aus dem Kriterienkatalog ist ein feiner Unterschied, der aber bei einem höherstufigen Modell nötig wäre, kaum formulierbar.

3. Ein höherstufiges Modell bräuchte ein sehr detailliertes Testautomatisierungstool-Know-how, um objektiv angewendet werden zu können. Dieses nötige, detaillierte Know-how über die zahlreichen am Markt vorhandenen Testautomatisierungstools ist in den meisten Fällen in einem Unternehmen nicht vorhanden.

4. Ein höherstufiges Modell könnte die MitarbeiterInnen schnell überfordern, die Bearbeitung eines solchen Auswahlprozesses würde eine nicht zumutbare, kostenintensive Zeit in Anspruch nehmen.

Beim 3-stufigen Modell werden dem Kriteriumerfüllungsgrad die Punktzahlen wie folgt zugeordnet:

- nicht erfüllt = 0 Punkte
- teilweise erfüllt = 1 Punkt
- vollständig erfüllt = 2 Punkte

Beim 3-stufigen Modell sollen die Bewertungen genau quantifiziert bzw. formuliert werden.

Bei den Kriterien, bei welchen eine Entscheidungsfrage zu beantworten ist (2-stufiges Modell), gibt es keine *teilweise erfüllt* (1 Punkt) Bewertung:

- Nein (Kriterium nicht erfüllt) = 0 Punkte
- Ja (Kriterium voll erfüllt) = 2 Punkte

Hierbei werden bei einer *Ja*-Antwort dem Kriteriumerfüllungsgrad auch 2 Punkte zugeordnet, um die Kriterien bei *vollständig erfüllt* gleich zu bewerten wie beim 3-stufigen Modell.

Bei den Entscheidungsfragen sollen die Fragen so formuliert werden, dass die 0 Punkte dem schlechten Ergebnis zugeordnet werden.

Die bei einem Kriterium endgültig vergebenen Punktzahlen werden als **Kriteriumerfüllungsgradpunktzahl (KEGP)** definiert. Sie sind für die Auswertung der Auswahlmethodik nötig, die im Unterabschnitt 3.4 erörtert wird.

K-Nr.	Kriterium	Frage	Erläuterung	Bewertung
K-01	Recording	Gibt es eine Recordingmöglichkeit?	Die Möglichkeit, den Testfall aufzuzeichnen und zu speichern, um später die Aufzeichnung abspielen zu können und/oder die entstandenen Skripte weiterentwickeln zu können.	0 Punkte => Nein 2 Punkte => Ja
K-02	Personalressourcen	Gibt es mindestens drei MitarbeiterInnen, die mit dem TaT selbständig arbeiten können?	Es gibt MitarbeiterInnen innerhalb des Unternehmens und auch im Projekt selbst mit TaT Erfahrungen. Dieses interne Know-how sollte berücksichtigt werden.	0 Punkte => Nein 1 Punkt => es gibt weniger als drei MitarbeiterInnen 2 Punkte => es gibt mindestens drei MitarbeiterInnen

Tabelle 1: Beispiele für die Bewertung mit 2- bzw. 3-stufigem Modell (Quelle: eigene Darstellung, angelehnt an [2])

Die Tabelle 1 stellt beispielhaft ein Kriterium mit einer 2-stufigen Modell-Bewertung (K-01) und ein Kriterium mit einer 3-stufigen Modell-Bewertung (K-02) dar.

In folgendem Abschnitt wird die Gewichtung der Kriterien erklärt.

3.2.5 Gewichtung der Kriterien

Der individuelle Kriterienkatalog ist noch nicht einsatzbereit. Der Kriterienkatalog beinhaltet die Kriterien, die für das Projekt wichtige Anforderungen beschreiben. Jedoch sind diese Kriterien für das Projekt nicht gleich wichtig. Diese Ungleichheit der Kriterien wird durch deren **Gewichtung** berücksichtigt. Die Gewichtung ist ein Faktor, der in Prozenten

vergeben wird. Die Summe aller Gewichtungen soll 100 % ergeben. Mit der Gewichtung wird die Punktzahl des Kriteriumerfüllungsgrads des jeweiligen Kriteriums multipliziert. Damit bekommen die wichtigen Kriterien größere Anteile für das Endergebnis als die weniger wichtigen Kriterien.

Ein Beispiel soll den Sachverhalt verdeutlichen:

Beispiel: Kriterium A ist wichtiger als Kriterium B.

Das Kriterium A wurde vollständig erfüllt. Das Kriterium wird mit 2 Punkten bewertet. Angenommen das Kriterium hat eine Gewichtung von 10 %, dann wird dieses Kriterium 20 % der Gesamtentscheidung beisteuern, weil 10 % * 2 = 20 %.

Das Kriterium B wurde auch vollständig erfüllt. Das Kriterium wird mit 2 Punkten bewertet. Angenommen das Kriterium hat eine Gewichtung von 8 %, dann wird dieses Kriterium 16 % der Gesamtentscheidung beisteuern, weil 8 % * 2 = 16 %.

Mit diesem Beispiel wurde dargestellt, dass obwohl beide Kriterien die höchste Kriteriumerfüllungsgradpunktzahl bekommen haben, aufgrund der Gewichtung das wichtigere Kriterium A mit 20 % aber einen größeren Anteil zur Gesamtentscheidung beisteuert als das Kriterium B mit 16 %. Diese berechneten Werte werden bei der Auswertung auf 100% normiert (siehe Abschnitt 3.4).

Die vergebenen Gewichtungsprozentzahlen können sehr variieren, abhängig davon, wie viele Kriterien insgesamt gewichtet werden sollen. Das folgende Beispiel stellt ein mögliches Vorgehen bei der Gewichtungsvergabe dar:

Beispiel: Es sind 30 Kriterien, die gewichtet werden sollen.

Es werden fünf Wichtigkeitsgruppen in zwei Schritten gebildet.

1. Schritt: Zuerst bekommen die Kriterien je nach Wichtigkeit die Gewichtungen mit den Werten von 2 %, 4 % und 6 %.

2. Schritt: Es folgt eine weitere, feinere Aufteilung der Kriterien:

a. Die am wenigsten wichtigen Kriterien mit 2 % werden nochmals kritisch betrachtet. Dabei bekommen manche Kriterien, die doch etwas wichtiger sind, eine Gewichtung von 3 %.

b. Dasselbe erfolgt mit den Kriterien, die ursprünglich 4 % bekommen haben. Dabei bekommen manche Kriterien, die doch etwas wichtiger sind, eine Gewichtung von 5 %.

c. Die wichtigsten Kriterien behalten die Gewichtung von 6 %.

Bei der Vergabe der Gewichtungsprozentzahlen wird darauf geachtet, dass am Schluss der Gewichtungsvergabe unter dem Strich die Summe alle Gewichtungen 100 % ergibt. So kann am Ende des Auswahlprozesses, nach einer Normierung auf 100 %, die Aussage getroffen werden, mit wie viel Prozent ein Testautomatisierungstool alle Kriterien erfüllt. Dies wird im Abschnitt 3.4 detailliert erklärt.

Wenn in einem Auswahlprozess weniger oder mehr Kriterien gewichtet werden sollen, dann können die Gewichtungsprozentzahlen dementsprechend angepasst werden (z. B. von 6 % bis 10 % bei weniger Kriterien oder von 1 % bis 5 % bei mehr Kriterien).

Die Verteilung der Gewichtungen beeinflusst das Endergebnis sehr stark. Dementsprechend sollte die Bestimmung der Werte sehr sorgfältig unter Einbeziehung aller projektrelevanten Informationen stattfinden. Darüber hinaus sollten die Gewichtungen vom/von der ProjektleiterIn und/oder von erfahrenen QS-MitarbeiterInnen bestimmt werden.

Am Ende sollen die Kriterien und deren Gewichtungen die projektspezifischen Anforderungen an das Testautomatisierungstool strukturiert widerspiegeln. Der so entstandene gewichtete, individuelle Kriterienkatalog ist eine Art Fußabdruck des Projekts. Die Tabelle 2 stellt das Schema des gewichteten, individuellen Kriterienkataloges dar.

K-Nr.	Kriterium	Gewichtung	Frage	Erläuterung
K-25	Wartbarkeit	4 %	Ist ein Protokoll vorhanden?	Im Protokoll kann der Fehler schnell dargestellt werden.
K-27	Versionsverwaltungssystem (VCS)	6 %	Sind die TaT-Projekt-Artefakte mit *git*[5] verwaltbar?	
K-29	Schnittstelle zu Jira[6]	2 %	Hat das TaT eine Jira-Schnittstelle?	Zurzeit wird in VS3 das Jira-Ticketing-System für die Testfälle/Bugs/…/Testdokumentation benutzt.
…	…	…	…	…
K-j	…	…	…	…

Tabelle 2: Schema des individuellen Kriterienkataloges mit Gewichtungen (Quelle: eigene Darstellung, angelehnt an [2])

[5] **git** ist ein Open Source verteiltes Versionsverwaltungssystem [18].

[6] **Jira** Software ist eine Webanwendung zur Fehlerverwaltung, Problembehandlung und zum operativen Projektmanagement, die von Atlassian entwickelt wird [1].

Ausschlusskriterien: Es gibt eine sehr spezifische, noch zu beschreibende Situation. Zwischen den Kriterien sind einige, die sich als absolut notwendig herauskristallisieren. Sie werden als Ausschlusskriterien definiert. Diese Kriterien müssen vollständig erfüllt werden, sie stehen nicht zur Diskussion. Ein Ausschlusskriterium ist im obigen Kriterienkatalog nicht abbildbar. Die Gewichtung sollte allein bei diesem Kriterium 100 % sein, und damit müssten alle anderen Kriterien mit 0 % gewichtet werden. Ein Beispiel für das Ausschlusskriterium wären die Lizenzkosten mit folgender Erläuterung: „Es dürfen keine Lizenzkosten entstehen."

Als Lösung werden die Ausschlusskriterien aus dem individuellen Kriterienkatalog entfernt und in eine Ausschlusskriterium-Liste eingetragen. Im Katalog sollen nur die Kriterien mit einer Gewichtung kleiner als 100 % bleiben.

Die Ausschlusskriterien sind beim Vorselektieren der Testautomatisierungstools sehr hilfreich. Basierend auf diesen Kriterien ist es möglich, von den zahlreichen am Markt befindlichen Testautomatisierungstools einige auszuschließen. Damit wird die Anzahl der Testautomatisierungstools reduziert, die detailliert zu bearbeiten sind. Im Kapitel 3.3 wird dieses Vorgehen in der ganzen Methodik genauer erläutert.

3.3 Phase 3: Evaluierung

In der Evaluierungsphase werden die zu betrachtenden Testautomatisierungstools entlang des individuellen Kriterienkataloges analysiert und bewertet. In dieser Phase sind zwei gut abgrenzbare Tätigkeiten definierbar: (1) selektieren der zu betrachtenden TaTs und anschließend (2) die selektierten Tools mit verschiedenen Methoden evaluieren und bewerten.

3.3.1 Selektieren

Es gibt auf dem Markt zahlreiche Testautomatisierungstools. Auf der Webseite von *International Testing Board* sind aktuell 28 Testautomatisierungstools aufgelistet [58]. Die Liste ist mit großer Wahrscheinlichkeit nicht vollständig. Dieser Tatsache folgend ergibt sich die Frage:

Welche Testautomatisierungstools sollten in der Evaluierungsphase betrachtet werden?

Diese Frage selbst generiert zwei weitere Teilfragen:

1. Welche Listen mit Testautomatisierungstools sind vorhanden?
2. Wie ist die große Anzahl von Testautomatisierungstools zu selektieren?

Um die aktuell vorhandenen Listen zu finden, muss eine ausführliche Recherche durchgeführt werden. Im Internet sind zahlreiche Webseiten mit Testautomatisierungstool-Listen zu finden, eine davon ist das obige Beispiel [58]. Es sind z. B. Listen zu finden, die die meistbenutzten Testautomatisierungstools enthalten. Auf der anderen Seite sind Listen vorhanden, die die Testautomatisierungstools von Sponsoren enthalten. Die Überlegung ist, die für die Evaluierungsphase notwendige Testautomatisierungstool-Liste aus mehreren Quellen zu erstellen. Dabei ist es möglich, solche Testautomatisierungstools aufzulisten, die auf mehreren Listen dieser Quellen zu finden sind. Die Anzahl der Testautomatisierungstools kann individuell bestimmt werden.

Die aus der Sicht des Projekts oder des Unternehmens erwünschte Testautomatisierungstools, die in Betracht gezogen werden sollen, werden auch in die Liste aufgenommen.

Die entstandene Testautomatisierungstool-Liste wird mit den Ausschlusskriterien weiter selektiert. Die Ausschlusskriterien wurden im Abschnitt 3.2.5 beschrieben. Die TaTs, die die Ausschlusskriterien nicht erfüllen, werden aus der Testautomatisierungstool-Liste entfernt. Als Endergebnis entsteht eine **selektierte Testautomatisierungstool-Liste**.

3.3.2 Evaluieren und Bewerten

Die Testautomatisierungstools aus der selektierten Testautomatisierungstool-Liste werden an jedem Kriterium aus dem individuellen Kriterienkatalog mit Gewichtungen gemessen. Es gibt mehrere Methoden, die in dieser Phase dienlich sein können:

1. Die Demoversion des Testautomatisierungstools installieren und testen.
2. Die Testautomatisierungstools innerhalb eines Workshops vom Hersteller an praxisnahen Beispielen kennenlernen.
3. Im Internet nach Bewertungen von ExpertInnen suchen.
4. Video-Tutorials anschauen.
5. Die Dokumentationen des Testautomatisierungstools analysieren.
6. Relevante Informationen aus der Community sammeln.

Baumgartner et al. empfehlen, die zu evaluierenden Testautomatisierungstools zu downloaden, zu installieren und zu testen. Die im Buch empfohlenen Leitfäden werden an dieser Stelle stichpunktartig dargestellt, um eine Orientierung und Hilfestellung bei der Evaluierung zu geben [2]:

1. Vorbereitung der Szenarien
    - Start eines Browsers
    - Öffnen des SUT in der Testumgebung
    - Login mit einem Test-User
    - Anlegen eines Datenelements (z. B. Kunde)
    - Überprüfung der erfolgreichen Anlage
    - Löschen des Datenelements
    - Überprüfung des erfolgreichen Löschens
    - Logout
    - Schließen des Browsers
2. Installation des Werkzeugs und Erstellung des Arbeitsbereiches
3. Erstellen des ersten Szenarios
4. Refactoring und Parametrierung des ersten Szenarios
5. Umsetzung weiterer Szenarien
6. Integration in den Entwicklungsprozess

Die in der Bachelorarbeit konzipierte Methodik ist auf der Evaluierung des Testautomatisierungstools auch mit geringer Erfahrung im Bereich der Testautomatisierung ausgelegt. Die Grundlage dafür ist der individuelle Kriterienkatalog mit Gewichtungen, der vom/von der ProjektleiterIn und/oder QS-MitarbeiterInnen, die im Bereich der Testautomatisierung schon Kenntnisse besitzen, erstellt wird. Beim Evaluieren sind nun die genau formulierten Fragen und Erläuterungen zu beantworten. Dabei kann gezielt mit einer Recherche im Internet nach Antworten gesucht werden. Es wäre möglich, mit Handbüchern, Dokumentationen, Blogs und Tutorials zu einem Ergebnis zu kommen, wenn ausreichend Quellen gefunden werden können. Der weitere Weg des Evaluierens wäre, die kostenlose Demoversion zu installieren und zu testen.

Der individuelle Kriterienkatalog mit Gewichtungen soll eine objektive und ressourcenschonende Testautomatisierungstool-Evaluierung ermöglichen.

Die Erkenntnisse der Evaluierung eines Testautomatisierungstools sind die Grundlagen der Bewertung jedes Kriteriums. Die Bewertungen sollen in Form von Kriteriumerfüllungsgradpunktzahlen, die im Unterabschnitt 3.2.4 erklärt werden, für sämtliche Kriterien festgehalten werden. Das geschieht mit der Hilfe der Entscheidungsmatrix. Die Entscheidungsmatrix ist eine für die Auswahlmethodik konzipierte Tabelle, die im nächsten Unterabschnitt vorgestellt wird.

3.3.3 Die Entscheidungsmatrix

Die Entscheidungsmatrix wird aufgrund des gewichteten, individuellen Kriterienkataloges konstruiert. Ein beispielhaftes Schema der Entscheidungsmatrix stellt die Tabelle 3 dar.

K-Nr.	Kriterium	Gewichtung	Tool-1	Tool-2	Tool-3	…	Tool-i
K-01	Lizenzkosten für den/die Testentwicklerln	10 %	2	2	1	…	x
K-02	Übertragbarkeit von Entwicklerlizenzen	10 %	1	0	0	…	y
K-03	Runtime-Umgebung	8 %	2	0	1	…	z
…	…	**…**	…	…	…	…	…
K-j	…	**…**	…	…	…	…	…
Ergebnis			**34,20 %**	**45,34 %**	**28,87 %**	**…**	**…**

Tabelle 3: Schema der Entscheidungsmatrix (Quelle: eigene Darstellung, angelehnt an [2])

Die Kriterien sind in der Entscheidungsmatrix in der Spalte *Kriterium* aufgelistet. Zu jedem Kriterium gehört eine *Gewichtung* in Prozent in der gleichnamigen Spalte. Die Tabelle wird mit weiteren Spalten je zu evaluierendem Testautomatisierungstool erweitert.

Die Kriteriumerfüllungsgradpunktzahlen werden in der Spalte des gerade evaluierten Testautomatisierungstools und in der Zeile des gerade zu bewertenden Kriteriums eingetragen.

3.3.4 Anzahl von Kriterien

Es stellt sich die Frage, ob eine optimale Anzahl von Kriterien für den individuellen Kriterienkatalog existiert. Hierfür hat sich bei der Literaturrecherche keine eindeutige Antwort ergeben. Im Weiteren werden einige Überlegungen beschrieben.

Eine Begrenzung der Anzahl der zu bewertenden Kriterien wurde in der Entscheidungstheorie analysiert. Es gibt keine eindeutige Quantifizierung, es wird kein exakter Wert empfohlen. Die Anzahl der abgeleiteten und in den individuellen Kriterienkatalog aufgenommenen Kriterien soll situationsabhängig betrachtet werden. Es könnte eine bestimmte Anzahl an Kriterien des individuellen Kriterienkataloges für jedes Projekt geben, die zu einer erfolgreichen Toolauswahl beitragen könnten [30].

Zu wenige Kriterien könnten zu einer sehr vereinfachten Abbildung der Anforderungen, die vom Projekt an das Testautomatisierungstool gestellt werden, führen. Damit könnte keine optimale Auswahl erreicht werden. Zu viele Kriterien könnten das eine oder andere wichtige Kriterium unterdrücken, d. h. die „kleinen“ Kriterien könnten die „großen“ Kriterien übertönen. Bei zu vielen Kriterien wären einige Kriterien mit sehr niedriger Gewichtung bewertet. Mit der Bearbeitung von zu vielen Kriterien würde durchaus einiges an Ressourcen, Zeit und Geld unnötig verbraucht werden.

3.4 Phase 4: Auswertung und Auswahl

Bei der Auswertung der Entscheidungsmatrix werden die Ergebnisse pro Testautomatisierungstool berechnet und in die unterste Zeile *Ergebnis* der Entscheidungsmatrix pro Testautomatisierungstool eingetragen.

1. Schritt: Zuerst werden die Produkte mit Faktoren *Gewichtung* und *Kriteriumerfüllungsgradpunktzahl (KEGP)* pro Kriterium berechnet.

2. Schritt: In diesem Schritt wird die Summe von allen Produkten aus dem 1. Schritt berechnet. Anschließend wird das Ergebnis mit der maximalen Kriteriumerfüllungsgradpunktzahl (max KEGP) auf 100 % normiert**.** Bei dem für diese Arbeit gewählten 3-stufigen Modell ist die maximale Kriteriumerfüllungsgradpunktzahl die 2.

Die Berechnungen aus den beiden Schritten können mit der folgenden Formel zusammengefasst werden (Quelle: eigene Formel):

$$Ergebnis(Tool_i) = \frac{1}{max\ KEGP} \sum_{j=1}^{n} \left(Gewichtung_j * KEGP_{(Tool_i, Kriteriumj)} \right)$$

Mit der Normierung soll ein besseres Verständnis des Ergebnisses erreicht werden: Durch Dividieren mit der maximalen Kriteriumerfüllungsgradpunktzahl in der obigen Formel wird das Ergebnis genau angeben, zu welchem Grad (in %) das jeweilige Testautomatisierungstool die Kriterien erfüllt.

Die Auswertung bzw. die erreichte Rangordnung ist für jedes Testautomatisierungstool aus der Zeile *Ergebnis* ablesbar (Tabelle 3). Das Testautomatisierungstool mit der größten Prozentzahl erfüllt die aufgestellten Kriterien und damit auch die vom Projekt gestellten Anforderungen am meisten. Damit liegt eine Empfehlung für das Testautomatisierungstool, das für das Projekt am besten geeignet ist, vor.

Es könnte vorkommen, dass zwei oder mehrere Testautomatisierungstools als Ergebnis einen identischen Prozentwert erreichen. In dem Fall sind die Bewertungen der Kriterien noch einmal zu revidieren. Das Testautomatisierungstool, das mehr wichtige Kriterien erfüllt, kann bevorzugt empfohlen werden.

4 Auswahl eines Testautomatisierungstools für das VS3-Web-Projekt

In diesem Kapitel wird die in Kapitel 3 beschriebene Methodik an einem praxisnahen Beispiel dargestellt. Hierbei handelt es sich um das VS3-Web-Anwendungsprojekt, das in der Einleitung dieser Arbeit beschrieben wurde. Die mit der Methodik festgelegten Phasen bzw. Schritte werden angewendet. Abgeschlossen wird dieses Kapitel mit einer Testautomatisierungstool-Empfehlung.

4.1 Phase 1: Grundsatzentscheidung

Die Grundsatzentscheidung ist die Initialphase des Auswahlprozesses. Hierbei geht es nach Baumgartner et al. um die „grundsätzliche Entscheidung über den Einsatz eines Werkzeugs“ [2]. Beim VS3-Web-Anwendungsprojekt geht es um ein komplexes Softwareprodukt in der Immobilienverwaltung, das viele Module beinhaltet und über drei Jahre entwickelt wird. Das Projekt wird als agiles Software-Entwicklungsprojekt durchgeführt. Dabei werden die Kundenwünsche stets beachtet. Deshalb werden in kurzen Intervallen lauffähige Versionen erstellt. Bei der Implementierung eines neuen Features spielen die Regressionstests eine zentrale Rolle. Dabei wird getestet, ob die schon implementierten Features bzw. Module weiterhin fehlerfrei funktionieren. Wegen der Komplexität des Softwareprodukts würde das manuelle Durchführen der Regressionstests sehr viel Zeit in Anspruch nehmen. Deshalb wurde der Einsatz eines Testautomatisierungstools im VS3-Web-Anwendungsprojekt beschlossen.

4.2 Phase 2: Anforderungen und Kriterien

In der zweiten Phase des Auswahlprozesses sollen zuerst die Anforderungen identifiziert werden, die vom VS3-Web-Anwendungsprojekt an das zukünftige Testautomatisierungstool gestellt werden. Dieser Schritt wird im nächsten Unterabschnitt beschrieben.

4.2.1 Identifikation der Anforderungen

Das VS3-Web-Anwendungsprojekt wurde analysiert, die Anforderungen, die vom Projekt an das zukünftige Testautomatisierungstool gestellt werden, identifiziert und in die Liste der identifizierten Anforderungen eingetragen (Tabelle 4). Als erste Quelle für die Analyse diente die VS3-Projektdokumentation. Die aus der Projektdokumentation identifizierten Anforderungen wurden mit dem Projektleiter besprochen und ergänzt. Am Ende ist die folgende Liste der identifizierten Anforderungen entstanden:

A-Nr.	Anforderung
A-01	Auf dem Markt etabliertes Testautomatisierungstool
A-02	Erlernbarkeit soll schnell und einfach sein
A-03	Es soll ein Record/Playback-Testautomatisierungstool sein
A-04	Stabilität und Zuverlässigkeit der Testsuite[7]
A-05	Übersichtlichkeit der Testsuite
A-06	Nachvollziehbarkeit der Testläufe
A-07	Nachvollziehbarkeit der Testergebnisse
A-08	Erweiterbarkeit, Änderbarkeit, Wartbarkeit und Usability der Testsuite – Möglichkeit, Elemente bzw. Zeilen mit unterschiedlichen Funktionalitäten in den Testfall einzufügen, zu verschieben, zu löschen und wiederherzustellen
A-09	Als Betriebssystem (OS) des Rechners ist *Microsoft Windows* vorgesehen
A-10	Als Versionsverwaltungssystem (VCS) ist *git* vorgesehen
A-11	Als CI/CD-Pipeline[8] ist *Jenkins*[9] vorgesehen
A-12	In iSYS wird *Jira* verwendet. Die direkte Kommunikation zwischen Testautomatisierungstool und Jira ist erwünscht (automatisch Ticket erstellen)
A-13	Für die verschiedenen Testfälle werden verschiedene Daten benötigt. Das Testautomatisierungstool soll das *datengetriebene Testing* unterstützen
A-14	PDF-Dokumente nach Layout und Inhalt prüfen
A-15	Lizenzkosten sollen im Rahmen sein
A-16	Die Möglichkeit, die Testfälle in einer Runtime-Umgebung auszuführen, um die Entwicklerlizenzen optimiert nutzen zu können, soll vorhanden sein
A-17	Die VS3-Anwendung ist eine Webanwendung

Tabelle 4: Die identifizierten Anforderungen (Quelle: eigene Darstellung)

[7] **Testsuite** "Die Zusammenstellung (Aggregation) mehrerer Testfälle für den Test einer Komponente oder eines Systems, bei der Nachbedingungen des einen Tests als Vor-bedingungen des folgenden Tests genutzt werden können" [22].

[8] Eine **CI/CD-Pipeline** umfasst mehrere Schritte, die zur Bereitstellung einer neuen Softwareversion ausgeführt werden müssen [25]. CI/CD steht für Continuous Integration/Continuous Delivery. Continuous Integration bezieht sich in der Regel auf das Integrieren, Erstellen und Testen von Code innerhalb der Entwicklungsumgebung. Continuous Delivery baut darauf auf und befasst sich mit den letzten Schritten, die für die Produktionsbereitstellung erforderlich sind [14].

[9] **Jenkins** ist ein webbasiertes Open Source Continuous Integration (CI) System [26].

Aus diesen Anforderungen wurden die Kriterien abgeleitet, die im nächsten Unterabschnitt dargestellt werden.

4.2.2 Ableitung der Kriterien

Die weiterführenden detaillierten Analysen der identifizierten Anforderungen führten zur Ableitung der Kriterien. Der „Kriterienkatalog zur Testwerkzeugauswahl“ von Baumgartner et al. (siehe Anhang) [2] wurde als Hilfestellung, wie in der Methodik in Unterabschnitt 3.2.3 beschrieben, angewendet. Die abgeleiteten Kriterien wurden mit dem Projektleiter besprochen und ergänzt. Die Tabelle 5 visualisiert, aus welcher Anforderung welches Kriterium abgeleitet wurde. Die zu den Kriterien gehörenden Fragen, Bewertungen und ggf. Erläuterungen sind in Unterabschnitt 4.2.3 formuliert. In der Spalte Anforderungsnummer (A-Nr.) ist die Nummer der jeweiligen Anforderung aus der Tabelle der identifizierten Anforderungen (Tabelle 4) eingetragen, aus welcher das Kriterium abgeleitet wurde. In der Spalte Kriterium-Nummer (K-Nr.) wurden die Nummern der abgeleiteten Kriterien eingetragen.

A-Nr.	K-Nr.	Kriterium
A-01	K-01	Testversion
A-01	K-02	Evaluierung im Einsatzumfeld
A-01	K-03	Marktpräsenz
A-01	K-04	Community
A-01	K-05	Support
A-01	K-06	Update
A-02	K-07	Dokumentation
A-02	K-08	Videotutorials
A-02	K-09	Personalressourcen
A-03	K-10	Record/Playback-Funktion
A-04	K-11	Debuggen
A-04	K-12	Komponentenerkennung
A-04	K-13	Exception Handling
A-04	K-14	Warten auf Events
A-05	K-15	Testsuitestruktur
A-06	K-16	Log-Datei
A-07	K-17	Reporting
A-08	K-18	Programmiersprache des TaTs
A-08	K-19	Benutzeroberfläche (GUI)
A-08	K-20	Usability_01
A-08	K-21	Usability_02
A-08	K-22	Usability_03

A-Nr.	K-Nr.	Kriterium
A-08	K-23	Wartbarkeit
A-09	K-24	Betriebssystem (OS)
A-10	K-25	Versionsverwaltungssystem (VCS)
A-11	K-26	Integration in Jenkins
A-12	K-27	Schnittstelle zu Jira
A-13	K-28	Datengetriebenes Testen
A-14	K-29	PDF-Prüfung
A-15	K-30	Entwickler-Lizenzkosten im ersten Jahr
A-15	K-31	Entwickler-Lizenzkosten in der Einsatzperiode
A-15	K-32	Floating-Lizenzen
A-15	K-33	Runtime-Lizenzkosten
A-15	K-34	Lizenzdauer für die Runtime-Umgebung
A-16	K-35	Runtime-Umgebung
A-16	K-36	Virtualisierte Umgebung
A-17	K-37	Browserunterstützung

Tabelle 5: Ableitung der Kriterien von identifizierten Anforderungen (Quelle: eigene Darstellung)

4.2.3 Gewichtung der Kriterien

Bei der Gewichtung der Kriterien wurde als erstes überlegt, welche Kriterien, aus der Sicht des Projekts, vom Testautomatisierungstool vollständig erfüllt werden müssen. Diese Kriterien sind die Ausschlusskriterien, die jeweils eine Gewichtung von 100 % haben. Sie sind im Unterabschnitt 3.2.5 beschrieben. Diese wurden identifiziert und in die Ausschlusskriterien-Liste eingetragen. Zu jedem Kriterium wird eine Frage und, wenn nötig, eine Erläuterung formuliert. Bei den Ausschlusskriterien ist immer eine Entscheidungsfrage zu beantworten. Die Ausschlusskriterien sind in der folgenden Tabelle aufgelistet:

K-Nr.	Kriterium	Frage	Erläuterung
K-01	Testversion	Gibt es eine Testversion des Tools?	Die TaTs sollen eine kostenlose Testversion über min. vier Wochen haben.
K-11	Debuggen	Die Möglichkeit von Debuggen ist gegeben.	
K-12	Komponentenerkennung	Unterstützt das TaT die Element-/Komponentenerkennung über die ID?	Für die Element-/Komponentenerkennung wird vorzugsweise die ID verwendet.

K-Nr.	Kriterium	Frage	Erläuterung
K-18	Programmiersprache des TaTs	Kann das Tool in JavaScript programmiert werden?	Im Unternehmen und auch im Projekt ist JavaScript-Know-how vorhanden.
K-24	Betriebssystem	Laufen alle Komponenten des TaTs unter Windows?	OS in VS3 ist Microsoft Windows.
K-26	Integration in Jenkins	Kann das TaT in die Jenkins CI/CD-Pipeline integriert werden?	Für nahtlose CI/CD-Prozesse ist es oft notwendig bzw. sehr hilfreich, Tests direkt in die damit verbundenen Pipelines einzubetten. Dazu gehört einerseits der Start der entsprechenden Testsuiten, aber auch das Auslesen von Testergebnissen und die Ablage von Berichten.
K-28	Datengetriebenes Testen	Gibt es eine Excel-Data-Driven Möglichkeit?	
K-30	Entwickler-Lizenzkosten im ersten Jahr	Liegen die Entwickler-Lizenzkosten im ersten Jahr unter 3.000 EUR pro Entwickler?	Entwickler-Lizenzkosten im ersten Jahr = einmaliger Kauf + jährliche Supportkosten
K-37	Browserunterstützung	Unterstützt das TaT das Testen von einer Web-Applikation in Firefox und Chrome?	

Tabelle 6: Die Ausschlusskriterien-Liste für das VS3-Web-Anwendungsprojekt (Quelle: eigene Darstellung, angelehnt an [2])

Bei den restlichen abgeleiteten Kriterien, die nicht zu den Ausschlusskriterien gehören, sind neben der Fragen und ggf. der Erläuterungen, auch die Bewertungen formuliert. Anschließend sind die Gewichtungen festgelegt und in gleichnamigen Spalten eingetragen. All diese Schritte sind auch mit dem Projektleiter abgestimmt worden. Die 28 Kriterien, die zu gewichten waren, wurden nach Wichtigkeit sortiert. Die wichtigeren Kriterien bekamen höhere Gewichtungen, die weniger wichtigen Kriterien bekamen niedrigere Gewichtungsprozente. Es wurde eine Gewichtungsprozentskala von 2 % bis 6 % angewendet. Um am Ende des Auswahlprozesses die Aussage treffen zu können, wie gut (in %) ein Testautomatisierungstool die Kriterien erfüllt, wurde darauf geachtet, dass die Summe aller Gewichtungen 100 % ergibt. Die folgende Tabelle 7 stellt den endgültigen individuellen Kriterienkatalog mit Gewichtungen der Kriterien dar.

K-Nr.	Kriterium	Gewichtung	Frage	Erläuterung	Bewertung
K-02	Evaluierung im Einsatzumfeld	2 %	Kann die Evaluierung vor Ort und am zu testenden Objekt stattfinden?		0 Punkte => Nein 2 Punkte => Ja
K-03	Marktpräsenz	2 %	Ist das TaT schon seit mindestens 5 Jahren auf dem Markt?		0 Punkte => weniger als 1 Jahr 1 Punkt => von 1 bis 5 Jahre 2 Punkte => mehr als 5 Jahre
K-04	Community	3 %	Ist in der letzten Woche oder im letzten Monat eine Aktivität in der Community vorhanden?		0 Punkte => nicht vorhanden 1 Punkt => im letzten Monat ist eine Aktivität vorhanden 2 Punkte => in letzter Woche ist eine Aktivität vorhanden
K-05	Support	4 %	Gibt es Support-Möglichkeit über E-Mail und Telefon?		0 Punkte => kein Support 1 Punkt => per E-Mail 2 Punkte => per E-Mail und Telefon
K-06	Update	4 %	Gab es in den letzten vier Monaten oder im letzten Jahr ein Update?		0 Punkte => kein Update 1 Punkt => Update im letzten Jahr 2 Punkte => Update in den letzten vier Monaten

K-Nr.	Kriterium	Gewichtung	Frage	Erläuterung	Bewertung
K-07	Dokumentation	4 %	Sind die TaT-Dokumentationen (Handbücher/Tutorials) für ein Selbststudium in deutscher und/oder englischer Sprache in ausreichender Menge vorhanden?	Der Know-how-Aufbau ist im Selbststudium-Modus geplant, der Selbststudium-Modus wird angestrebt. Zu betrachten sind Menge und Sprache.	0 Punkte => keine Dokumentation in deutscher und/oder englischer Sprache vorhanden 1 Punkt => es gibt Handbücher oder Tutorials in deutscher und/oder englischer Sprache 2 Punkte => es gibt Handbücher und Tutorials in deutscher und/oder englischer Sprache
K-08	Videotutorials	4 %	Gibt es mindestens 10 h Videotutorials auf der Webseite des TaTs oder auf YouTube in deutscher und/oder englischer Sprache?	Die Abläufe und Vorgehensweisen im Werkzeug sind durch illustrative Videotutorials und Beispiele dargestellt.	0 Punkte => keine Videotutorials 1 Punkt => weniger als 10 h Videotutorials 2 Punkte => mindestens 10 h Videotutorials
K-09	Personalressourcen	3 %	Gibt es mindestens drei MitarbeiterInnen, die mit dem evaluierten TaT arbeiten können?	Es gibt MitarbeiterInnen innerhalb des Unternehmens, die mit dem TaT schon gearbeitet haben. Dieses interne Know-how sollte berücksichtigt werden.	0 Punkte => Nein 1 Punkt => es gibt weniger als drei MitarbeiterInnen 2 Punkte =>es gibt mindestens drei MitarbeiterInnen
K-10	Record/Playback-Funktion	6 %	Hat das TaT eine Record/Playback-Funktion?		0 Punkte => Nein 2 Punkte => Ja

K-Nr.	Kriterium	Gewichtung	Frage	Erläuterung	Bewertung
K-13	Exception Handling	6 %	Existiert Exception Handling?	Nach einem Fehler bzw. in einer unerwarteten Situation während des Testdurchlaufs kann darauf reagiert werden und die Testsuite weiter ausgeführt werden.	0 Punkte => Nein 2 Punkte => Ja
K-14	Warten auf Events	6 %	Gibt es die Möglichkeit auf ein Event (z. B. Dokument geladen oder Komponente existiert) zu warten?	Wenn eine Vorbedingung erfüllt ist, wird ein Event ausgelöst, auf das gewartet wird.	0 Punkte => Nein 2 Punkte => Ja
K-15	Testsuitestruktur	6 %	Gibt es die Möglichkeit, die Testsuite modular aufzubauen?	Gliederung in Testfallsätze und Testfälle (Baumansicht).	0 Punkte => Nein 2 Punkte => Ja
K-16	Log-Datei	2 %	Ist es möglich, eine konfigurierbare Log-Datei zu erstellen?	Um den Testlauf nachvollziehen zu können.	0 Punkte => Nein 1 Punkt => mit Plug-in oder External Library 2 Punkte => wenn die Möglichkeit im TaT integriert ist
K-17	Reporting	3 %	Gibt es im TaT auch einen Reportgenerator mit eigener Gestaltungsmöglichkeit?	Die Testläufe sollen dokumentiert werden in Form von Reports. Zu den erwarteten Dokumentationen gehören auch die Testergebnisse. Gestaltung auch nach Kundenwunsch.	0 Punkte => Nein 1 Punkt => es gibt eine Reportgenerierung 2 Punkte => es gibt Reports nach Kundenwunsch
K-19	Benutzeroberfläche (GUI)	6 %	Hat das TaT eine GUI?		0 Punkte => Nein 2 Punkte => Ja

K-Nr.	Kriterium	Gewichtung	Frage	Erläuterung	Bewertung
K-20	Usability_01	2 %	Ist es möglich, Elemente bzw. Zeilen mit unterschiedlichen Funktionalitäten im Testfall einzufügen?	Elemente bzw. Zeilen mit unterschiedlichen Funktionalitäten: Ablaufsteuerung, Mausklick, Try, Catch usw.	0 Punkte => Nein 2 Punkte => Ja
K-21	Usability_02	2 %	Ist es möglich, in der Testsuite Zeilen zu verschieben?	Änderbarkeit der Testsuite.	0 Punkte => Nein 2 Punkte => Ja
K-22	Usability_03	2 %	Ist es möglich, in der Testsuite das Löschen rückgängig zu machen?	Änderbarkeit der Testsuite.	0 Punkte => Nein 2 Punkte => Ja
K-23	Wartbarkeit	4 %	Ist ein Protokoll in Baumansicht vorhanden?	Im Protokoll kann der Fehler schnell dargestellt werden.	0 Punkte => Nein 2 Punkte => Ja
K-25	Versionsverwaltungssystem (VCS)	6 %	Sind die TaT-Projekt-Artefakte mit *git* verwaltbar?		0 Punkte => Nein 2 Punkte => Ja
K-27	Schnittstelle zu Jira	2 %	Hat das TaT eine Jira-Schnittstelle?	Zurzeit wird in VS3 das Jira-Ticketing-System für die Testfälle, Bugs, Testdokumentation usw. benutzt.	0 Punkte => Nein 2 Punkte => Ja
K-29	PDF-Prüfung	2 %	Können PDF-Dateien auf Inhalt und Gestaltung geprüft werden?		0 Punkte => Nein 2 Punkte => Ja

K-Nr.	Kriterium	Gewichtung	Frage	Erläuterung	Bewertung
K-31	Entwickler-Lizenzkosten in der Einsatzperiode	6 %	Liegen die Entwickler-Lizenzkosten in der Einsatzperiode unter 1.500 EUR/Jahr pro Entwickler?	Das Projekt wurde auf drei Jahre geplant. Die Entwickler-Lizenzkosten in der Einsatzperiode werden nach der folgenden Formel berechnet: Entwickler-Lizenzkosten in der Einsatzperiode = (einmaliger Kauf/3) + jährliche Supportkosten.	0 Punkte => die Kosten sind mehr als 1.500 EUR/Jahr pro Entwickler 1 Punkt => die Kosten sind zwischen 0 EUR und 1.500 EUR 2 Punkte => das TaT ist ein Open-Source-Tool (Kosten = 0 EUR)
K-32	Floating-Lizenzen	2 %	Gibt es eine einfache Möglichkeit (z. B. per Lizenz-Datei), beliebige Arbeitsplätze für das TaT freizuschalten?	Die Lizenzen sollten nicht personengebunden sein und sollten einfach von Arbeitsplatz zu Arbeitsplatz übertragbar sein.	0 Punkte => Nein 2 Punkte => Ja
K-33	Runtime-Lizenzkosten	2 %	Sind die Runtime-Lizenzkosten im vorgegebenen Rahmen?	Die jährlichen Runtime-Lizenzkosten sollten pro Lizenz weniger als 500 EUR betragen.	0 Punkte => die Lizenzkosten sind über 500 EUR 1 Punkt => die Lizenzkosten sind zwischen 0 EUR und 500 EUR 2 Punkte => das TaT ist ein Open-Source-Tool (Lizenzkosten = 0 EUR)
K-34	Lizenzdauer für die Runtime-Umgebung	2 %	Gibt es Runtime-Lizenzierungsmöglichkeiten pro Monat?	Die Runtime-Lizenzen werden in einem begrenzten Zeitraum für die Lasttests notwendig.	0 Punkte => Nein 2 Punkte => Ja

K-Nr.	Kriterium	Gewichtung	Frage	Erläuterung	Bewertung
K-35	Runtime-Umgebung	3 %	Gibt es eine Möglichkeit, unabhängig von der Entwicklungsumgebung die automatisierten Tests laufen zu lassen?	Die Tests sollten von der Testentwicklungsumgebung unabhängig ausführbar sein.	0 Punkte => Nein 2 Punkte => Ja
K-36	Virtualisierte Umgebung	4 %	Können die Testfälle in Docker laufen?	Die automatisierten Tests sollen auch in virtualisierter Umgebung laufen.	0 Punkte => Nein 2 Punkte => Ja

Tabelle 7: Der individuelle Kriterienkatalog für das VS3-Web-Anwendungsprojekt mit Gewichtungen (Quelle: eigene Darstellung, angelehnt an [2])

4.3 Phase 3: Evaluierung

In der dritten Phase des Auswahlprozesses wurden zuerst die Testautomatisierungstools selektiert, was im nächsten Unterabschnitt beschrieben wird.

4.3.1 Selektieren

Für die Ausgangsliste der zu evaluierenden Testautomatisierungstools wurde eine Suche im Internet nach Testautomatisierungstool-Auflistungen vorgenommen. Dabei stellte sich heraus, dass es zahlreiche unterschiedliche Auflistungen über die gängigsten Tools gibt. Ausgesucht wurden die folgenden Quellen:

1. Gartner: Software Test Automation Reviews and Ratings [17]
2. The Forrester Wave ™: Continuous Functional Test Automation Suites [59]
3. International Testing Board: Testautomatisierungstools [58]
4. Die besten Tools für die Testautomatisierung (2021) [36]
5. Test Automation Landscape in 2020 [28]

In die Testautomatisierungstool-Liste wurden die Testautomatisierungstools aufgenommen, die in vier von fünf der obigen Auflistungen enthalten waren. Darüber hinaus wurden die vier Testautomatisierungstools zu der Liste hinzugefügt, die aus der Sicht der

iSYS Software GmbH relevant sind. Die Testautomatisierungstool-Liste wurde in alphabetischer Reihenfolge sortiert. Jedes Testautomatisierungstool, das in der erstellten Liste enthalten ist, wurde auf Ausschlusskriterien überprüft. Mit der Erfüllung eines Ausschlusskriteriums wurde das Testautomatisierungstool aus der Evaluierung ausgeschlossen. Zu der erstellten Tabelle wurde eine weitere Spalte hinzugefügt und das entsprechende Ausschlusskriterium eingetragen. Die entstandene Liste mit dem ggf. zutreffenden Ausschlusskriterium ist in der Tabelle 8 enthalten.

TaT-Nr.	Testautomatisierungstool	Ausschlusskriterium
TaT-01	Eggplant	-
TaT-02	IBM Rational Functional Tester (RFT)	-
TaT-03	Katalon Studio	-
TaT-04	Parasoft	-
TaT-05	Protractor	-
TaT-06	QF-Test	-
TaT-07	Ranorex Studio	Entwickler-Lizenzkosten im ersten Jahr (K-30)
TaT-08	Selenium	-
TaT-09	TestCafe	-
TaT-10	TestCafe Studio	-
TaT-11	TestComplete	Entwickler-Lizenzkosten im ersten Jahr (K-30)
TaT-12	Tricentis Tosca	Entwickler-Lizenzkosten im ersten Jahr (K-30)

Tabelle 8: Liste der Testautomatisierungstool-Kandidaten unter Einbeziehung der Ausschlusskriterien (Quelle: eigene Darstellung)

Aus der Tabelle 8 ist zu entnehmen, dass von den insgesamt zwölf Testautomatisierungstools neun kein Ausschlusskriterium erfüllen und damit evaluiert werden könnten. Dennoch wurden in dieser Bachelorarbeit lediglich vier Testautomatisierungstools evaluiert, nämlich die aus der Sicht der iSYS Software GmbH für das VS3-Web-Anwendungsprojekt infrage kommen: QF-Test, Selenium, TestCafe und TestCafe Studio. Diese vier Testautomatisierungstools werden in folgenden Unterabschnitten aus der Sicht der abgeleiteten Kriterien bezüglich des VS3-Web-Anwendungsprojekts beschrieben und bieten somit keine vollständige Darstellung der möglichen Tools.

4.3.2 QF-Test

QF-Test ist eine etablierte Software zur Testautomatisierung für Java-, Web- und Windows-Desktop-Anwendungen mit einer grafischen Benutzeroberfläche (GUI) [38].

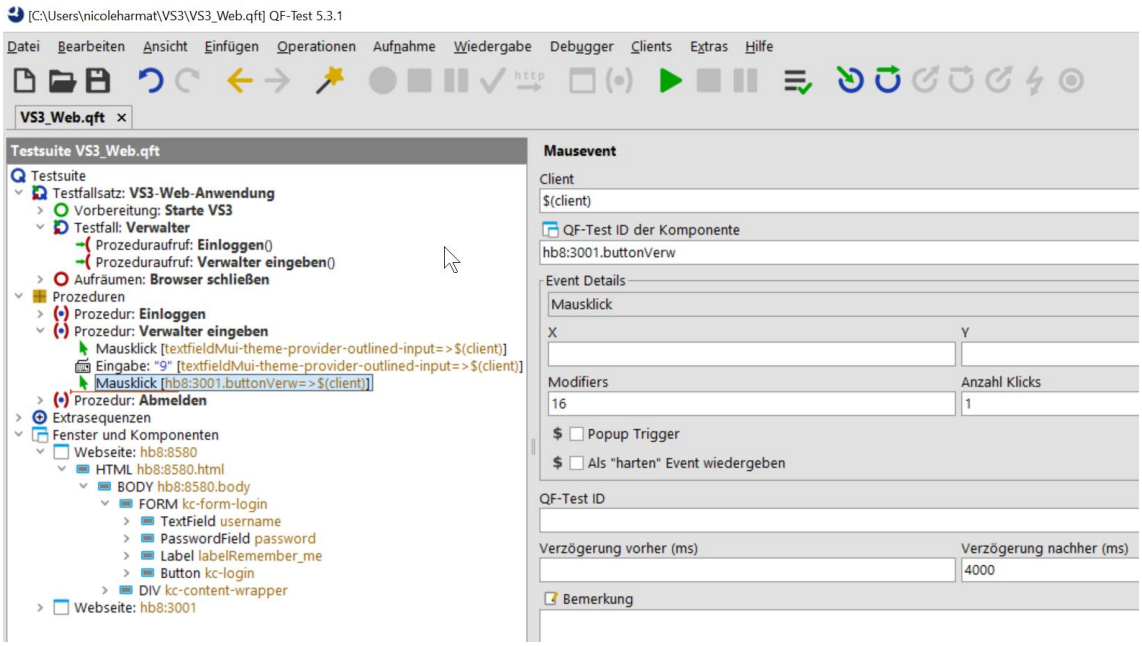

Abbildung 8: Die GUI von QF-Test (Quelle: eigener Screenshot)

QF-Test ist seit 20 Jahren auf dem Markt und wurde von der Quality First Software GmbH mit Sitz in Geretsried[10] entwickelt. Die Software steht zum Downloaden zur Verfügung und kann als Demoversion ohne Zeitlimit und kostenfrei getestet werden. Nur das Speichern der Testfälle ist dabei nicht möglich. Es gibt eine weitere kostenlose Testversion, die den vollen Funktionsumfang mit einem freien Zugang zum Support-Team bietet. Diese Testversion kann mit der Testlizenz vier Wochen lang getestet werden. Nach der Kontaktaufnahme mit dem Support-Team erfolgte die Zusendung der Testversion-Lizenzen in dem hier beschriebenen Fall innerhalb von einer Stunde. Eine Kontaktaufnahme mit dem Support-Team ist über E-Mail, Kontaktformular oder telefonisch möglich. Die Evaluierung kann in der Demo- oder Testversion mit der VS3-Webanwendung erfolgen. Außerdem wird von der QF Software GmbH jeden Montag zu einer festen Uhrzeit ein kostenloses Starter-Webinar angeboten. Die QF-Test-Community ist aktiv, es sind monatliche Einträge z. B. auf der eigenen Webseite in einer Mailingliste vorhanden. Updates finden mindestens alle zwei Monate statt. Letztes Update war im Juni 2021.

QF-Test hat eine umfangreiche und genaue Dokumentation in deutscher und englischer Sprache in Form eines Handbuchs, das im PDF- und HTML-Format auf der Webseite von QF-Test zur Verfügung steht [38]. Das Handbuch weist an vielen Stellen mit einem Link

[10] Geretsried in Bayern, Deutschland

auf dem Thema entsprechende Videotutorials in deutscher oder englischer Sprache hin. Das Unternehmen hat insgesamt mehrere Stunden Videos erstellt und auf der Webseite des QF-Tests und in YouTube zur Verfügung gestellt.

Die Record- und Playback-Funktionen sind aus der Menü- oder Werkzeugleiste des QF-Test Anwendungsfensters zu starten. Für die Wiedererkennung der grafischen Komponenten, die für die Stabilität der Testsuite sorgt, bietet QF-Test viele Konfigurationsmöglichkeiten, wie z. B. *Name übertrifft alles*. Dabei ist es notwendig, dass bei der Implementierung seitens der EntwicklerInnen eindeutige Namen der GUI-Komponenten der VS3-Webanwendung in den Quellcode eingepflegt werden.

QF-Test hat viele Exceptions, mit welchen Fehlerzustände behandelt werden können. Im Handbuch von QF-Test sind 53 mögliche Exceptions aufgelistet.

QF-Test hat eine sehr überschaubare Testsuitestruktur (Abbildung 8). Testfälle werden in eine grafische Baumansicht implementiert, die eine grafische Programmiersprache darstellt, in der es auch möglich ist, Ablaufsteuerungselemente, wie z. B. *if* oder *while*, einzubinden. Die Zeilen dieser Baumansicht werden in QF-Test *Knoten* genannt. Es ist möglich, die Knoten mit verschiedenen Funktionalitäten per Drop-down-Menü auszusuchen, einzufügen, zu verschieben oder zu löschen und wiederherzustellen.

Ein Protokoll wird nach jedem Testlauf erstellt, um Fehler schnell finden zu können. Mit einem Mausklick auf das Fehler-Ikon wird genau die Stelle/Zeile im Testsuite gezeigt, wo sich der Fehler befindet.

Einen Report mit einer Pie-Chart-Darstellung für erfolgreiche und fehlgeschlagene Testfälle zu generieren, ist auch möglich. Der Report kann individuell nach Kundenwunsch gestaltet werden.

QF-Test kann mit dem git Versionsverwaltungssystem verwaltet werden, womit Teamarbeit ermöglicht ist. Eine Schnittstelle zu Jira ist ebenfalls vorhanden. PDF-Dateien können laut Handbuch auch auf Komponenten und deren Eigenschaften getestet werden.

Warten auf verschiedene Events, die QF-Test anbietet, wie z. B. Laden eines Dokumentes, ist eine Art von Knoten, die in einem Testfall verwendet werden kann.

Die Lizenzkosten betragen einmalig 2095 EUR. Darüber hinaus wird ein jährlicher Pflegevertrag für 505 EUR angeboten, der die Updates des QF-Test TaTs und den Support beinhaltet.

Die Runtime-Lizenzkosten betragen einmalig 1045 EUR. Der Pflegevertrag dafür kostet jährlich 155 EUR.

Die Lizenzen können auch gemietet werden: Dabei kostet die Entwicklerlizenz 1090 EUR, die Runtime-Lizenz 470 EUR pro Jahr [37].

„Alle QF-Test Lizenzen sind floating, d.h. nicht an individuelle Rechner oder Anwender gebunden (im Gegensatz zu node locked Lizenzen, die an einen User gebunden sind)“ [37].

Auf Webseite von QF-Test ist ein Fallbeispiel über Einsatz von Docker in Verbindung mit QF-Test enthalten [40].

QF-Test stellt PDF-Dokumente zum Downloaden bereit, wie Produktbroschüre [38], Produktbeschreibung und Lizenzmodelle [37], und eine Checkliste [39], die bei der Evaluierung hilfreich waren. Sie geben einen Überblick über die Features von QF-Test, wie z. B. die Lizenzenzmodelle, Testvorbereitung, Testdurchführung, Reportgenerierung, Fehleranalyse, Skripting, Prozeduren, Testmanagement, Keyword- und Data-Driven Testing usw.

4.3.3 Selenium

Selenium hat sich als das meisteingesetzte Testautomatisierungstool für Webanwendungen bewährt. Es ist eine Open-Source-Software und existiert seit 2004. Selenium beinhaltet drei Komponenten, die zusammen ein vielseitiges Testsystem bilden und ermöglichen, die Webbrowser zu automatisieren [29]. Die drei Komponenten sind: Selenium IDE, Selenium WebDriver und Selenium Grid [46]. Um Selenium in JavaScript Programmiersprache nutzen zu können, wurden zuerst node.js und npm auf dem Rechner installiert. Als nächstes wurden die Bibliotheken für Selenium WebDriver mit folgenden Befehlen installiert:

```
npm init
npm install -selenium-webdriver
```

Selenium WebDriver ist im W3C-Standard aufgenommen und unterstützt alle gängigen Browser [48]. Die WebDriver-Technologie wurde in Absatz 2.3.1 erläutert.

„[Selenium] WebDriver selbst hat keine Testfunktionen: Dieser kann nicht Werte vergleichen, feststellen ob [sic] ein Test ok ist oder fehlschlägt [sic] und er besitzt keine Funktionen bezüglich Reporting oder kennt auch nicht die Angenommen/Wenn/Dann

Grammatik“ [49]. In Kombination mit einem Testautomatisierungsframework, wie beispielsweise Mocha[11] für JavaScript, werden die Befehle implementiert. Über diese Befehle wird der WebDriver gesteuert und die entsprechenden Schritte des Tests werden in der GUI des Browsers ausgeführt [49].

Um Selenium IDE nutzen zu können, wurde die Software als Webbrowsererweiterung zu Chrome und Firefox hinzugefügt [47]. Darüber hinaus wurden jeweils die browserspezifischen Driver, GeckoDriver für Firefox und ChromeDriver für Chrome, heruntergeladen und im Systempfad hinzugefügt [47]. Als IDE[12] wurde Visual Studio Code (VS Code) installiert.

Die Selenium IDE hat eine Record/Playback-Funktion zum Aufnehmen von Testfällen. Selenium IDE zeichnet mehrere Locatoren für jedes Element auf, mit dem es interagiert. Ein Element kann über einen Locator bei der Wiedergabe (engl. Playback) identifiziert werden. Wenn ein Locator während der Wiedergabe fehlschlägt, werden die anderen ausprobiert, bis einer erfolgreich ist [48]. Die GUI von Selenium mit der Möglichkeit, einen Locator auszuwählen, ist in der Abbildung 9 zu sehen.

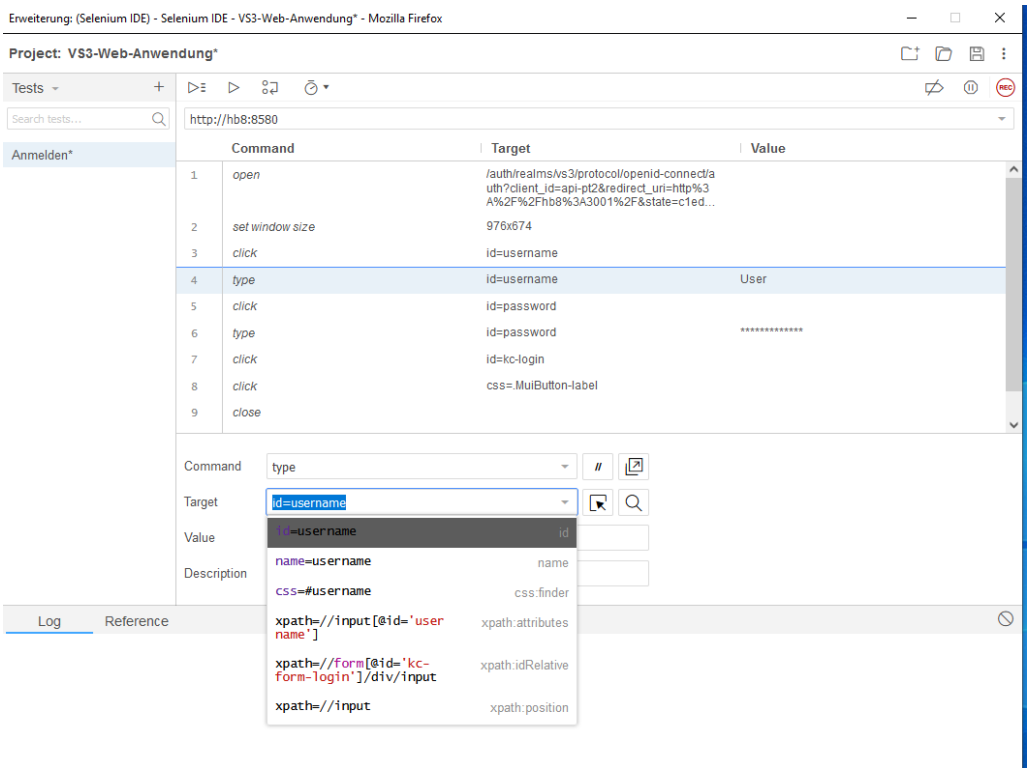

Abbildung 9: Das GUI von Selenium IDE (Quelle: eigener Screenshot)

[11] **Mocha** ist ein flexibles Open-Source-Testframework, welches JavaScript verwendet und zur Durchführung von automatisierten Tests in Node.js und Browsern verwendet wird [45].

[12] **IDE** (integrierte Entwicklungsumgebung) ist eine Sammlung von Computerprogrammen, mit denen die Aufgaben der Softwareentwicklung möglichst ohne Medienbrüche bearbeitet werden können. [https://de.wikipedia.org/wiki/Integrierte_Entwicklungsumgebung].

Die getätigten Interaktionen mit der VS3-Webanwendung wurden aufgenommen und gespeichert. Die Datei wurde in eine Skript-Datei konvertiert und in Visual Studio Code geöffnet. Hier bestand die Möglichkeit, den Testcode zu bearbeiten, zu erweitern oder zu ändern. Eine Darstellung des Codes in VS Studio ist auf der Abbildung 10 zu sehen.

Die Selenium-Community ist sehr aktiv. Auf der Webseite von Selenium ist z. B. die *Official User Group* verlinkt. Hier werden mehrere Einträge am Tag eingestellt. Alle zwei Wochen findet online ein *Public Project Meeting* statt. Das letzte Update fand vor zwei Jahren statt. Selenium 4 befindet sich in der Entwicklung. Die letzte Betaversion wurde im Juni 2021 veröffentlicht [47].

Selenium stellt auf der Webseite eine englisch- und teilweise deutschsprachige Dokumentation bereit. Darüber hinaus sind zahlreiche Tutorials, Videotutorials und Beiträge auf verschiedenen Webseiten und in YouTube zu finden.

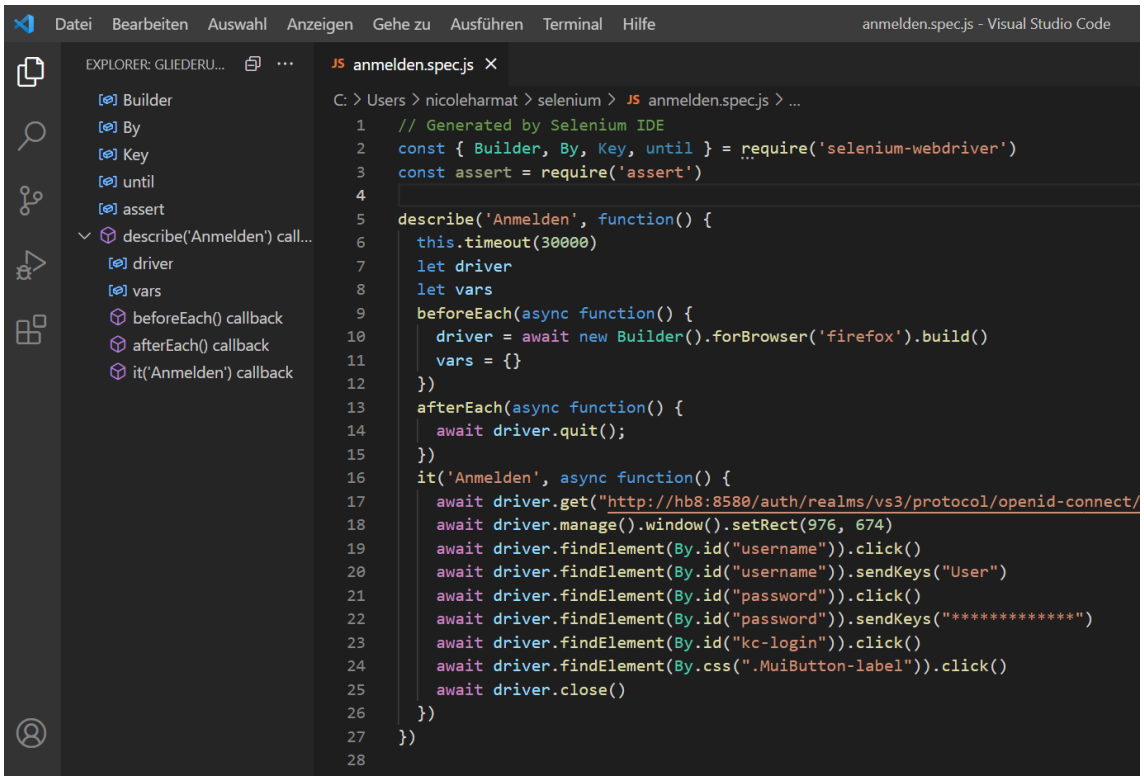

Abbildung 10: Aus Selenium IDE exportierte Datei in VS Code geöffnet (Quelle: eigener Screenshot)

Exceptions können in VS Code-IDE in Testfälle implementiert werden.

Warten auf ein Event kann auf drei Arten implementiert werden [50]:

1. Explizites Warten – die Programmausführung wird angehalten oder der Thread eingefroren, bis eine bestimmte Bedingung erfüllt ist.

2. Implizites Warten – die Programmausführung wird angehalten oder der Thread eingefroren, bis das gesuchte Element gefunden wird. Dies kann nützlich sein, wenn bestimmte Elemente auf der Webseite nicht sofort verfügbar sind und einige Zeit zum Laden benötigen.

3. Fließendes Warten – definiert die maximale Wartezeit für eine Bedingung sowie die Häufigkeit, mit der die Bedingung überprüft werden soll.

Die Testfälle für die Testautomatisierung des VS3-Projekts können mit der Unterstützung von z. B. Mocha modular aufgebaut werden.

Die Reporterstellung bei Selenium ist nur möglich, wenn ein Reportingframework von Drittanbietern in das Testautomatisierungsframework von Selenium integriert wird.

In Selenium ist die Log-Dateierstellung nicht integriert. Mit Mocha-Framework ist das Loggen über die Konsole möglich [34]. Ein Protokoll mit der Möglichkeit, die Fehler in einer grafischen Baumansicht darzustellen, ist nicht möglich.

Nachdem ein Testfall mit Selenium IDE erstellt worden war, wurden einige Funktionalitäten getestet mit folgendem Ergebnis:

1. Zeilen auszuwählen mit verschiedenen Funktionen ist in Selenium IDE möglich über Drop-down-Menü,

2. Verschieben der Zeilen ist auch möglich,

3. nach Löschen einer Zeile ist eine Wiederherstellung nicht möglich.

Selenium kann mit git verwaltet werden. Eine Integration mit Jira ist auch möglich. Auf der Webseite von Docker sind zahlreiche Docker-Images mit Selenium vorhanden [10].

Selenium ist eine kostenfreie Open-Source-Software.

4.3.4 TestCafe und TestCafe Studio

TestCafe ist ein plattformübergreifendes, Open-Source-Testautomatisierungstool zum Testen von Webanwendungen [53]. TestCafe Studio ist eine kommerzielle Web-Test-IDE, die auf der Engine des TestCafe Open-Source-Testautomatisierungstools entwickelt wurde [57]. Demnach werden beide Softwares in diesem Unterabschnitt zusammen analysiert. Bei der Bewertung in der Entscheidungsmatrix hat jedes Tool eine eigene

Spalte bekommen. Die Produkte haben viele gemeinsame Merkmale und Funktionen. In der folgenden Tabelle sind die wichtigsten Eigenschaften von TestCafe und TestCafe Studio dargestellt:

Eigenschaft	TestCafe	TestCafe Studio
Plattformübergreifend und browserübergreifend	✓	✓
Keine Notwendigkeit für WebDriver oder Browser Plug-ins	✓	✓
Testfälle in JavaScript oder TypeScript schreiben	✓	✓
Stabile Testläufe wegen *Smart Assertion Query Mechanism*	✓	✓
Schnellere Testläufe wegen *Automatic Waiting Mechanism*	✓	✓
Schnellere Testläufe wegen *Concurrent Test Execution*	✓	✓
Benutzerdefinierte Report-Plug-ins	✓	✓
Integration in CI Systeme	✓	✓ *
Open-Source-Software	✓	
Record/Playback-Funktion		✓
Interactive Test Editor		✓
Automatische Selektoren-Generierung		✓
Code Editor (IDE)		✓

* mit Verwendung von TestCafe

Tabelle 9: Die wichtigsten Eigenschaften von TestCafe und TestCafe Studio (Quelle: eigene Darstellung, angelehnt an [55])

TestCafe und TestCafe Studio wurden von Developer Express Inc. (DevExpress), einem Unternehmen mit Sitz in Kalifornien, USA, entwickelt.

TestCafe kann als Open-Source-Testautomatisierungstool eigenständig eingesetzt werden. Beim Einsatz von TestCafe Studio ist allerdings die Installation von TestCafe notwendig, um die Testfälle in einen CI/CD-Prozess integrieren zu können.

Um TestCafe verwenden zu können, ist es notwendig, node.js Framework und den npm Pakagemanager zu installieren. Die Installation selbst wird vollständig von npm übernommen. Die Initialisierung und Installierung von TestCafe wurden mit folgenden Befehlen durchgeführt [54]:

```
npm init
npm install -save-dev testcafe
```

Damit ist das Testautomatisationstool einsatzbereit. Mit einer geeigneten IDE, z. B. Visual Studio Code, kann das VS3-Testprojekt in JavaScript implementiert werden.

TestCafe Studio kann unter Windows, Linux und MacOS installiert werden. Um eine Webanwendung evaluieren zu können, wird vom Hersteller eine 30-tägige Testversion angeboten. Nach der Installation war das Produkt ohne weiteren Konfigurationsbedarf einsetzbar. Die GUI von TestCafe Studio mit der Möglichkeit, einen der automatisch generierten Selektoren auszuwählen, ist in der Abbildung 11 zu sehen.

Beide Testautomatisationstools haben eine grundlegende Dokumentation in englischer Sprache auf eigener Webseite. Ein kostenfreies Handbuch ist nicht vorhanden. Es sind für TestCafe mehrere Stunden Videotutorials auf YouTube vorhanden. TestCafe Studio hat wenig Videos. Eine Kontaktaufnahme mit dem Support-Team ist über E-Mail oder Kontaktformular möglich. Für beide TaTs finden regelmäßig Updates statt. Das letzte Update fand für TestCafe im Juni 2021 statt [56], für TestCafe Studio in April 2021 [8].

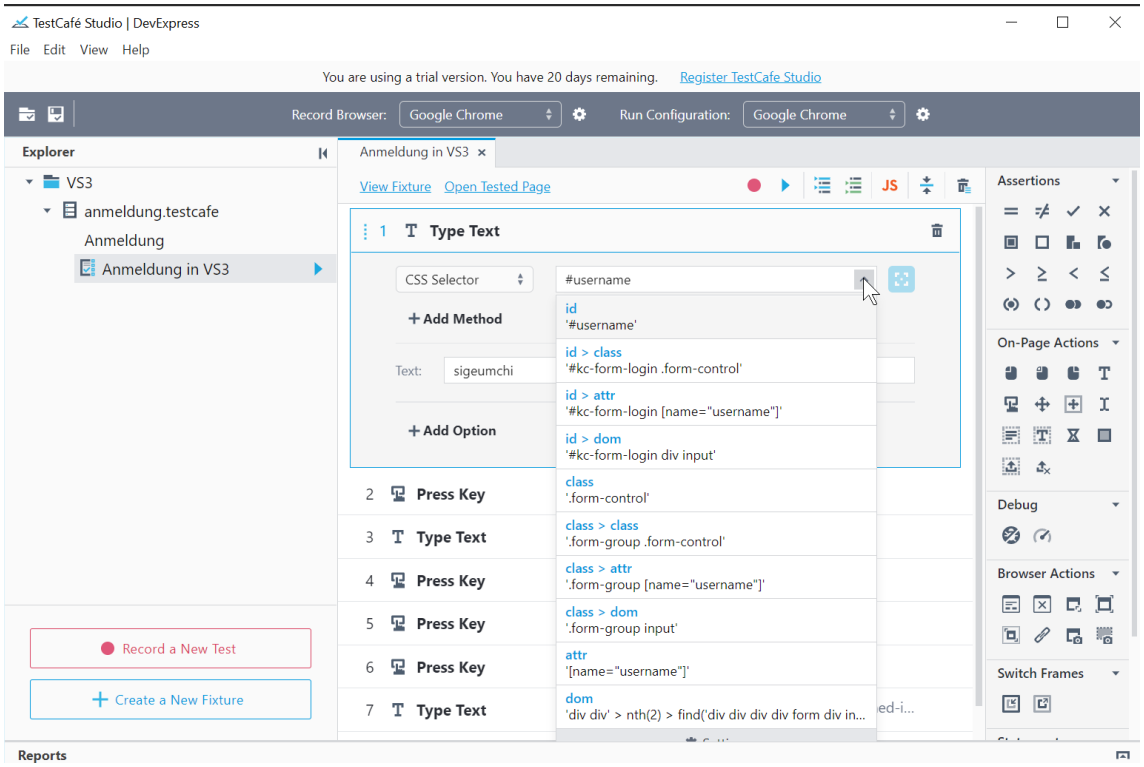

Abbildung 11: Die GUI von TestCafe Studio (Quelle: eigener Screenshot)

Der browserübergreifende Einsatz wird mit einem im Hintergrund laufenden Proxy-Server ermöglicht. TestCafe basiert nicht wie einige andere Testautomatisierungstools, z. B. Selenium, auf der Fernsteuerung des Browsers mit dem WebDriver[13]. Der verdeckte Proxy-Server transformiert die HTML- und JavaScript-Codes. Mit der Transformation werden die für die Testdurchführung notwendigen Codefragmente in den Originalcode

[13] Die WebDriver-Technologie ist im Unterabschnitt 2.3.1 erläutert.

injiziert. Damit werden die Web-Applikationen in jedem Browser, der HTML5 unterstützt, testbar. Diese Proxy-Server-Technologie ist im Unterabschnitt 2.3.1 erläutert.

Der *Automatic Waiting Mechanism* ist eine vorteilhafte Eigenschaft von TestCafe. Eine Webseite bzw. die Elemente der Webseite werden asynchron aufgebaut. Deshalb kann es vorkommen, dass bei Aufruf der Webseite ein Element nicht gleich erreichbar ist. Die Aufbauzeiten sind von vielen Rahmenbedingungen abhängig und deswegen immer unterschiedlich. Wenn z. B. innerhalb eines Testfalls ein Klick auf eine in der Webseite noch nicht vorhandene Button getätigt wird, dann entsteht ein Fehler. Es gibt mehrere Techniken, um einen zu frühen Klick auf ein Element zu vermeiden. Diese Techniken brauchen meistens zusätzliche Programmcodes und verursachen längere Wartezeiten. Mit dem im TestCafe implementierten *Automatic Waiting Mechanism* wird, ohne zusätzlichen Programmcode, innerhalb einer definierten Time-out-Zeit das ausgewählte Element mehrfach angesprochen, bis es erscheint. Mit dieser Technik wurde die Testlaufzeit optimiert.

Die Integration in den CI/CD-Prozess ist mit TestCafe möglich. Die mit dem TestCafe Studio aufgenommenen Testfälle können als JavaScript-Datei exportiert und mit TestCafe ausgeführt werden. Somit können auch die Tests, die mit TestCafe Studio erfasst wurden, in eine CI/CD-Pipeline integriert werden. Die folgende Abbildung visualisiert die mit Visual Studio Code geöffnete JavaScript-Datei.

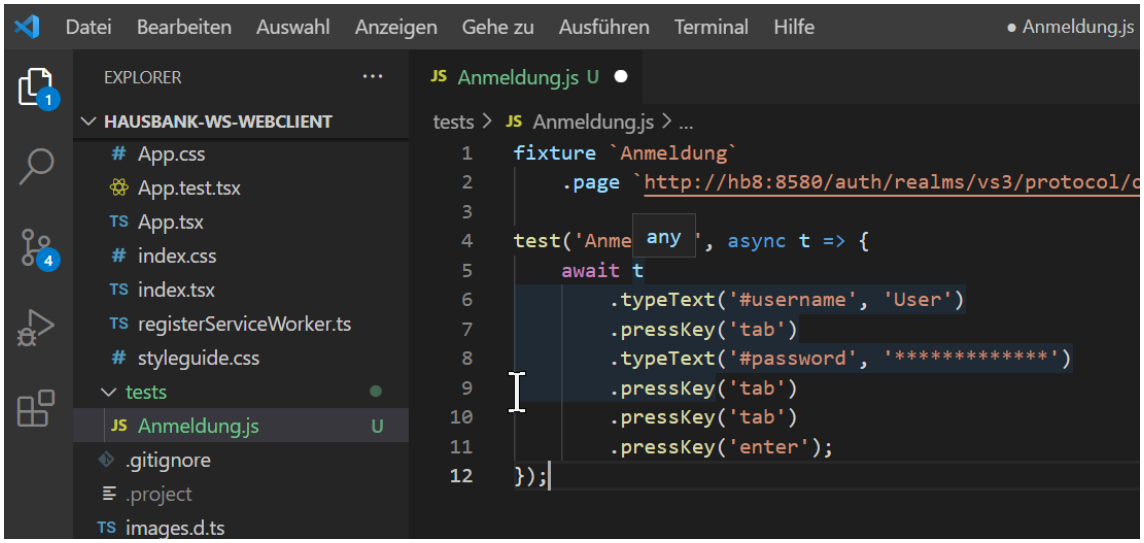

Abbildung 12: In TestCafe Studio aufgenommener Testfall in VS Code geöffnet (Quelle: eigener Screenshot)

Die Datei kann auch direkt in TestCafe Studio mit dem integrierten Code Editor IDE geöffnet und bearbeitet werden. Somit ist ein Exception Handling in beiden TaTs implementierbar. Die Testsuitestruktur ist modular aufgebaut und hat eine Baumansicht. Die

Zeilen sind verschiebbar. Es gibt bei beiden Varianten die Möglichkeit, Zeilen in Code einzufügen. Das Löschen einer Zeile kann nicht rückgängig gemacht werden. Eine Log-Datei und ein Report können nur über ein Plug-in erstellt werden. Die Verwaltung der Versionen über git ist nur bei TestCafe möglich. Eine Schnittstelle zu Jira ist nicht vorhanden.

Auf der Webseite von Docker sind Docker-Images mit TestCafe vorhanden [11]. Mit TestCafe Studio ist kein Image zu finden.

TestCafe ist eine kostenfreie Open-Source-Software.

TestCafe Studio kostet ca. 210 EUR mit Community-Support bzw. TestCafe Studio Pro ca. 420 EUR pro Jahr mit vollem Support. Bei der Verlängerung der Lizenz kostet TestCafe Studio ca. 85 EUR und TestCafe Studio Pro ca. 170 EUR pro Jahr (die Preise sind auf der Webseite von TestCafe Studio in Dollar angegeben) [57]. Die Lizenzen sind keine Floating-Lizenzen, sie sind an bestimmte Personen gebunden.

4.3.5 Bewertung und die Entscheidungsmatrix

Die Bewertung erfolgte pro Kriterium für jedes in die Entscheidungsmatrix aufgenommene und evaluierte Testautomatisierungstool. Die in der Bewertung entstandenen Kriteriumerfüllungsgradpunktzahlen (KEGPs) wurden in die entsprechenden Zellen der Matrix eingetragen. Die Kriteriumerfüllungsgradpunktzahlen wurden basierend auf (1) den in den vorherigen Evaluierungsschritten für Testautomatisierungstools ermittelten Informationen, (2) gesammelten Erfahrungswerten und (3) den im Kriterienkatalog beschriebenen Bedingungen vergeben. Folgende Entscheidungsmatrix mit Bewertungen und Ergebnissen ist entstanden (Tabelle 10):

K-Nr.	Kriterium	Gewichtung	TaT-06 QF-Test	TaT-08 Selenium	TaT-09 TestCafe	TaT-10 TestCafe Studio
K-02	Evaluierung im Einsatzumfeld	2 %	2	2	2	2
K-03	Marktpräsenz	2 %	2	2	2	2
K-04	Community	3 %	2	2	2	2
K-05	Support	4 %	2	0	1	1
K-06	Update	4 %	2	1	2	2
K-07	Dokumentation	4 %	2	2	1	1
K-08	Videotutorials	4 %	2	2	2	1
K-09	Personalressourcen	3 %	2	0	0	0
K-10	Record/Playback-Funktion	6 %	2	2	0	2
K-13	Exception Handling	6 %	2	2	2	2
K-14	Warten auf Events	6 %	2	2	2	2
K-15	Testsuitestruktur	6 %	2	2	2	2
K-16	Log-Datei	2 %	2	1	1	1

K-Nr.	Kriterium	Gewich-tung	TaT-06 QF-Test	TaT-08 Selenium	TaT-09 TestCafe	TaT-10 TestCafe Studio
K-17	Reporting	3 %	2	0	0	0
K-19	Benutzeroberfläche (GUI)	6 %	2	2	0	2
K-20	Usability_01	2 %	2	0	0	2
K-21	Usability_02	2 %	2	0	0	2
K-22	Usability_03	2 %	2	2	2	0
K-23	Wartbarkeit	4 %	2	0	0	0
K-25	Versionsverwaltungssystem (VCS)	6 %	2	2	2	0
K-27	Schnittstelle zu Jira	2 %	2	2	0	0
K-29	PDF-Prüfung	2 %	2	0	0	0
K-31	Entwickler-Lizenzkosten in der Einsatzperiode	6 %	1	2	2	1
K-32	Floating-Lizenzen	2 %	2	2	2	0
K-33	Runtime-Lizenzkosten	2 %	1	2	2	1
K-34	Lizenzdauer für die Runtime-Umgebung	2 %	0	2	2	0
K-35	Runtime-Umgebung	3 %	2	2	2	2
K-36	Virtualisierte Umgebung	4 %	2	2	2	0
Ergebnis			**94,0 %**	**77,0 %**	**65,0 %**	**59,0 %**

Tabelle 10: Die Entscheidungsmatrix mit den Bewertungen und Ergebnissen (Quelle: eigene Darstellung, angelehnt an [2])

Die Ergebnisse wurden mit dem Einsatz der Formel, die im Abschnitt 3.4 definiert ist, berechnet. Das QF-Test-Testautomatisierungstool erfüllte mit 94,0 % die Kriterien aus dem Kriterienkatalog.

Die Empfehlung: Das QF-Test Testautomatisierungstool wird für den Einsatz im Projekt empfohlen.

5 Diskussion

Bei der Evaluierung der Testautomatisierungstools wurden einige relevante Erkenntnisse gewonnen:

- die verschiedenen Testautomatisierungstools haben in verschiedenen Teilbereichen ihre Stärken und
- es sind technische und nichtfunktionale Kriterien erkennbar,
- die Testautomatisierungstools erfüllen größtenteils die geforderten technischen Kriterien,
- die Testautomatisierungstools befinden sich noch immer in einem sehr dynamischen Entwicklungsprozess, viele neue Produkte werden entwickelt und am Markt eingeführt.

Die gewonnenen Erkenntnisse bei der Evaluierung zeichnen einige Bereiche für mögliche weiterführende Untersuchungen auf, die in den folgenden drei Abschnitten erläutert werden.

5.1 Optimierung von Kriterien

Wegen der zahlreichen am Markt vorhandenen Testautomatisierungstools, deren Zahl in der nahen Zukunft voraussichtlich weiterwachsen wird, werden die Ausschlusskriterien an Bedeutung gewinnen. Höchstwahrscheinlich werden viele technische Kriterien als Ausschlusskriterien aufgenommen, weil diese im Vergleich zu nichtfunktionalen Kriterien stärker standardisierbar sind. Die nichtfunktionalen Kriterien werden aber den Unterschied zwischen den evaluierenden Testautomatisierungstools definieren. Diese werden von verschiedenen Aspekten und Projekteigenschaften, wie z. B.: Wie viele QS-MitarbeiterInnen sind geplant, welches QS-Know-how ist vorhanden, welche Programmierkenntnisse haben die MitarbeiterInnen, Projektbudget, Laufzeit und Komplexität des Projekts usw., beschrieben. In einer weiterführenden Optimierung könnten diese Überlegungen und Erkenntnisse eingebunden werden.

5.2 Optimieren des Gewichtungsprozesses

Die Gewichtung von Kriterien spielt in der erstellten Methodik eine zentrale Rolle. Die Gewichtung von Kriterien wurde in dieser Arbeit mit Expertenwissen durchgeführt.

Die Ergebnisse von Raulamo-Jurvanen et al. deuten darauf hin, dass mindestens vier ExpertInnen befragt werden sollten, um ein zuverlässiges Ergebnis zu bekommen. Die Befragung von sieben oder mehr ExpertInnen könnte zu einem guten bzw. exzellenten Ergebnis führen [43]. Eine Empfehlung für die Gewichtung von Kriterien ist daher, so viele QS-MitarbeiterInnen, die am Projekt arbeiten, wie möglich miteinzubeziehen.

Die Abhängigkeit von Expertenwissen ist aus den vorherigen Überlegungen offensichtlich. Darüber hinaus gibt es weitere Faktoren, die eine optimale Gewichtung negativ beeinflussen könnten. Zum Beispiel könnte die gesamte Gewichtung der Kriterien, die sich auf den Preis beziehen, überproportional groß oder klein sein. Um diese und ähnliche Verzerrungen im Gewichtungsprozess zu minimalisieren, wäre ein formalisiertes Verfahren durchaus denkbar.

5.3 Erweiterbarkeit – Anpassungsfähigkeit

Mit der in dieser Bachelorarbeit konzipierten Methodik wurden vier ausgewählte TaTs, nämlich QF-Test, Selenium, TestCafe und TestCafe Studio, evaluiert. Unter Einbeziehung weiterer Testautomatisierungstools aus der Liste der Testautomatisierungstool-Kandidaten (Tabelle 8) würden die Ergebnisse für die bereits evaluierten Tools unberührt bleiben. Damit ist eine horizontale Erweiterung der Entscheidungsmatrix möglich. Die bis dahin in den Auswahlprozess investierten Aufwände blieben erhalten. Lediglich die neu aufgenommen Testautomatisierungstools müssten evaluiert werden.

Die Änderungen im Bereich Kriterien und Gewichtungen sind komplexer, aber möglich. Auch in diesem Fall bleiben die in den Prozess investierten Aufwände erhalten. Es gibt grundsätzlich zwei Szenarien, die zu einer vertikalen Änderung oder Erweiterung der Entscheidungsmatrix führen können:

- Bei eventuellen Gewichtungsverschiebungen werden neue Kriteriengewichtungen ermittelt und in die Entscheidungsmatrix eingetragen. Das neue Ergebnis entsteht automatisch ohne erneute Testautomatisierungstool-Evaluierung.

- Bei eventuell neuen Kriterien werden diese aufgenommen und die Gewichtungen angepasst. Der Anpassungsbedarf entsteht, weil die Summe von allen Gewichtungen weiterhin 100 % sein sollte. Die neuen Kriterien sollten für alle TaTs evaluiert werden.

6 Zusammenfassung und Ausblick

Die konzipierte Methodik zur Auswahl eines Testautomatisierungstools ermöglicht es, Testautomatisierungstools nach einem einheitlichen Schema zu bewerten und die Resultate zu vergleichen. Der Kriterienkatalog ist dazu geeignet, plausible und zutreffende Aussagen zu treffen, ob und wie weit ein Testautomatisierungstool im Kontext eines bestimmten Projekts einsetzbar ist. Die Evaluation erfolgte am Beispiel eines Software-Entwicklungsprojekts in der Immobilienverwaltung und hat gezeigt, dass der Ansatz in der Praxis anwendbar ist. Bei der konzipierten Methodik wird der beschriebene individuelle Kriterienkatalog für ein Projekt vom/von der ProjektleiterIn und/oder von erfahrenen QS-MitarbeiterInnen, die im Projekt involviert sind, erstellt. Die weitere Arbeit, die das Einholen der Informationen aus der grauen Literatur sowie die Evaluierung der Testautomatisierungstools aufgrund des individuellen Kriterienkataloges beinhaltet, kann von weiteren MitarbeiterInnen erfolgen, auch wenn diese eventuell nur geringe Erfahrung in der Testautomatisierung haben.

Diese Methodik ist vielseitig einsetzbar. Sie kann auf Projekte mit sehr unterschiedlichen Kontexten angewendet werden. Das Projekt kann eine Webanwendung, ein BI-, Mobile-App- oder IoT-Projekt usw. sein. Die im jeweiligen Projekt gestellten Anforderungen werden dementsprechend jeweils sehr unterschiedlich sein wie auch die daraus abgeleiteten Kriterien. Basierend auf diesen Kriterien wird mit in der Methodik beschriebenem Vorgehen ein Ergebnis erreicht. Dieses Ergebnis stellt die Empfehlung für jenes TaT dar, das die Kriterien im jeweiligen Kontext am meisten erfüllt.

Nach der Analyse der Entscheidungsmatrix war zu erkennen, dass die evaluierten TaTs die technischen Kriterien größtenteils erfüllen. Bei den etablierten Testautomatisierungstools stellt sich daher weniger die Frage, ob sie die technischen Kriterien erfüllen, sondern eher, ob sie den nichtfunktionalen Kriterien gerecht werden. Unter den nichtfunktionalen Kriterien, die vom Projekt an das Testautomatisierungstool gestellt werden, sind z. B. folgende zu nennen: Lizenzkosten, Usability, Erlernbarkeit usw. Daraus folgt eine mögliche Überlegung, die konzipierte Methodik zu verbessern, um ein genaueres, differenzierteres Auswahlergebnis zu bekommen. Der Weg dahin könnte über die Optimierung der Kriterienableitungen und über die Standardisierung der Kriteriengewichtungen führen.

7 Anhang – Standardkriterienkatalog zur Testautomatisierungstool-Auswahl

„Kriterienkatalog zur Testwerkzeugauswahl“ (Quelle: [2])

Kriterium	Frage	Erläuterung
Allgemeine Kriterien		
Werkzeugtyp		
Testprozesse	Entspricht das Werkzeug der Struktur der Testprozesse im Einsatzumfeld? Unterstützt das Werkzeug eine Phase des Testprozesses, in der es Mängel gibt?	Reine Testfallspezifikationstools können keine Testfallauswertung unterstützen.
Testmethodik	Unterstützt das Werkzeug Testmethoden, die schon im Einsatz sind?	Unterstützt das Werkzeug z.B. die Äquivalenzklassenmethode?
Testressourcen	Kann das Werkzeug vom derzeitigen Personal ohne größere Probleme eingesetzt werden oder sind massive Schulungen notwendig?	Für ein Werkzeug wären Programmierkenntnisse notwendig, diese sind aber im derzeitigen Testteam nicht vorhanden.
Overhead	Wie hoch ist der Zusatzaufwand durch den Einsatz des Tools? Wie sieht der Business Case aus?	Eine komplexe Testfallverwaltung für einen Testfallkatalog aus zehn Testfällen zu verwenden, erzeugt wohl mehr Aufwand, als damit eingespart wird. (Bisher: 5 Tage Test, jetzt: nur mehr 4 Tage Test, dafür 7 Tage Verwaltungsaufwand)
Integration in vorhandene Systemlandschaft	Wie kann sich die neue Komponente ins bestehende System integrieren? Datenaustausch?	Proprietäre Formate machen z.B. standardisierte Auswertungen schwer.
Evaluierungshilfen		
Demo- bzw. Evaluierungsversionen	Gibt es Evaluierungs- bzw. Demoversionen des Tools?	Die meisten Toolhersteller bieten kostenfreien Zugang zu zeitlich begrenzten Evaluierungsversionen ihrer Tools.
Tool-demonstration	Gibt es eine Toolpräsentation seitens des Herstellers?	Eventuell Live- oder Onlinepräsentation
Dokumentation während der Evaluierung	Wie viel der Online- und Gesamtdokumentation ist während der Evaluierung verfügbar?	Üblicherweise sind alle Dokumentationen und Supportdienstleistungen verfügbar.

→

Kriterium	Frage	Erläuterung
Allgemeine Kriterien		
Evaluierungshilfen (Fortsetzung)		
Evaluierung im Einsatzumfeld	Kann die Evaluierung vor Ort und am zu testenden Objekt stattfinden?	
Infrastruktur		
Systemlandschaft	Was für Clients und Server mit welcher Umgebung sind zum Einsatz des Tools vonnöten?	Die meisten kommerziellen Tools setzen auf Windows auf – wichtig, falls Plattform-unabhängigkeit gewünscht ist. Zumindest Server sollten im Allgemeinen plattformun-abhängig sein.
Anschaffungen	Sind Arbeitsplätze vorhanden oder müssen Anschaffungen getätigt werden? Wenn ja, wie viele für eine Evaluierungsphase bzw. für den Echteinsatz?	
Usability		
Benutzungs-oberfläche	GUI? Command-Line? Drag & Drop?	Oftmals soll der Fachbereich eingebunden werden; dafür kann eine intuitive Benutzungsoberfläche von Vorteil sein.
Wizards und Workflows	Sind die gängigsten Prozesse oder Aktionsserien in bequemen Schritt-für-Schritt-Aktivitäten vorbereitet?	Data Driving von aufgezeichneten Testskripten; finden von relevanten Plug-ins zur Testunterstützung gewisser Applikationen.
Reporting	Können die von dem Werkzeug generierten Dokumente direkt verwertet werden? Können Reports auch über definierte Teilaspekte generiert werden?	Testlogs von Automatisierungswerkzeugen sind oft schwer lesbar bzw. mit nicht direkt relevanter Information überladen. Der Umfang und Detailgrad der Protokolle sollen dem Bedarf entsprechen.
Customizing	Kann das Werkzeug den Ansprüchen gemäß konfiguriert werden? Expertenmodus?	Schriftgröße, Farbcodes usw.
Antwortzeit-verhalten	Hat das Werkzeug Antwortzeiten, die im guten oder erträglichen Bereich liegen?	Fehler- und Taskverwaltungssysteme mit vielen Usern haben oft hohe, von Benutzern als störend empfundene Reaktionszeiten.
Multiuser-Fähigkeit	Können die Daten zentral gespeichert und von mehreren Usern bearbeitet werden?	In größeren Projekten ist es oft notwendig, dass mehrere User gleichzeitig auf einen Datenstamm zugreifen und ihn bearbeiten.
Sprache des Tools	Ist das Werkzeug in einer der Nutzergruppe angemessenen Sprache verfügbar?	Deutsch, Englisch, weitere Sprachen

→

Kriterium	Frage	Erläuterung
Allgemeine Kriterien		
Skalierbarkeit/Stabilität		
Verlässlichkeit	Wie hoch ist das Risiko eines Ausfalls im realen Einsatz?	Ein System, das nicht verwendet werden kann, ist nicht brauchbar.
Belastbarkeit	Wie viele User können technisch am System arbeiten?	Multiuser-Tools haben oft eine praktisch begrenzte Anzahl an gleichzeitigen Usern.
Skalierbarkeit	Sollte das System erweitert werden (neue Komponenten, mehr User), wie viel Mehraufwand ist vonnöten, um diese Erweiterung durchzuführen?	Etwa Migration auf einen stärkeren Server
Dokumentation		
Aktualität	Ist die Dokumentation auf die neueste Version des Tools bezogen?	Zwischen Version 7.0 und 8.0 eines Tools kann sich viel verändert haben.
Vollständigkeit	Deckt die Dokumentation alle Aspekte des Tools ab?	Schwer zu findende Funktionen umfassen oft einen großen Teil der Funktionalität eines Tools.
Tutorials	Sind die Abläufe und Vorgehensweisen im Werkzeug durch illustrative Tutorials und Beispiele dargestellt?	Praktische Beispiele erleichtern oft den Umgang mit Tools ungemein.
Sprache der Dokumentation	Ist die Dokumentation in einer der Nutzergruppe angemessenen Sprache verfügbar?	Deutsch, Englisch, weitere Sprachen
Schulung		
Qualität der Schulung	Sind die angebotenen Schulungen vom Inhalt her so gestaltet, dass man danach mit dem Werkzeug sinnvoll arbeiten kann? Sind die Trainer entsprechend qualifiziert?	
Flexibilität des Schulungsunternehmens	Wie flexibel ist das Unternehmen, Schulungen in einer praxisnahen Umgebung abzuhalten?	Kann man davon ausgehen, dass nach einer Woche Schulung das Team fähig ist, die täglich gestellten Aufgaben selbstständig zu meistern? Oder ist nach der Schulung noch weiterer Support notwendig?
Qualität der Schulungsunterlagen	Sind die Schulungsunterlagen derart gestaltet, dass sie auch zu einem späteren Zeitpunkt als Referenz genutzt werden können?	Schulungsunterlagen bieten für viele User den direktesten Zugang zur Produktdokumentation.
Sprache der Schulung	Ist die Schulung in einer der Nutzergruppe angemessenen Sprache verfügbar?	Deutsch, Englisch, weitere Sprachen

→

Kriterium	Frage	Erläuterung
Allgemeine Kriterien		
Support/Wartung		
Qualität der Supportplattform	Gibt es Onlineformulare, Foren, Knowledge Bases? Wie lange ist die Reaktionszeit bei schweren/ leichten Problemen?	Knowledge Bases und Userforen bieten oft rasche Lösungen bei einfacheren Problemen. Bei schwereren ist dann guter Support gefragt.
Updatefrequenz und Aufwand	Wie oft kommen Updates heraus? Wie aufwendig ist es, auf eine neue Version umzusteigen?	Zu häufige Updates erzeugen Overhead oder werden ignoriert, zu seltene Updates bedeuten oft persistente Probleme.
Consultingteam/ Support/ Community	Wie viele Mitarbeiter unterstützen den Kunden? Vor Ort?	Viele größere Hersteller haben keine regionalen Niederlassungen, während lokale Hersteller ihr gesamtes Team in greifbarer Nähe haben.
Support alter Versionen	Werden alte Versionen weiterhin unterstützt?	Üblicherweise gibt es einige Jahre Support für ältere Versionen.
Sprache des Supports	Ist der Support in einer der Nutzergruppe angemessenen Sprache verfügbar?	Deutsch, Englisch, weitere Sprachen
Strategische Kriterien		
Standardsoftware	Ist das Werkzeug ein Standard-werkzeug im Unternehmen?	Eventuell gibt es Unternehmensstandards, die an bestimmte Anbieter gebunden sind.
Open-Source-Werkzeug	Ist das Werkzeug offen und frei?	In geschlossenen Systemen ist die Einarbeitung neuer Features dem Hersteller vorbehalten und damit von ihm abhängig.
Hersteller	Wie lange gibt es das Werkzeug schon am Markt? Wie erfolgreich ist der Hersteller? Ist das Anwendungsgebiet ein Schwerpunkt des Anbieters?	Beispielsweise ist zu prüfen, ob gewährleistet ist, dass der Toolhersteller den Support auch über die gesamte Anwendungsdauer durchführen kann.
Partnerschaften	Gibt es Partnerschaften mit relevanten Komponenten-herstellern?	Partnerschaften deuten oft auf guten Support gewisser Fremdhersteller-komponenten hin.
Referenzen		
Erfahrung im Umfeld	Wie viele vergleichbare Projekte wurden bisher durchgeführt?	Erfahrung im Umfeld hilft bei der Effizienz der Umsetzung und im Support.
Erfolg vergleichbarer Projekte/ externe Erfahrungsberichte	Wie sehen die Erfahrungsberichte oder Erfolgsquantifizierungen in vergangenen Projekten aus?	Wie gut sind vergleichbare Projekte bisher abgelaufen? Stolpersteine?

→

Kriterium	Frage	Erläuterung
Kriterien Testautomatisierungswerkzeuge		
Kompatibilität mit Testobjekt		
Grundlegende Architektur	Wird das zugrunde liegende System, auf dem die getestete Applikation läuft, unterstützt? Geschieht dies integriert oder über ein Add-in?	.NET, HTML, Java, VB, SAP usw.
Verwendete Bibliotheken und GUI-Elemente (Standard)	Werden die grundlegenden GUI-Elemente und Interaktionsmedien unterstützt? Geschieht dies integriert oder über ein Add-in?	Listen, Buttons, Terminal-Windows
Verwendete Bibliotheken und GUI-Elemente (komplex/Third Party)	Werden die erweiterten/selbstgestrickten/komplexeren GUI-Elemente unterstützt? Geschieht dies integriert oder über ein Add-in?	Grids, Logikbausteine, Drag & Drop Interfaces
Verwendete Kombinationen von Bibliotheken und GUI-Elementen	Können die notwendigen Unterstützungsmodule gleichzeitig aktiv sein?	Oftmals schließen sich Add-ins bzw. Komponentenunterstützungsmodule gegenseitig aus.
Zeichensätze und Sprachen	Werden Applikationen mit allen notwendigen Zeichensätzen und Sprachen unterstützt?	Griechisch, Kyrillisch, Chinesisch usw.
Data Interfaces		
Datenverwaltung	Können Testdaten direkt im Automatisierungswerkzeug gehalten und verwaltet werden?	
Datenbefüllung der Tests	Können Standarddatenbankverbindungen genutzt werden?	Eventuell wird die automatische Befüllung der Testfallparameter über eine Standarddatenbank durchgeführt.
Test der Datenbank der Applikation	Wird die Datenbankschnittstelle der Applikation unterstützt?	Automatischer Test legt einen Kunden an – ist dieser auch tatsächlich danach in der Datenbank?
Zugriffsart auf die Daten	Gibt es die Möglichkeit, SQL-Statements, Cursors usw. zu verwenden?	SQL & Co. machen den effizienten und einfachen Zugriff auf Daten möglich.
Programmierung		
Sprachstil	An welche gängige Sprache ist die Programmierung angelehnt?	Normalerweise dient eine Programmiersprache als Grundlage für das Automatisierungssystem.
Lesbarkeit	Wie lesbar und editierbar sind die Skripte für Nichtprogrammierer?	Einige Tools bieten anschauliche Strukturansichten, die auch ohne weitere Programmierkenntnisse Bearbeitungen ermöglichen.
Mächtigkeit der Sprache	Wie mächtig ist die Programmiersprache tatsächlich?	Klassen, Identifier, Exception Handling usw.

→

Kriterium	Frage	Erläuterung
Kriterien Testautomatisierungswerkzeuge		
Programmierung (Fortsetzung)		
Features der Entwicklungs-umgebung	Was für Features bietet die IDE?	»Go to Definition«, Autocomplete usw.
Struktur der Entwicklungs-umgebung und/oder Sprache	Werden Bibliotheken/wieder-verwertbare Module usw. grafisch aufbereitet?	Gibt es die Möglichkeit, strukturiert zu arbeiten (abseits von Tabulatoreinschüben)?
Objektidentifikation		
Dynamische Objekterkennung	Können Objekte zur Laufzeit generiert und erkannt werden?	Oft entstehen Objekte, die erkannt werden sollen, erst aus der Datensteuerung des Skripts (z.B. neu angelegte Kunden, die eigene Buttons oder Links in Webapplika-tionen erhalten). Dies bezeichnet die Fähigkeit, im Code zur Laufzeit Objekte zur Erkennung zu definieren.
Statische Objekt-erkennung	Gibt es ein Repository, in dem statischen Objekten logische Bezeichnungen zugeordnet werden können?	Logische Namen statt langer und wartungs-intensiver ID-Strings erleichtern die Über-sicht. Dies bezeichnet die Möglichkeit, Objekte statisch bzw. mit Wildcards in Objektlisten zur Verwendung im Code zu definieren. Dies ist bei unveränderlichen GUI-Komponenten hilfreich, wie zum Beispiel bei Standardmenüeinträgen (Speichern, Speichern unter, Schließen usw.).
Erkennungs-dimensionen und -logik	Wie funktioniert die Definition der Erkennungskriterien?	Vordefinitionen für gewisse Erkennungs-kriterien für Komponententypen, wie z. B., dass eine Liste an ihrem Label und ihrer Anzahl an Einträgen erkannt werden kann.
Erkennung von logischen Objekten beim Recording	Werden bereits in der statischen Map angelegte logische Objekte beim Recording als solche erkannt?	Das nachträgliche Ersetzen der logischen Objekte durch ihre statischen Gegenstücke ist oft mühsam.
Verwaltung/Integration		
Testfallverwaltung	Ist eine (brauchbare) Testfall-verwaltung integriert?	Es kann oft hilfreich sein, Testfallverwaltung und Testautomatisierung in einem Werkzeug zu haben, speziell für das Mapping von fachlichen Testfällen zu Skripten und Daten sowie zur Verwaltung von Abhängigkeiten.
Integration mit vorhandenen externen Testfall-verwaltungen	Wie gut läuft hier das Mapping der Testfälle zu Skripten ab? Wie werden die Ergebnisse integriert?	Hier geht es beispielsweise um das Starten von bestimmten Testfällen zu spezifischen Komponenten und die Integration der Ergebnisse.

→

Kriterium	Frage	Erläuterung
Kriterien Testautomatisierungswerkzeuge		
Verwaltung/Integration (Fortsetzung)		
Pipeline-Integration	Kann das Automatisierungswerkzeug einfach in eine CI/CD-Pipeline integriert werden?	Für nahtlose CI/CD-Prozesse ist es oft notwendig bzw. sehr hilfreich, Tests direkt in die damit verbundenen Pipelines einzubetten. Dazu gehört einerseits der Start der entsprechenden Testsuiten, aber auch das Auslesen von Testergebnissen und die Ablage von Berichten.
Kompatibilität mit skalierbarer Infrastruktur	Kann für die automatische Testdurchführung eine dynamische Infrastruktur (z.B. Docker) genutzt werden?	Einige Werkzeuge benötigen vollwertige & GUI-basierte Umgebungen (z.B. Windows Desktop) für die Durchführung von Tests. Diese sind schwer bzw. aufwendig und teuer in Cloud- oder hoch skalierbaren Infrastrukturen einzubetten. Zum Beispiel sind Headless-Durchführungen in Docker-basierten Infrastrukturen in vielen modernen Architekturen und Entwicklungsumgebungen deutlich einfacher umsetzbar.
Stabilität		
Exception Handling	Existiert Exception Handling?	Kann nach einem Fehler bzw. in einer unerwarteten Situation während des Testdurchlaufs darauf reagiert bzw. weitergemacht werden?
Automatisches Exception Handling	Existiert ein automatisches Exception Handling?	Einige Tools stellen z.B. für Webapplikationen gewisse automatische Mechanismen zur Wiederherstellung der Testumgebung zur Verfügung.
Umsetzung eines Zustandsmodells	Kann ein funktionierendes Zustandsmodell umgesetzt werden?	Einige Tools haben bereits ein Zustandsmodell integriert, in anderen kann dies leicht umgesetzt werden.
Reaktionen auf Probleme des Tools selbst	Gibt es einen externen Testtreiber, der auch das Werkzeug neu starten kann?	Sollte das Automatisierungswerkzeug selbst Probleme bereiten, kann in manchen Testautomatisierungssuiten auch dieses neu gestartet werden.
Bibliotheken/Out-of-the-box-Lösungen		
Zugriff auf externe Bibliotheken und Schnittstellen	Können externe Standardbibliotheken verwendet werden?	Zugriff auf externe Bibliotheken und Schnittstellen erleichtert das Leben oft ungemein, z.B. beim Beenden eines Prozesses.
Zugriff durch externe Programme	Können externe Prozesse das Werkzeug ansteuern? Welche? Wie geschieht dies?	Integration in Build-Prozess, Scheduler, Debugger, Frontends usw.

→

Kriterium	Frage	Erläuterung
Kriterien Testautomatisierungswerkzeuge		
Bibliotheken/Out-of-the-box-Lösungen (Fortsetzung)		
Interne Bibliotheken	Werden für die wichtigsten Aktionen und Operationen im Test entsprechend komfortabel nutzbare Funktionalitäten mitgeliefert?	Tools unterstützen oft die einfachsten Operationen, wie Substring-Replacement, nicht out-of-the-box. Wichtige Features im Test sind Stringoperationen, Zeit- und Datumsoperationen, Berechnungen, Rundungen usw.
Benutzer-definierte Bibliotheken	Wie effizient kann selbst geschriebener Code ausgeführt werden?	Beispielsweise werden gewisse benutzer-erstellte Funktionen zur effizienteren Ausführung kompiliert.
Community-Bibliotheken	Wie sieht die Unterstützung durch die Community im Bibliotheks-bereich aus?	Einige Tools haben eine rege Community, in der auch viele selbst gestrickte Bibliotheken freigegeben werden. Auch kostenpflichtige Bibliotheken sind erhältlich.
Dokumentation		
Kommentare und Tags	Können bequem Kommentare, Tags usw. Skripten, Files oder Blöcken zugewiesen werden?	Gute Kommentierung ist bei jeglicher Entwicklung wichtig.
Automatische Kommentierung	Werden sinnvolle Kommentare automatisch ergänzt, wenn Funktionalität aufgezeichnet wird oder Wizards verwendet werden?	Verschafft beim nachträglichen Bearbeiten mehr Klarheit.
Nachvollzieh-barkeit	Kann man am Code direkt erkennen, »was denn nun eigentlich geschieht«?	Bei aufgezeichneten Statements kommen z.B. Screenshots hinzu.
Logging		
Testdurchlauflogs	Sind die internen Logs in irgendeiner Struktur enthalten?	Baumstrukturen usw. sorgen in den oft sehr vollgepackten Logs für mehr Übersicht.
Pass und Fail Flags	Können Negativtests bei Fehler-meldungen als »Passed« markiert werden?	Einige Tools markieren bei Fehlermeldungen den Testfall automatisch als »Failed«. Nicht gut für Negativtests.
Nachvollzieh-barkeit	Kann man im Log direkt erkennen, »was denn nun eigentlich geschieht«?	Zum Beispiel werden bei jeder ausgeführten Aktion bzw. bei jedem Check Screenshots hinzugefügt.
Recording		
Recording-Modi	Können nur logische Aktionen oder nur analoge Aktionen aufgezeichnet werden? Beides?	Logisch: »Hauptfenster.click«. Analog: »Click(180,123)«, also auf Koordinaten. Oft ist es gut, zur Sicherheit beide Möglichkeiten zu haben, auch wenn erstere Variante weit wichtiger ist (z.B. wenn vereinzelt Objekte nicht unterstützt bzw. erkannt werden).

→

Kriterium	Frage	Erläuterung
Kriterien Testautomatisierungswerkzeuge		
Recording (Fortsetzung)		
Erkennung aller logischen Aktionen	Werden z.B. in unterstützten Grids die Zugriffe auch per Zellen-ID erkannt?	Oft zeichnen Tools bei Objekten, die per Add-in identifiziert werden, nur Analog-informationen (also Koordinaten, Tasten-drücke) auf.
Checking		
Kriterien	Welche Kriterien können für Checks herangezogen werden?	Können z.B. auch nicht als Erkennungs-kriterien definierte Attribute (Tabellen-inhalte, Screenshot-Segmente usw.) als Checkkriterien definiert werden?
Recording und Check in einem Durchlauf	Muss zuerst aufgezeichnet werden und dann werden Checks einge-führt?	Kann z.B. zwischen Recording- und Check-modus hin- und hergesprungen werden, ist die Aufzeichnung eines Testfalls in einem Durchlauf möglich.
Trennung Check-container und Checkdaten	Können Checks dynamisch mit Daten versorgt werden?	Viele Programme speichern die bei den Checks erwarteten Werte statisch ab. Datengesteuerte Tests funktionieren aber eleganter, wenn diese Werte dynamisch auch aus den Daten gelesen werden können.
Aufbereitung »Soll vs. Ist«	Können die Ergebnisse von fehl-geschlagenen Checks anschaulich angezeigt werden?	Checks auf Tabellen: Viele Werte, nur wenige davon verschieden. Verschiedene Werte sollten deutlich sichtbar markiert werden.
Automatisches Update von Checkdaten	Können Checkdaten automatisch aktualisiert werden?	Beispielsweise wird ein »Update«-Modus unterstützt, indem eine Ausführung des Skripts die Sollwerte auf die nun erhaltenen Werte stellt.
Events		
Kriterien	Können Events nach bestimmten Kriterien ausgelöst werden?	Wenn gewisse Kriterien erfüllt sind, wird ein Event ausgelöst, auf das gewartet bzw. reagiert werden kann. Eignet sich z.B. für Recovery oder Synchronisierung.
Asynchronität	Können Events auch asynchron Codeteile triggern?	Einige Tools verwenden Events als »warte auf etwas« zur Synchronisation, lassen aber asynchrones Verwerten der Eventmasken nicht zu.
Files und Versionskontrolle		
Fileformate	Welche Formate werden verwendet?	Textfiles? XML? JPG? Datenbank? Für Maps, Skripte, Screenshots, Reports usw.
Speicherplatz	Wie viel Platz brauchen Skripte/Logs/Reports?	Beim Debugging usw. entstehen oft Unmengen an Logs und Skriptversionen, die das Repository anschwellen lassen.

→

Kriterium	Frage	Erläuterung
Kriterien Testautomatisierungswerkzeuge		
Files und Versionskontrolle (Fortsetzung)		
Versions-management	Gibt es ein integriertes Versions-management? Wie gut ist es?	Manche Tools haben ein integriertes Versionsmanagement, verwenden aber eine Datenbank zum Speichern der Komponenten, was ein externes Versions-management schwer macht. Andere haben keines und erschweren durch ihren Datei-aufbau die Verwendung eines externen Versionsmanagements.
Externes Versions-management	Wie leicht ist es, die Testauto-matisierung in ein externes Versionsmanagement zu integrieren (CVS[a], SVN[b])?	
Parallelität		
Testbarkeit	Gibt es eine Möglichkeit, parallel ablaufende Prozesse zu testen?	Mehrere Instanzen des Testtools, Möglich-keit, parallel Code auszuführen, usw.
Ausführung	Ist es möglich, parallelisierten Code in einer Instanz des Tools zu erstellen?	Direkte Parallelität in den Testskripten, nur eine Instanz des Werkzeugs notwendig.
Parallele Test-durchführung	Können mehrere Tests parallel ausgeführt werden?	Dies kann die Durchführungsdauer deutlich reduzieren, erfordert aber Aufmerksamkeit bei der Implementierung und Gruppierung der Testsuite und der Testdaten.
Kosten und Lizenz		
Kosten-adäquatheit	Stehen die Kosten des Tools in Relation zur Projektgröße bzw. zu dem zu erwartenden Nutzen?	Eine Applikation, deren Entwicklung 500 € nicht übersteigen soll, mit einem Testwerk-zeug testen, das im Jahr 10.000 € kostet?
Kosten in der Evaluierungs-phase	Gibt es eine kostenfreie Evaluierungslizenz? Eine günstige?	
Kosten in der Anschaffung	Wie ist der Basispreis des Tools?	Direkte Anschaffungskosten der Software exkl. Userlizenzen, Wartung und Support
Supportkosten	Wie hoch sind die Kosten für den Support des Tools?	Kosten nur Support
Schulungs-, Consulting- und Coaching-Kosten	Was kostet eine adäquate Ausbildung der zuständigen Personen? (Trainer-Tagessatz)	Vor allem in der Testautomatisierung ist oft Zusatzwissen gefordert.
Wartungs- und Updatekosten	Wie hoch sind die Kosten bei Point-Releases? Major Releases?	Bei manchen Herstellern wird eine Lizenz für das Werkzeug gekauft: Sowohl Point-Releases als auch volle Updates sind inkludiert.

a. CVS: Concurrent Versions System.
b. SVN: Apache Subversion.

→

Kriterium	Frage	Erläuterung
Finanzielle Kriterien		
Kosten und Lizenz (Fortsetzung)		
Lizenztypen	Gibt es Floating-Lizenzen? Per-Seat-Lizenzen? Preise?	
Lizenzverwaltung	Muss ein eigener Lizenzserver angeschafft werden?	Viele Toolanbieter bieten Versionen sowohl als Stand-alone als auch als Client-Lizenz-server-Variante an.
Infrastruktur	Muss ein eigener Lizenzserver angeschafft werden? Andere infrastrukturelle Kosten?	Sämtliche Applikationen, die Floating-Lizenzen voraussetzen.
Skalierbares Lizenzmodell	Wie skalieren die Lizenzkosten bei parallelisierten Testdurchläufen oder dynamischer Skalierung der Infrastruktur?	Wenn z.B. zur Parallelisierung zwanzig Docker-Instanzen mit Werkzeug und Tests gestartet werden, können unter Umständen auch zwanzig Lizenzen notwendig sein. Freie Skalierbarkeit, günstige Durchführungslizenzen oder Pay-per-Use-Modelle können hier kostensparend wirken.

8 Literaturverzeichnis

[1] Atlassian Jira, Jira Software, URL: https://confluence.atlassian.com/jirasoftwareserver085/jira-software-server-8-5-documentation-981155994.html

[2] Baumgartner, Manfred; Gwihs, Stefan; Seidl, Richard; Steirer, Thomas; Wendland, MarcFlorian (2021): Basiswissen Testautomatisierung. Aus- und Weiterbildung zum ISTQB® Advanced Level Specialist – Certified Test Automation Engineer, 3., aktualisierte und überarbeitete Auflage, Heidelberg: dpunkt.verlag

[3] Baumgartner, Manfred; Klonk, Martin; Pichler, Helmut; Seidl, Richard; Tanczos, Siegfried (2018): Agile Testing, München: Carl Hanser Verlag

[4] Benzies, Karen M.; Premji, Shahirose; Hayden, K. Alix; Serrett, Karen (2006): State-of-the-Evidence Reviews: Advantages and Challenges of Including Grey Literature, In: World views on Evidence-Based Nursing, Wiley Online Library, Volume 3, Issue 2, 55–61

[5] Bucsics, Thomas; Baumgartner, Manfred; Seidl, Richard; Gwihs, Stefan (2015): Basiswissen Testautomatisierung. Konzepte, Methoden und Techniken. 2. Auflage, Heidelberg: dpunkt.verlag

[6] Capgemini (2015): world-quality-report-2015-16, URL: https://www.capgemini.com/resources/world-quality-report-2015-16-germany/

[7] Cerioli, Maura; Leotta, Maurizio; Ricca, Filippo (2020): What 5 Million Job Advertisements Tell Us about Testing: a Preliminary Empirical Investigation, In: The 35th ACM/SIGAPP Symposium on Applied Computing (SAC '20) Brno, Czech Republic. ACM, New York, NY, USA, 1586–1594

[8] Component Source, Anwendungen URL: https://www.componentsource.com/de/product/testcafe-studio/releases

[9] Devada, DZone Developer Community URL: https://dzone.com/articles/testcafe-e2e-testing-tool

[10] DockerHub, Images, Selenium, URL: https://hub.docker.com/r/selenium/standalone-chrome

[11] DockerHub, Images, TestCafe, URL: https://hub.docker.com/r/testcafe/testcafe

[12] Döring, Nicola; Bortz, Jürgen (2016): Forschungsmethoden und evalution, 5. Auflage, Berlin, Heidelberg: Springer

[13] embarc GmbH, Zörner, Stefan (2018): „Was ist eigentlich Architekturbewertung?", Informatik Aktuell, URL: https://www.informatik aktuell.de/entwicklung/methoden/was-ist-eigentlich-architekturbewertung.html#c27123

[14] Fowler, Martin (2013): Continuous Delivery, https://martinfowler.com/bliki/ContinuousDelivery.html

[15] Garousi, Vahid; Mäntylä, Mika V. (2016): When and what to automate in software testing? A multi-vocal literature review. In: Information and Software Technology 76, 92–117

[16] Garousi, Vahid; Felderer, Michael; Mäntylä, Mika V. (2019): Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. In: Information and Software Technology 106, 101–121

[17] Gartner: Test Automation Reviews and Ratings, URL: https://www.gartner.com/reviews/market/software-test-automation

[18] git Verwaltungskontrollsystem, URL: https://git-scm.com/

[19] Heidilyn V. Gamido, Marlon V. Gamido (2019): Comparative review of the features of automated software testing tools, In: International Journal of Electrical and Computer Engineering (IJECE), Vol. 9, No. 5, 4473–4478

[20] Hooda, Itti, & Chhillar, Rajender Singh (2015): Software test process, testing types and techniques. International Journal of Computer Applications, 111(13), 10–14

[21] ISTQB® International Software Testing Qualifications Board, URL: https://www.istqb.org/

[22] ISTQB® Glossary, URL: https://glossary.istqb.org/app/de/search/

[23] ISTQB® (2016): Worldwide software testing practices report 2015–2016, URL: https://www.istqb.org/references/surveys/istqb-worldwide-software-testing-practices-report-2015-2016.html

[24] iSYS Software GmbH, URL: https://www.isys.de/

[25] Jenkins Pipeline, URL: https://www.jenkins.io/doc/book/pipeline/

[26] Jenkins Continuous Integration System, URL: https://www.jenkins.io/

[27] Kaur, Harpreet; Gupta, Gagan (2013): Comparative Study of Automated Testing Tools: Selenium, Quick Test Professional and Testcomplete, In: Int. Journal of Engineering Research and Applications, Vol. 3, Issue 5, 1739–1743

[28] Katalon (2020): Test Automation Landscape in 2020, URL: https://www.katalon.com/test-automation-landscape-2020/

[29] Kleuker, Stephan (2019): Qualitätssicherung durch Softwaretests, 2. erweiterte und aktualisierte Auflage, Wiesbaden: Springer Vieweg

[30] Laux, Helmut; Gillenkirch, Robert M.; Schenk-Mathes, Heike Y. (2018): Entscheidungstheorie, Berlin, Heidelberg: Springer

[31] Leotta, M.; Stocco, A.; Ricca, F.; Tonella, P. (2015): Using Multi-Locators to Increase the Robustness of Web Test Cases. In: 2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST), 1–10

[32] Liggesmeyer, Peter (2009): Software-Qualität; Testen, Analysieren und Verifizieren von Software, 2. Auflage, Heidelberg: Spektrum Akademischer Verlag

[33] Maslow, Abraham Harold (1966): The Psychology of Science: A Reconnaissance, New York: Maurice Bassett Publishing

[34] Open JS Fondation, Mocha, URL: https://openjsf.org/projects/

[35] Passos, Carol; Braun, Ana Paula; Cruzes, Daniela S.; Mendonca, Manoel (2011): Analyzing the Impact of Beliefs in Software Project Practices, International Symposium on Empirical Software Engineering and Measurement, 444–452

[36] Qcentric, Die besten Tools für die Testautomatisierung (2021), URL: https://q-centric.com/blogs/die-besten-tools-fuer-die-testautomatisierung-2021

[37] QFS GmbH, QF-Test, Preise, URL: https://www.qfs.de/produkt/preise.html

[38] QFS GmbH, QF-Test, Produkt URL: https://www.qfs.de/produkt/qf-test.html

[39] QFS GmbH, QF-Test, Checkliste, URL: https://www.qfs.de/fileadmin/Web-data/pdf/checkliste-qf-test.pdf

[40] QFS GmbH, QF-Test, Fallstudien, https://www.qfs.de/unternehmen/referenzen/fall-studien/janitza.html

[41] Raulamo-Jurvanen, Päivi (2017): Decision Support for Selecting Tools for Software Test Automation. In: *SIGSOFT* Softw. Eng. Notes 41 (6), 1–5

[42] Raulamo-Jurvanen, Päivi; Mäntylä, Mika; Garousi, Vahid (2017): Choosing the Right Test Automation Tool, EASE '17: Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering, 21–30

[43] Raulamo-Jurvanen Päivi; Hosio, Simo; Mäntylä, Mika V. (2019): Practitioner Evaluations on Software Testing Tools, EASE '19: Proceedings of the Evaluation and Assessment on Software Engineering, 57–66

[44] Ricca, Filippo; Stocco, Andrea (2021): Web Test Automation: Insights from the Grey Literature. In: Tomás Bures, Riccardo Dondi, Johann Gamper, Giovanna Guerrini, Tomasz Jurdziński, Claus Pahl et al. (ed.): SOFSEM 2021. 47th International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM 2021, Bolzano-Bozen, Italy, January 25–29, 2021, Proceedings. Cham, 2021. Cham: Springer International Publishing AG (Lecture Notes in Computer Science Ser, v.12607), 472–485

[45] SIMPLYTEST GmbH, testautomatisierung.org, Mocha, URL: https://www.testautomatisierung.org/lexikon/mocha-testing/

[46] Selenium, Selenium Projekts URL: https://www.selenium.dev/projects/

[47] Selenium, Download, URL: https://www.selenium.dev/downloads/

[48] Selenium, Locating elements, URL: https://www.selenium.dev/documentation/en/webdriver/locating_elements/

[49] Selenium, Understanding the components, URL: https://www.selenium.dev/documentation/de/webdriver/understanding_the_components/

[50] Selenium, Waits, URL: https://www.selenium.dev/documentation/en/webdriver/waits/

[51] Söllner, Dierk (2017): DevOps in der Praxis – Handlungsfelder für eine erfolgreiche Zusammenarbeit von Entwicklung und Betrieb. HMD 54, 189–204

[52] Testautomatisierung gewusst wie (2019), URL: https://testautomatisierung-gewusst-wie.de/testautomatisierung-von-webanwendungen-mit-selenium/

[53] TestCafe, URL: https://testcafe.io/

[54] TestCafe, Install TestCafe, URL: https://testcafe.io/documentation/402834/guides/basic-guides/install-testcafe

[55] TestCafe, FAQ, URL: https://testcafe.io/402636/faq#i-have-heard-that-testcafe-does-not-use-selenium-how-does-it-operate

[56] TestCafe, Release Notes, URL: https://testcafe.io/release-notes

[57] TestCafe Studio, URL: https://www.devexpress.com/products/testcafestudio/

[58] TEQS Consulting GmbH, Testautomatisierung-Tools, URL: https://www.testing-board.com/testautomatisierung-tools/

[59] The Forrester Wave ™: Continuous Functional Test Automation Suites, Q2 2020 (2020), URL: https://go.forrester.com/blogs/the-forrester-wave-continuous-functional-test-automation-suites-q2-2020-is-live/

[60] Weiss, Johannes; Schill, Alexander; Richter, Ingo; Mandl, Peter (2016): Literature Review of Empirical Research Studies within the Domain of Acceptance Testing, In: Software Engineering and Advanced Applications (SEAA), 42th Euromicro Conference on IEEE, 181–188

Alle URLs wurden zuletzt am 24.07.2021 geprüft.

9 Tabellenverzeichnis

10 Abbildungsverzeichnis