Vergleich verschiedener Tools zur automatischen Durchführung von Testfällen für unterschiedliche Programme innerhalb von Verbundtests

Hochschule für Telekommunikation Leipzig
University of Applied Sciences

Vorgelegt von:
Robert Schnorr

Matrikel-Nummer: 133135
geboren am 18. Juni 1993 in Geldern
eingereicht am: 16. November 2016

Erstprüfer: Prof. Dr.-Ing. Sabine Wieland

Zweitprüfer: Frau Nicole Weyen

Hochschule für Telekommunikation Leipzig (FH)
Kommunikations- und Medieninformatik (dual)

Abschlussarbeit zur Erlangung des akademischen Grades
Bachelor of Engineering

Kurzfassung

Die vorliegende Bachelorarbeit befasst sich mit dem Vergleich verschiedener Tools zur automatischen Durchführung von Testfällen für unterschiedliche Programme innerhalb von Verbundtests.
Dabei wird zunächst darauf eingegangen, welche Vorteile ein Softwaretest mit sich führt und welche Ansätze zur Automatisierung solcher Softwaretest bereits bestehen. Aus Zeitgründen liegt der Schwerpunkt dieser Arbeit auf der Verwendung von Automatisierungstools für GUI-basierte Verbundtests.
Außerdem beschränkt sie sich auf Automatisierungstools, welche auf dem Betriebssystem Microsoft Windows 7 ausgeführt werden können. Diese Arbeit richtet sich an Softwareunternehmen, welche einen Verbund mehrerer Applikationen mit unterschiedlichen GUI-Technologien automatisiert testen möchten.
Als Referenzumgebung für Untersuchungen gilt dabei die Systemverbundtestumgebung „AD/CS“ des T-Systems Softwareprojektes „AD“ (Außendienstdisposition der deutschen Telekom).

Im Vorfeld dieser Untersuchungen steht eine generelle Betrachtung der auf dem Markt verfügbaren Automatisierungstools, unabhängig davon ob Sie dem zuvor genannten Schwerpunkt entsprechen.
Auf Basis dieser Betrachtung folgt ein Vergleich von drei bis vier - dem Schwerpunkt entsprechenden - Tools.
Zur Einschränkung der Auswahl werden die Kriterien unterstützte Technologien, benötigte Infrastruktur, Nutzerfreundlichkeit, Skalierbarkeit, Hilfe und Support, Integration und Anbindung als auch Kosten und Lizenz herangezogen.
Zusätzlich erfolgt die Bewertung auf Basis von Anforderungen der Referenzumgebung.
Der Fokus liegt auf Untersuchungen zur folgenden These: „Es gibt keine geeignete Software zur Testautomatisierung, welche alle Technologien und Anforderungen eines Verbundtests in vollem Umfang erfüllen könnte, und noch schneller arbeitet als ein menschlicher Tester“.
Zur Beantwortung dieser These werden die Kriterien Installationsaufwand, Ausführbarkeit von Testfällen, Fehleranfälligkeit der Software, Zeitdauer im Vergleich zum manuellen

Testen, Import-/Exportmöglichkeiten der Software und Verwendbarkeit für Tester ohne Programmierkenntnisse herangezogen.
Außerdem werden, zur Findung einer Empfehlung an die Zielgruppe, verschiedene Sichtweisen von Personen betrachtet, die an einem Softwaretest beteiligt sind.

Im Rahmen des Vergleichs kann die These durch die Software HP Unified Functional Testing (UFT) als erfüllt angesehen werden. Generell überzeugten die anderen untersuchten Automatisierungstools mit einigen Einschränkungen.
Einige der Tools zeigten sich als fehleranfällig, nur für Programmierer geeignet oder nur für bestimmte Technologien geeignet.

Robert Schnorr
November 2016

Abstract

This thesis deals with the comparison of different tools for automatic execution of test cases for different programs within integration tests. Therefore, I lift up the benefits of a software test and which kinds of such automated software test already exist.
For reasons of time, I set the main focus of this thesis on the use of automation tools for GUI-based integration tests. In addition, I confine myself to automation tools that are based on the Microsoft Windows 7 operating system.
My thesis is dedicated to software companies, that want to test several applications with different GUI technologies within a integration test automatically.
A reference environment for my studies is the system-integration-test-environment “AD/CS“ of the T-Systems Software Project “AD“ (External Service disposition of Deutsche Telekom).

In advance of these investigations is a general consideration of available automation tools, regardless of they are fitting my previously mentioned conditions.
On the basis of this consideration, a comparison of three to four Tools - corresponding to my focus - is following.
To limit my selection, I prefer the criteria: supported technologies, required infrastructure, user-friendliness, scalability, help and support, integration and connection and also the cost and license.
In addition, I selected the tools on the basis of the requirements of my reference environment.
I dedicated to spent my investigations on the following thesis: “There is no suitable software for test automation, which supports all technologies and requirements of a intergration test and still works faster than a human tester“.
To answer this thesis I investigated the criteria: installation, executability of test cases, susceptibility to error, time of automatic testing in comparison to the manual testing, import/export possibilities and usability for testers without programming knowledge.
Moreover, I consider to find a recommendation to my target group and also to different views of in a software test involved persons.

According to my comparison, I come to the conclusion that the software HP Unified functional testing (UFT) fits to my thesis.
In general, the other investigated automation tools also convinced, but with some restrictions. Some tools were error-prone, others only usable for programmers or only for certain technologies.

Robert Schnorr
November 2016

Inhaltsverzeichnis

Abbildungsverzeichnis **ix**

Tabellenverzeichnis **xi**

Abkürzungsverzeichnis **xiii**

1 Einführung **1**

1.1 Ausgangssituation und Problemstellung 1

1.1.1 Organisatorische Einordnung des Musterverbundes 1

1.1.2 Problemstellung 2

1.1.3 Theoretischer Bezug 3

1.1.4 Historie des Testens 5

1.2 These der Arbeit 6

1.3 Abgrenzung der Themenstellung 7

1.4 Die Zielgruppe 8

2 Hauptteil **9**

2.1 Kriterien für die Untersuchung 9

2.1.1 Kriterien für Automatisierungs-Produkte 9

2.1.2 Kommerzielle Produkte vs. Open-Source-Produkte 25

2.2 Anforderungen an die Software für den Verbundtest 26

2.3 Kriterien für die Untersuchung 28

2.4 Auswahl der Software 28

2.5 Untersuchung von Software für Testautomatisierung 29

2.5.1 Installationsaufwand 29

2.5.2 Erstellung eines ersten Testfalls 31

2.6 Mögliche Testfallabdeckung 39

2.6.1 Durchführbarkeit von Testfällen 40

2.6.2 Zeitdauer im Vergleich zum manuellen Testen 50

2.6.3 Fehleranfälligkeit der Software 52
2.6.4 Import-/Exportmöglichkeiten der Software 56
2.7 Verwendbarkeit für Tester ohne Programmierkenntnisse 58

3 Schlussteil 61
3.1 Zusammenfassung der Untersuchungsergebnisse 61
3.1.1 Untersuchte Kriterien nach Vorauswahl der Software 61
3.2 Empfehlungen an die Zielgruppe 64
3.2.1 Allgemein 64
3.2.2 Ohne Programmierkenntnisse 66
3.2.3 Mit Programmierkenntnissen 67
3.2.4 Webbasierte GUIs 68
3.2.5 Geringes Budget 69
3.3 Fazit und Ausblick 70

Literaturverzeichnis 73

4 Selbstständigkeitserklärung 81

Anhang A Glossar 83

Anhang B Installation der Softwarewerkzeuge 85
B.1 Ranorex Studio 85
B.2 Telerik Studio 94
B.3 SikuliX 102
B.4 HP Unified Functional Testing (UFT) 103

Anhang C Kommerzielle Testsoftware 113

Abbildungsverzeichnis

1.1 Testverbund AD 2
1.2 Software-Lebenszyklus 6

2.1 geladener CS-Web-Client 26
2.2 geladener Auftrag im AD-Client 27
2.3 Startansicht von Ranorex Studio 31
2.4 Startansicht von Telerik Test Studio 33
2.5 Startansicht von SikuliX 35
2.6 Startansicht der HP UFT 37
2.7 geladener Auftrag im AD-Client 41
2.8 Test des AD-Clients mit SikuliX 42
2.9 disponierter Auftrag im CS-Client 44
2.10 Test des CS-Clients im Telerik Test Studio 45
2.11 Test des AD/CS-Verbundes im Ranorex Studio 47
2.12 Test des AD/CS-Verbundes im HP UFT 48
2.13 HP UFT stoppt aufgrund eines Fehlers 53
2.14 Ranorex Studio stoppt aufgrund eines Fehlers 54
2.15 SikuliX stoppt aufgrund eines Fehlers 55
2.16 Import: Data Driven Testing im HP UFT 57
2.17 HP UFT Test-Flow 59

B.1 Installation des Ranorex Studios (1) 85
B.2 Installation des Ranorex Studios (2) 86
B.3 Installation des Ranorex Studios (3) 87
B.4 Installation des Ranorex Studios (4) 88
B.5 Installation des Ranorex Studios (5) 89
B.6 Installation des Ranorex Studios (6) 90
B.7 Installation des Ranorex Studios (7) 91

B.8 Installation des Ranorex Studios (8) 92
B.9 Installation des Ranorex Studios (9) 92
B.10 Installation des Ranorex Studios (10) 93
B.11 Installation des Ranorex Studios (11) 93
B.12 Installation des Telerik Test Studios (1) 94
B.13 Installation des Telerik Test Studios (2) 95
B.14 Installation des Telerik Test Studios (3) 96
B.15 Installation des Telerik Test Studios (4) 97
B.16 Installation des Telerik Test Studios (5) 98
B.17 Installation des Telerik Test Studios (6) 99
B.18 Installation des Telerik Test Studios (7) 100
B.19 Installation des Telerik Test Studios (8) 101
B.20 Installation von SikuliX 102
B.21 Installation der HP UFT (1) 103
B.22 Installation der HP UFT (2) 104
B.23 Installation der HP UFT (3) 105
B.24 Installation der HP UFT (4) 106
B.25 Installation der HP UFT (5) 107
B.26 Installation der HP UFT (6) 108
B.27 Installation der HP UFT (7) 109
B.28 Installation der HP UFT (8) 110
B.29 Installation der HP UFT (9) 111

Tabellenverzeichnis

1.1 Perioden des Softwaretests 5

2.1 Punkteverteilung 10
2.2 Kriterium: Typ/Technologie 11
2.3 Kriterium: Benötigte Infrastruktur 13
2.4 Kriterium: Nutzerfreundlichkeit 15
2.5 Kriterium: Skalierbarkeit 17
2.6 Kriterium: Hilfe/Support 19
2.7 Kriterium: Integration/Anbindung 21
2.8 Kriterium: Kosten und Lizenz 23
2.9 Software-Gesamtbewertung nach Punkten 28
2.10 Installationsaufwand 30
2.11 Test der Software anhand des AD-Clients 43
2.12 Test der Software anhand des CS-Clients 46
2.13 Test des Software-Verbundes AD/CS 49
2.14 Zeitdauer im Vergleich zum manuellen Testen 50
2.15 Fehleranfälligkeit der Software 52
2.16 Import-/Exportmöglichkeiten der Software 56
2.17 Verwendbarkeit ohne Programmierkenntnisse 58

3.1 Gesamt-Punkteverteilung 62
3.2 Pro & Kontra für HP UFT 65
3.3 Pro & Kontra für Ranorex Studio 66
3.4 Pro & Kontra für Telerik Test Studio 68
3.5 Pro & Kontra für SikuliX 69

Abkürzungsverzeichnis

AD Projekt Außendienst Client/Server

MA Mitarbeiter Außendienst

Tk Technischer Kundendienst (DTTS)

DTTS Deutsche Telekom Technischer Service

ANTK Auftragnehmeranbindung Technischer Kundendienst

WMSTK Workflow Management System Technischer Kundendienst

Persdispo Personaldisposition Technischer Kundendienst

CS ClickSchedule

Tesy Termin Erinnerungs System (Kundenaufträge)

Smile Service, Montage, Information und Lenkung (Kundenaufträge)

Tel IT Telekom IT

PersDispo-TK PersonalDisposition Technischer Kundendienst

KVWS Kapazitätsverwaltungssystem der Arbeitszeiten für TK MA

ERP Enterprise-Resource-Planning

bzw. beziehungsweise

z. B. zum Beispiel

div. diverse

UFT Unified Functional Testing

Gantt Instrument des Managements, das die zeitliche Abfolge von Aktivitäten grafisch in Form von Balken auf einer Zeitachse darstellt

(G)UI (grafische) Benutzeroberfläche

ALM Application lifecycle management / Kombination aus der Entwicklung und Betreuung von Applikationen

DOM Document Object Model

vollumfänglich testbar Bedeutung hier: Alle für eine Software spezifizierbaren Testfälle können auch getestet werden. Im Sinne einer Automatisierung kann Software vollumfänglich/im vollen Umfang testen, wenn diese alle Testfälle ausführen kann, die auch ein menschlicher Tester „ausführen“ (testen) könnte.

Keyword Driven Begriffserklärung siehe https://www.stickyminds.com/article/keyword-driven-testing

Software-Lebenszyklus Begriffserklärung siehe http://de.ccm.net/contents/574-software-lebenszyklus

An dieser Stelle möchte ich all jenen danken, die durch ihre Unterstützung zum Gelingen dieser Bachelorarbeit beigetragen haben. Diese Bachelorarbeit ist im Rahmen des Abschlusses meines Studiums Kommunikations- und Medieninformatik an der Hochschule für Telekommunikation in Leipzig entstanden. Ich habe diese Bachelorarbeit kooperativ im Rahmen meiner Arbeit im Systemtest des Projektes AD bei der T-Systems in Essen geschrieben. Die Durchführung meiner Untersuchungen war teilweise beschwerlich, aber dennoch war ich nach ausführlichen Auswertungen in der Lage, meine Leitfrage zu beantworten. Während der gesamten Zeit standen mir meine betriebliche Betreuerin Frau N. Weyen und seitens der Hochschule Frau Professorin S. Wieland stets helfend zur Seite. Ich bedanke mich im Besonderen für ihre Geduld und die gewissenhafte Beantwortung meiner Fragen. Bei den anderen Kollegen seitens T-Systems möchte ich mich gerne für die angenehme Zusammenarbeit bedanken. Ich konnte mich oft mit ihnen über meine Untersuchungen und Ergebnisse austauschen. Auch der Austausch mit meinen Freunden und meiner Familie half mir bei der Erstellung dieser Arbeit. Sie waren immer für mich da, wenn ich nicht weiterwusste oder die Motivation verlor. Außerdem bedanke ich mich an dieser Stelle bei Frau M. Verhülsdonk für ihre Mühen zum Korrekturlesen. Im Besonderen möchte ich natürlich meinen Eltern danken. Ihre Ratschläge und die einfühlsamen Worte haben mir immer sehr geholfen.

Robert Schnorr
November 2016

Kapitel 1

Einführung

1.1 Ausgangssituation und Problemstellung

1.1.1 Organisatorische Einordnung des Musterverbundes

Das Projekt AD bezieht sich auf eine IT-Anwendung, die innerhalb der Deutschen Telekom von der DTTS GmbH genutzt wird. Das Ziel der Anwendung ist es, die Aufträge aus den Vorsystemen (SMILE, TESY und WMS-TK) anzunehmen und zu bearbeiten. Weiterhin sind an dem AD-Server die IT-Anwendungen PersDispo-TK(CS) für die Auftragsdisposition, KVWS für die Personalkapazitäts-Verwaltung und der AD-Client zur Auftragsausführung für eigene Außendienst-Mitarbeiter bzw. ANTK für die Abarbeitung durch Subunternehmer angebunden.

Der AD-Server hat eine Oberfläche, das sogenannte Admin-Portal. Das Admin-Portal ist die Oberfläche zur Pflege von zentralen Referenzdaten für den Anwendungsverbund. Um diesen Anwendungsverbund zu testen, gibt es mehrere Testumgebungen wie Komponententest, Integrationstest, Systemtest, Lasttest und Abnahmetest. Jede dieser Umgebungen stellt einen AD-Server dar, an dem die unterschiedlichen Partnersysteme real oder in Form eines Simulators eingebunden sind. Um eine geeignete Aussage der Tests auf den Wirkbetrieb zu gewährleisten, ist eine wirkbetriebsnahe Konfiguration der Umgebungen sehr wichtig.

Der Systemtest AD, welcher hier als Musterverbundtest dienen soll, verfügt über kein reales WMS-TK bzw. ANTK, beide Systeme werden mittels eines Simulators nachempfunden, nutzen dafür jedoch die selben Schnittstellen wie beim Echtsystem. Aus Zeitgründen beschränken sich die Tests auf den kleinsten Verbund innerhalb der Umgebung, welcher durch die Komponenten AD-Server, AD-Client und CS dargestellt wird. Die Schnittstellen zu ANTK und WMS-TK werden dabei zwar aktiv benutzt, aber nicht im Rahmen eines Tests

mit untersucht, weshalb insgesamt keine Schnittstellen getestet werden, sondern lediglich GUI-basiert getestet wird.

1.1.2 Problemstellung

Gegenstand dieser Arbeit ist es zu erforschen, mit welchen Softwarewerkzeugen es möglich ist, innerhalb eines Verbundtests alle Softwarekomponenten ausgiebig und automatisch in Form sogenannter Autotests (Automatisierung der Testdurchführung per Definition nach ISTQB Glossar) zu durchlaufen und auf Fehler zu untersuchen. Dabei wird im Speziellen der Systemtestverbund des T-Systems Projekts AD mit Anbindung an die Software WMSTK, CS und ANTK (siehe Abbildung 1.1) betrachtet.

Abbildung 1.1 Testverbund „AD“

Ziel der Arbeit

Diese Arbeit beschäftigt sich damit, auf welche Weise ein hohes Maß an Softwarequalität und Automatisierung von Testaktivitäten innerhalb eines Verbundtests gewährleistet werden kann. Untersucht werden verschiedene Softwarelösungen für diese Problematik, nach Möglichkeit soll sich dabei auf kostenfreie Software beschränkt werden, welche kommerziell verwendet werden darf.

Außerdem stellt sich die Frage, welcher Aufwand aufgebracht werden muss, um eine geeignete Software zum Einsatz zu bringen und welche Schritte notwendig sind, um einen Test automatisch abarbeiten zu können. Tests sind innerhalb der letzten Jahrzehnte immer wichtiger für die Softwareentwicklung geworden, da Software immer komplexer und daher auch fehleranfälliger geworden ist.

Immer mehr, vor allem größere Unternehmen suchen Lösungen, um Tests häufig durchführen zu können und zeitgleich eine vollständige Testabdeckung garantieren zu können. Die Testautomatisierung bietet die Möglichkeit, eine vollständige Testabdeckung zu erreichen mit dem Ziel, dabei jeden möglichen Testfall beliebig oft und möglichst genau abzudecken. Die Problematik liegt dabei in der Automatisierung von festgelegten Testfällen. Untersucht wird, mit welcher Software Testfälle über mehrere Systeme optimal automatisiert werden kann unter Beachtung von Zeit und Aufwand.

1.1.3 Theoretischer Bezug

Definition

Dieser Arbeit legen die Definitionen des Glossars des ISTQB (International Software Testing Qualifications Board) zugrunde. Im Besonderen zu nennen sind die Begriffe Integrationstest, Systemtest und Testdurchführung. Die dortige Definition der Testautomatisierung möchte ich ergänzen zu: „*Einsatz von Softwarewerkzeugen zur Durchführung oder Unterstützung von Testaktivitäten, z. B. Testmanagement, Testentwurf, Testdatengenerierung, Testausführung und Soll/Ist-Vergleich* mit dem Ziel einer vollständigen Testabdeckung auf Verbund-Ebene in einem möglichst kurzen Zeitintervall.“ ISTQB AISBL [40]

Ansätze der Testautomatisierung

Es gibt eine Reihe von Ansätzen. Laut der TeQS Consulting Unternehmergesellschaft (haftungsbeschränkt) kann eine Testautomatisierung GUI-basiert, schnittstellenbasiert z.B. http oder basierend auf Unit-Tests auf Codeebene arbeiten. Chece [5]

Bei der GUI-basierten Variante wird die Nutzeroberfläche einer Software, welche getestet werden soll, anhand von vorher definierten Maus- und Tastatur-gesten angesteuert. Es erfolgt eine Simulation der Mausklicks und der Mauspositionswechsel, sowie von Tastureingaben, nach Vorgabe, wie ein Mensch besagte Oberfläche durch Eingaben und Klicks testen würde. Je nach Art der Software kann diese Oberfläche HTML, Text oder Bildelemente enthalten. HTML-Elemente könnten mittels DOM Objektstruktur direkt angesteuert, ausgelesen und verändert werden. Bei der textbasierten GUI kann mit Hilfe eines Parsers gearbeitet werden

und bei der bildbasierten Oberfläche wird eine Bild/Text-erkennung zur Überprüfung der korrekten Darstellung der Oberfläche inklusive der dargestellten Werte verwendet.

Laut Greiter wird folgendes behauptet: *„Wird nur per GUI getestet, verlaufen 75% aller Testautomatisierungsprojekte im Sande“* Greiter [29]. Bei der Schnittstellen-basierten Variante werden die Nachrichten auf der zu testenden Schnittstelle nachsimuliert, die Übertragung der Nachrichten kann je nach Art der Software gewissen Anforderungen unterliegen. Beispiele für Übertragungsarten sind SOAP-Nachrichten, HTTP-Nachrichten und Queue-Nachrichten. Bei der Variante des Unit-Tests auf Codeebene (Komponententest) wird die kleinste Softwareeinheit, zum Beispiel eine Klasse oder eine einzelne Methode, auf Funktionalität getestet.

Einige Arten von Software (Testframeworks) sind zu kompliziert und deshalb nicht für Softwaretester ohne Programmierkenntnisse geeignet. Es ist stets zu beachten, dass sich auch komplexe Testfälle automatisieren lassen müssen. Laut einer Studie von Carl Nagle (CN16 [7]) scheiterten sehr viele solcher Projekte aufgrund unterschiedlicher Anforderungen an die Software in Bezug auf Wartbarkeit durch verschiedene Gruppen, Kosten für die Entwicklung und die zur Verfügung stehende Zeit, um überhaupt Softwareänderungen in Testfälle umsetzen zu können. Tester würden sich schwer tun, wenn sie keine Programmierer sind. Die Sicht auf die Software ist eine andere, als ein Programmierer diese hat. Für eine vollständige Umsetzung der Testfälle ist ausreichend Zeit einzuplanen, es darf nicht „nebenbei“ betrieben werden und es muss darauf geachtet werden, das eine vollumfänglich geeignete Software genutzt wird, welche technisch in der Lage ist, alle denkbaren Testszenarien unterstützen zu können. CN16 [7]

Exkurs Data Driven Testing

Grundsätzlich kann ein Testfall wiederholt mit denselben Daten betrieben werden.

Die Praxis zeigt, dass bestimmte Fehler erst ersichtlich werden, wenn die getesteten Daten auch einer gewissen Variabilität unterliegen. Ein Beispiel aus der Praxis ist, dass die zu testende Software in einem Textfeld Umlaute falsch erfasst oder die Länge der Eingabe in ein solches Textfeld bei der Weiterverarbeitung Probleme hervorrufen kann, weil z. B. die Tabelle in einer Datenbank die Daten nicht annehmen kann.

Um solche „datenabhängige“ Fehler provozieren zu können, muss der Testlauf mit verschiedenen Testdaten wiederholt werden. Im Idealfall wird dazu die Technik des „Data Driven Testings“, bei welchem der allgemeine Testlauf mit definierten Variablen erstellt wird, verwendet.

Diese Variablen lassen sich anschließend anhand einer Datentabelle verwalten. Für jeden Durchlauf kann die Automatisierungssoftware auf einen neuen variablen Datensatz

zugreifen, welcher in Form einer Datenbank, Excel-Tabelle oder Ähnlichem verwaltet werden kann. Software [78]

1.1.4 Historie des Testens

Es lässt sich mutmaßen, dass es bereits seit Beginn der ersten Software eine Art Test gegeben hat. Laut einem Artikel von David Gelperin und Bill Hetzel aus dem Jahr 1988 ist der Softwaretest in folgende Phasen und Ziele einzuordnen:

Tabelle 1.1 Perioden des Softwaretests

von	bis	deutsch	englisch
-	1956	debugging-orientierte Periode	debugging oriented period
1957	1958	demonstrations-orientierte Periode	demonstration oriented period
1979	1982	destruktions-orientierte Periode	destruction oriented period
1983	1987	evaluations-orientierte Periode	evaluation oriented period
1988	-	präventions-orientierte Periode	prevention oriented period

Bis zum Jahr 1956 reichte die debugging-orientierte Periode, in dieser Zeit gab es keine wesentliche Unterscheidung zwischen Test und Debugging. Das Debugging war gleichzusetzen mit dem Test der Software. Mit steigenden Anforderungen an die Software wurde die demonstrations-orientierte Periode (1957-1958) eingeführt, deren Ziel es war darzulegen, dass Software gewissen Anforderungen genügt. Laycock [43] Gelperin and Hetzel [27]

Erste Unternehmen verstanden die Notwendigkeit von Softwaretests, hatten aber keine eigenen Kapazitäten, so dass erste Testlabore entstanden. Witte [103] S.4

Innerhalb der destruktions-orientierten Periode (1979-1982) lag das Hauptaugenmerk im Softwaretest auf dem Auffinden von Softwareanomalien. Mit Beginn des Jahres 1983 wurden verschiedene Standards zur Softwareentwicklung veröffentlicht mit dem Ziel, Fehler zu einem frühen Zeitpunkt des Software-Lebenszyklus zu erkennen, da aus negativen Erfahrungen Erkenntnisse entstanden, wie wichtig ein frühzeitiger Softwaretest ist.

Im Allgemeinen ist ein Software-Lebenszyklus[Abbildung 1.1] in die Phasen Planung, Analyse, Design, Entwicklung, Testen und Ausliefern einzuteilen, wobei es Abweichungen geben darf. Während dieses Software-Lebenszyklus soll eine Produkt-Evaluation (Review auf Benutzbarkeit) durchgeführt werden und die Software in Hinblick auf ihre Qualitätsanforderungen untersucht werden.

Abbildung 1.2 Sechs-Phasen-Modell / Software-Lebenszyklus

Ab 1988 begann die präventions-orientierte Periode, deren Ziel es war, die Einhaltung der Spezifikation einer Software zu untersuchen und mögliche Anomalien vorsorglich zu erkennen. Laycock [43] Gelperin and Hetzel [27]

Ein genauer Zeitpunkt, wann die Automatisierung von Tests einsetzte, ist nicht bekannt. Das manuelle Testen wurde zu komplex, um Software von Hand umfangreich testen zu können.

1.2 These der Arbeit

Zur Auswahl der geeigneten Softwarewerkzeuge liegt der Fokus ausschließlich auf den System-Verbundtest des T-Systems Projektes „AD“ und dabei wird bei den eigentlichen Untersuchungen nur jene Software betrachtet, welche zur Testautomatisierung für den AD-Verbundtest geeignet ist.

Die folgende These wird dazu in den Raum gestellt: „Es gibt keine geeignete Software zur Testautomatisierung, welche alle Technologien und Anforderungen eines Verbundtests in vollem Umfang erfüllen könnte, und noch schneller arbeitet als ein menschlicher Tester.“

Im Zuge dieser Bachelorarbeit sollen die nachfolgenden Leitfragen beantwortet werden mit Hinblick auf eine Validierung der zuvor aufgestellten These:

- Welche Software zur Testautomatisierung gibt es?
- Welche Technologie deckt diese Software ab?
- Wie ist die Bedienbarkeit dieser Software?
- Welchen Zeit- und Arbeitsaufwand verlangt diese Software?
- Wie groß ist der Nutzen dieser Software?

Dem Anhang können Untersuchungen und tiefergreifende Informationen, die den Rahmen dieser Bachelorarbeit sprengen würden, entnommen werden.

1.3 Abgrenzung der Themenstellung

Testautomatisierung umfasst je nach Definition nicht nur die Durchführung des eigentlichen Testfalls, sondern kann auch die Erstellung, Auswertung und Dokumentation eines Testfalls bedeuten.

Die folgenden Ausführungen konzentrieren sich vor allem auf die Untersuchung von Tools zur automatischen Durchführung von Testfällen, um den Zeitrahmen dieser Arbeit einhalten zu können.

Außerdem beruhen die Untersuchungen auf Basis des Verbund-Systemtests des T-Systems Projektes „AD“. Eine Untersuchung in anderen Verbundumgebungen wie Lasttest, Komponententest, Integrationstest, Abnahmetest oder sogar anderen Softwareprojekten erfolgt nicht. Es wird so untersucht, dass die Ergebnisse auch auf andere Projekte und Umgebungen übertragen werden können.

Zudem wird das Hauptaugenmerk nicht auf dem Testprozess als solchem liegen, es wird folglich nicht im Detail auf die Planung, Überwachung und Steuerung von Testfällen eingehen. Vielmehr beschäftigt sich diese Arbeit mit den Möglichkeiten der Ausführung von Testfällen zur optimalen Integration innerhalb eines solchen Testprozesses.

Die betrachteten Testfälle sollen allgemein ein durchschnittliches Szenario innerhalb des Verbund-Systemtests darstellen. Dabei wird jedoch nicht detailliert auf die Erstellung und das Design von Testfällen eingehen. Es wird vielmehr der Testfall als solches vorausgesetzt, wobei jedoch Aufschluss darüber geben wird, welche Arten von Testfällen durch welches Tool abgebildet und automatisiert getestet werden können.

Im Übrigen wird auf die verwendete Technik der jeweiligen Software zur Automatisierung eingehen. Der Fokus liegt dort auf der Sichtweise eines Testers anstatt eines Entwicklers. Um es mit dem Beispiel der Blackbox darzulegen, wird unter anderem untersucht, wie die Software arbeitet, aber nicht welche Funktionen im Inneren der Software dafür vonnöten sind. So wird Software innerhalb ihrer Art und des Einsatzgebietes im Rahmen dieser Arbeit unterschieden, jedoch wird nicht darauf eingegangen, wie diese Software entwickelt wurde.

1.4 Die Zielgruppe

Diese Arbeit richtet sich an Softwareunternehmen, welche einen Verbund mehrerer Applikationen mit unterschiedlichen GUI-Technologien automatisiert testen möchten. Beschränkt wird sich dabei aus Zeitgründen auf Automatisierungstools, welche auf dem Betriebssystem Microsoft Windows 7 ausgeführt werden können.

Die Anforderung dieser Zielgruppe ist es eine Automatisierungslösung zu finden, welche jeden durch einen Tester ausführbaren Testfall automatisiert ausführen kann.

Kapitel 2

Hauptteil

2.1 Kriterien für die Untersuchung

2.1.1 Kriterien für Automatisierungs-Produkte

Die Bewertung erfolgte ausschließlich anhand der jeweiligen vom Hersteller zur Verfügung gestellten Informationen.

Untersucht wurden Softwarewerkzeuge unter den Kriterien:

- unterstützte Technologien
- benötigte Infrastruktur
- Nutzerfreundlichkeit
- Skalierbarkeit
- Hilfe und Support
- Integration und Anbindung
- Kosten und Lizenz

Die Punkteverteilung erfolgte dabei wie folgt:

Tabelle 2.1 Punkteverteilung

Punkte	Begründung
0	erfüllt die Software nicht
1	erfüllt die Software nur teilweise oder nicht ab Werkseinstellung*
2	erfüllt die Software in vollem Umfang
3 oder mehr	besondere Gewichtung des Kriteriums

**: Erst erfüllt durch die Installation zusätzlicher Erweiterungen oder Drittanbieter-Software.*

Tabelle 2.2 Kriterium: Typ/Technologie

Software	**erfüllt**								**Summe**
	ERP	Schnitt-stellen	Web-Services	GUIs	Browser	Silver-light	Units	Besonder-heiten*	
HP Unified Functional Testing (UFT)	1	2	0	2	2	2	0	0	9
VisualStudio Coded UI Tests	0	0	0	1	1	2	0	0	4
Ranorex Automation Framework	1	0	0	2	2	2	0	1(a)	8
Micro Focus SilkTest	1	0	0	2	2	0	0	0	5
QFTest	0	0	0	1	2	0	0	0	3
Telerik Test Studio	0	2	1	1	2	2	0	0	8
IBM Rational Functional Tester	1	2	0	2	2	2	0	0	9
eggPlant Functional	0	1	0	2	2	2	0	0	6
Tricentis TOSCA TestSuite	1	2	2	2	2	1	0	0	9
expecco	1	1	0	2	0	0	0	0	4
TestComplete	0	0	0	1	1	0	1	0	3
RapidRep	0	2	2	0	0	0	0	0	4
TTworkbench	0	2	2	0	0	0	0	0	4
Parasoft Software Testing Tools	0	1	1	0	0	0	1	1(b)	4
Worksoft Certify	0	0	0	2	2	0	0	0	4
SikuliX	0	0	0	2	2	1	0	1(c)	6
Selenium	0	0	0	0	2	0	0	0	2
CasperJS	0	0	0	0	1	0	0	0	1
Protractor	0	0	0	0	1	0	0	0	1
SoapUI	0	2	2	0	0	0	0	0	4
Jubula/GUIdancer	0	0	0	2	1	0	0	0	3

*Besonderheiten = Zusatzpunkte für a) Flash/Flex-Tests, b) UNIT-/Penetration-Tests und c) Test basiert auf Bildschirmausgabe

veraltet

Unterstützte Technologien

Es gibt Softwarewerkzeuge, welche eine einzige Technologie unterstützen wie zum Beispiel Web Services, HTML- oder andere GUIs. SoapUI, RapidR[illegible]nd TTworkbench unterstützen lediglich Web Services wie SOAP und andere ähnliche Schnit[illegible]n.

Andere Tools wie HP Unified Functional Testin[illegible], IBM Rational Functional Tester, Tricentis TOSCA TestSuitec und expecco hingegen kön[illegible] Vielzahl an Technologien in einem Programm bzw. einer Sammlung an Progra[illegible]n testen, inklusive Erweiterungen wie zum Beispiel Silverlight und Enterprise-Res[illegible]e-Plan[illegible]g Software (SAP und weitere).

Die meisten Programme spezialisieren sich [illegible] die Automatisierung von GUIs egal welcher Form, dabei bieten einige Tools e[illegible]hnologieabhängige Erkennung der Oberfläche. So werden zum Beispiel die Inhalte v[illegible] Textfeldern als Text erkannt und somit sind diese Inhalte für die Überprüfung v[illegible] Testfällen geeignet. Eine andere Technik ist die der Bilderkennung. Dabei ist es nebensäch[illegible]us welcher Technologie eine GUI besteht, die Software fertigt einen Screensh[illegible]t e[illegible]es Ausschnittes der Oberfläche an und vergleicht diesen mit einem zuvor aufgezeichn[illegible]ete [illegible]nshotausschnitt.

In dieser Kategorie s[illegible]iden die Softwareprodukte HP Unified Functional Testing (UFT), Telerik Test Stu[illegible] Tricentis TOSCA TestSuite mit jeweils neun Punkten am besten ab. Sie bieten die größte Auswahl an unterstützen Technologien.

veraltet

veraltet

Tabelle 2.3 Kriterium: Benötigte Infrastruktur

Software	**erfüllt**							**Summe**
	Microsoft Windows	Unix	Linux	MacOS	Java JRE Installation	Node.Js	Phantomjs	Summe
HP Unified Functional Testing (UFT)	2	0	0	0	0	0	0	2
VisualStudio Coded UI Tests	2	0	0	0	0	0	0	2
Ranorex Automation Framework	2	0	0	0	0	0	0	2
Micro Focus SilkTest	2	0	0	0	0	0	0	2
QFTest	2	2	0	0	0	0	0	4
Telerik Test Studio	2	0	0	0	0	0	0	2
IBM Rational Functional Tester	2	0	2	0	0	0	0	2
eggPlant Functional	2	0	2	2	0	0	0	6
Tricentis TOSCA TestSuite	2	0	0	0	0	0	0	2
expecco	2	0	0	0	0	0	0	4
TestComplete	2	0	0	0	0	0	0	2
RapidRep	2	2	2	0	0	0	0	6
TTworkbench	2	0	2	0	0	0	0	4
Parasoft Software Testing Tools	1	1	1	1	0	0	0	4
Worksoft Certify	2	0	0	0	0	0	0	2
SikuliX	0	0	0	0	8	0	0	8
Selenium	0	0	0	0	8	0	0	8
CasperJS	0	0	0	0	0	0	4	4
Protractor	0	0	0	0	0	4	0	4
SoapUI	2	0	2	2	0	0	0	6
Jubula/GUIdancer	0	0	0	0	8	0	0	8

Java JRE Installation, Node.Js und Phantomjs wurden höher bewertet, da diese plattformunabhängig sind

Benötigte Infrastruktur

Nachforschungen ergaben, dass alle betrachteten Tools unter einem Microsoft Windows Betriebssystem lauffähig sind. Einige erfordern dafür das .NET Framework oder eine aktuelle Java-Installation (JRE). Als ein Beispiel benötigt die Software von Ranorex **mindestens** Microsoft Windows XP mit Microsoft Windows Installer 3.1, Microsoft .NET Framework 4.0 und Internet Explorer 7 auf einem System mit 1GHz Prozessor und 512MB RAM. Ranorex [73] Microsoft [47]

Einige der Tools wie SikuliX, Selenium und Jubula basieren auf Java und sind somit betriebssystemunabhängig. Es wird lediglich eine aktuelle Java-Installation (JRE) vorausgesetzt. Das Betriebssystem Mac OS X wird nur von wenigen Applikationen wie eggPlant Functional und SoapUi unterstützt. Die Tools JasmineJS und Protractor fallen etwas aus dem Rahmen. Sie basieren jeweils auf den JavaScript-Umgebungen Node.js oder Phantomjs und sind somit betriebssystemunabhängig. Hierzu [31] Foundation [26]

Wird allerdings Linux bzw. Unix als Betriebssystem der Wahl bevorzugt, kann in etwa die Hälfte der Applikationen auch unter Linux verwendet werden, besonders die betriebssystemunabhängigen Testwerkzeuge.

In dieser Kategorie dominieren die Softwareprodukte SikuliX, Selenium und Jubula/GUIdancer aufgrund ihrer Plattformunabhängigkeit. Es wird lediglich eine Installation der Java Runtime Environment(JRE) vorausgesetzt.

veraltet

veraltet

Tabelle 2.4 Kriterium: Nutzerfreundlichkeit

Software	erfüllt					Summe
	Drag & Drop	Steuerung über GUI	Record-Tool	Steuerung über Schnittstelle	Script-Code	
HP Unified Functional Testing (UFT)	4	2	0	2	1	9
VisualStudio Coded UI Tests	0	2	2	1	1	5
Ranorex Automation Framework	4	2	2	2	1	11
Micro Focus SilkTest	0	2	0	0	2	4
QFTest	0	2	2	2	1	7
Telerik Test Studio	4	2	2	2	1	11
IBM Rational Functional Tester	0	2	2	2	1	7
eggPlant Functional	0	2	2	0	2	6
Tricentis TOSCA TestSuite	0	2	2	2	1	7
expecco	4	1	0	0	2	7
TestComplete	4	2	0	0	2	8
RapidRep	0	0	0	0	2	2
TTworkbench	0	1	2	0	2	5
Parasoft Software Testing Tools	0	1	1	0	2	4
Worksoft Certify	0	2	2	0	1	5
SikuliX	0	2	1	0	2	5
Selenium	0	1	1	0	2	4
CasperJS	0	0	0	0	2	2
Protractor	0	0	0	0	2	2
SoapUI	0	1	1	0	2	4
Jubula/GUIdancer	0	2	1	0	1	4

Drag & Drop wurde höher bewertet, weil dies die nutzerfreundlichste Möglichkeit der Steuerung darstellt.

Nutzerfreundlichkeit

Bei der Nutzerfreundlichkeit überzeugen die kostenpflichtigen kommerziellen Testautomatisierungswerkzeuge. Die meisten von ihnen verfügen über sogenannte „Record & Play“-Funktionalitäten, mit deren Hilfe es möglich ist, manuelle Testabläufe innerhalb einer Oberfläche aufzuzeichnen, zu Testfällen zu verarbeiten und später als Testdurchführung wieder ausführen zu können. Die höherpreisigen Applikationen wie HP Unified Functional Testing, Ranorex Automation Framework oder TestComplete verfügen zudem über Drag & Drop-Funktionalitäten, sodass Testfälle aus einzelnen Testelementen via GUI leicht kombiniert oder neu erstellt werden können.

Neben diesen auch für Tester zu verstehenden Methoden unterstützen die meisten Testautomatisierungswerkzeuge eine Steuerung bzw. Programmierung über eine Schnittstelle oder Script-Code. Ein anderer Aspekt der Nutzerfreundlichkeit spiegelt sich in den unterstützten Technologien wider. Einige Testautomatisierungswerkzeuge arbeiten als ein Framework aus mehreren Komponenten zusammen und können somit ohne spürbaren Wechsel der Software unterschiedliche Technologien unterstützen, wodurch Zeit eingespart wird und sich damit die Effektivität erhöht.

Am benutzerfreundlichsten sind die Softwareprodukte Ranorex Automation Framework und Telerik Test Studio zu bewerten. Die beiden Applikationen ermöglichen das Arbeiten per Drag & Drop innerhalb einer GUI und zusätzlich können Testfälle über eine Schnittstelle eingestellt werden.

veraltet

Tabelle 2.5 Kriterium: Skalierbarkeit

Software	erfüllt			Summe
	Erweiterungen	mehrere (virtuelle) Instanzen	Remote-Ausführung	
HP Unified Functional Testing (UFT)	2	2	2	6
VisualStudio Coded UI Tests	1	0	0	1
Ranorex Automation Framework	0	2	2	4
Micro Focus SilkTest	2	0	2	4
QFTest	2	2	0	4
Telerik Test Studio	2	2	2	6
IBM Rational Functional Tester	2	2	1	5
eggPlant Functional	0	2	2	4
Tricentis TOSCA TestSuite	2	2	1	5
expecco	2	0	0	2
TestComplete	2	2	1	5
RapidRep	2	0	0	2
TTworkbench	2	0	1	3
Parasoft Software Testing Tools	2	2	0	4
Worksoft Certify	0	0	0	0
SikuliX	1	0	0	1
Selenium	0	1	0	1
CasperJS	0	0	0	0
Protractor	0	1	0	1
SoapUI	0	1	0	1
Jubula/GUIdancer	1	0	0	1

veraltet

Skalierbarkeit

Generell bieten die meisten Applikationen Möglichkeiten der Erweiterbarkeit durch zum Beispiel das Ausführen mehrerer (virtueller) Instanzen oder Plugins, um den Nutzungsumfang der Software zu erhöhen. Am Beispiel HP Unified Functional Testing zeigt sich, dass mehrere Instanzen einer Testausführung parallel auf Rechnern, Mobilgeräten und (realen und virtuellen) Servern möglich sind. Die verteilte Testausführung mehrerer Tests gleichzeitig ermöglicht einen wirkbetriebsähnlichen Test bzw. einen Test unter Last. Außerdem ist es möglich unterschiedliche Betriebssysteme gleichzeitig zu testen, wodurch eine Erhöhung der Effektivität und Einsparung von Zeit erreicht wird. Enterprise [19]

Die höherpreisigen Applikationen wie HP Unified Functional Testing, Ranorex Automation Framework, IBM Rational Functional Tester oder Telerik Test Studio arbeiten dabei nicht nur lokal, sondern ermöglichen einen Test via Remote. So müssen zusätzliche Instanzen nicht separat lokal installiert werden und eine Skalierung kann von einer zentralen Installation aus verwaltet werden. Die Ausführung von Remote Tests kann dabei entweder direkt nativ in der Applikation unterstützt oder durch die Installation von Erweiterungen ermöglicht werden. Ranorex [72], Packard [52], Smartbear [75], IBM [34]

Generell ist im Punkt Skalierbarkeit auf die Lizenzierung einzugehen. Für jede Instanz muss unter Umständen eine eigene sogenannte Floating-Lizenz erworben werden.

Die größte Möglichkeit zur Skalierung der Testautomatisierung bieten die Softwareprodukte HP Unified Functional Testing und Telerik Test Studio.

veraltet

Tabelle 2.6 Kriterium: Hilfe/Support

veraltet

Software	erfüllt				
	Dokumentation	bezahlte Schulungen	bezahlter Support	Community	Summe
HP Unified Functional Testing (UFT)	2	2	2	0	6
VisualStudio Coded UI Tests	2	2	2	0	6
Ranorex Automation Framework	2	2	2	0	6
Micro Focus SilkTest	2	2	2	2	8
QFTest	2	2	2	0	6
Telerik Test Studio	2	2	2	2	8
IBM Rational Functional Tester	2	2	2	2	8
eggPlant Functional	2	2	2	2	8
Tricentis TOSCA TestSuite	2	2	2	2	8
expecco	2	2	1	1	6
TestComplete	2	2	2	2	8
RapidRep	2	2	2	0	6
TTworkbench	2	2	2	0	6
Parasoft Software Testing Tools	2	2	2	2	8
Worksoft Certify	2	2	2	0	6
SikuliX	2	0	0	2	4
Selenium	2	1	1	2	6
CasperJS	2	0	0	2	4
Protractor	2	0	0	2	4
SoapUI	2	2	2	2	8
Jubula/GUIdancer	2	2	2	2	8

1 Punkt = Kriterium erfüllt, aber nur mäßig oder nur durch Drittanbieter

Hilfe und Support

Eine Benutzerdokumentation/Handbuch ist für jede der in Tabelle 2.6 aufgeführten Software-Applikationen verfügbar. Darüber hinaus werden die Open-Source-Applikationen alle von einer großen Community betreut, gepflegt und erklärt. Das Wissen dieser Communities ist online meistens in Form eines Blogs oder Wikis kostenfrei verfügbar und kann von Jedermann dort erweitert oder bearbeitet werden. Eine großen „Schwarmintelligenz“ trägt dazu bei, dass die betreffende Software umfangreich beschrieben ist und neue Ansätze für mögliche Neuerungen innerhalb der Software vorangetrieben werden. Diesen Ansatz einer Community verfolgen auch einige kommerzielle Hersteller wie Micro Focus, Telerik, IBM, eggPlant, Tricentis, Smartbear und Parasoft.

Zusätzlich bieten alle kommerziellen Hersteller zusätzlichen Service, von einem kostenpflichtigen höherwertigen Support bis hin zu mehrtägigen Software-Schulungen zu ihrem Produkt. Bei einigen größeren Anbietern wie HP, QFS und Ranorex ist der Benutzer gezwungen, sofern ihm die Dokumentation nicht genügt, Support und Schulungen als kostenpflichtiges Zusatzprodukt zu erwerben. So kostet ein Einführungskurs für Ranorex und HP ab £695.00 (808.97 €) [Online] netto, bei QFS ab 695 € netto. Edgewords [18] QFS [67]

In dieser Kategorie gibt es keine klare „Sieger“-Software. Es lässt sich sagen, dass die kommerziellen Hersteller gegen Bezahlung den besseren Support bieten und zum Teil sogar eine Community pflegen.

veraltet

Tabelle 2.7 Kriterium: Integration/Anbindung

Software	erfüllt	Integration von				Summe
	Testfallimport	Continuous Integration Tools	Testmanagement -Tools	IDE z. B. Visual Studio	weitere Tools	
HP Unified Functional Testing (UFT)	2	2	2	0	1	7
VisualStudio Coded UI Tests	1	1	1	2	1	6
Ranorex Automation Framework	2	2	2	2	2	10
Micro Focus SilkTest	2	2	2	2	2	10
QFTest	2	2	2	0	2	8
Telerik Test Studio	2	2	2	2	2	10
IBM Rational Functional Tester	2	2	2	2	2	10
eggPlant Functional	0	2	2	0	2	6
Tricentis TOSCA TestSuite	2	2	2	2	2	10
expecco	0	0	2	0	0	2
TestComplete	0	2	2	0	0	4
RapidRep	0	0	2	0	1	3
TTworkbench	0	0	0	0	3	3
Parasoft Software Testing Tools	0	0	0	0	0	0
Worksoft Certify	0	2	0	0	0	2
SikuliX	0	0	0	0	3	3
Selenium	0	0	0	0	3	3
CasperJS	0	0	0	0	1	1
Protractor	0	0	0	0	1	1
SoapUI	0	1	1	0	1	3
Jubula/GUIdancer	0	1	0	2	0	3

3 Punkte bei weitere Tools = Kriterium erfüllt, aber nur durch Plugins (oder eigene Programmierung)

veraltet

Integration und Anbindung

Idealerweise sollte sich ein Tool zur Testautomatisierung in den bereits existierenden Workflow (Testmanagement) und die Ressourcen (Testdaten) eines Unternehmens einbinden lassen. Viele der in Tabelle 2.7 aufgeführten Tools verfügen über Möglichkeiten, Testfälle automatisch aus bereits existierenden Daten zum Beispiel einer Excel-Tabelle zu importieren und zu verarbeiten. In diesem Zusammenhang können einige Applikationen wie HP Unified Functional Testing, VisualStudio Coded UI Tests, Ranorex Automation Framework und Micro Focus SilkTest die Methodik des Keyword Driven Tutorialspoint [101] Ranorex [70] oder Data-Driven Testing Tutorialspoint [102] Ranorex [68] verwenden, um aus einem Testfall weitere mögliche Testfälle hervorbringen zu können.

Einige Tools verfügen sogar über die Möglichkeit, mittels einer Schnittstelle aus einer Entwicklungsumgebung wie zum Beispiel Visual Studio direkt Testfälle zu erzeugen.

Einen weiteren wichtigen Aspekt der optimalen Integration erfüllen die meisten kostenpflichtigen Tools, indem sie von sich aus Schnittstellen zur Anbindung an Continuous-Integration-, Testmanagement- bzw. Fehlermanagement-Tools bereitstellen.

Da die meisten Open-Source-Applikationen das eigenständige Weiterentwickeln von Plugins ermöglichen oder Drittanbieter solche Plugins auf den Markt gebracht haben, ist es auch für solche Software möglich. Hierbei ist jedoch mit einem höheren Aufwand zur Anbindung an andere Tools zu rechnen.

Die beste Integration in andere Softwareapplikationen bieten die Tools Ranorex Automation Framework, Micro Focus SilkTest, Telerik Test Studio, IBM Rational Functional Tester und Tricentis TOSCA TestSuite.

veraltet

Tabelle 2.8 Kriterium: Kosten und Lizenz

Software	erfüllt	Bewertung (0 - 5)
HP Unified Functional Testing (UFT)	30 Tage Test (ab $600 je User/3 Monate)	2
VisualStudio Coded UI Tests	90-Tage-Test, Teil von VisualStudio (ab $499 je User)	2
Ranorex Automation Framework	30-Tage-Test (ab 1.990 € netto je User)	2
Micro Focus SilkTest	45-Tage-Test (5.679 € je User/Jahr)	2
QFTest	30-Tage-Test (ab 2.475 € netto je User)	2
Telerik Test Studio	30-Tage-Test (ab $2,499 netto je User)	2
IBM Rational Functional Tester	30-Tage-Test (ab 6.977 € netto je User)	2
eggPlant Functional	5-Tage-Test (Preise nur auf Anfrage)	1
Tricentis TOSCA TestSuite	1[illegible]-Tage-Test (Preise nur auf Anfrage)	1
expecco	Demo und Preise nur auf Anfrage	0
TestComplete	30-Tage-Test (ab 1.067 € netto je User)	2
RapidRep	6-Monate-Test (ab [illegible] € netto je User)	3
TTworkbench	14-Tage-Test [6-Monats-Test für Universitäten] (Preise nur auf Anfrage)	1
Parasoft Software Testing Tools	Demo und Preise nur auf Anfrage	0
Worksoft Certify	Demo und Preise nur auf Anfrage	0
SikuliX	kostenlos	5
Selenium	kostenlos	5
CasperJS	kostenlos	5
Protractor	kostenlos	5
SoapUI	kostenlos oder PRO-Version (ab 539 € netto je User/Jahr)	4
Jubula/GUIdancer	kostenlos	5

Je länger die Testversion lauffähig ist und je günstiger die Softwarelizenz, desto mehr Punkte

veraltet

Kosten und Lizenz

Grundsätzlich gibt es verschiedene Lizenztypen. Unterschieden werden kann unter den aufgeführten Tools zwischen den Lizenzmodellen Open Source, Node-Locked- oder Floating-Lizenz. Eine Open-Source-Lizenz ermöglicht dem Nutzer, die Software für beliebige Zwecke zu verwenden, sie zu verändern und den Quellcode zu analysieren. Dadurch, dass sie selbst für kommerzielle Zwecke kostenfrei ist, können beliebig viele Instanzen installiert werden, ohne dass Lizenzkosten anfallen. c´t [16]

Unter einer Node-Locked-Lizenz wird verstanden, dass die Software nur auf einem bestimmten Rechner installiert und betrieben werden darf. Dabei gibt es Lizenzen, die an physikalische Prozessoren oder virtuelle Instanzen gebunden sind. Möchte ein Unternehmen mehrere Instanzen einer solchen Software ausführen, müssen unter Umständen neue Lizenzen erworben werden.

Ein weiteres Lizenzmodell ist die Floating-Lizenz, mit welcher es möglich ist, dass die Software auf unterschiedlichen Systemen installiert werden kann. Es wird dabei eine feste Anzahl von Anwendern festgelegt, die gleichzeitig die Software ausführen dürfen. CHIP [6]

Je nach Umfang einer Software verlangen Hersteller bis zu 7000 € pro Benutzer und Jahr, wobei einige Hersteller lediglich individuelle Angebote auf Anfrage anbieten. Generell bietet jeder kommerzielle Anbieter eine Demo- bzw. Testversion, der Umfang und die mögliche Testdauer kann dabei von Anbieter zu Anbieter unterschiedlich sein. Der mögliche Testzeitraum einer solchen Software kann dabei von fünf bis zu 90 Tagen reichen oder im Rahmen einer Light-Version wie bei SoapUi zeitlich unbegrenzt sein.

In dieser Kategorie gewinnen die kostenfreien Applikationen mit jeweils fünf Punkten.

veraltet

Die Tabellen basieren auf Grundlage der folgenden Quellen: Thomas Bucsics [91], Borland [2], Perriault and contributors [60], Perriault and contributors [61], Parasoft [53], Hocke [32], Hocke [33], LP [45], BREDEX [4], IBM [36], Focus [23], Focus [24], BREDEX [3], Interactive [37], Foundation [25], Etestinghub [20], Neumann [51], Deutschland [17], Protractor [63], Ben-Hur [1], Ranorex [73], Ranorex [71], Ranorex [74], Partner [58], Partner [59], Partner [57], Project [62], Corporation [9], Corporation [14], Corporation [12], Corporation [13], Corporation [11], Corporation [15], IST [38], IST [39], Technologies [87], Technologies [86], TestPlant [90], TestPlant [89], TestPlant [88], Tricentis [95], Tricentis [92], Tricentis [96], Tricentis [97], Tricentis [98], Tricentis [94], Tutorialspoint [99], IT-Management [41], Harlan [30], GitHub [28], Microsoft [49], Microsoft [46], LP [44], Software [76], Software [85], Software [77], Software [81], Software [84], Tricentis [93], Microsoft [48], IBM [35], ComponentSource [8], eXept Software [22], eXept Software [21], Parasoft [55], Parasoft [56], Parasoft [54], QFS [64], QFS [66], QFS [65], Software [82], Software [79], Software [80], Software [83], Microsoft [50], Worksoft [105], Worksoft [106], Worksoft [104]

Hinweis: Unter der Sektion „Auswahl der Software“ ist eine Gesamtbewertung der Software unter Berücksichtigung einer doppelten Wertung für Lizenzkosten, Nutzerfreundlichkeit und technologischer Unterstützung zu finden. Diese doppelte Wertung innerhalb dieser Kriterien wurde projektspezifisch entschieden, da die Software zum einen gewisse Technologien unterstützen muss und zum anderen von einem Tester bedienbar sein sollte. Das dritte Kriterium der Lizenz wurde hier aus Gründen der Wirtschaftlichkeit herangeführt. Es ist im Sinne des Unternehmens T-Systems gewünscht, dass die Software zur Testautomatisierung preiswert im Vergleich zum Nutzen dieser Software ausgewählt sein soll.

2.1.2 Kommerzielle Produkte vs. Open-Source-Produkte

Kommerzielle Produkte bieten oftmals Vorteile wie eine „hübschere“, einfachere Bedienung über eine grafische Oberfläche, umfangreichere Auswertungen und grafische Darstellungen von Messwerten. Zudem garantiert der Hersteller je nach Vertrag die Unterstützung des Kunden bei Verwendung des Produktes und oft ist die Integration von anderen Produkten desselben Herstellers kinderleicht umzusetzen.

Kommerzielle Produkte verfügen über eine Sammlung an zur Verfügung stehenden Technologien für die Ausführung von Testfällen, viele solcher Produkte bestehen dabei aus mehreren einzelnen Programmen, die in einer sogenannten Testsuite zusammengestellt sind. Auf der Gegenseite stehen die Open-Source-Produkte, welche vollumfänglich kostenfrei genutzt werden können, selbst für kommerzielle Zwecke.

Meistens werden solche Projekte von einer Community getragen anstatt von einem einzigen Hersteller, wodurch die Programme je nach Größe der Community ständig weiterentwickelt werden und jeder mit gängigen Programmierkenntnissen zu Erweiterungen beitragen kann.

2.2 Anforderungen an die Software für den Verbundtest

Der AD-Client und der CS-Client sollten plattformübergreifend getestet werden. Technologisch basiert der AD-Client auf einer GUI aus HTML-Elementen, welche in einer WIN32-GUI angezeigt werden. Der CS-Client hingegen ist ein reiner Web-Client, welcher auf Basis von Silverlight im Internet Explorer ausgeführt wird.

Die Automatisierungs-Software muss in der Lage sein, verschiedene GUIs und im speziellen Silverlight- und HTML-Elemente zu erkennen und zu verarbeiten.

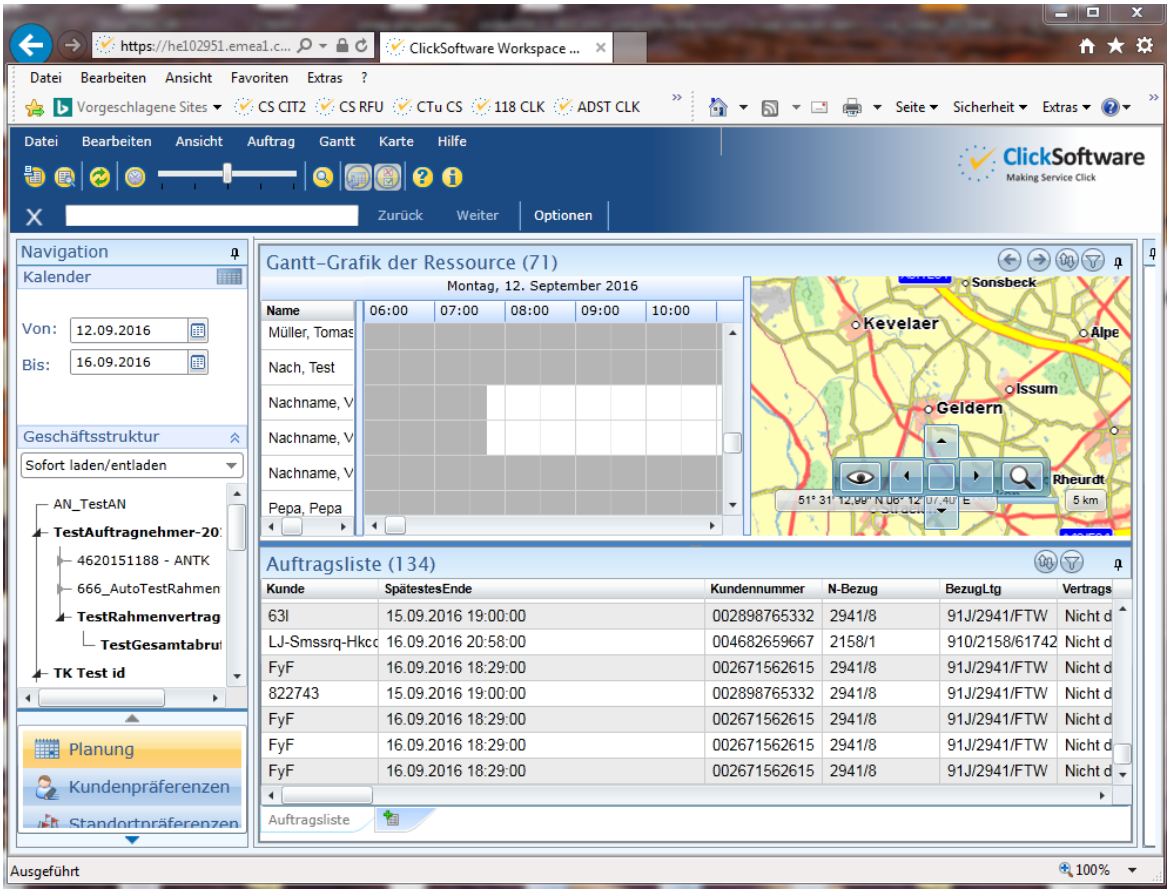

Abbildung 2.1 geladener CS-Web-Client

Die Abbildung 2.1 zeigt dabei die Silverlight-Oberfläche des CS-Clients. Dargestellt ist ein Gantt zur Personaldisposition, auf dem Aufträge, welche vom AD an CS weitergeleitet wurden, durch den Disponenten verarbeitet werden sollen. Ein Auftrag muss dabei anhand einer eindeutigen ID identifiziert werden und durch Drag & Drop von der Auftragsliste auf einen Benutzeraccount eines Außendienstmitarbeiters auf dem Gantt „abgelegt“ werden. Sobald der Auftrag einem Benutzer zugeordnet ist und an erster Position und zeitlich in unmittelbarer Zukunft auf dem Gantt „liegt“, kann der Benutzer sich am AD-Client anmelden und besagten Auftrag zur Bearbeitung laden. Die Abbildung 2.2 zeigt den AD-Client mit einem

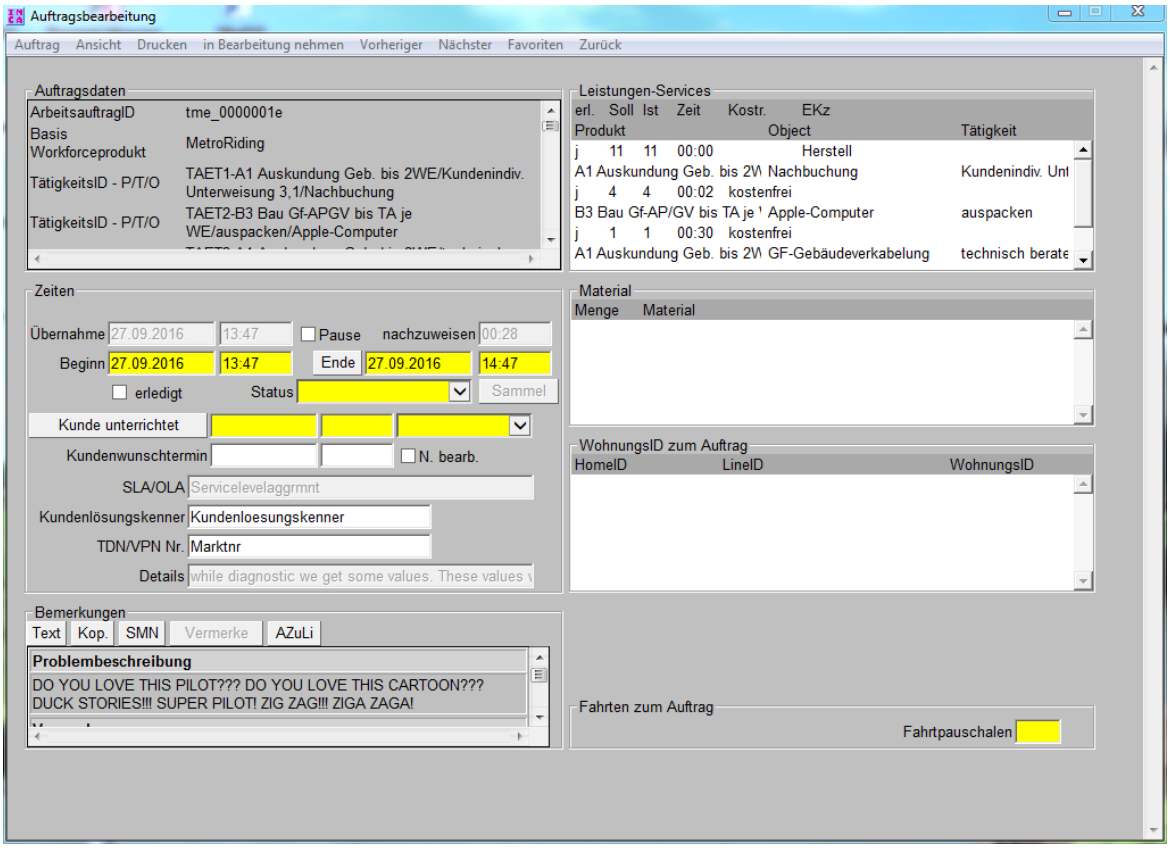

Abbildung 2.2 geladener Auftrag im AD-Client

geladenen, zur Bearbeitung des durch besagten Benutzer (Techniker) geöffneten Auftrags. Die Auftragsmaske besteht aus einer Reihe von HTML-Feldern, welche Informationen für den Techniker bereithalten oder vom Techniker im Sinne der Weiterverarbeitung ausgefüllt werden müssen.

Sobald der Techniker die im Auftrag beschriebenen Tätigkeiten ausgeführt hat, muss im AD-Client der Auftrag „abgeschlossen“ werden. Das heißt, alle Informationen werden im Auftragsformular eingegeben und an den AD-Server versendet.

2.3 Kriterien für die Untersuchung

Die für diese Arbeit definierten Kriterien zur Auswahl der Werkzeuge wurden auf Basis der Interessen des T-Systems Projektes „AD“ in Bezug auf die Testautomatisierung der Verbundtest-Umgebung AD entworfen und sind somit nicht als allgemeingültig zu betrachten.

Untersucht wird die mögliche Integration verschiedener Automatisierungswerkzeuge in den Verbund-Systemtest „AD/CS“, dabei wird die Installation, die Nutzerfreundlichkeit, die Anbindung an andere Systeme, die Qualität der Softwaredokumentation und der Nutzungsumfang/Erfüllbarkeit der Testfälle analysiert.

2.4 Auswahl der Software

Tabelle 2.9 Software-Gesamtbewertung nach Punkten

Software	Gesamt mit Wertung (absteigend)
Telerik Test Studio	69
Ranorex Automation Framework	64
HP Unified Functional Testing (UFT)	61
IBM Rational Functional Tester	61
Tricentis TOSCA TestSuite	59
eggPlant Functional	50
QFTest	5
Micro Focus SilkTest	46
TestComplete	45
Jubula/GUIdancer	44
SoapUI	42
SikuliX	42
Selenium	40
VisualStudio Coded UI Tests	37
TTworkbench	36
expecco	36
RapidRep	35
Parasoft Software Testing Tools	32
Worksoft Certify	28
Protractor	26
CasperJS	25

Die für die Anforderungen wichtigen Kriterien **technologische Unterstützung**, **Lizenzkosten** und **Nutzerfreundlichkeit** wurden innerhalb der Wertung doppelt gezählt.

veraltet

Im Rahmen von Untersuchungen wurden die Applikationen Ranorex Studio, Telerik Test Studio HP Unified Functional Testing (UFT) und SikuliX ausgewählt, weil diese den Anforderungen entsprechen. Außerdem wird, um ausreichend und aussagekräftig Testen zu können, für Untersuchungen eine Testversion von mindestens drei Wochen Testzeit benötigt.

Eine Begründung für die doppelte Wertung der genannten Kriterien befindet sich im Hinweis auf Seite 25.

Die Applikationen von HP, Ranorex und Telerik wurden aufgrund der höchsten Wertung nach Tabelle 2.9 ausgewählt. Da nach Möglichkeit auch eine kostenfreie Software untersucht werden sollte, wurde sich als viertes Tool für SikuliX entschieden. Wichtig bei der Auswahl einer geeigneten Software war auch, dass alle dieser Tools mit Silverlight kompatibel sind. Laut Herstelleraussagen unterstützen die Applikationen Silverlight bzw. sind wie SikuliX unabhängig von der Art und Technologie der Benutzeroberfläche.

Die Entscheidung viel gegen Visualstudio, IBM, eggPlanet und Triscentis, auch wenn diese Silverlight unterstützen würden, weil die Nutzerfreundlichkeit dieser Softwarewerkzeuge im Vergleich zu den recht teuren Lizenzgebühren nicht dem Optimum entspricht.

2.5 Untersuchung von Software für Testautomatisierung

2.5.1 Installationsaufwand

Die Ranorex 6.1-Installation erforderte Visual C++ Runtime, der Installer hat alles bereitgestellt und die Installation benötigte 20 Minuten. Plugins für Google Chrome, Firefox und Internet Explorer wurden automatisch mit installiert und mussten beim jeweiligen Browserstart einmalig aktiviert werden.

Das Telerik 2016.2 benötigte zehn Minuten zur Installation. Plugins für Google Chrome, Firefox und Internet Explorer wurden automatisch installiert und mussten beim jeweiligen Browserstart einmalig aktiviert werden.

Für die Software SikuliX wird zunächst eine Java Runtime Environment benötigt, sofern diese schon installiert ist, kann der Setup durch eine GUI-Anwendung direkt gestartet werden. Während des Setups werden vier verschiedene Jar-Files erstellt. Die Installation als solche benötigt wenige Minuten. Außer einer kurzen Information im Setup gibt es keine Hilfestellung, dafür ist aber auf der Webseite des Herstellers eine Quickstart-Anleitung beschrieben. Bei der Installation kann der Nutzer auswählen, ob er die sikuliXeigene IDE für Python oder Ruby oder aber eine Integration in eine vorhandene IDE wie Eclipse oder Netbeans nutzen will, wobei die letztere Variante auch noch die Umsetzung der Scripte in Java ermöglicht. Standard ist jeweils Python (Jython) als Skriptsprache.

Die Installation der Software HP inklusive Microsoft Access Database Engine benötigte in etwa 2,5 Stunden, wobei zwei Stunden Zeit für den Download der mehrere Gigabyte großen Datei dazugerechnet werden. Ähnlich der Software von Ranorex und Telerik wurden Plugins für Google Chrome, Firefox und Internet Explorer automatisch installiert und mussten beim jeweiligen Browserstart einmalig aktiviert werden. Sowohl Installationsprozess als auch die Software waren automatisch in der Sprache des Benutzers, in diesem Falle Deutsch.

Die folgende Tabelle soll den Installationsaufwand anhand der Kriterien Zeit, Vorbedingungen und Nutzerfreundlichkeit darstellen:

Tabelle 2.10 Installationsaufwand

Software	Installationsdauer (ohne Vorbedingungen)	Vorbedingungen	Nutzer-freundlichkeit des Installers	Summe
Ranorex	20 Min. (2)	Visual C++ Runtime (0)	3	5
Telerik	10 Min. (2)	keine (2)	3	7
SikuliX	2 Min. (2)	Java Runtime Environment (0)	1	3
HP UFT	4,5 Std. (0)	MS Access DB Engine (0)	3	3

Die kostenpflichtigen Applikationen überzeugen durch einen grafischen und benutzerfreundlichen Installationsprozess. Die kostenfreie Applikation SikuliX ist minimal schwerer zu installieren. Bei allen Applikationen ist der Installationsprozess innerhalb der jeweiligen Dokumentation verständlich beschrieben. Die unkomplizierteste Installation war die des Telerik Test Studios. Eine jeweilige genaue Abfolge der Installation ist dem Anhang unter Punkt 2 zu entnehmen.

2.5.2 Erstellung eines ersten Testfalls

Erstellung eines ersten Testfalls in Ranorex Studio

Nach dem Start von Ranorex Studio bietet die Software eine übersichtliche Menüführung und eine Reihe von Hilfsdialogen, die den Benutzer durch das Programm führen, dargestellt ist dies in Abbildung 2.3.

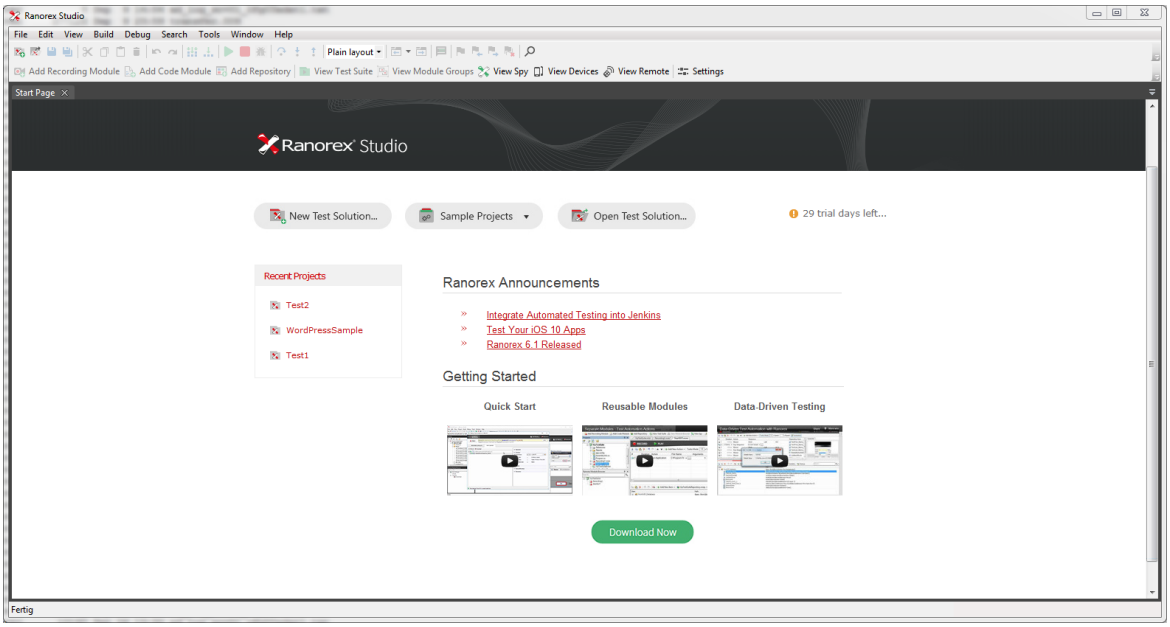

Abbildung 2.3 Startansicht von Ranorex Studio

Unter dem Menüpunkt „Help“ befinden sich für den Nutzer wichtige Informationen zur Verwendung der Software. Die Dokumentation und die Software selbst sind nur auf Englisch verfügbar. Mit über 600 Seiten ist diese sehr umfangreich und mit Bildern belegt dargestellt. Um einen ersten Testfall zu erstellen, kann der Benutzer einen Beispieltestfall laden, sofern dieser den Kriterien der zu untersuchenden Software ähnelt. Es gibt zu jeder Kategorie wie Desktop, Mobile oder Browser einen Testfall, der auf ein Minimum reduziert ist, wie zum Beispiel das Anmelden und Verfassen eines Posts in einer Demo-Wordpress-Installation.

Um einen komplett eigenen Test zu erstellen, führt das Programm den Benutzer durch verschiedene Masken. Der Benutzer kann sich dafür entscheiden, ob das zu erzeugende Projekt in C# oder VBNet angelegt und verwaltet werden soll. Einem Projekt können mehrere sogenannter Solutions untergeordnet sein. Aufgebaut ist das Ganze in einer Art Klassenstruktur. Es gibt Dateien und Ordner für Konfiguration, Test-Berichte, Referenzen auf Plugins und den eigentlichen „Testschritten“.

Ein Testschritt ist in diesem Fall nicht als die kleinste atomare Aktion wie ein Klick auf eine Oberfläche zu deuten, sondern entspricht einem übergeordneten Schlüsselwort wie zum Beispiel „Login“ oder „Logout“. Diese Schlüsselwörter können als Bausteine innerhalb einer Maske zu neuen Interaktionen/Testfällen per Drag & Drop oder durch Copy & Paste zusammengetragen werden. Die einfachste Art, einen Testfall anzulegen, ist es, diesen manuell zu testen und durch den eingebauten Recorder von Ranorex zu erfassen. Dabei unterstützt der Recorder verschiedene Modi wie bildbasierte oder textbasierte Erkennung und eine manuelle oder automatische Mausverfolgung. Als Grundeinstellung verfolgt der Recorder alle Aktionen mit der Maus und der Tastatur automatisch. Dank einer Benutzeroberfläche des Recorders kann die Aufzeichnung besagter Testschritte jederzeit abgeschlossen oder nur pausiert werden.

Abgeschlossene Aufzeichnungen werden als Code und als atomare Testschritte in einer GUI angelegt und können somit auf Basis der gewählten Programmiersprache oder per Drag & Drop und anhand von Auswahlmenüs bearbeitet werden. Die Auswahlmenüs erscheinen jeweils mit Klick auf einen der atomaren Testschritte. Bei einem Klick zum Beispiel auf die Aktion „Mouse“ kann der Benutzer die Bewegung der Maus (klicken, verschieben, gedrückt halten, loslassen), aber auch die zu interagierende Taste (keine, links, rechts, mitte, Zusatzbutton1, Zusatzbutton2) bearbeiten.

Negativ fällt auf, dass die Software sehr viele Systemressourcen benötigt und sich während des ersten Testens einige Male aufgehängt hatte und neugestartet werden musste.

Erstellung eines ersten Testfalls im Telerik Test Studio

Nach dem Start von Telerik Test Studio 2016.2 erwartet den Benutzer eine Auswahlmaske, dargestellt ist dies in Abbildung 2.4.

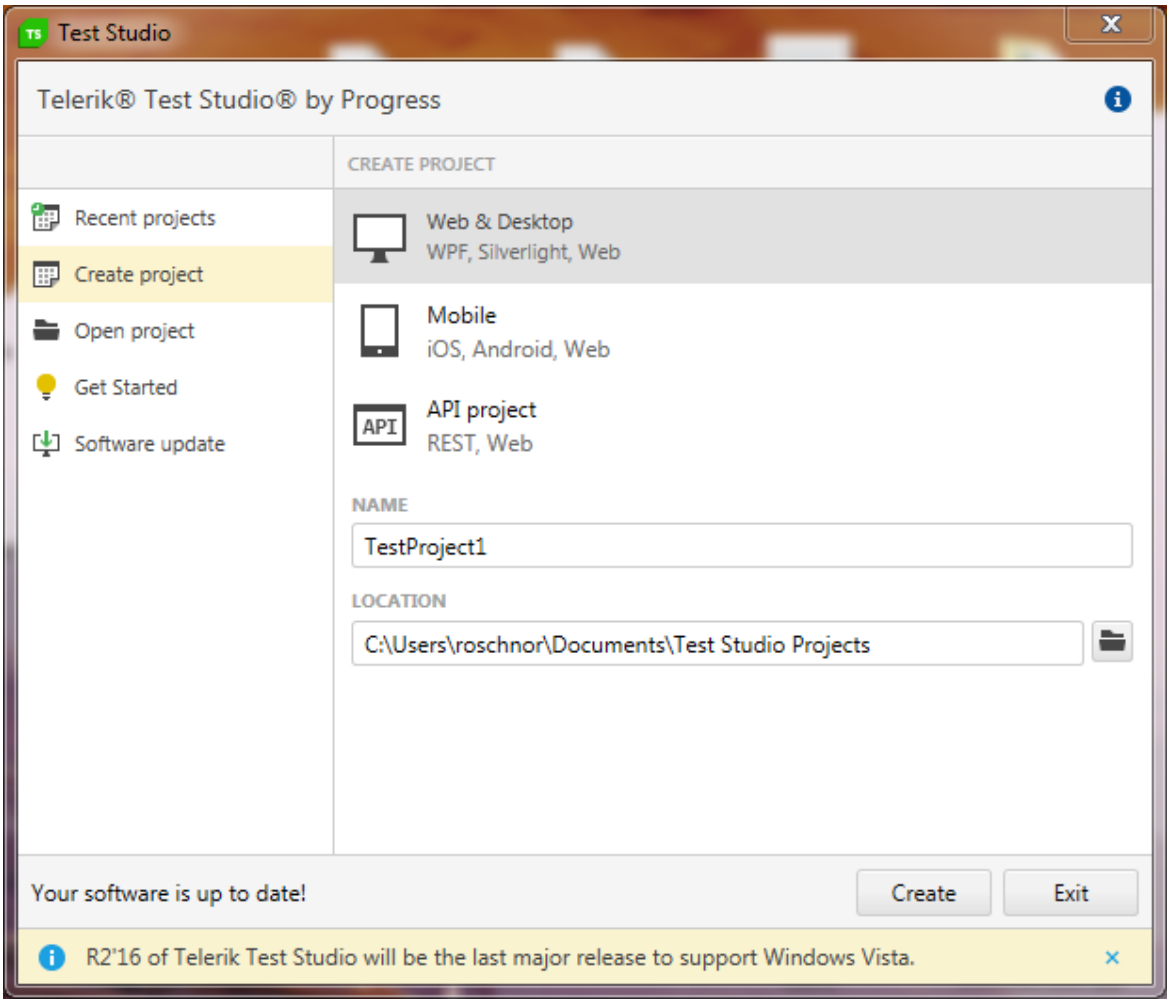

Abbildung 2.4 Startansicht von Telerik Test Studio

So ist es möglich, Beispielprojekte für Web & Desktop, Mobile Testing oder API Testing über den Auswahlpunkt „Get Started“ anzulegen. Die Projekte werden dabei jeweils durch ein Einführungsvideo erläutert. Weitere Auswahlpunkte sind Updates der Software, Öffnen von Projekten, Übersicht über aktuelle Projekte und das Erstellen von neuen Projekten. Es wird sich an dieser Stelle mit dem Punkt „Create Project“ befasst.

Der Benutzer hat die Möglichkeit auszuwählen, für welche Art von Technologie das Testprojekt erstellt werden soll. Zur Auswahl stehen WPF, Silverlight und Web-Applikationen unter der Kategorie „Web & Desktop“, iOs, Android und Web unter der Kategorie „Mobile“, als auch REST und Web unter der Kategorie „API project“. Zu Testzwecken wurde sich für

die Kategorie „Web & Desktop“ entschieden und dort ein erstes Projekt mit dem Namen „TestProject1“ erstellt. Die Software bietet für die einfache Erstellung von Testfällen einen Record-Modus, mit dessen Hilfe können manuell ausgeführte Testschritte atomar aufgezeichnet werden. Der Recorder benötigt sehr viele Systemressourcen, so dass es ratsam ist, keine weiteren Applikationen ausführen zu lassen.

Generell fügt sich der Recorder gut in Webseiten und andere GUIs ein. Zu Beginn sollte er mit einem einfachen Testlauf wie z. B. der Aufruf von Google und Simulation von Suchanfragen und Klicks „aufgewärmt“ werden. Nach einer Aufwärmphase ist der Recorder in der Lage, sämtliche atomaren Aktionen bis hin zum komplizierten Drag & Drop einzeln aufzuzeichnen und als Testschritte in einem „Storyboard“ zu speichern. Die einzelnen atomaren Testschritte lassen sich mittels Drag & Drop neu zusammenstellen und durch einen Doppelklick in ihren Einstellungen bearbeiten.

Innerhalb dieser Einstellungen können nachträglich Wartezeiten definiert werden. Ansonsten verwendet das Programm für die Ausführung der Testschritte vordefinierte Werte wie zum Beispiel 15 Sekunden. So kann bei einer Mausgeste die simulierte Taste (keine, links, rechts, mitte, Zusatzbutton1, Zusatzbutton2) ausgewählt werden. Bei einem Tastendruck kann zusätzlich die Häufigkeit und Dauer des Drückens vorgegeben werden.

Jeder der Testschritte kann beliebig umbenannt, gelöscht oder dupliziert werden. Durch einen zusätzlichen „Step Builder“ kann der Benutzer Aktionen oder Überprüfungen von oder auf Elemente manuell dem Test hinzufügen wie zum Beispiel den Klick auf ein Element, den Aufruf einer URL oder die Überprüfung des Textwertes eines Textelementes.

Erstellung eines ersten Testfalls in SikuliX

SikuliX startet als eine Entwicklungsumgebung. Wie auf Abbildung 2.5 zu sehen, ist die Software automatisch auf Deutsch umgestellt worden, lediglich die Hilfe bzw. Dokumentation sind auf Englisch.

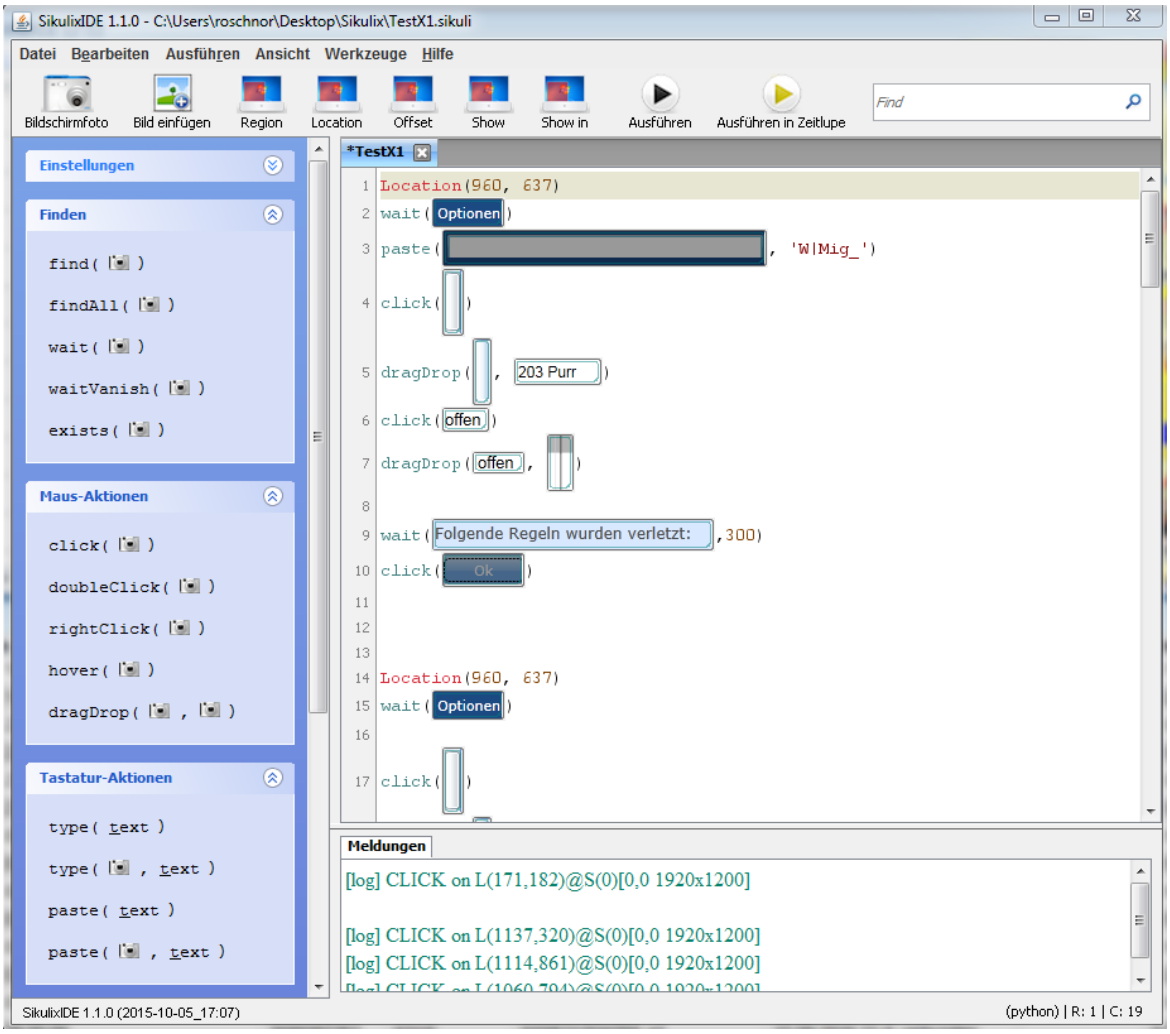

Abbildung 2.5 Startansicht der SikuliX

Einen Recorder zur Aufzeichnung von manuellen Testschritten gibt es in SikuliX nicht, jedoch findet sich ein Nutzer, der nur wenige Programmierkenntnisse hat, mit etwas Ausprobieren in der Software zurecht. Die Software arbeitet mit einer bildbasierten Objekterkennung, wobei es vordefinierte Aktionen gibt, die aus einem Menü ausgewählt werden können. Jeder Aktion wird, je nach Typ der Aktion, entweder ein Bildschirmausschnitt des Elementes oder Text übergeben. Nach dem Auswählen einer bildbasierten Aktion wird der Benutzer dazu aufgefordert, einen Bildschirmausschnitt mit einem „Snipping Tool“ auszuwählen. Mit

einem Mouse-Over über die Menüeinträge erfährt der Benutzer, was eine jeweilige Aktion simuliert.

Innerhalb des Editors ist es möglich, solche Aktionen zu kopieren, zu löschen oder zu bearbeiten. Die Bearbeitung erfolgt auf Grundlage der ausgewählten Skriptsprache und unter Beachtung der Dokumentation. Durch einen Klick auf den dargestellten Bildschirmausschnitt innerhalb der Funktion öffnet sich eine neue Benutzeroberfläche, in welcher der Benutzer den „Klickpunkt“ und die Genauigkeit der Bilderkennung einstellen kann. SikuliX untersucht im Rahmen der Bilderkennung den ganzen Bildschirm, was bei einem einfachen Test zur fehlerhaften Erkennung von Elementen führte. So ist es zum Beispiel möglich, dass ein ähnlicher Button in der Applikation mehrfach auftritt und damit auch mehrfach und fälschlicherweise als gesuchtes Objekt erkannt werden kann.

Bei mehreren ähnlichen Elementen verwendet die Software in diesem ersten Beispieltest immer das Element mit der größten Objektgleichheit und bei nahezu identischen Objekten das auf der Bildschirmfläche von oben links nach unten rechts als erstes auftretende Objekt.

Erstellung eines ersten Testfalls in HP UFT

Ähnlich zu SikuliX startet die Software von HP, wie auf Abbildung 2.6 zu sehen, automatisch auf Deutsch, lediglich die Hilfe bzw. Dokumentation sind auf Englisch.

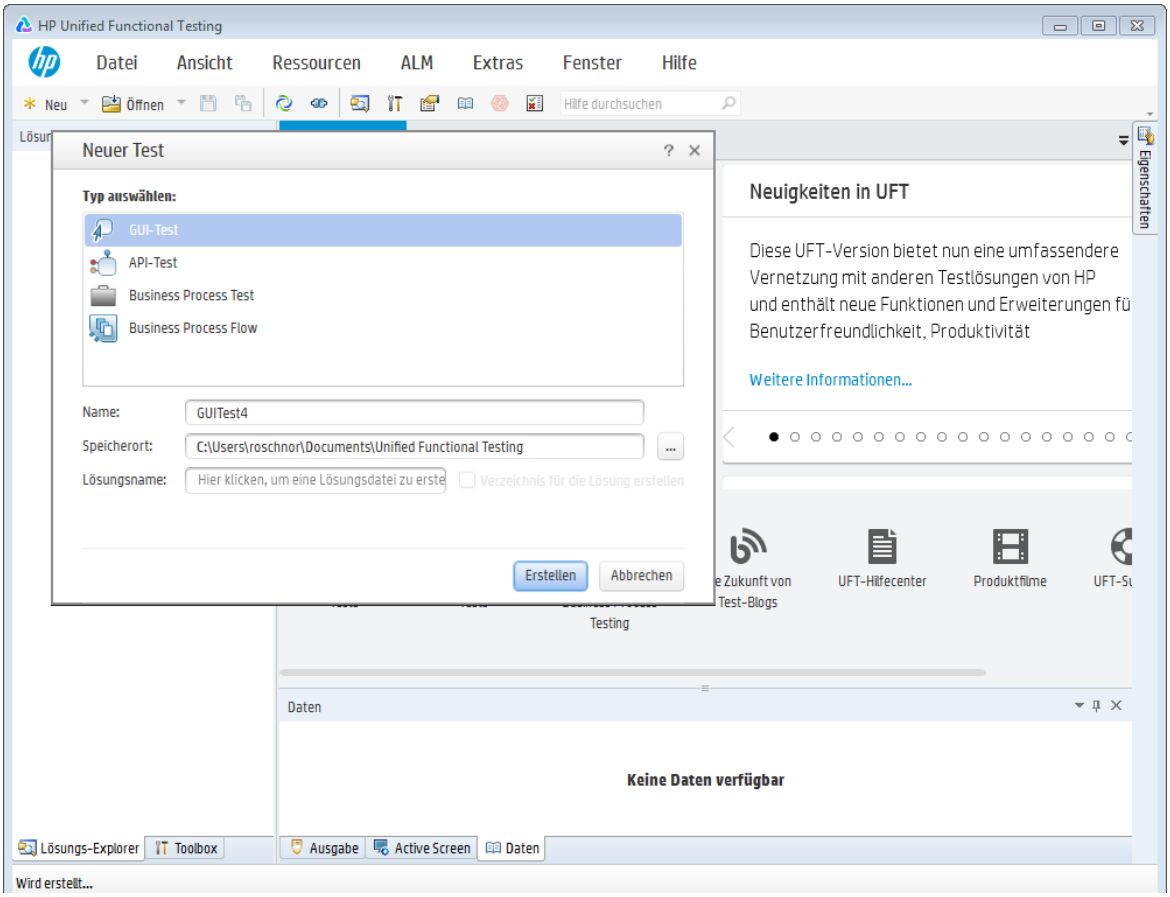

Abbildung 2.6 Startansicht der HP UFT

Die Startansicht der HP Unified Functional Testing zeigt dem Nutzer direkt wichtige Verknüpfungen zur Hilfestellung wie zum Beispiel Produktfilme oder ein Forum. Unter dem Menüpunkt Datei hat der Benutzer die Möglichkeit, Tests zu öffnen, Beispieltests wie einen GUI Test von HP Sprinter zu laden oder eigene Tests zu erstellen. Einzelne Tests lassen sich gruppiert als eine „Lösung“ zusammenfassen, wobei jeder dieser Tests auch aus einzelnen Aktionen bestehen kann. Eine Aktion ist zum Beispiel das Anmelden an einem Client einer Software mit allen dafür notwendigen Testschritten. Als Testart kann jeweils GUI-Test, API-Test, Business Process Test oder Business Process Flow ausgewählt werden.

Die jeweiligen einzelnen Aktionen lassen sich zu neuen Tests zusammenstellen. Diese Zusammenstellung kann über eine Schnittstelle, als Programm-Code oder innerhalb einer GUI via Drag & Drop erfolgen. Die Darstellung eines Tests in der GUI-Variante entspricht in

etwa einem Prozess-Diagramm inklusive Start, Ende und den jeweiligen Pfaden der verknüpften Aktionen. Insgesamt stehen dem Nutzer auch verschiedene Möglichkeiten der Planung, Analyse und Aufbereitung zur Verfügung. Selbst eine Integration von Testdaten, anderen Ressourcen und im Speziellen ALM-Tools wie zum Beispiel Jira bietet diese Software werksseitig an.

Die Erstellung eines Tests kann innerhalb von UFT auf mehrere Arten erfolgen. Neben der Variante des Programmierens und dem Zusammenstellen von Aktionen via Drag & Drop bietet die Software von HP einen integrierten Recorder zur Aufzeichnung von manuellen Aktionen und Umwandlung in atomare und automatisierbare Testschritte. Dabei unterstützt der Recorder mehrere Modi für alle Arten von zu automatisierenden Applikationen. Neben dem Standard-Modus, der Elemente nur anhand der Bildschirmposition erkennt, und einem bildbasierten Modus (ähnlich zu SikuliX) gibt es zusätzliche Modi, die Mausaktionen quasi analog aufzeichnen (nicht als einzelne Schritte) oder UI-basiert ähnlich zu DOM arbeiten.

Die jeweiligen Testschritte lassen sich zusätzlich wie in SikuliX als Funktionen mit Übergabeparameter bearbeiten, kopieren oder löschen. Zusätzliche Testschritte lassen sich dabei über einen Testschrittgenerator innerhalb der jeweiligen Aktion ergänzen.

2.6 Mögliche Testfallabdeckung

Im Rahmen der Anforderungen an einen Verbundtest wird an dieser Stellen von Testfällen ausgegangen, die einen Auftrag von der Disposition in CS, der Bearbeitung im AD bis zur Überprüfung in CS durchlaufen.

Ein möglicher Testfall sieht dabei wie folgt aus:

Lokale Vorbedingung:

Es wird ein WMS-TK-OrderLine-Auftrag erstellt. Da dieser Auftrag korrekt auf den benutzten Techniker vorgeschlagen werden soll, müssen bei der Auftragserstellung folgende Punkte beachtet werden:

Vorbedingungen:

- Der zu verwendende Techniker wird als benötigter Techniker gesetzt.
- Der Techniker muss die benötigten Skill(s) für den Auftrag haben.
- Der Techniker muss die benötigten Auftragsklassen besitzen.
- Der Kundenstandort muss innerhalb des Aktionsradius des Technikers sein.
- Der Techniker besitzt einen Kalender.
- Der Auftrag wurde mittels Simulator nach AD eingespielt und befindet sich anschließend in CS im Status „offen“.

Testdurchführung:

1. Der Tester meldet sich in der Rolle Disponent am ClickSchedule Client an.
2. Der Tester prüft, ob der orderLine–Auftrag in CS den SubDistrict erreicht hat (Auftragsliste).
3. Der Tester disponiert den Auftrag aus der lokalen Vorbedingung auf den ADM.
4. Der Tester meldet sich am AD-Client an und wechselt in die Auftragsbearbeitung mit dem Button „Auftrag laden“ im Hauptmenü.
5. Der Tester füllt alle Pflichtfelder des Auftrags mit plausiblen Daten und wählt dabei keinen Rückgabegrund aus.

6. Der Tester schließt den Auftrag ab (Menü: Auftrag > Abschließen).
7. Der Tester schickt den Auftrag zum Server (Versenden). - Das „Versenden“ erfolgt aus der Hauptmaske.
8. Der Tester meldet sich vom AD-Client ab.
9. Der Tester wechselt zum ClickSchedule-Client und überprüft die Statuswerte des Auftrags.
10. Der Tester meldet sich vom ClickSchedule-Client ab.

Im Folgenden wird zunächst die mögliche Umsetzung dieses Testfalls anhand der Kriterien Durchführbarkeit, Zeitdauer und Fehleranfälligkeit untersucht.

2.6.1 Durchführbarkeit von Testfällen

Im Rahmen der Untersuchung der Durchführbarkeit einer Automatisierung des zuvor genannten Testfalls lassen sich zunächst folgende Aspekte betrachten:

- Verwendbarkeit der Software für die Automatisierung des AD-Clients
- Verwendbarkeit der Software für die Automatisierung des CS-Clients
- Wechsel zwischen den Applikationen und fehlerfreier Durchlauf des gesamten Testfalls (drei Durchgänge)

Dabei werden jeweils null bis fünf Punkte für die Einfachheit der Testfall-Erstellung, der Einfachheit der Änderung von Testdaten und für die Ausführbarkeit des Testfalls vergeben. Null bedeutet, dass einer dieser Aspekte nicht anwendbar gewesen ist. Ansonsten ist fünf die höchstmögliche Punktzahl im Optimalfall. Sofern von diesen Applikationen keine besonders positiv zu bewerten ist, erhält der „Gruppensieger“ drei Punkte, der „Zweitplatzierte“ zwei Punkte und der „Drittplatzierte“ einen Punkt innerhalb des jeweiligen Aspektes unter Betrachtung von Zeitaufwand und Fehlerhäufung.

Verwendbarkeit der Software für die Automatisierung des AD-Clients

Die folgende Abbildung zeigt den geladenen Auftrag im AD-Client, nachdem dieser im CS-Client bereits angewiesen wurde. Dies entspricht dem Punkt fünf des zuvor festgelegten Beispiel-Testfalls.

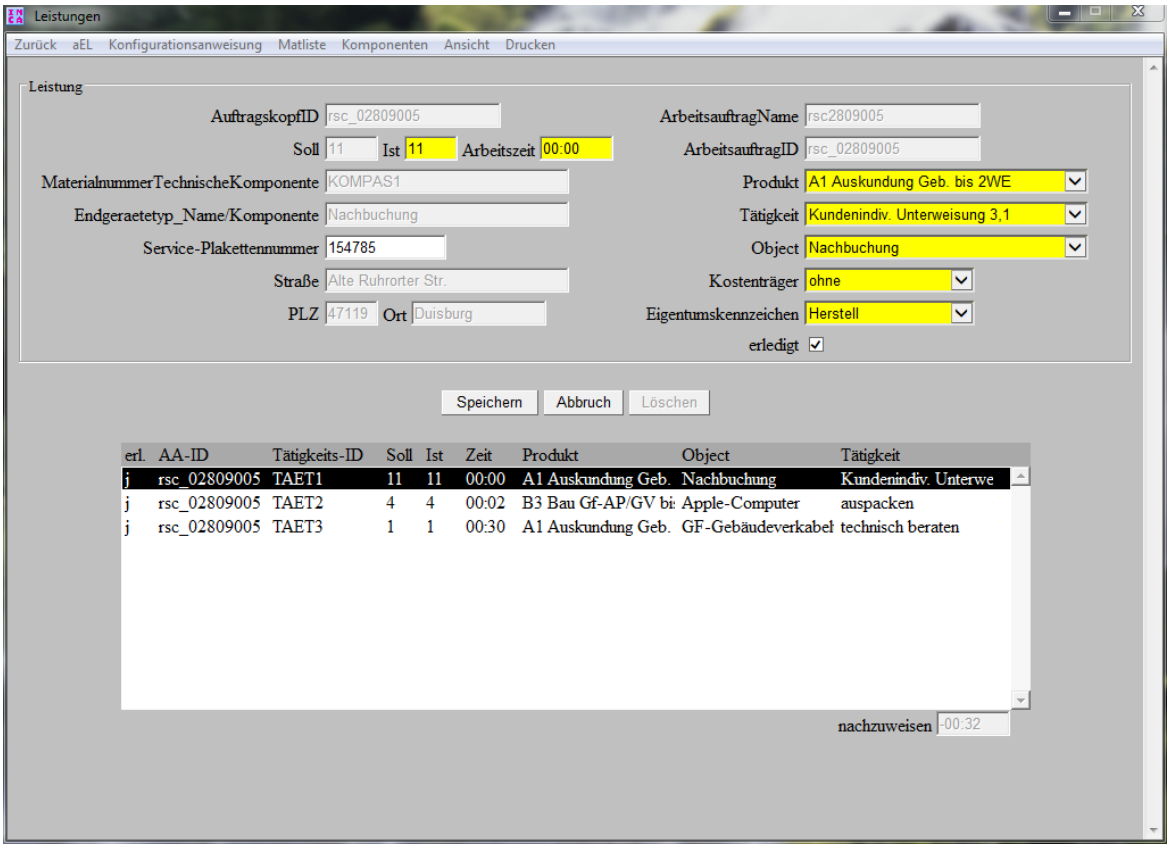

Abbildung 2.7 geladener Auftrag im AD-Client

Anders als ursprünglich erwartet, ist das Telerik Test Studio inkompatibel zum Html-basierten AD-Client. Auf Grund dessen scheidet die Software für die Untersuchung des gesamten Testfalls aus.

Die Applikationen Ranorex Studio und HP UFT liegen in etwa gleich auf. Beide benötigten in diesem Beispiel eine Stunde für die Erstellung des Testfalls und wenige Minuten für etwaige Anpassungen wie zum Beispiel die Änderung des verwendeten Technikers bei der Anmeldung. Was die Ausführungszeit betrifft, war SikuliX mit vier anstatt sechs Minuten (Ranorex) bzw. sieben Minuten (HP UFT) etwas schneller. Dafür war die Erstellung des Testfalls nicht so komfortabel wie beim Ranorex Studio. Ranorex zeichnete dabei anders als HP UFT direkt die „Wartezeiten“ zwischen Testschritten auf. Bei HP UFT mussten diese als Funktion manuell ergänzt werden.

Der integrierte Recorder von Ranorex ermöglichte eine sehr benutzerfreundliche Erstellung der Testschritte, indem er den manuellen Test anhand der DOM-Struktur des HTML-basierten AD-Clients direkt im Ranorex Studio abbildete. An einer Stelle war ein manuelles Eingreifen vonnöten, da es sich um ein HTML-Listenelement mit variablem Bezeichner handelte und diese Variabilität erst dem Ranorex Studio „beigebracht“ werden musste. Diese geforderte Variabilität konnte sehr einfach mittels Klicken auf den Testschritt und einem eigens dafür ausgelegten Auswahlmenü festgelegt werden. Bei der Applikation HP UFT gibt es einen benutzerfreundlichen Recorder, jedoch ist die Nachbearbeitung und die Erstellung zusätzlicher Testschritte als komplex zu bewerten. Es gibt einen Testschrittgenerator, aber Kenntnisse in der Programmiersprache VBScript sind von Vorteil.

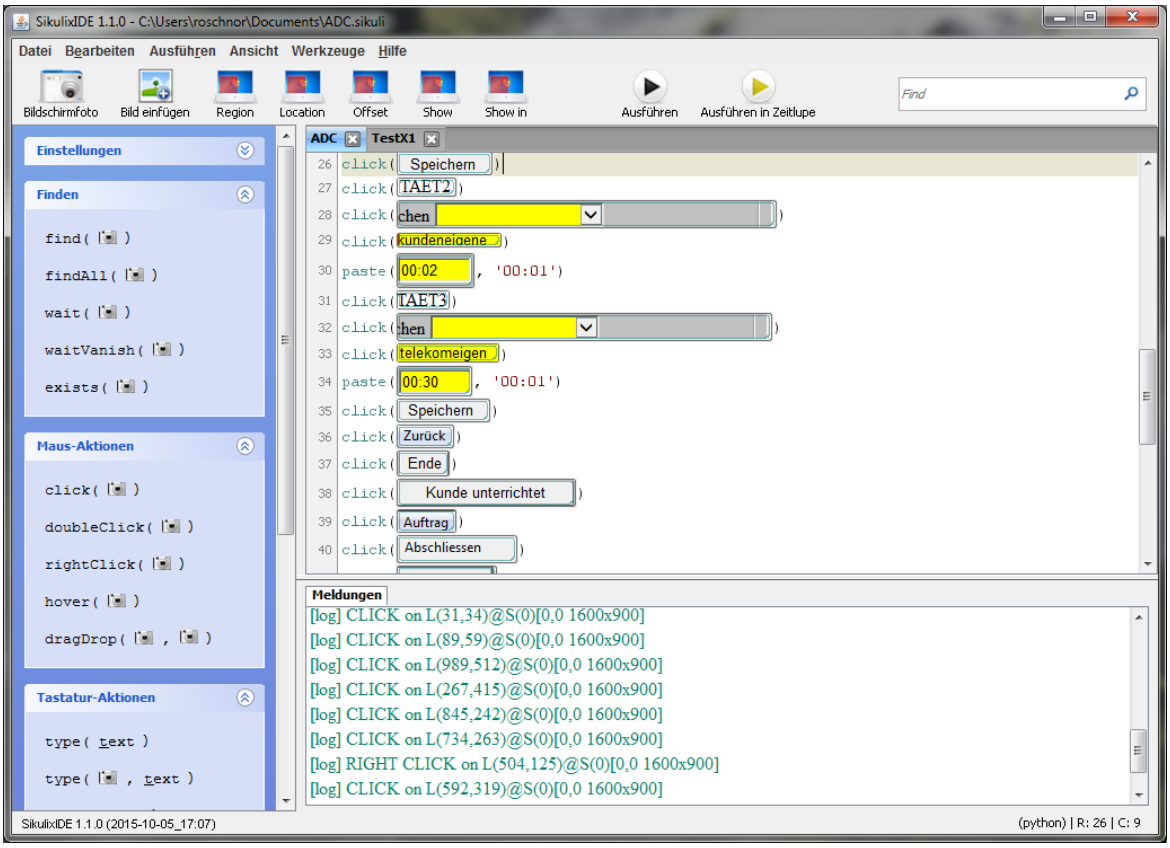

Abbildung 2.8 Test des AD-Clients mit SikuliX

Ein anderer wichtiger Aspekt war die Fehlerfreiheit der Ausführung, welche von allen drei Programmen ohne Probleme beherrscht wurde. Anders als zuvor kam es beim Testen der GUI des AD-Clients zu keinen Abstürzen von Ranorex und SikuliX. Außerdem liefen die Testfälle bei drei Durchläufen mit den jeweiligen Tools fehlerfrei ab. Um diese Fehlerfreiheit zu erreichen, war es wichtig, wie in Abbildung 2.8 zu sehen ist, den Bildschirm-Auswahlbereich möglichst groß zu wählen, damit SikuliX die gesuchten Elemente besser identifizieren kann.

Die nachfolgende Tabelle zeigt den direkten Vergleich der Tools in Bezug auf Testfall-Erstellung, Änderung und Ausführung.

Tabelle 2.11 Test der Software anhand des AD-Clients

Software	Testfall-Erstellung	Änderung von Testdaten	Ausführbarkeit	Summe
Ranorex	3	3	2	8
Telerik	0	0	0	0
SikuliX	2	2	3	7
HP UFT	2	3	3	8

Fazit zu diesem Abschnitt: Sowohl das Ranorex Studio als auch HP UFT und SikuliX eignen sich zum automatischen Testen des AD-Clients. Die Software von Telerik scheidet aufgrund von Inkompatibilität aus.

Verwendbarkeit der Software für die Automatisierung des CS-Clients

Zur Untersuchung findet die zuvor definierte Testdurchführung, im Genauen die Testschritte ein bis drei und 10 Verwendung. Beschränkt wird sich dabei auf die Disposition eines Auftrags auf einen Techniker, der keine Aufträge an einem zuvor definierten Datum auf seinem Gantt hat. Für diesen Testlauf wurde der 02.10.2016 als Datum und die zu diesem Zeitpunkt aktuelle Uhrzeit von 15 Uhr gewählt, welche im CS-Client mit einer Linie auf dem Gantt veranschaulicht wird (siehe Abbildung 2.9).

Um diese Position auf dem Gantt „auffindbar“ zu machen, musste innerhalb des CS-Clients das Gantt nach unten und nach rechts gescrollt werden. Die eingestellte Auflösung des Bildschirms betrug 1600 x 900 Pixel und der CS-Client wurde im maximierten Internet Explorer ausgeführt. Zur optimalen Nutzung des CS-Clients wurde in HP UFT das Silverlight/WPF Add-In nachträglich installiert und aktiviert.

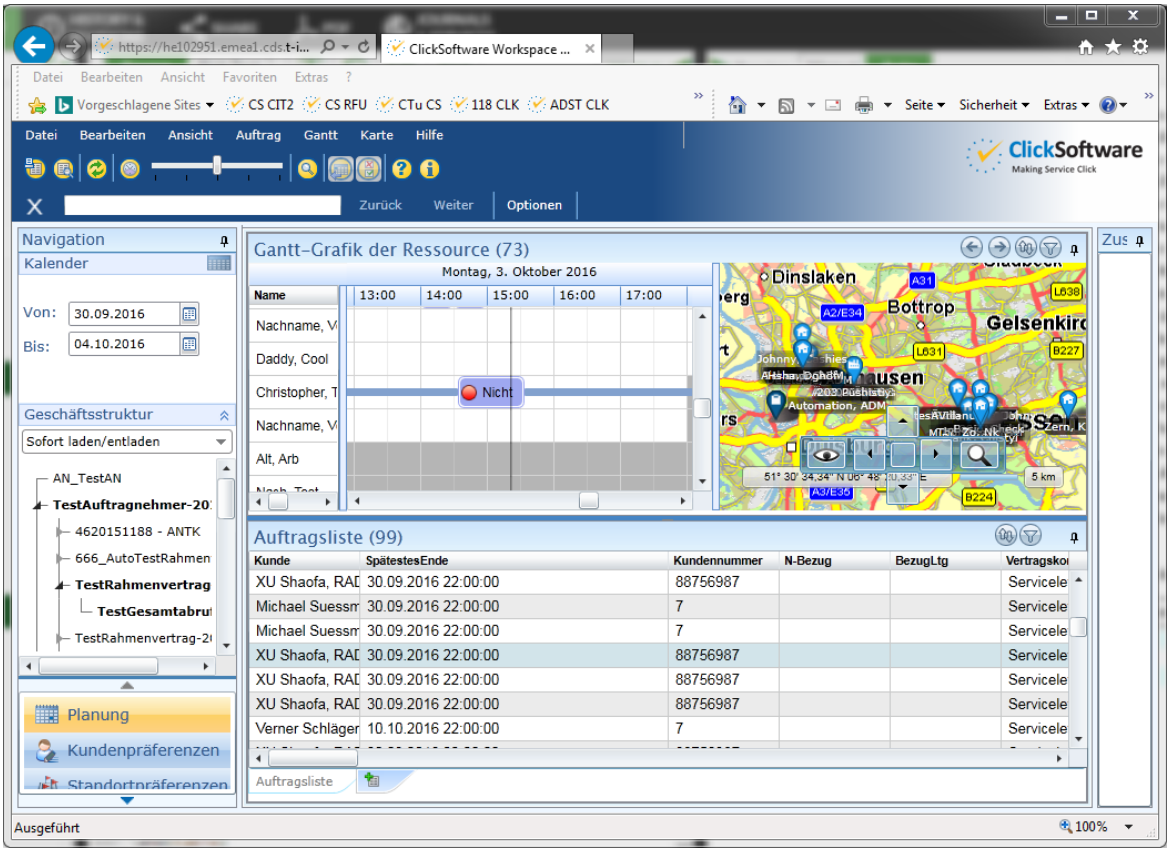

Abbildung 2.9 disponierter Auftrag im CS-Client

Mit 20 Minuten Aufwand zur Erstellung dieses Testfalls schneidet die Software von Telerik am besten ab. Im Punkt Ausführbarkeit erzielt die Software das beste Ergebnis der untersuchten Software. Es kam zu keinen Programmabbrüchen, die Software wartete auto-

matisch eine vordefinierte Zeit auf das Erscheinen von Elementen und selbst die Erkennung und Verschiebung des Auftrags aus der Auftragsliste auf das Gantt des zu verwendenden Technikers mittels Drag & Drop führte zu keinem Problem.

Die Abbildung 2.10 zeigt die jeweiligen automatisch erstellten Testschritte dieses Testfalls innerhalb der Software Telerik Test Studio.

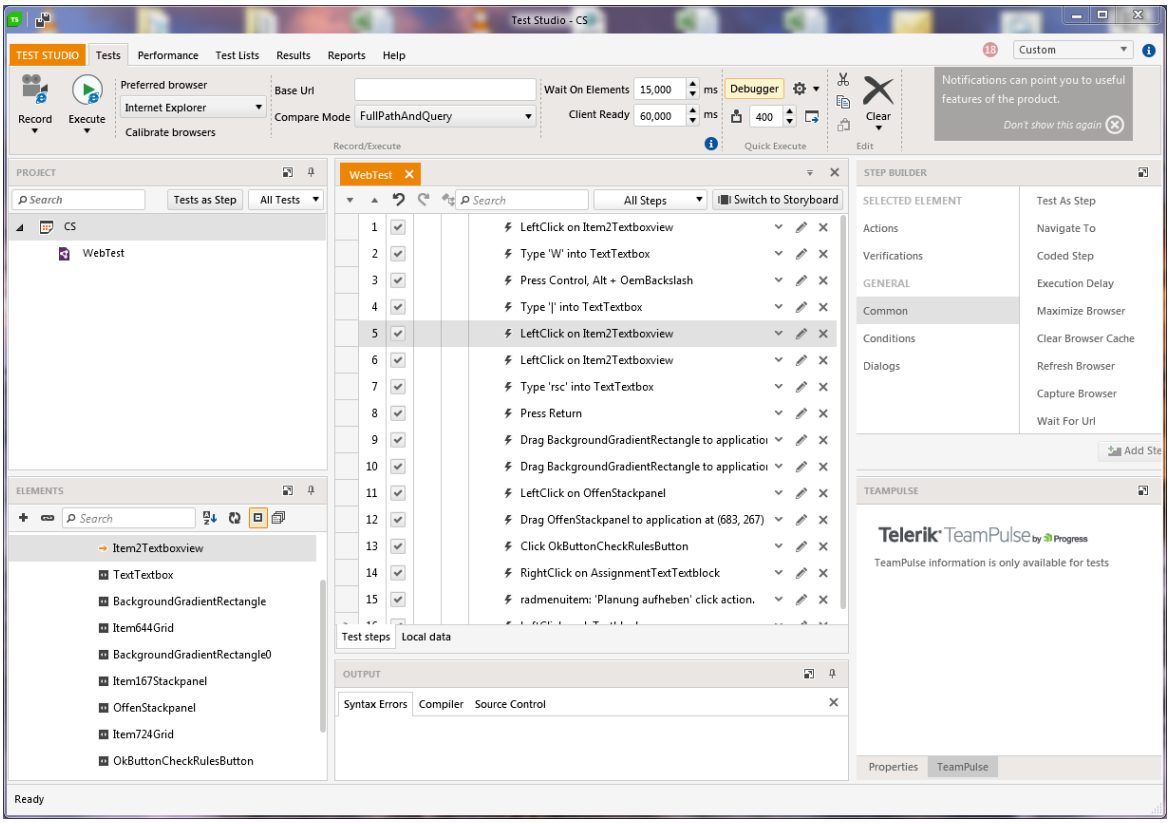

Abbildung 2.10 Test des CS-Clients im Telerik Test Studio

Die Applikation SikuliX lag bei der Erstellung des Testfalls und dessen Ausführung leicht hinter der Applikation von Telerik, da diese keinen automatischen Recorder besitzt und somit Mehrarbeit bei der Erstellung des Testfalls anfällt. Bei der Ausführung war SikuliX bei drei von drei Durchläufen fehlerfrei, jedoch etwas langsamer als das Telerik Test Studio. Ähnlich wie die Software von Telerik konnte das HP UFT mit einer komfortablen Aufzeichnung von einzelnen Testschritten punkten, aber es mussten notwendige Lade- bzw. Wartezeiten manuell in den Testfall implementiert werden.

Bei der Software von Ranorex kam es hingegen bei der Erstellung des Testfalls zu Problemen, der Recorder von Ranorex musste aufgrund von Programmabstürzen des Öfteren neu gestartet werden und nicht jeder Testschritt wurde auf Anhieb erkannt und musste manuell nachbearbeitet werden. Lediglich, was die Anbindung von Testdaten bzw. deren

Änderung betrifft, kann die Software durch ein System von Variablen punkten. Dabei können Testdaten aus einer Datenbank, Excel-Datei oder Ähnlichem importiert werden. Ranorex [68]

Diesen Vorteil bietet auch die Software von Telerik. Corporation [10] Bei der Ausführung des Testfalls hat die Software von Ranorex am schlechtesten abgeschnitten, in zwei von drei Durchläufen kam es zu fehlerhaftem Verhalten beim Scrollen der Scrollbars des CS-Clients und dadurch zum Abbruch des Programms. Auch die Software von HP bietet die Möglichkeit vom Import solcher Testdaten, jedoch ist die Einbindung als kompliziert zu bewerten, da diese teilweise implementiert werden müssen und nicht per GUI ausgewählt werden können.

Die nachfolgende Tabelle zeigt den direkten Vergleich der Tools in Bezug auf Testfall-Erstellung, Änderung und Ausführung.

Tabelle 2.12 Test der Software anhand des CS-Clients

Software	Testfall-Erstellung	Änderung von Testdaten	Ausführbarkeit	Summe
Ranorex	1	3	1	5
Telerik	3	3	3	9
SikuliX	2	2	2	6
HP UFT	3	2	3	8

Fazit zu diesem Abschnitt: Die Software von Telerik überzeugte in allen untersuchten Bereichen am meisten. Prinzipiell gelingt es mit jeder der untersuchten Applikationen mit etwas Aufwand zum gewünschten Ergebnis und damit zu einer möglichen Automatisierung des CS-Clients zu kommen.

Wechsel zwischen den Applikationen und fehlerfreier Durchlauf des gesamten Testfalls (drei Durchgänge)

Für den Test des AD/CS-Verbundes wird wieder von jeweils maximierten Fenstern (Internet Explorer und AD-Client) bei einer Bildschirmauflösung von 1600 x 900 Pixeln ausgegangen. Ansonsten gilt hier, dass in HP UFT das Silverlight/WPF Add-In aktiviert wurde.

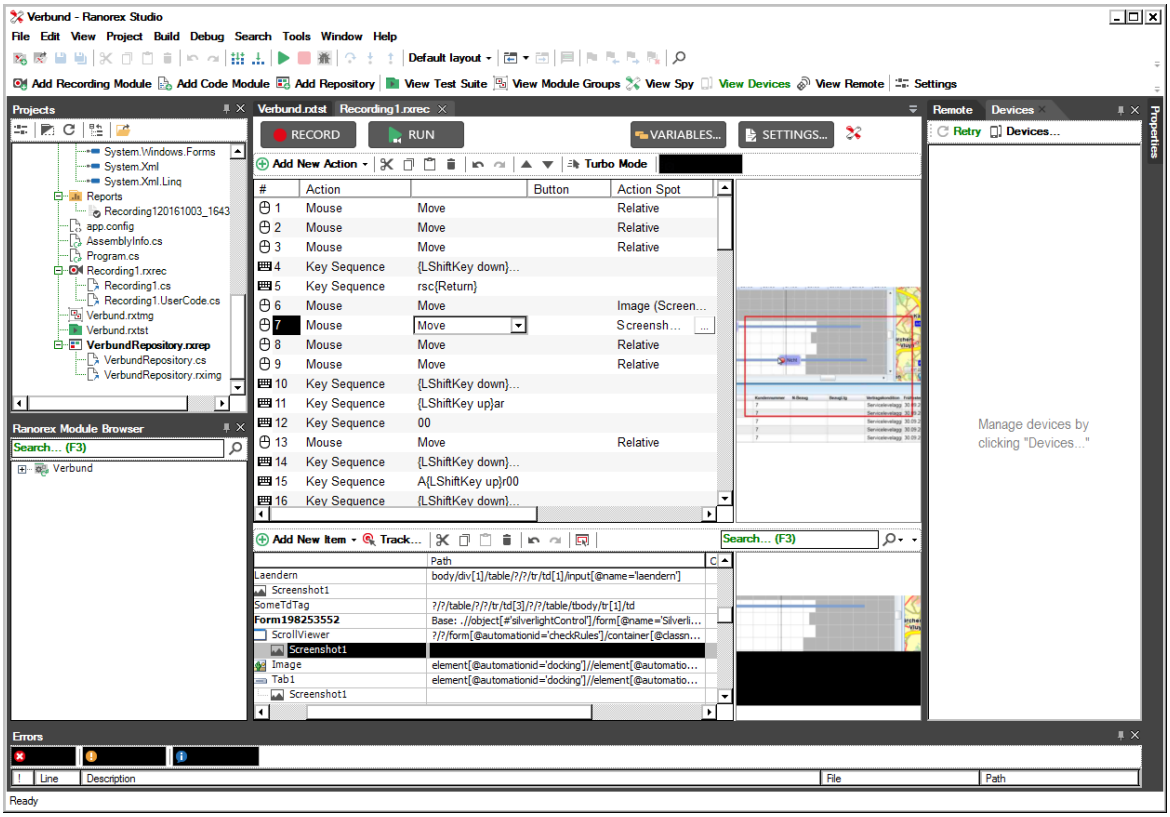

Abbildung 2.11 Test des AD/CS-Verbundes im Ranorex Studio

Die Abbildung zeigt den Verbund-Testfall als einzelne Testschritte durch den Ranorex Recorder aufgezeichnet. Die Software von Ranorex und SikuliX erfüllen in etwa beide gleich gut die automatische Testausführung dieses Testfalls.

Die Software von Telerik scheidet aufgrund der Inkompatibilität zum AD-Client aus.

Das Ranorex Studio und das HP UFT gewinnen an dieser Stelle durch die Unterstützung von Data Driven Testing, sodass sich eine mögliche Änderung von Testdaten etwas komfortabler einbinden lässt. Bei der Erstellung des Testfalls hat die Applikation von SikuliX weniger Probleme mit Softwareabstürzen als das Ranorex Studio. Die Erstellung in SikuliX benötigt etwa zehn Minuten mehr Aufwand als durch den Ranorex Recorder. Da dieser Recorder nicht alle Elemente direkt erkannte und Aktionen zum Teil von Hand nachbearbeitet

werden mussten, lassen sich die beiden Applikationen als gleichwertig bewerten, was die Nutzerfreundlichkeit bei der Erstellung dieses Testfalls betrifft.

Während der Testausführung kam es bei beiden Programmen in einem der drei Testdurchläufe zu einem Abbruch. SikuliX hatte einmal einen Fehler bei der Erkennung einer Schaltfläche und hat stattdessen eine ähnliche andere Schaltfläche angeklickt. Ranorex hingegen hatte ein Problem beim Drag & Drop von der Auftragsliste auf das Gantt und ist an dieser Stelle abgebrochen. Beide Applikationen zeigen bei einem Abbruch jeweils den Testschritt an, zu dessen Zeitpunkt das Programm abgebrochen hatte. SikuliX hat bei diesen Durchgängen Fehlverhalten nicht erkannt und „wahllos“ weiterhin Maus- und Tastaturgesten ausgeführt.

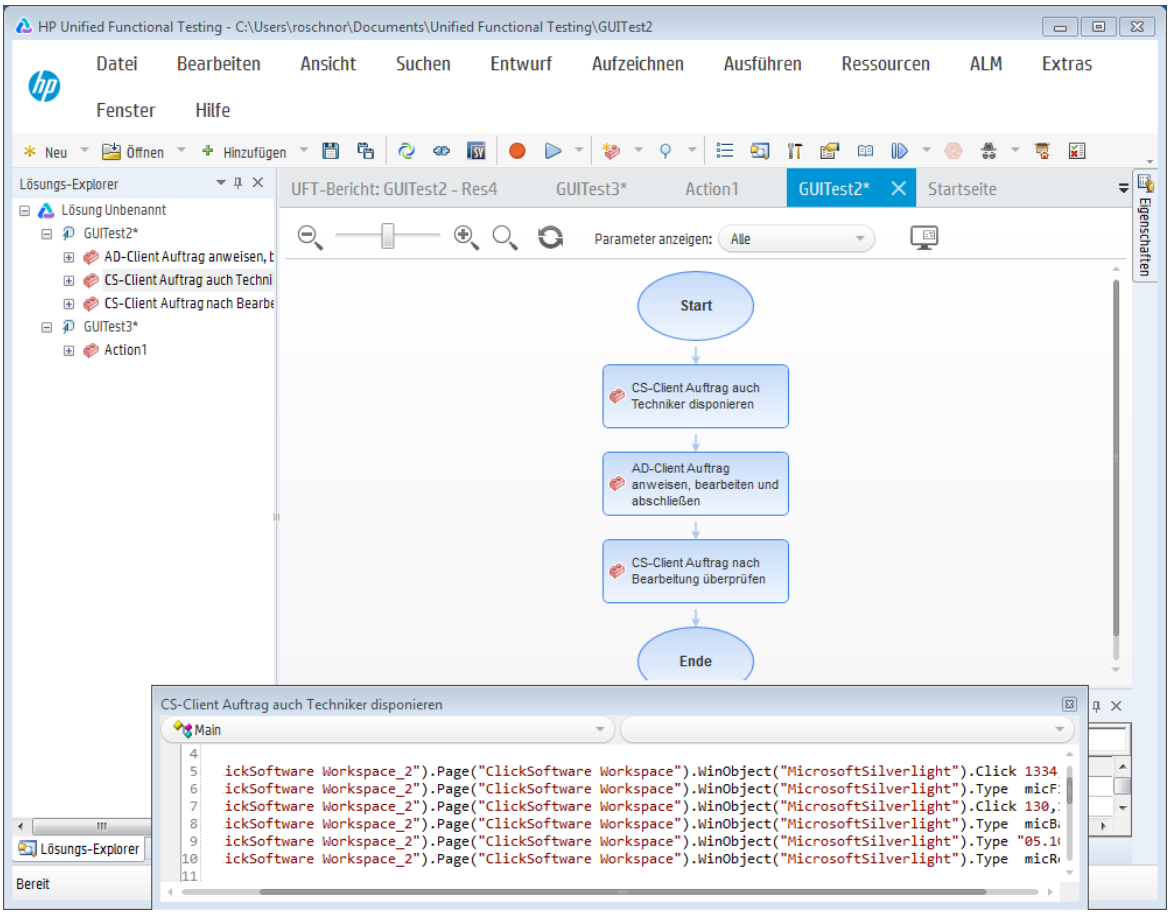

Abbildung 2.12 Test des AD/CS-Verbundes im HP UFT

Das HP UFT schneidet an dieser Stelle etwas besser ab, da die Testausführung komplett fehlerfrei und am flüssigsten abgelaufen ist. Dafür war die Erstellung des Testfalls am kompliziertesten.

Es mussten Programmaufrufe, Wartezeiten etc. manuell in den Testfall implementiert werden. Der innerhalb von HP UFT vorhandenen Debugger ermöglichte dabei eine benutzerfreundliche Überprüfung der einzelnen Testschritte.

Die nachfolgende Tabelle zeigt den direkten Vergleich der Tools in Bezug auf Testfall-Erstellung, Änderung und Ausführung.

Tabelle 2.13 Test des Software-Verbundes AD/CS

Software	Testfall-Erstellung	Änderung von Testdaten	Ausführbarkeit	Summe
Ranorex	2	2	2	6
Telerik	0	0	0	0
SikuliX	2	1	2	5
HP UFT	1	2	3	6

Fazit zu diesem Abschnitt: Die Software von Telerik kann für diesen Test nicht verwendet werden. Die Software von Ranorex und SikuliX hingegen erfüllen in etwa beide gleich gut die automatische Testausführung dieses Testfalls. Insgesamt erfolgte unter der Software HP UFT die beste (fehlerfreie) automatische Testausführung eines Verbundtestfalls.

2.6.2 Zeitdauer im Vergleich zum manuellen Testen

Auf Grund der Inkompatibilität vom Telerik Test Studio wird dieses für die weiteren Untersuchungen nicht in Betracht gezogen.

An dieser Stelle erfolgt ein Vergleich des manuellen Testens mit den Zeiten für Programmstart, Testfall-Erstellung und Testfall-Ausführung durch die verschiedenen Tools.

Tabelle 2.14 Zeitdauer im Vergleich zum manuellen Testen

Software	Programmstart	Testfall-Erstellung	Testfall-Ausführung	Summe
manuell	0	0	12	12
Ranorex	2	108	12	122
SikuliX	1	119	10	130
HP UFT	6	140	10	156

Angaben jeweils in Minuten

Wie in der Tabelle erkennbar, wird für den Verbund-Testfall aus Sektion 2.6 bei Verwendung von HP UFT am meisten Zeit benötigt. Die Software benötigt beim Programmstart mit sechs Minuten im Vergleich am längsten zu Ranorex mit zwei Minuten und SikuliX mit einer Minute. Gemessen wurde die Zeitdauer vom Start der Software bis zum Laden des gespeicherten Tests. Auch in Bezug auf die Erstellung des automatischen Tests benötigt die Software von HP mit 140 Minuten am meisten. Der Grund hierfür ist, dass viele Testschritte durch manuelle Funktionen ergänzt wurden, so dass es in diesem Fall in etwa 30 Minuten Mehrarbeit im Vergleich zu Ranorex Studio erforderlich war.

Wenn dagegen die eigentliche Ausführungszeit des Tests betrachtet wird, muss gesagt sein, dass der Ranorex Recorder automatisch die Wartezeiten des manuellen Vortests aufzeichnet und in den Testfall integriert, diese Wartezeiten in HP UFT oder SikuliX aber manuell gesetzt sein müssen. Sowohl Ranorex als auch SikuliX interpretieren die erstellten Wartezeiten in diesem Test je nach Einstellung als maximale Wartezeit, so dass zum Beispiel definiert werden kann, der Test soll fortgesetzt werden, sobald ein bestimmtes Element auftaucht oder verschwindet bzw. sich verändert. Ansonsten kennt SikuliX nur einen normalen Durchlauf und einen Durchlauf in Zeitlupe, weitere Geschwindigkeiten können nicht gesetzt werden. Innerhalb von HP UFT und Ranorex Studio lassen sich feste Wartezeiten zwischen jedem Schritt definieren, standardmäßig liegen diese bei 15 Sekunden. Ist ein Element innerhalb dieser Zeit nicht geladen, wird der Test mit Erkennung des Fehlers sofort beendet.

Die Untersuchung hat gezeigt, dass bei einem einfachen Testfall die Automatisierung das zehn-15-fache der manuellen Testfall-Ausführungszeit benötigt. Die Zeit für die Konzeption des Testfalls und Einarbeitung eines Mitarbeiters in diesen manuellen Testfall bzw. die

Automatisierungssoftware wird dabei vernachlässigt, weil ohne diese Vorbedingungen auch kein automatischer Test erfolgen könnte. Ein sehr wichtiger Aspekt ist die Dauer, bis ein Testfall automatisiert ist bzw. die Bedingungen, unter denen ein automatischer Test Vorteile mit sich bringt.

Dabei lässt sich sagen, dass die Applikationen von Ranorex und HP zwar kaum Zeit bei der Ausführung einsparen, aber die Vorteile der Programme liegen darin, dass diese anders als ein Mensch rund um die Uhr (auch nach definiertem Zeitplan) eingesetzt werden könnten. Außerdem bieten diese beiden Tools das automatische oder drag-&-drop-basierte Erstellen von neuen möglichen Testfällen, wodurch wiederum Zeit eingespart werden kann. SikuliX hingegen kann ein Script ausführen, welches jeweils von einem Menschen gestartet werden muss. Der Vorteil hier beschränkt sich darauf, dass die Ausführungszeit optimiert wird.

2.6.3 Fehleranfälligkeit der Software

In dieser Sektion erfolgt eine Bewertung aufgrund von Programmabstürzen und Fehlverhalten bei unerwarteten Werten (andere Positionen von Elementen, andere Werte von Textfeldern) bzw. Änderung der Bildschirmauflösung.

Dabei wird für jeden aufgetretenen Fehler ein Punkt vergeben. Insgesamt wird jeder Testlauf in Bezug auf das Fehlverhalten dreimal wiederholt, wobei Programmabstürze in Summe bei allen neun Durchgängen (dreimal Verschiebung von Elementen, dreimal Änderung der Bildschirmauflösung und dreimal Änderung von Textfeldern) betrachtet werden.

Als Programmabsturz zählt auch, wenn die Software nicht mit einer Fehlermeldung abbricht, wie in den Abbildungen zu sehen ist, sondern unbestimmt weiterhin irgendwelche Aktionen ausführt, die nicht geplant waren (Beispiel: das Anklicken nicht vorgegebener Schaltflächen).

Tabelle 2.15 Fehleranfälligkeit der Software

Software	Programm-absturz	andere Positionen von Elementen	andere Werte von Textfeldern	Änderung der Bildschirm-auflösung	Summe
Ranorex	2	1	1	1	5
SikuliX	3	3	1	2	9
HP UFT	1	1	0	1	3

Angaben jeweils für drei Durchläufe

Grundsätzlich zeigt sich, dass keine der getesteten Applikationen komplett fehlerfrei arbeitet, es lässt sich eine Häufung der Fehleranfälligkeit bei SikuliX erkennen.

Dabei wurden jeweils die folgenden Faktoren getestet:

- Änderung der Datumsvorauswahl im CS-Client
- Änderung der Zeitvorgabe im AD-Client
- Änderung der Suchfeldvorbelegung im CS-Client
- Änderung der Scrollbarposition im CS-Client
- Änderung der Auftragsreihenfolge innerhalb der Auftragsliste im CS-Client
- Änderung der Ausgangsgröße des Gantts im CS-Client
- Änderung der Bildschirmauflösung auf 1680 x 1050 Pixel

- Änderung der Bildschirmauflösung auf 1920 x 1200 Pixel
- Änderung der Bildschirmauflösung auf 800 x 600 Pixel

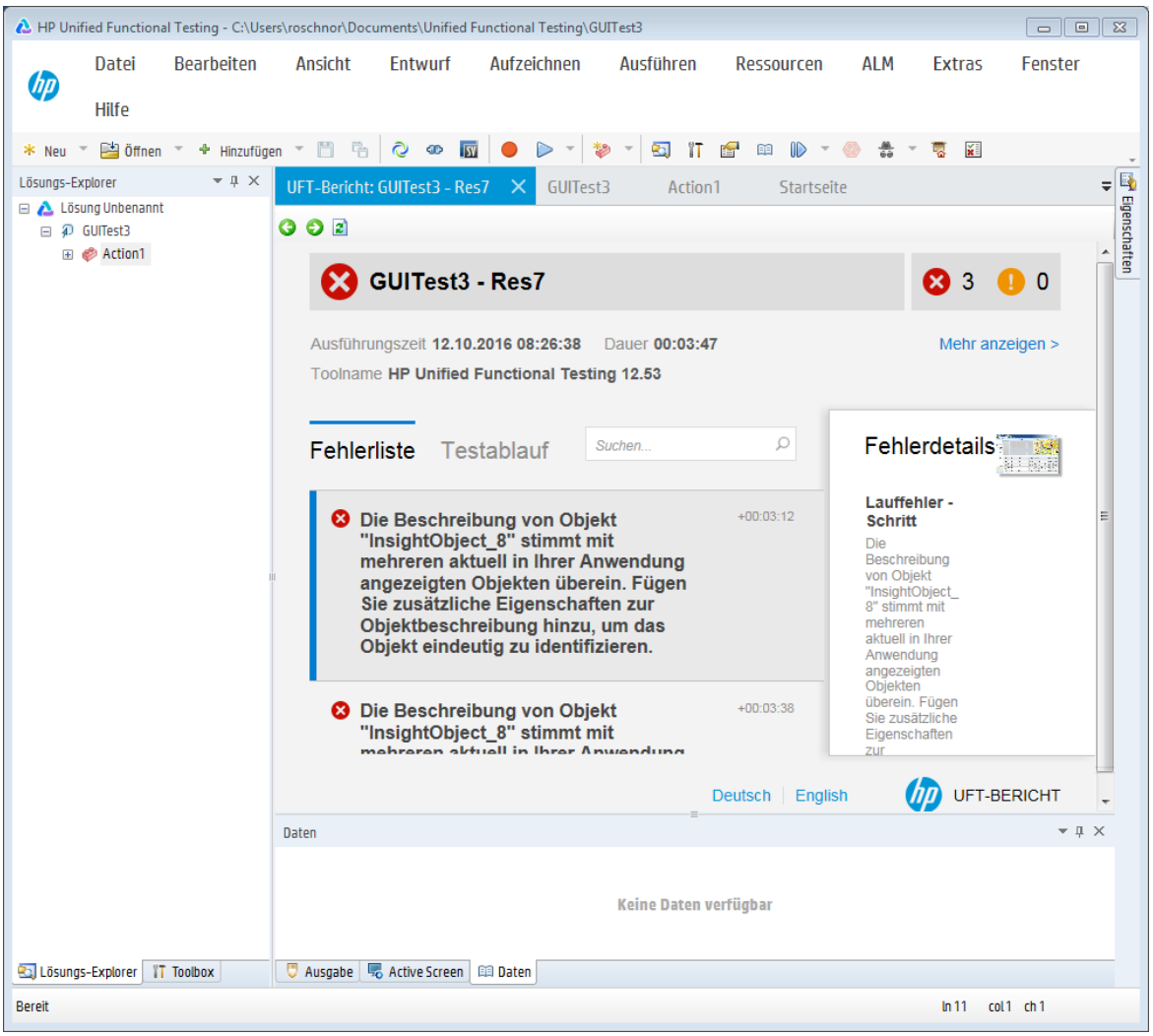

Abbildung 2.13 HP UFT stoppt aufgrund eines Fehlers

Am wenigsten Fehler traten bei der HP UFT auf. Textfelder wurden in allen untersuchten Fällen erkannt und bei Änderung der Bildschirmauflösung oder dem Verschieben von Elementen trat jeweils nur einmal ein Fehlverhalten auf.

Die Abbildung zeigt einen Fehlerbericht in HP UFT, nachdem die Software aufgrund eines Fehlers gestoppt war.

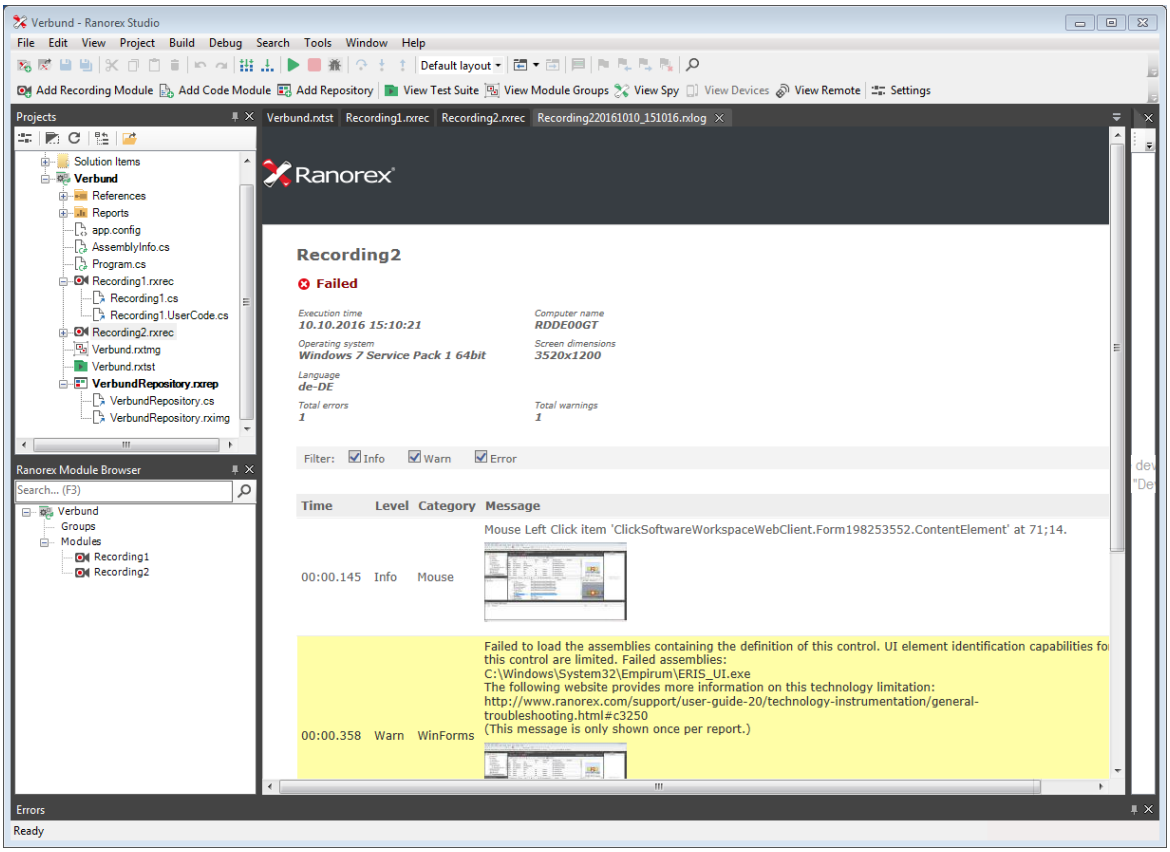

Abbildung 2.14 Ranorex Studio stoppt aufgrund eines Fehlers

Beim Ranorex Studio hingegen traten bei einigen Änderungen der Ausgangssituation direkt Fehlverhalten auf. Die Software erkannte bestimmte Elemente nicht. Dabei kam es nicht auf die Position der Elemente innerhalb der GUI an.

Die Abbildung zeigt einen Fehlerbericht im Ranorex Studio, nachdem die Software aufgrund eines Fehlers gestoppt war.

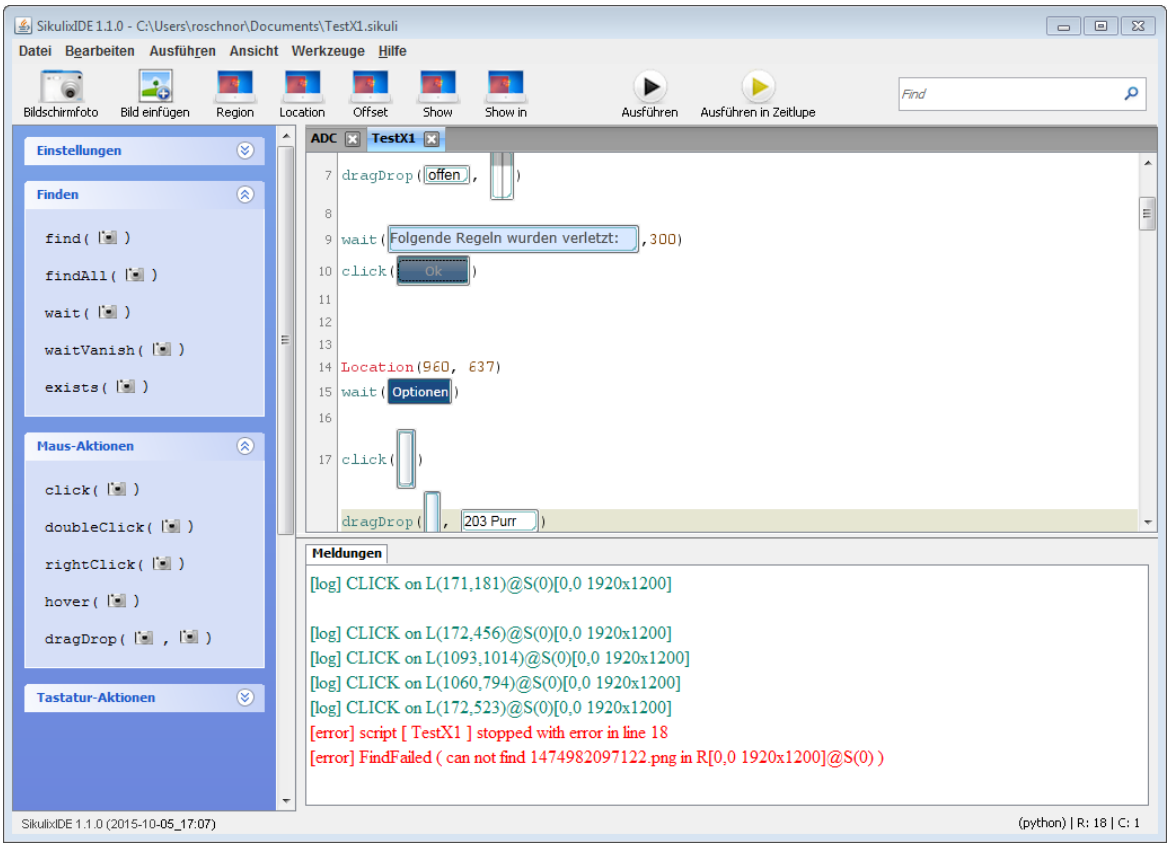

Abbildung 2.15 SikuliX stoppt aufgrund eines Fehlers

SikuliX erzeugte bei diesem Test die meisten Fehler. Kleine Änderungen an der Position von Elementen oder an der Bildschirmauflösung reichten aus, damit SikuliX fehlerhaft agierte. Insgesamt kam es bei allen drei Applikationen zu Programmabstürzen bzw. wurden innerhalb von SikuliX Fehler nicht erkannt und mit falschen Elementen weitergearbeitet.

Die Abbildung zeigt einen Fehlerbericht in SikuliX, nachdem die Software aufgrund eines Fehlers gestoppt war.

An dieser Stelle ist anzumerken, dass die hier getesteten Fehlerquellen nicht allgemein gültig sind. Je nachdem, wie viel zusätzlicher Programmieraufwand bei der Erkennung von Fehlern betrieben wird, desto wahrscheinlicher ist es, dass bestimmte Fehler nicht auftreten bzw. das Programm mit ihnen umzugehen weiß. Sowohl das Ranorex Studio als auch HP UFT lassen zum Beispiel die Möglichkeit des „Continue on Fail“ zu, also die Fehlerbehandlung bei definierten Testschritten. Tutorialspoint [102] Ranorex [69]

2.6.4 Import-/Exportmöglichkeiten der Software

In Anbetracht dessen, dass durch eine Zunahme der Integrationsmöglichkeiten von Software, besonders mit Blick auf das Management eines Softwareprojektes, die Qualität und Performance der eingesetzten IT-Systeme gesteigert werden kann (IT&PRODUCTION [42]), wird im Folgenden untersucht, welche Optionen zum Datenaustausch durch die jeweiligen Applikationen bereitgestellt werden.

Innerhalb der Untersuchung wird nicht darauf eingegangen, wie und mit welchem Aufwand die dafür benötigten Daten von ihrer Art her für den Prozess aufbereitet werden mussten, z. B. durch einen Wrapper, sondern es wird jeweils ein Punkt für die generelle Erfüllung von direkten Möglichkeiten für den Import bzw. Export von Testfällen, Objekten, Fehlerberichten oder der Möglichkeit einer Integration von Application-Management-Software vergeben.

Als „Objekte“ wird in diesem Fall die Menge aller zu testenden Elemente, also Schaltflächen, Textfelder etc. bezeichnet. Dabei sollte die Objektstruktur z. B. als DOM-Pfad so gespeichert sein, dass Elemente innerhalb von Testfällen wiedererkannt werden können, damit die Software in der Lage ist, aus einem Testfall automatisch den Testlauf zu erstellen.

Tabelle 2.16 Import-/Exportmöglichkeiten der Software

Software	Testfall-import	Testfall-export	ALM-Integration	Testbericht-export	Objekt-import	Objekt-export	Summe
Ranorex	1	1	0	1	1	1	5
SikuliX	0.5	0.5	0	0	0	0	1
HP UFT	1	1	1	1	1	1	6

Das kostenlose Automatisierungstool SikuliX erfüllt dieses Kriterium nahezu gar nicht; es gibt lediglich eine Möglichkeit, die in SikuliX erstellten Scripte als Archiv abzuspeichern bzw. zu laden. Eine Option zur Integration mit anderer Software ist zum aktuellen Zeitpunkt nicht bekannt, aber es ist jedem Entwickler möglich, eine eigene Schnittstelle für SikuliX zu entwickeln.

Die besten Integrationsmöglichkeiten in dieser Untersuchung bietet die Applikation HP UFT. Wie der Tabelle 2.16 entnommen werden kann, besitzt die Software werksseitig Import- bzw. Exportoptionen für Testfälle (Testdaten), Berichte und Test-Objektstrukturen, sowie eine Anbindung zu ALM-Applikationen. Durch definierte Schnittstellen können Entwickler eigene Erweiterungen zur Integration von Software entwickeln.

Auch die Software von Ranorex bietet werksseitig Import- bzw. Exportoptionen. Es ist keine direkte Anbindung zu ALM-Applikationen möglich. Das Ranorex Studio besitzt ähnlich zum HP UFT definierte Schnittstellen, mit deren Hilfe ein Entwickler eigene Erweiterungen mit der Software verknüpfen kann. Auf diesem Wege ist es möglich Drittanbieter-Software zu integrieren.

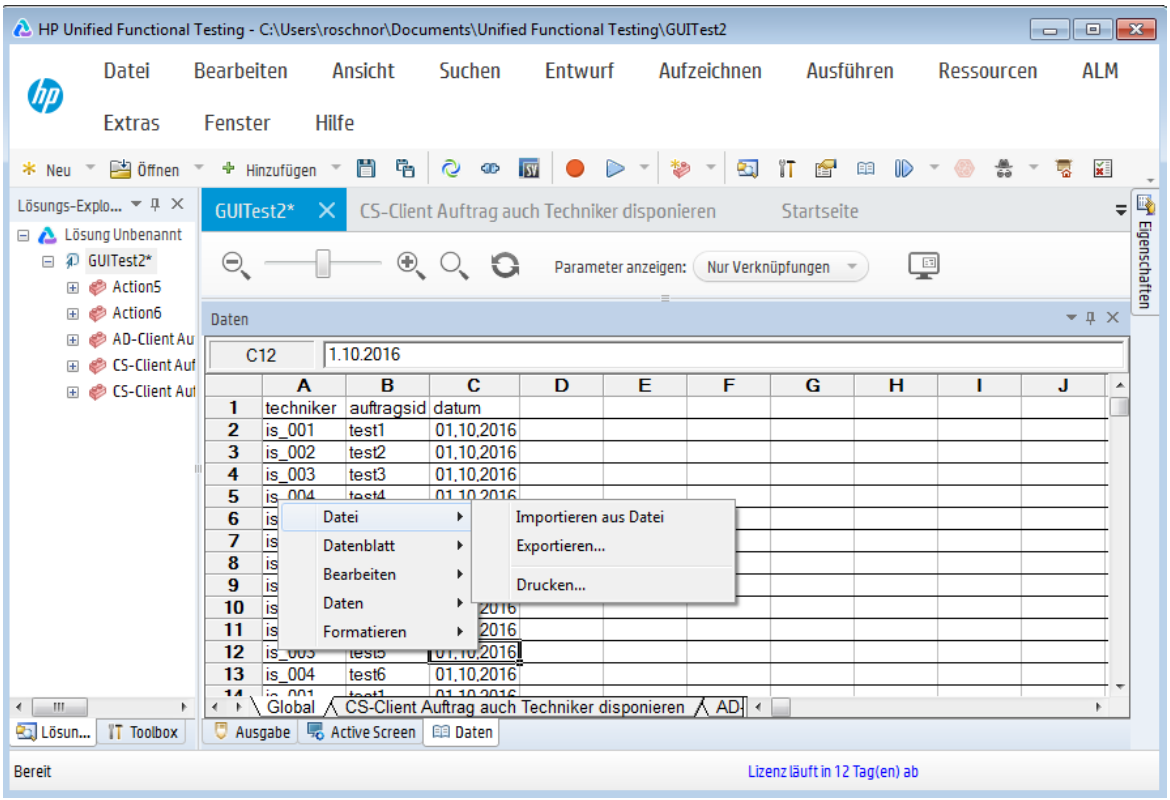

Abbildung 2.16 Import: Data Driven Testing im HP UFT

Die beiden kostenpflichtigen Automatisierungswerkzeuge von HP und Ranorex verfügen nicht nur über die reine Möglichkeit des Testfall-Imports, sondern Testfälle können auch auf Basis eines Imports von Keywords (Keyword Driven) bzw. Datasets (Data Driven) generiert werden (Abbildung 2.16).

An dieser Stelle wird nochmal auf das Kapitel 2.6 verwiesen.

2.7 Verwendbarkeit für Tester ohne Programmierkenntnisse

Grundsätzlich können alle vorgestellten Applikationen von Testern ohne Programmierkenntnisse verwendet werden. Es gibt besonders bei der Erstellung neuer Testfälle einige Situationen, in denen Programmierkenntnisse von Vorteil sind. So zum Beispiel kann der in HP UFT eingebaute Recorder zwar einen manuellen Testlauf aufzeichnen, jedoch funktioniert dies nur bei einfachen Testfällen wie dem Aufruf einer Suchanfrage bei Google wirklich fehlerfrei. In der Mehrzahl ist es erforderlich, dass diese aufgezeichneten Testschritte nachbearbeitet werden müssen. Diese Nachbearbeitung erfolgt innerhalb von HP UFT rein scriptbasiert auf Basis von VB.Net. Ein Vorteil ist an dieser Stelle für Programmierer, jedoch nicht für Tester, dass diese Testschritte über eine Schnittstelle programmiert werden könnten.

Tabelle 2.17 Verwendbarkeit ohne Programmierkenntnisse

	Erstellung	Ausführung	Neu-Kombination	Summe
Ranorex	3	3	2	8
SikuliX	2	2	1	5
HP UFT	1	2	3	6

Die Tabelle zeigt eine Bewertung zwischen null (nicht erfüllt) bis drei Punkten (optimal erfüllt) für Erstellung, Ausführung und Änderung von Testfällen innerhalb der Applikationen.

Das Ranorex Studio hat einen genaueren Recorder als HP UFT integriert, der innerhalb eines manuellen Testschrittes die notwendige Wartezeit von sich aus erkennt. Außerdem bietet die Software eine eigene GUI für die benutzerfreundliche Bearbeitung von diversen Testschritten. So lassen sich hier Wartezeiten, genauere Elementeigenschaften und Fehlerumgehungsmöglichkeiten editieren. Genau wie die Software von HP bietet Ranorex einen Objektinspektor, mit dessen Hilfe Elemente leichter identifiziert werden können. Die Auswahl erfolgt durch das Anklicken des Elementes und einer Auswahlmaske.

Bei SikuliX gibt es keinen Recorder und es ist nur eine scriptbasierte Oberfläche vorhanden. Aber dadurch, dass es „nur“ 14 verschiedene Funktionen gibt und ansonsten die Funktionalität stark eingeschränkt ist, ist diese Software sehr schnell zu verstehen. Es gibt zwei Geschwindigkeitsmodi innerhalb von SikuliX, so dass die Ausführung nicht optimal anpassbar ist. Was die Möglichkeit der Neukombination von Testfällen betrifft, können Teile eines SikuliX-Scriptes innerhalb von SikuliX kopiert und durch Einfügen neu zusammen gestellt werden. Es ist nicht möglich, auf eine benutzerfreundliche Art neue Testfälle zusammenzustellen.

An dieser Stelle punkten die Applikationen von Ranorex und HP, welche jeweils Testschritte in unterschiedlichen Ressourcen verpacken können, die sich wahlweise per Drag & Drop oder ähnlichen Funktionen neu zusammenstellen lassen.

Innerhalb von HP UFT, wie in der Abbildung 2.17 zu sehen, erfolgt die Gliederung dieser Ressourcen in Solution, Test und Action.

Innerhalb von Ranorex Studio heißen diese Solution, Testcase und Modul.

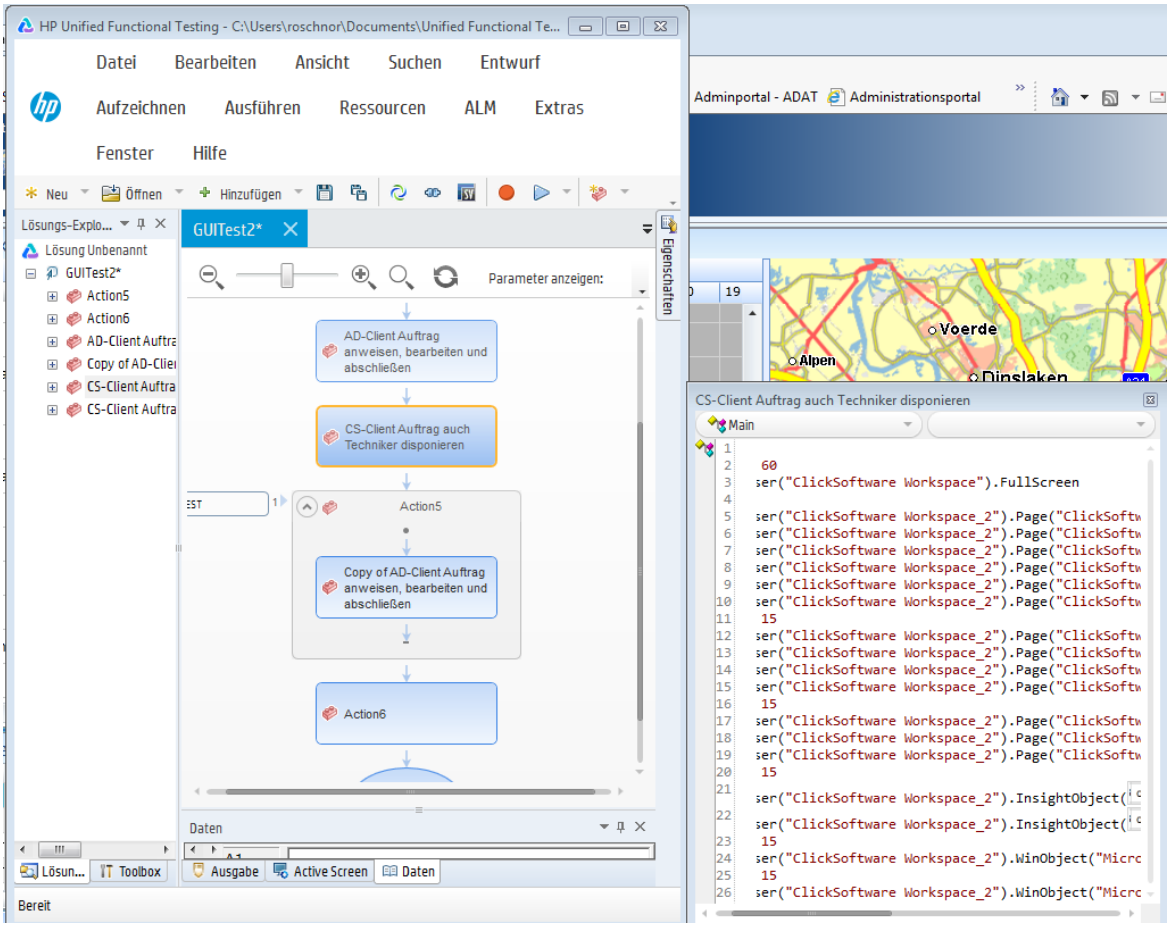

Abbildung 2.17 HP UFT Test-Flow

Abschließend lässt sich feststellen, dass die Erstellung komplett neuer Testfälle für einen Tester innerhalb des Ranorex Studios aufgrund des Recorders einfacher zu handhaben ist. Das HP UFT ist benutzerfreundlicher gehalten in der Erstellung neuer Testfälle aus bereits existierenden Komponenten.

Kapitel 3

Schlussteil

3.1 Zusammenfassung der Untersuchungsergebnisse

Im Zuge der Untersuchung lassen sich die Ergebnisse aus den einzelnen Tests in Summe wie folgt zusammenfassen. Dabei wird für den jeweiligen Gewinner eines Kriteriums jeweils drei Punkte, für den Zweitplatzierten jeweils zwei Punkte und für den Drittplatzierten jeweils ein Punkt vergeben.

Die nachfolgenden Kriterien dienten dieser Untersuchung als Bewertungsfaktoren. An dieser Stelle ist zu erwähnen, dass die Auswahl der zu untersuchenden Software bereits durch Abschnitt 2.4 erläutert wurde.

3.1.1 Untersuchte Kriterien nach Vorauswahl der Software

- Installationsaufwand (Kapitel 2.5.1)
- Test der Software anhand des AD-Clients (Kapitel 2.6.1)
- Test der Software anhand des CS-Clients (Kapitel 2.6.1)
- Test des Software-Verbundes AD/CS (Kapitel 2.6.1)
- Zeitdauer im Vergleich zum manuellen Testen (Kapitel 2.6.2)
- Fehleranfälligkeit der Software (Kapitel 2.6.3)
- Import-/Exportmöglichkeiten der Software (Kapitel 2.6.4)
- Verwendbarkeit für Tester ohne Programmierkenntnisse (Kapitel 2.7)

Tabelle 3.1 Gesamt-Punkteverteilung

Kriterium	Software		
	Ranorex	SikuliX	HP UFT
Installationsaufwand	3	2	2
Test der Software anhand des AD-Clients	3	2	3
Test der Software anhand des CS-Clients	1	2	3
Test des Software-Verbundes AD/CS	3	2	3
Zeitdauer im Vergleich zum manuellen Testen	3	2	1
Fehleranfälligkeit der Software	2	1	3
Import-/Exportmöglichkeiten der Software	2	0	3
Verwendbarkeit für Tester ohne Programmierkenntnisse	3	1	1
Summe	20	12	19

Wie der Tabelle entnommen werden kann, liegen die beiden kostenpflichtigen Applikationen von HP (19 Punkte) und Ranorex (20 Punkte) in etwa gleich auf.

Die kostenlose Software SikuliX liegt, unter Betrachtung der möglichen Punkte von acht bis 24, mit 12 Punkten im unteren Mittelfeld.

Es zeigt sich, dass keine der Applikationen die Maximalpunktzahl erreicht, vielmehr überzeugen die einzelnen Automatisierungsprogramme in jeweils unterschiedlichen Kriterien.

So hatte die Software von Ranorex besonders Probleme im Umgang mit dem CS-Client, konnte dafür die anderen Kriterien gut bis sehr gut erfüllen.

Die Software von HP ist als komplexer zu betrachten, was sich darin bemerkbar macht, dass es für einen Tester ohne Programmierkenntnisse schwierig ist, Testfälle zu erstellen, und vom zeitlichen Aufwand her die anderen Applikationen mehr überzeugen.

Wenn der Aufwand außer Acht gelassen wird und sich auf die Resultate der eigentlichen Testdurchführung gestützt wird, gewinnt an dieser Stelle die Software von HP. Das HP UFT war von der Fehleranfälligkeit, der Integrationsmöglichkeiten und im Besonderen von der Genauigkeit bei der Ausführung von automatischen Testläufen am besten.

Unter Betrachtung der Erstellung und automatischen Ausführung eines Tests, lässt sich die Software SikuliX als ausreichend einstufen, da diese trotz einiger Abzüge im Bedienungskomfort eine fast fehlerfreie, wenn nur einfache Automatisierung ermöglicht.

3.2 Empfehlungen an die Zielgruppe

Entsprechend der in Abschnitt 1.4 definierten Zielgruppe lässt sich eine allgemeine Aussage finden und zusätzlich lässt sich diese für spezielle Erwartungen an solche Software ergänzen.

Es gibt keine Software zur Automatisierung von GUI-basierten Verbundtests, welche alle Erwartungen unterschiedlicher Gruppen (Tester, Testentwickler, Testmanagement, Geschäftsführung) vollständig erfüllt.

Vielmehr gibt es für unterschiedliche Erwartungen eine jeweils andere passende Software.

Dabei ist davon auszugehen, dass ein Tester keine Programmierkenntnisse hat, ein Testentwickler allerdings sehr wohl, das Testmanagement sowohl eine optimale Ausführung als auch Auswertung von Testdurchläufen verlangt und die Geschäftsführung möglichst wenig Zeit und Geld investieren möchte.

Im Folgenden werden unterschiedliche Empfehlungen ausgesprochen.

3.2.1 Allgemein

Unter Berücksichtigung aller Faktoren lässt sich sagen, dass sich die Software HP UFT am besten eignet für eine Automatisierung von Softwaretests. Wie der Tabelle 3.2 entnommen werden kann, überzeugt die Software von HP sowohl durch eine geringe Fehleranfälligkeit bei der Ausführung von Testfällen als auch durch eine Vielzahl an Möglichkeiten zur Integration anderer Tools bzw. dem Import/Export von Testfällen.

Neben dem Import von Testfällen bietet die Software dem Benutzer eine Programmierschnittstelle, mit welcher eine Art eigener API zur Erstellung neuer Testfälle implementiert werden kann. Für einen Tester ohne Programmierkenntnisse ist diese Software schwierig zu bedienen. Einige Aspekte erfordern den Einsatz einer Programmierung.

Dank der Möglichkeit, GUI-basiert aus atomaren Testschritten neue Testfälle via Drag & Drop zu erstellen, kann die Software an dieser Stelle von Testern ohne Programmierkenntnisse verwendet werden. Zudem steht einem solchen Tester ein Record-Tool (Recorder) zur Verfügung, mit dessen Hilfe neue Testschritte anhand eines manuellen Durchlaufs aufgezeichnet werden können.

Insgesamt arbeitet dieser Recorder mit verschiedenen Modi, so dass technologieunabhängig bzw. durch Erweiterungen auf bestimmte Technologien angepasst gearbeitet werden kann. Diese Erweiterungen müssen zum Teil kostenpflichtig erworben werden, wodurch allerdings eine Verbesserung der Erkennung von Elementen wie z.B. Schaltflächen ermöglicht wird. Ansonsten arbeitet die Software HP UFT anhand von Mauspositionen oder auf Basis einer bildbasierten Objekterkennung. Zu den so erstellten Testschritten müssen innerhalb von HP zusätzliche Funktionen, z.B. Wartezeiten, Bedingungen etc., programmiert werden.

Neben dem Recorder kann die Software auf Basis von bereits existenten Testfällen und Integration einer Datenquelle, z.B. Excel, neue Testfälle mit variablen Daten erstellen. Diese Methodik wird als Data Driven Testing bezeichnet. Die folgende Tabelle zeigt Vor- und Nachteile von HP UFT.

Tabelle 3.2 Pro & Kontra für HP UFT

Pro	Kontra
viele Erweiterungen verfügbar	Erweiterungen müssen gekauft werden
mehrere (virtuelle) Instanzen	Schulungen nur gegen Gebühr
Remote-Ausführung	nicht optimal für reine Tester
technologieunabhängig (auch Silverlight)	hoher Anschaffungspreis
div. Import-/Export-Möglichkeiten	abhängig von Microsoft Windows
Schnittstellen (auch Steuerung)	Programmierung erforderlich
Drag & Drop	Support nur kostenpflichtig über den Hersteller
GUI-basierte Neuerstellung von Testfällen	
geringe Fehleranfälligkeit	
Record-Tool	
Data Driven Testing	
werksseitige Integration div. Tools	

Unabhängig von der Integration solcher Datenquellen bietet die Applikation werksseitig sowohl Schnittstellen zu ALM- als auch zu Continuous Integration Tools. Die Software ist damit gut geeignet zur Verwendung innerhalb eines Softwareprojektes, da aus Projektmanagementsicht alle Erwartungen erfüllt werden. Berichte werden ausführlich generiert und erkannte Mängel innerhalb der zu testenden Software können direkt einen Nachtest auslösen. Es bedarf dafür der Verwendung von gewissen Tools bzw. der eigenständigen Programmierung einer solchen Schnittstelle.

Ein anderer Aspekt von HP UFT ist die Verwendbarkeit als virtuelle Instanz. Unter Umständen ist hier eine zusätzliche Lizenz für das Programm erforderlich. Virtuelle Instanzen können unter HP UFT Remote oder auf einem lokalen Rechner ausgeführt werden.

Das Negative an dieser Software sind sowohl die hohen Anschaffungskosten pro Benutzer als auch, dass es Schulungen und Support nur kostenpflichtig über den Hersteller gibt.

3.2.2 Ohne Programmierkenntnisse

Für Benutzer ohne Programmierkenntnisse empfiehlt sich die Verwendung des Ranorex Studios. Die Software erfordert keine Programmierkenntnisse und bietet nahezu denselben Funktionsumfang wie HP UFT, jedoch mit der Einschränkung, dass weniger Erweiterungen zur Verfügung stehen. Außerdem erwies sich das Ranorex Studio als fehleranfällig bei der Ausführung von Testfällen. Die folgende Tabelle zeigt Vor- und Nachteile des Ranorex Studios.

Tabelle 3.3 Pro & Kontra für Ranorex Studio

Pro	Kontra
optimal für Tester	wenige Erweiterungen verfügbar
keine Programmierkenntnisse notwendig	fehleranfällig bei der Ausführung
mehrere (virtuelle) Instanzen	Schulungen nur gegen Gebühr
Remote-Ausführung	hoher Anschaffungspreis
technologieunabhängig (auch Silverlight)	abhängig von Microsoft Windows
div. Import-/Export-Möglichkeiten	Support nur kostenpflichtig über den Hersteller
Schnittstellen (auch Steuerung)	
Drag & Drop	
Record-Tool	
Data Driven Testing	
GUI-basierte Neuerstellung von Testfällen	

Was die Erstellung neuer Testfälle betrifft, kann die Software von Ranorex mit HP gut mithalten. Es gibt diverse Import- bzw. Export-Möglichkeiten im Besonderen bei Verwendung des sogenannten Data Driven Testings.

Genau wie bei HP verfügt das Ranorex Studio über einen integrierten Recorder zur Aufzeichnung manueller Testschritte. Über eine Schnittstelle oder GUI-basiert via Drag & Drop können neue Testfälle erstellt werden.

Dadurch, dass Ranorex Studio rein bildbasiert arbeiten kann, arbeitet diese Software auf Wunsch technologieunabhängig. Die Software erfordert ein Microsoft Windows-Betriebssystem. Insgesamt ist es dem Benutzer möglich, virtuelle Instanzen der Software lokal oder Remote auszuführen. Unter Umständen ist dafür eine zusätzliche Lizenz vonnöten.

Wie bei HP UFT bietet Ranorex einen Support bzw. Schulungen nur kostenpflichtig an. Außerdem entstehen für die Software hohe Anschaffungskosten.

Das einzige, was an Ranorex Studio im Vergleich zu HP UFT als besser zu bewerten ist, ist die Verwendbarkeit für Tester ohne Programmierkenntnisse.

Bei HP UFT sind Programmierkenntnisse erforderlich, so dass die Verwendung dieser Software Unternehmen mit Testentwicklern zu empfehlen ist.

3.2.3 Mit Programmierkenntnissen

Bei Benutzern mit Programmiererfahrung empfiehlt sich, wie bereits zuvor allgemein betrachtet, HP UFT. Für einfachere Projekte kann wahlweise das Ranorex Studio Verwendung finden. Es ist an dieser Stelle eine Frage der eigenen Interessen (Aufwand/Kosten/Nutzen), so dass hier die persönliche Entscheidung dem Benutzer überlassen wird.

3.2.4 Webbasierte GUIs

Obwohl es für den untersuchten Verbundtest aufgrund von Inkompatibilität ausscheidet, empfiehlt sich das Telerik Test Studio für rein webbasierte Tests von z. B. Webportalen innerhalb eines Browsers. Die folgende Tabelle zeigt Vor- und Nachteile des Telerik Test Studios.

Tabelle 3.4 Pro & Kontra für Telerik Test Studio

Pro	Kontra
optimal für reine Tester	wenige Erweiterungen verfügbar
mehrere (virtuelle) Instanzen	Schulungen nur gegen Gebühr
Remote-Ausführung	hoher Anschaffungspreis
div. Import-/Export-Möglichkeiten	abhängig von Microsoft Windows
Schnittstellen (auch Steuerung)	unterstützt nur Web-Technologien
Drag & Drop	
Record-Tool	
Data Driven Testing	
GUI-basierte Neuerstellung von Testfällen	
Support auch durch Community	
geringe Fehleranfälligkeit	

Die Software überzeugte bei Silverlight-basierten Webseiten mit einer geringen Fehleranfälligkeit bei der Testausführung. Außerdem zeichnete der eingebaute Recorder zur Aufzeichnung manueller Testschritte als einziger die jeweiligen Wartezeiten zwischen den Schritten mit auf.

Ähnlich zum Ranorex Studio und HP UFT unterstützt die Software verschiedene Möglichkeiten zur Erstellung von Testfällen, die da wären: schnittstellenbasiert, auf Basis einer Datenquelle (Data Driven Testing), GUI-basiert oder durch weitere Importmöglichkeiten.

Das Telerik Test Studio verfügt im Gegensatz zu anderen Applikationen über wenige Erweiterungen. Im Kern konzentriert sich die Software auf browserbasierte Tests, wobei diese als virtuelle Instanz oder Remote ausgeführt werden können. Die Software ist abhängig von einem Microsoft Windows-Betriebssystem.

Erwähnenswert ist an dieser Stelle die Community, so dass ein Benutzer nicht auf Support durch den Hersteller angewiesen ist. Schulungen und ausführlicher Support müssen beim Hersteller eingekauft werden. Der Benutzer muss einen hohen Anschaffungspreis im Vergleich zum Nutzen der Software tragen.

3.2.5 Geringes Budget

Wer nur wenige, sich nicht ändernde Testschritte automatisieren möchte oder ein sehr geringes Budget zur Verfügung hat, dem lässt sich die kostenlose Software SikuliX empfehlen. Die folgende Tabelle zeigt Vor- und Nachteile von SikuliX.

Tabelle 3.5 Pro & Kontra für SikuliX

Pro	Kontra
unabhängig vom Betriebssystem (erfordert Java) technologieunabhängig (auch Silverlight) kostenlos	keine Erweiterungen verfügbar keine Schulungen Support nur durch Community keine Integration kein Import/Export fehleranfällig bei der Ausführung

Wie der Tabelle entnommen werden kann, arbeitet die Software komplett auf Basis von Java, so dass die Ausführung an kein spezielles Betriebssystem gebunden ist. Mehrere virtuelle Instanzen oder eine Remote-Ausführung sind ausgeschlossen.

Es gibt für die Software bisher keine Erweiterungen oder Möglichkeiten zur Integration mit anderen Tools. Einem findigen Programmierer sei es vorbehalten, selbständig die Open Source Software unter Einhaltung der Lizenzbedingungen für seine Zwecke weiterzuentwickeln.

SikuliX arbeitet scriptbasiert auf Basis einer bildbasierten Objekterkennung. Der Benutzer muss zur Verwendung der Software einen aktiven Monitor angeschlossen haben, in welchem die Aktionen (Testschritte) als Bildschirmausschnitt aufgezeichnet bzw. später ausgeführt werden. Die Software zeichnet immer wieder den Bildschirm auf und vergleicht die definierten Ausschnitte. Anhand von Koordinaten wird der Mauszeiger bewegt und ein Klicken simuliert.

Insgesamt ist diese Art der Erkennung sehr fehleranfällig. Sind auf dem Bildschirm zwei oder mehrere ähnliche Elemente vorhanden, kann es zu Verwechselungen und damit zu Fehlern bei der Ausführung kommen. Es kommt dazu, dass ein Fehler dem Programm gar nicht auffällt und es fehlerhaft weiterhin Aktionen ausführt.

Was die Möglichkeiten von Schulungen oder Support betrifft, verfügt die Software lediglich über eine Community. Umfangreiche Schulungsmaßnahmen können nicht erworben werden.

3.3 Fazit und Ausblick

Unter Berücksichtigung der These „Es gibt keine geeignete Software zur Testautomatisierung, welche alle Technologien und Anforderungen eines Verbundtestes im vollen Umfang erfüllen könnte, und dabei noch schneller arbeitet als ein menschlicher Tester“ lässt sich das folgende Fazit aufstellen:

Die Software HP Unified Functional Testing (UFT) ist in der Lage, alle Technologien und Anforderungen eines Verbundtests im vollen Umfang zu erfüllen. Dabei kann es erforderlich sein, verschiedene Lizenzen und Erweiterungen für HP UFT zu erwerben.

Unter Betrachtung der Zusatzbedingung, dass diese Software schneller als ein menschlicher Tester arbeiten soll, kann dieser Bedingung nur zugestimmt werden, wenn der Aufwand für die Erstellung und Pflege der Testfälle außen vorgelassen und sich dabei auf die reine Ausführungszeit des Tests gestützt wird.

Die Erstellung und Pflege von Testfällen gestaltet sich, wie die Untersuchungen zeigen, als zeitintensiv und komplex. Ein unerfahrener Tester, wie ich es selbst bin, benötigt anfangs zur Erstellung von Testfällen innerhalb von HP UFT Zeit, um die Funktionsweisen und Möglichkeiten der Software zu verstehen.

Sobald der Test erstellt ist und ausgeführt werden muss, überzeugt HP UFT durch die folgenden Punkte:

- schnelle Testausführung
- Ausführung rund um die Uhr
- Ausführung auf mehreren Systemen parallel
- Data Driven Testing
- automatischer Testbericht

Außerdem kann innerhalb von HP UFT über eine Schnittstelle eine erneute Ausführung eines Testfalls z. B. im Rahmen eines Regressions- oder Nachtests, angestoßen werden. Es empfiehlt sich solche Automatisierungssoftware vor allem für die Ausführung von sich häufig wiederholenden Tests und weniger für Tests mit hoher Variabilität.

Letzteres und Freihandtests sollten zum jetzigen Zeitpunkt, im Besonderen beim Testen einer GUI, besser von einem menschlichen Tester ausgeführt werden. Unter Umständen können gewisse Fehler erst in einem Freihandtest erkannt werden, da ansonsten wiederholt mit denselben bzw. ähnlichen Testdaten getestet werden würde und damit ein Fehler durch bestimmte Eingaben gar nicht erst im Testlauf auftreten kann.

Software ist bisher nicht in der Lage solche Freihandtests abzubilden. Im Moment lassen sich programmierte Testfälle automatisch ausführen. Software ist aktuell nicht in der Lage selbständig mit einer Art künstlicher Intelligenz Testfälle durchführen.

Die Vorteile von HP UFT und ähnlichen Tools liegen in der Möglichkeit, bestimmte Testfälle immer und immer wieder und parallel zu jeder beliebigen Zeit ausführen zu können. Zudem verringern diese Tools durch integrierte Schnittstellen zu ALM- und anderen Managementapplikationen den Aufwand an Testmanagement für eine Software.

Im Allgemeinen lässt sich sagen, dass Testsoftware einen Menschen nicht hundertprozentig ersetzen kann. Durch immer komplexere Software, wie z. B. für Internet of Things, Industrie 4.0 oder cloudbasierte Systeme, müsste eine Testsoftware wie ein Mensch denken können, um wie ein Mensch testen zu können.

Der Unterschied zu einem Menschen ist der, dass ein Mensch unterbewusst seine komplette Umgebung wahrnimmt. Wenn er seinen Fokus auf die Ausführung eines Tests legt, fallen ihm trotzdem unterbewusst Fehler an anderen Stellen innerhalb der zu testenden Software auf. Auf Grundlage dieser Ausführungen lässt sich sagen, dass eine Automatisierung einen strikten Fokus auf den vordefinierten Testlauf hat und von diesem nicht abweicht.

Lediglich, wenn ein Fehler Auswirkungen auf den Testlauf als solchen hat, ein Programm z. B. abstürzt oder gewisse Elemente anders dargestellt werden, erkennt die Automatisierungssoftware, dass ein Fehler aufgetreten ist.

Um es allgemeiner zu beschreiben, lässt sich auf Grundlage von Testerfahrungen und Untersuchungen feststellen, dass beim Testen einer GUI der Mensch wesentlich komplexere Denkweisen nachvollziehen kann und somit Fehler erkennt, die eigentlich gar nicht getestet wurden.

Handelt es sich bei der getesteten „Oberfläche“ nicht um eine GUI, sondern eine maschinell lesbare Struktur, die nach festen Regeln gebildet wurde wie z. B. XML-Schnittstellen, ist es möglich, den Fokus der Automatisierungssoftware auf die Überprüfung der kompletten Struktur zu legen und somit alle Fehler anhand einer Überprüfung der zuvor genannten Regeln zu erkennen.

Abschließend gesagt, kann ich mir nicht vorstellen, dass in naher Zukunft Automatisierungssoftware den menschlichen Tester ersetzen kann, weil die Anforderungen an die Software permanent steigt. Ich bin davon überzeugt, dass solche Software für Tester eine Arbeitserleichterung darstellt.

Literaturverzeichnis

[1] Ben-Hur, Y. (2016). eggplant functional. [online] http://www.qatestingtools.com/testing-tool/eggplant-functional (Abgerufen am 10.09.2016).

[2] Borland (2016). Micro focus community. [online] http://community.microfocus.com/borland/ (Abgerufen am 17.09.2016).

[3] BREDEX (2016a). Jubula. [online] http://www.bredex.de/produkte/infos/jubula/ (Abgerufen am 15.09.2016).

[4] BREDEX (2016b). Welcome to the bredex testing resources portal. [online] http://testing.bredex.de (Abgerufen am 15.09.2016).

[5] Chece, S. (2016). Testautomatisierung definition, tutorial und artikel. [online] https://www.testing-board.com/testautomatisierung/ [oberer Teil] (Abgerufen am 25.08.2016).

[6] CHIP (2016). Überblick im lizenzdickicht. [online] http://www.chip.de/artikel/Software-Lizenzen-im-Griff-2_30249120.html (Abgerufen am 23.09.2016).

[7] CN16 (2016). Test automation frameworks. [online] http://safsdev.sourceforge.net/FRAMESDataDrivenTestAutomationFrameworks.htm (Abgerufen am 25.08.2016).

[8] ComponentSource (2016). Silk test. [online] https://www.componentsource.com/product/silktest/prices (Abgerufen am 17.09.2016).

[9] Corporation, P. S. (2016a). Buy test studio - automated testing software. [online] http://www.telerik.com/purchase/teststudio (Abgerufen am 17.09.2016).

[10] Corporation, P. S. (2016b). Data driven testing. [online] http://docs.telerik.com/teststudio/features/data-driven-testing/overview (Abgerufen am 02.10.2016).

[11] Corporation, P. S. (2016c). Manual testing with test studio. [online] http://www.telerik.com/teststudio/manual-testing (Abgerufen am 16.09.2016).

[12] Corporation, P. S. (2016d). New features in test studio | telerik products what´s new. [online] http://www.telerik.com/support/whats-new/teststudio (Abgerufen am 17.09.2016).

[13] Corporation, P. S. (2016e). Software testing tools, automated testing software | telerik. [online] http://www.telerik.com/teststudio (Abgerufen am 10.09.2016).

[14] Corporation, P. S. (2016f). Support and learning. [online] http://www.telerik.com/support (Abgerufen am 16.09.2016).

[15] Corporation, P. S. (2016g). System requirements for test studio. [online] http://www.telerik.com/teststudio/system-requirements (Abgerufen am 17.09.2016).

[16] c´t (2016). Open-source-lizenzen. [online] https://www.heise.de/ct/artikel/Open-Source-Lizenzen-221957.html (Abgerufen am 23.09.2016).

[17] Deutschland, P. (2016). Software-lösungen von parasoft in köln – analysen und problembehebung. [online] http://www.parasoft.de/produkte/parasoft-test.html (Abgerufen am 15.09.2016).

[18] Edgewords (2016). Public training courses. [online] http://www.edgewordstraining.co.uk/training-courses/ (Abgerufen am 22.09.2016).

[19] Enterprise, H. P. (2016). Unified functional testing. [online] https://saas.hpe.com/de-de/software/uft (Abgerufen am 22.09.2016).

[20] Etestinghub (2015). Software testing tools - win runner. [online] http://www.etestinghub.com/winrunner.php (Abgerufen am 15.09.2016).

[21] eXept Software (2016a). Alle technologien in einem tool. [online] https://www.exept.de/technologien.html (Abgerufen am 11.09.2016).

[22] eXept Software (2016b). expecco - tool zur testautomatisierung von softwaretests. [online] https://www.exept.de/produkt/expecco.html (Abgerufen am 11.09.2016).

[23] Focus, M. (2016a). Borland silk test. [online] http://www.borland.com/de-DE/Products/Software-Testing/Automated-Testing/Silk-Test (Abgerufen am 09.09.2016).

[24] Focus, M. (2016b). Borland silk test - integrations. [online] http://www.borland.com/de-DE/Products/Software-Testing/Automated-Testing/Silk-Test/Integration (Abgerufen am 17.09.2016.

[25] Foundation, E. (2016a). Community and professional support for jubula. [online] http://www.eclipse.org/jubula/support.php (Abgerufen am 16.09.2016).

[26] Foundation, N. (2016b). Download node.js. [online] https://nodejs.org/en/download/ (Abgerufen am 21.09.2016).

[27] Gelperin, D. and Hetzel, B. (1988). The grow of software testing. pages 1–9.

[28] GitHub (2016). Contributing protactor. [online] https://github.com/angular/protractor/blob/master/CONTRIBUTING.md#questions (Abgerufen am 16.09.2016).

[29] Greiter, D. G. (2013). Test automation tools. [online] http://www.greiterweb.de/spw/test_werkzeuge.htm [mittlerer Teil] (Abgerufen am 25.08.2016).

[30] Harlan, K. (2013). Using casperjs, drush and jenkins to test drupal. [online] https://designhammer.com/blog/using-casperjs-drush-and-jenkins-test-drupal (Abgerufen am 16.09.2016).

[31] Hidayat, A. (2016). Download phantomjs. [online] http://phantomjs.org/download.html (Abgerufen am 21.09.2016).

[32] Hocke, R. (2014). Sikuli / sikulix documentation for version 1.1+ (2014 and later). [online] http://sikulix-2014.readthedocs.io/en/latest/index.html (Abgerufen am 15.09.2016).

[33] Hocke, R. (2016). Sikulix by raiman. [online] http://sikulix.com/ (Abgerufen am 13.09.2016).

[34] IBM (2016a). Automate test cases by integrating rational functional tester with rational testmanager. [online] http://www.ibm.com/developerworks/rational/library/automate-test-cases/ (Abgerufen am 22.09.2016).

[35] IBM (2016b). Preise anzeigen und kaufen. [online] https://www-112.ibm.com/software/howtobuy/buyingtools/paexpress/Express?P0=E1&part_number=D53NFLL,D530BLL,D54SHLL,D0BGLLL,D0BGMLL,D0BGNLL&catalogLocale=de_DE&Locale=de_DE&country=DEU&PT=jsp&CC=DEU&VP=&TACTICS=&S_TACT=&S_CMP=&brand=SB03 (Abgerufen am 17.09.2016).

[36] IBM (2016c). Rational functional tester. [online] http://www-03.ibm.com/software/products/en/functional (Abgerufen am 10.09.2016).

[37] Interactive, M. (2016). Winrunner user guide. [online] http://www.cbueche.de/WinRunnerUserGuide.pdf (Abgerufen am 15.09.2016).

[38] IST, T. T. (2016a). Capture & replay empiric test case generation. [online] http://www.testingtech.com/download/datasheets/Capture-and-Replay.pdf (Abgerufen am 15.09.2016).

[39] IST, T. T. (2016b). Tt workbench brochure. [online] http://www.testingtech.com/download/flyer/TTbrochure.pdf (Abgerufen am 12.09.2016).

[40] ISTQB AISBL, G. T. B. e. (2013). German testing board (gtb glossar). [online] http://www.german-testing-board.info/service/information/glossar.html (Abgerufen am 25.08.2016).

[41] IT-Management, F. (2016). E2e-testing mit protractor. [online] https://blog.flavia-it.de/e2e-testing-mit-protractor-teil-3-konfiguration-von-ausfuehrung-und-umgebung/ (Abgerufen am 15.09.2016).

[42] IT&PRODUCTION (2014). Markterfolg mit software unterstützen zehn vorteile integrierter systeme. [online] http://www.it-production.com/index.php?seite=einzel_artikel_ansicht&id=61578 (Abgerufen am 20.10.2016).

[43] Laycock, G. T. (1993). The theory and practice of specification based software testing. pages 6–18.

[44] LP, H. P. E. D. (2016a). Pricing unified functional testing. [online] https://saas.hpe.com/de-de/buy/uft (Abgerufen am 16.09.2016).

[45] LP, H. P. E. D. (2016b). Unified functional testing (uft). [online] http://www8.hp.com/de/de/software-solutions/unified-functional-automated-testing/ (Abgerufen am 08.09.2016).

[46] Microsoft (2016a). Erstellen, bearbeiten und verwalten von tests der codierten ui. [online] https://msdn.microsoft.com/de-de/library/ff977233.aspx (Abgerufen am 08.09.2016).

[47] Microsoft (2016b). Microsoft .net framework 4 (standalone installer). [online] https://www.microsoft.com/en-US/download/details.aspx?id=17718 (Abgerufen am 21.09.2016).

[48] Microsoft (2016c). Selenium components for coded ui cross browser testing. [online] https://visualstudiogallery.msdn.microsoft.com/11cfc881-f8c9-4f96-b303-a2780156628d/ (Abgerufen am 08.09.2016).

[49] Microsoft (2016d). Verwenden von benutzeroberflächenautomatisierung zum testen des codes. [online] https://msdn.microsoft.com/de-de/library/dd286726.aspx (Abgerufen am 08.09.2016).

[50] Microsoft (2016e). Visual studio-kaufoptionen. [online] https://www.visualstudio.com/products/how-to-buy-vs (Abgerufen am 16.09.2016).

[51] Neumann, A. (2011). Test-framework selenium in version 2.0 erschienen. [online] http://www.heise.de/newsticker/meldung/Test-Framework-Selenium-in-Version-2-0-erschienen-1276752.html (Abgerufen am 15.09.2016).

[52] Packard, H. (2014). *Cross-browser functional testing best practices: HP Unified Functional Testing best practices series - Technical white paper*. HP, Palo Alto.

[53] Parasoft (2016a). Parasoft forum. [online] http://forums.parasoft.com/ (Abgerufen am 15.09.2016).

[54] Parasoft (2016b). Parasoft training & professional services. [online] https://www.parasoft.com/service/professional-services/ (Abgerufen am 15.09.2016).

[55] Parasoft (2016c). Soatest api testing. [online] https://www.parasoft.com/product/soatest/ (Abgerufen am 12.09.2016).

[56] Parasoft (2016d). Supported environments. [online] https://www.parasoft.com/product/soatest/#supp_env (Abgerufen am 08.09.2016).

[57] Partner, F. F. S. (2016a). Rapidrep test suite zur testautomatisierung - das back-end automatisiert testen. automatisiert, wirtschaftlich, zuverlässig. [online] http://www.rapidrep.com/de/uebersicht-test-suite (Abgerufen am 12.09.2016).

[58] Partner, F. F. S. (2016b). Support - wir sind immer für sie da - rapidrep. [online] http://www.rapidrep.com/de/support (Abgerufen am 15.09.2016).

[59] Partner, F. F. S. (2016c). Systemanforderungen der rapidrep test suite. [online] http://www.rapidrep.com/de/systemanforderungen-test-suite (Abgerufen am 14.09.2016).

[60] Perriault, N. and contributors (2016a). Installation casperjs 1.1.0-dev documentation. [online] http://docs.casperjs.org/en/latest/installation.html (Abgerufen am 14.09.2016).

[61] Perriault, N. and contributors (2016b). Testing casperjs 1.1.0-dev documentation. [online] http://docs.casperjs.org/en/latest/testing.html (Abgerufen am 14.09.2016).

[62] Project, S. (2012). Introduction - selenium documentation. [online] http://www.seleniumhq.org/docs/01_introducing_selenium.jsp (Abgerufen am 13.09.2016).

[63] Protractor (2016). Tutorial. [online] http://www.protractortest.org/#/tutorial (Abgerufen am 14.09.2016).

[64] QFS (2016a). Faq: Häufig gestellte fragen zu qf-test. [online] https://www.qfs.de/qf-test/faq.html (Abgerufen am 16.09.2016).

[65] QFS (2016b). Gui testautomatisierung für java und web. [online] https://www.qfs.de/qf-test/testautomatisierung-mit-qf-test.html (Abgerufen am 09.09.2016).

[66] QFS (2016c). Preise für lizenzen & pflegevertrag. [online] https://www.qfs.de/qf-test/preise.html (Abgerufen am 17.09.2016).

[67] QFS (2016d). Schulung und beratung. [online] https://www.qfs.de/qf-test-support/schulung-beratung.html (Abgerufen am 22.09.2016).

[68] Ranorex (2016a). Data driven testing. [online] http://www.ranorex.com/support/user-guide-20/lesson-3-data-driven-testing.html (Abgerufen am 23.09.2016).

[69] Ranorex (2016b). Error behavior settings. [online] http://www.ranorex.com/support/user-guide-20/lesson-4-ranorex-test-suite.html (Abgerufen am 10.10.2016).

[70] Ranorex (2016c). Keyword driven testing. [online] http://www.ranorex.com/blog/keyword-driven-test-automation-framework/ (Abgerufen am 23.09.2016).

[71] Ranorex (2016d). Preise und lizenzmodelle | ranorex premium und runtime lizenzen. [online] http://www.ranorex.de/preise.html (Abgerufen am 16.09.2016).

[72] Ranorex (2016e). Ranorex remote. [online] http://www.ranorex.com/remote-testing.html (Abgerufen am 22.09.2016).

[73] Ranorex (2016f). Test automation user guide | ranorex – automated testing tutorial. [online] http://www.ranorex.com/support/user-guide-20.html (Abgerufen am 09.09.2016).

[74] Ranorex (2016g). Wie funktioniert ranorex | ranorex - gui software test automatisierung. [online] http://www.ranorex.de/wie-funktioniert-testautomatisierung.html (Abgerufen am 09.09.2016).

[75] Smartbear (2016). Distributed testing tutorial. [online] https://support.smartbear.com/articles/testcomplete/distributed-testing-tutorial/ (Abgerufen am 22.09.2016).

[76] Software, S. (2016a). Automated software testing. [online] https://smartbear.com/product/testcomplete/overview/ (Abgerufen am 11.09.2016).

[77] Software, S. (2016b). Automated web application testing. [online] https://smartbear.com/product/testcomplete/web-module/overview/ (Abgerufen am 11.09.2016).

[78] Software, S. (2016c). Introduction to data-driven testing. [online] https://smartbear.com/learn/automated-testing/introduction-to-data-driven-testing/ (Abgerufen am 22.10.2016).

[79] Software, S. (2016d). Latest release of soapui. [online] https://www.soapui.org/downloads/latest-release.html (Abgerufen am 14.09.2016).

[80] Software, S. (2016e). Purchase soapui ng pro licenses. [online] https://www.soapui.org/store.html (Abgerufen am 17.09.2016).

[81] Software, S. (2016f). Smartbear support. [online] https://support.smartbear.com/ (Abgerufen am 16.09.2016).

[82] Software, S. (2016g). Soapui | functional testing for soap and rest apis. [online] https://www.soapui.org/ (Abgerufen am 14.09.2016).

[83] Software, S. (2016h). Support. [online] https://www.soapui.org/support.html (Abgerufen am 17.09.2016).

[84] Software, S. (2016i). System requirements. [online] https://support.smartbear.com/viewarticle/82006/ (Abgerufen am 15.09.2016).

[85] Software, S. (2016j). Testcomplete platform pricing. [online] https://smartbear.com/product/testcomplete/pricing/ (Abgerufen am 17.09.2016).

[86] Technologies, S. (2016a). Consultancy - professional support in all test phases. [online] http://www.testingtech.com/services/consultancy.php (Abgerufen am 15.09.2016).

[87] Technologies, S. (2016b). Ttworkbench - the reliable test automation platform. [online] http://www.testingtech.com/products/ttworkbench.php (Abgerufen am 12.09.2016).

[88] TestPlant (2016a). Cross platform browser compatability testing with eggplant. [online] http://www.testplant.com/explore/testing-use-cases/cross-browser-testing-2/ (Abgerufen am 16.09.2016).

[89] TestPlant (2016b). eggplant functional. [online] http://www.testplant.com/eggplant/testing-tools/eggplant-developer/ (Abgerufen am 16.09.2016).

[90] TestPlant (2016c). eggplant functional downloads. [online] http://www.testplant.com/dlds/eggplant-functional/ (Abgerufen am 17.09.2016).

[91] Thomas Bucsics, Manfred Baumgartner, R. S. S. G. (2015). *Basiswissen Testautomatisierung – Konzepte, Methoden und Techniken*. 2.Auflage dpunkt Verlag, Heidelberg.

[92] Tricentis (2016a). The continuous testing company | tricentis. [online] http://www.tricentis.com/de/ (Abgerufen am 15.09.2016).

[93] Tricentis (2016b). System requirements for tricentis tosca testsuite. [online] https://documentation.tricentis.com/en/930/index.htm#systemanforderungen/systemrequirements.htm%3FTocPath%3D_____6 (Abgerufen am 17.09.2016).

[94] Tricentis (2016c). Technology integration. [online] http://www.tricentis.com/de/tricentis-tosca-testsuite/technology-integration/ (Abgerufen am 08.09.2016).

[95] Tricentis (2016d). Tosca public training course. [online] http://www.tricentis.com/academy/courses/tosca-public-training/ (Abgerufen am 16.09.2016).

[96] Tricentis (2016e). Tosca testsuite overview. [online] http://www.tricentis.com/de/resource-assets/tosca-testsuite-overview/ (Abgerufen am 10.09.2016).

[97] Tricentis (2016f). Tricentis tosca testsuite. [online] http://www.tricentis.com/de/tricentis-tosca-testsuite/ (Abgerufen am 10.09.2016).

[98] Tricentis (2016g). Unterstützte technologien & applikationen. [online] http://www.tricentis.com/de/tricentis-tosca-testsuite/supported-technology/ (Abgerufen am 10.09.2016).

[99] Tutorialspoint (2016a). Capture/replay tool. [online] http://www.tutorialspoint.com/software_testing_dictionary/capture_replay_tool.htm (Abgerufen am 15.09.2016).

[100] Tutorialspoint (2016b). Data driven testing. [online] https://www.tutorialspoint.com/software_testing_dictionary/data_driven_testing.htm (Abgerufen am 23.09.2016).

[101] Tutorialspoint (2016c). Keyword driven testing. [online] https://www.tutorialspoint.com/software_testing_dictionary/keyword_driven_testing.htm (Abgerufen am 23.09.2016).

[102] Tutorialspoint (2016d). Qtp - error handling (vorgänger von uft). [online] https://www.tutorialspoint.com/qtp/qtp_error_handling.htm (Abgerufen am 10.10.2016).

[103] Witte, F. (2016). *Testmanagement und Softwaretest*. Springer Vieweg, Landshut.

[104] Worksoft (2016a). Web application testing. [online] https://www.worksoft.com/solutions/web-application-testing (Abgerufen am 13.09.2016).

[105] Worksoft (2016b). worksoft certify and sap solution manager. [online] https://www.worksoft.com/files/resources/Worksoft-Datasheet-Certify-And-SAP-Solution-Manager.pdf (Abgerufen am 14.09.2016).

[106] Worksoft (2016c). Worksoft services. [online] https://www.worksoft.com/services (Abgerufen am 15.09.2016).

Sämtliche Webquellen finden Sie als Screenshots auf der CD zu dieser Arbeit.

Kapitel 4

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die von mir an der Hochschule für Telekommunikation Leipzig (FH) eingereichte Abschlussarbeit zum Thema „*Vergleich verschiedener Tools zur automatischen Durchführung von Testfällen für unterschiedliche Programme innerhalb von Verbundtests*“ vollkommen selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Essen, den 13.11.2016

Robert Schnorr

Anhang A

Glossar

Bitte nutzen Sie das Glossar unter

https://www.testautomatisierung.org/testglossar-test-deutsch/

Hinweis zur Quelle:
ISTQB / German Testing Board (GTB Glossar) – Version 2.2 vom 19. April 2013
(http://www.german-testing-board.info/service/information/glossar.html)

Abgerufen am 24.08.2016 von der Seite
https://www.testautomatisierung.org/testglossar-test-deutsch/

Anhang B

Installation der Softwarewerkzeuge

B.1 Ranorex Studio

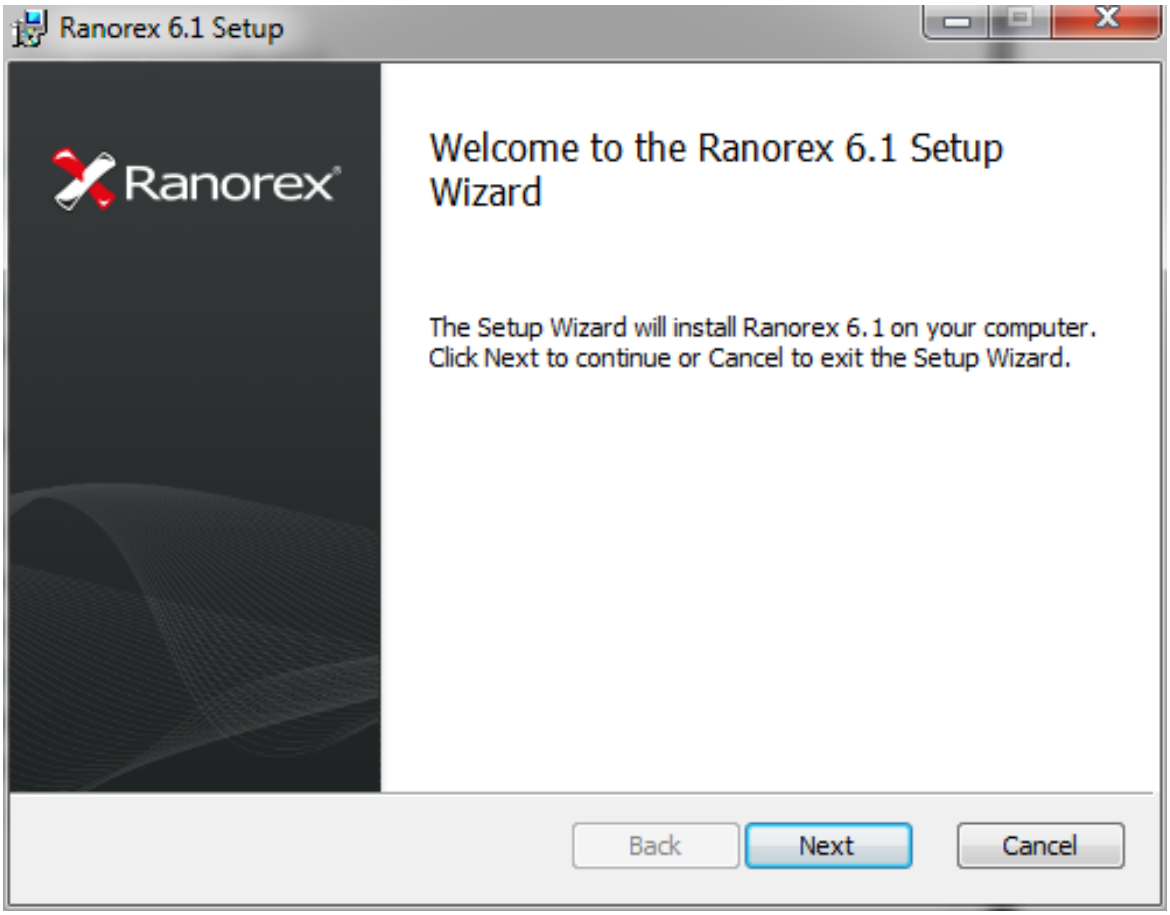

Abbildung B.1 Installation des Ranorex Studios (1)

Abbildung B.2 Installation des Ranorex Studios (2)

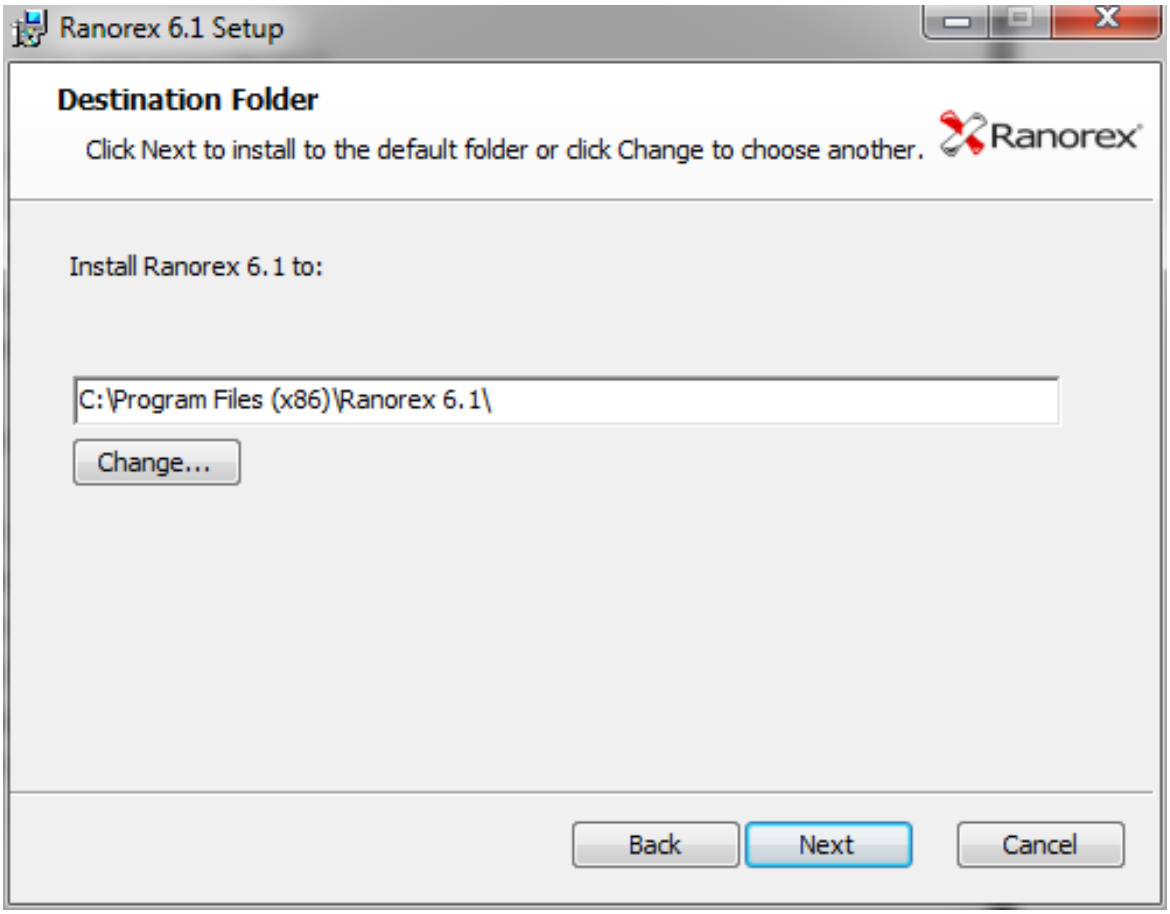

Abbildung B.3 Installation des Ranorex Studios (3)

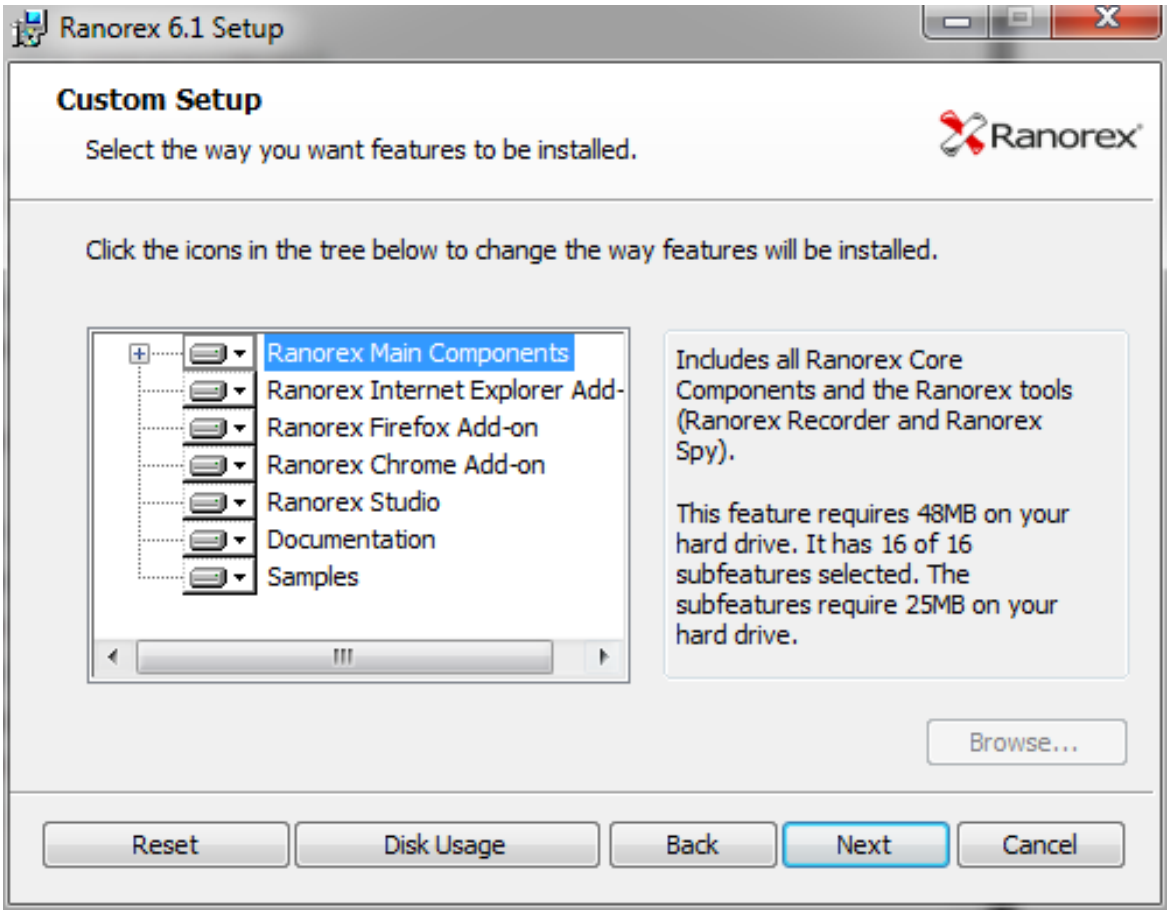

Abbildung B.4 Installation des Ranorex Studios (4)

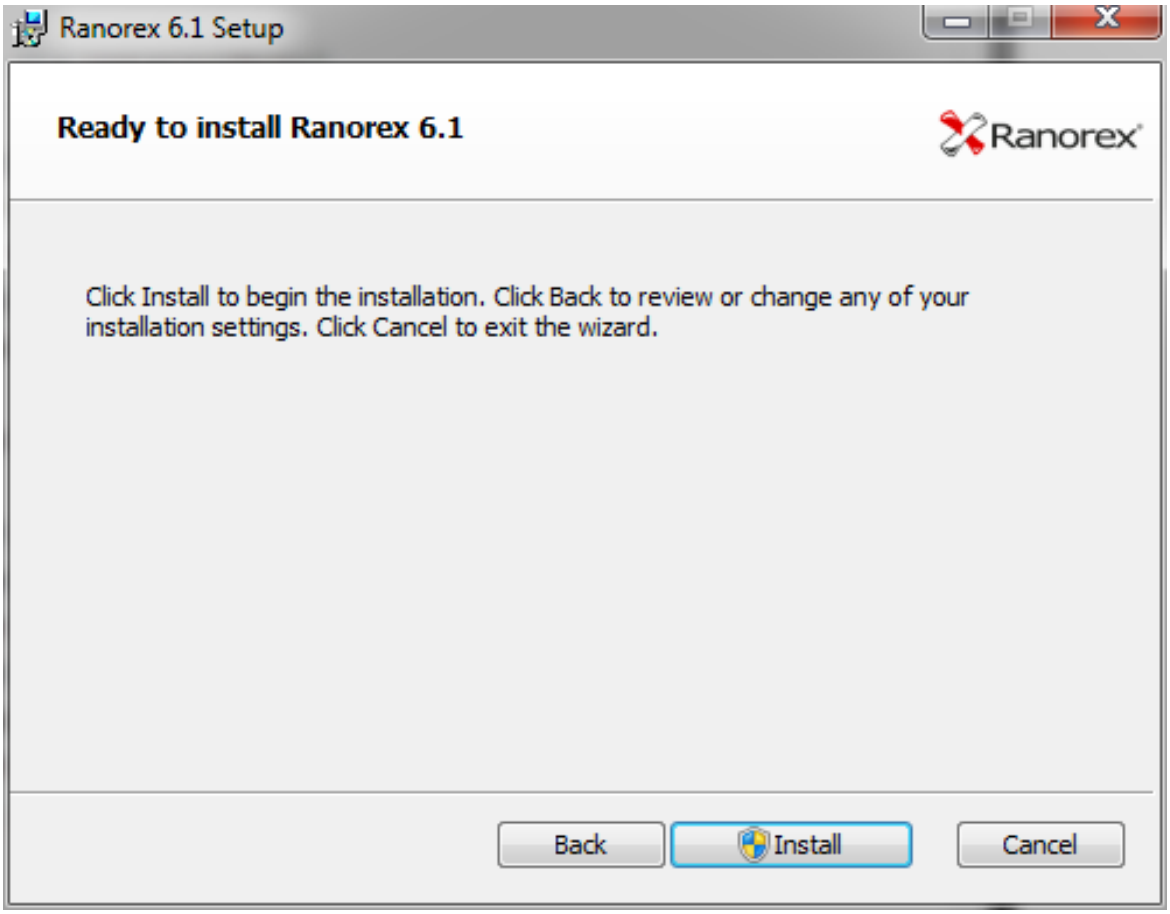

Abbildung B.5 Installation des Ranorex Studios (5)

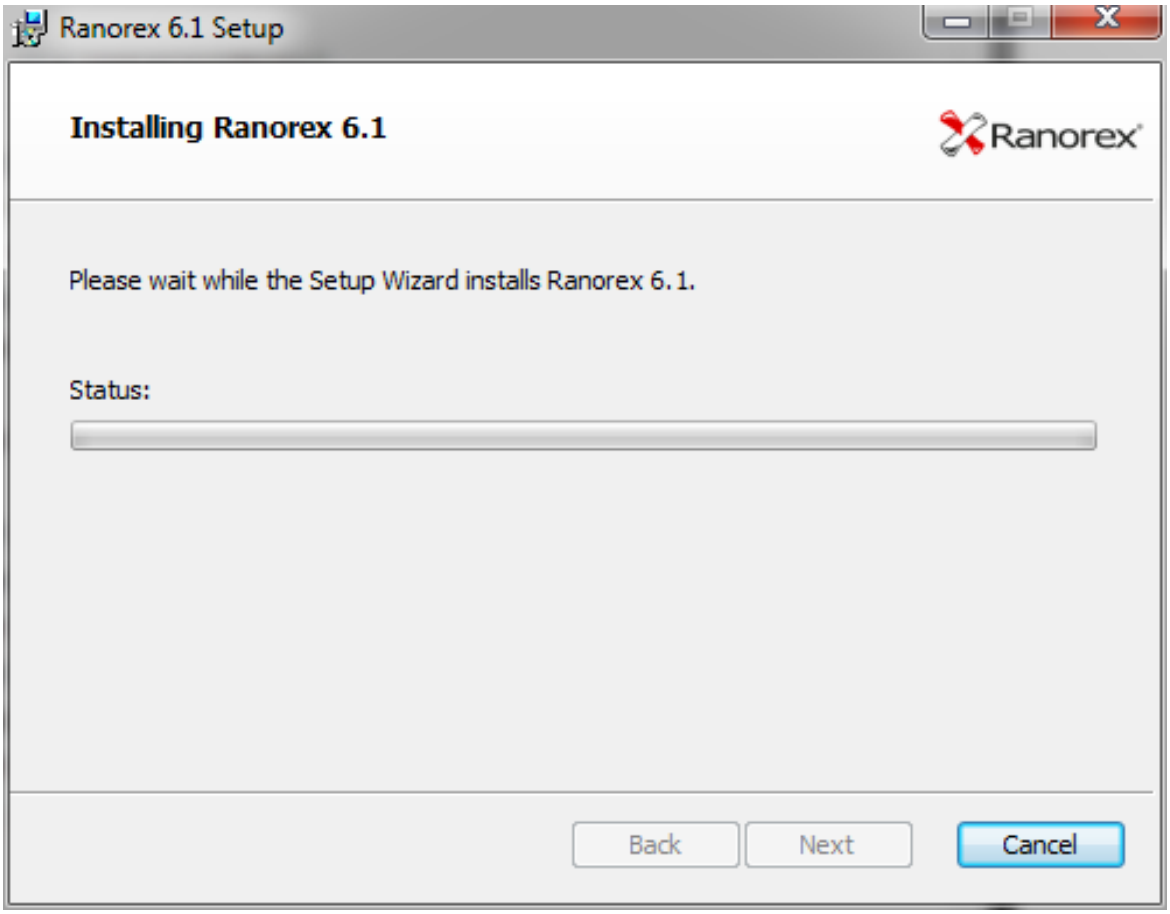

Abbildung B.6 Installation des Ranorex Studios (6)

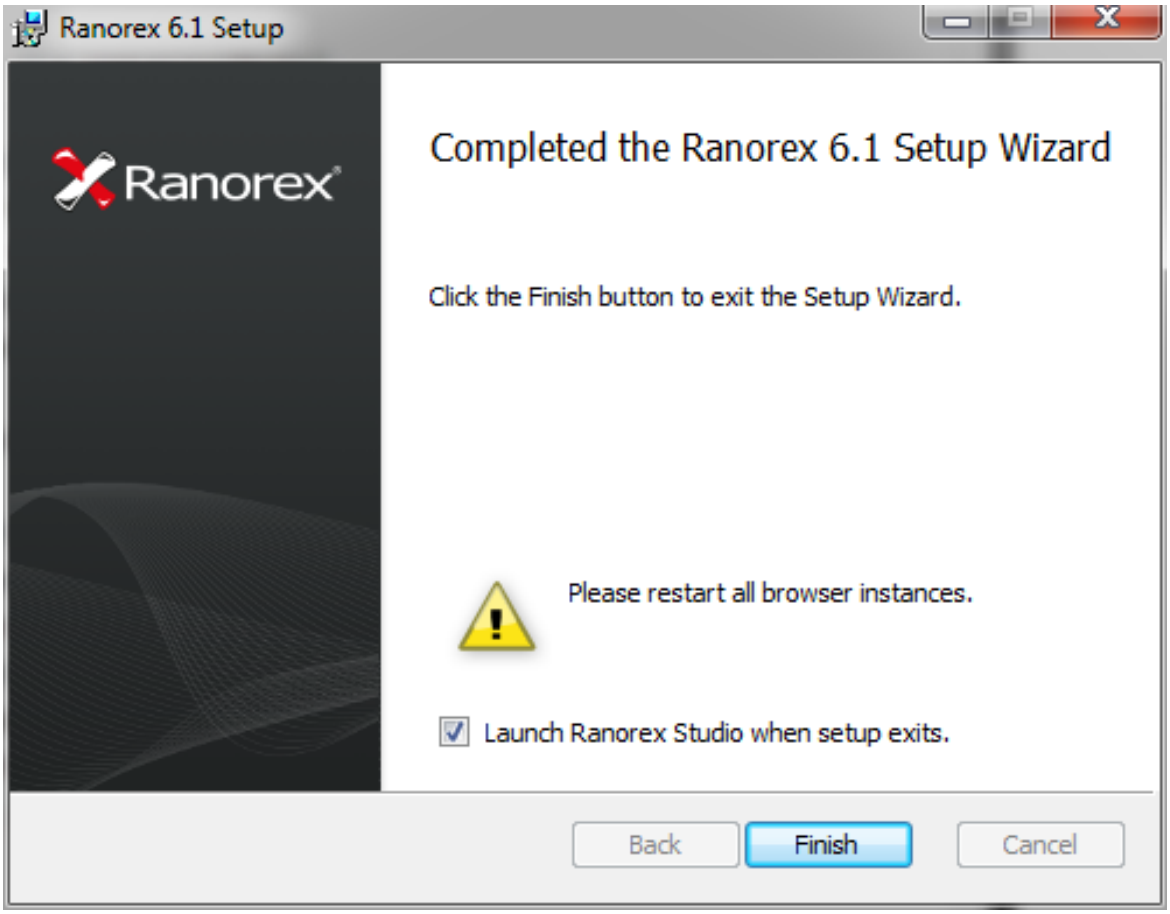

Abbildung B.7 Installation des Ranorex Studios (7)

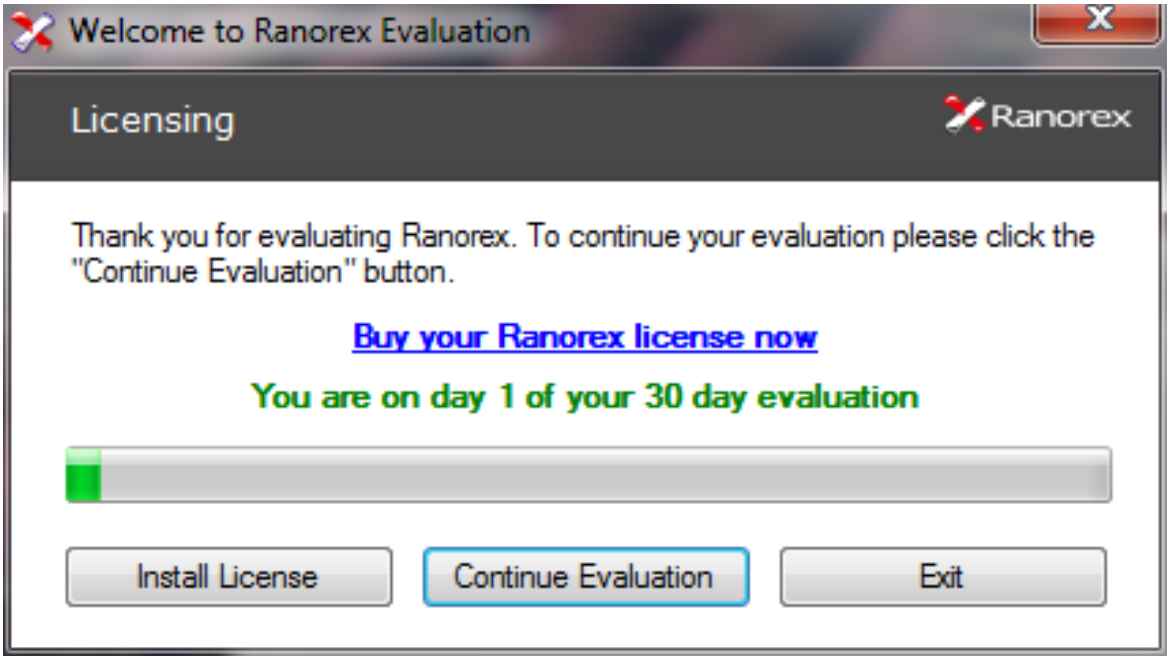

Abbildung B.8 Installation des Ranorex Studios (8)

Abbildung B.9 Installation des Ranorex Studios (9)

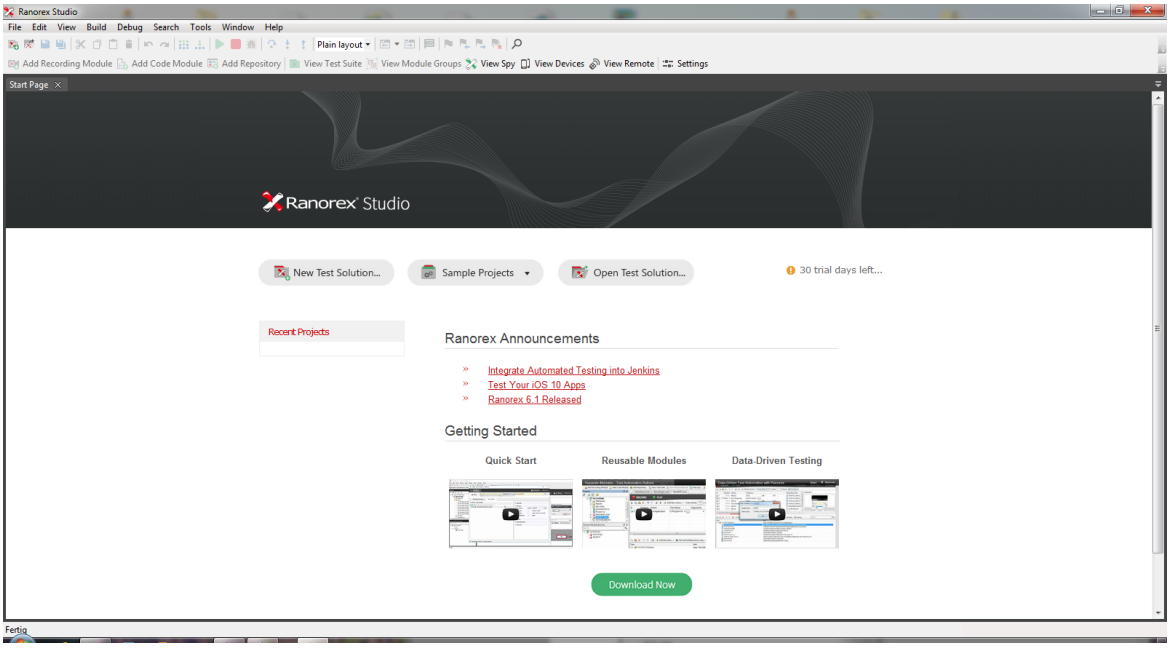

Abbildung B.10 Installation des Ranorex Studios (10)

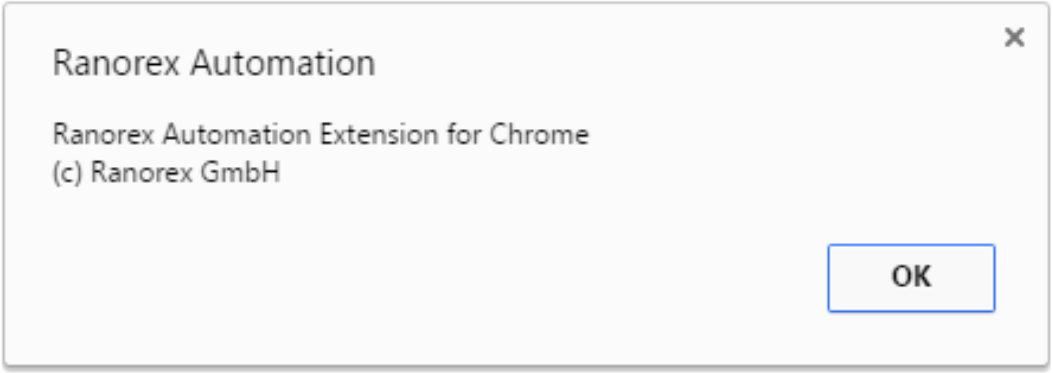

Abbildung B.11 Installation des Ranorex Studios (11)

B.2 Telerik Studio

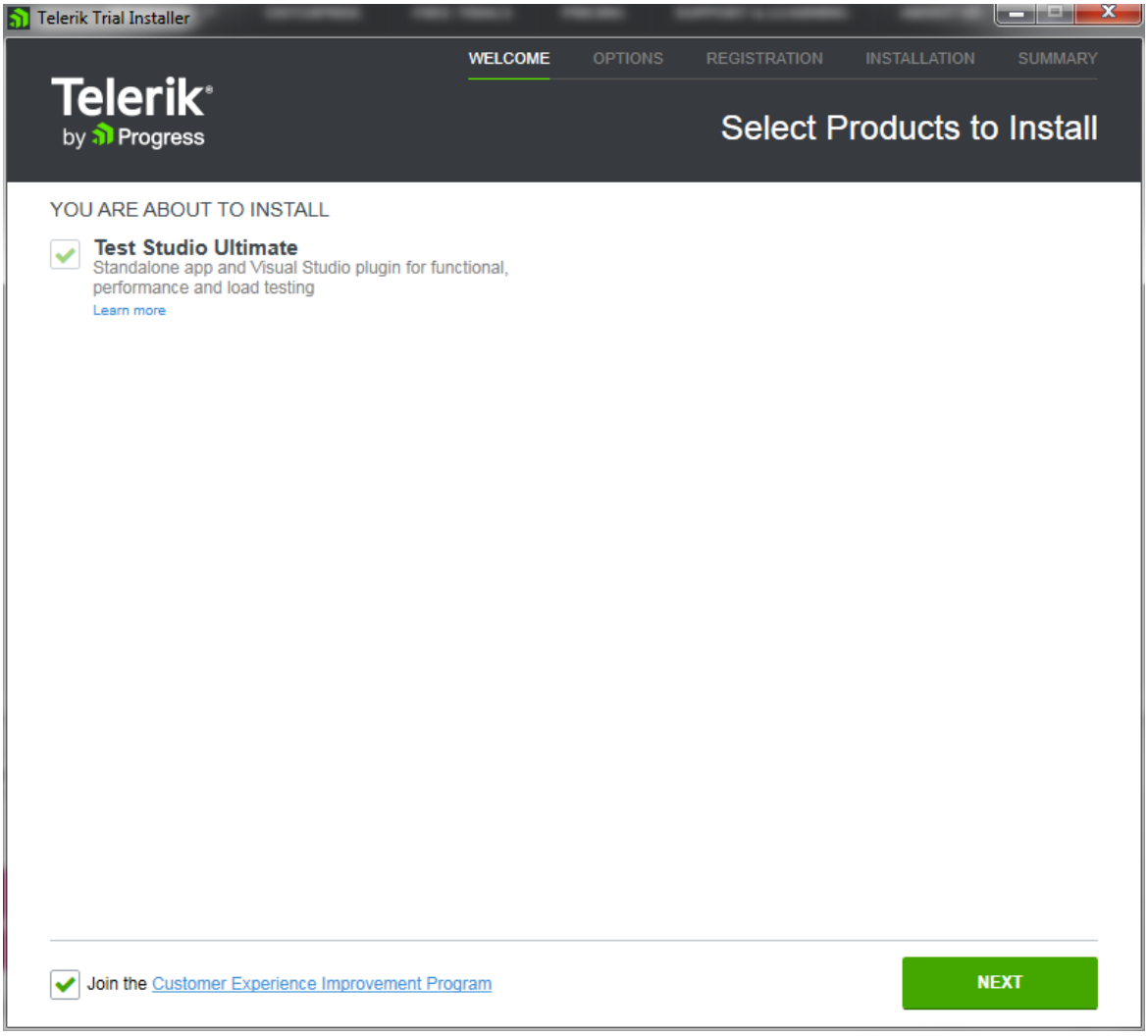

Abbildung B.12 Installation des Telerik Test Studios (1)

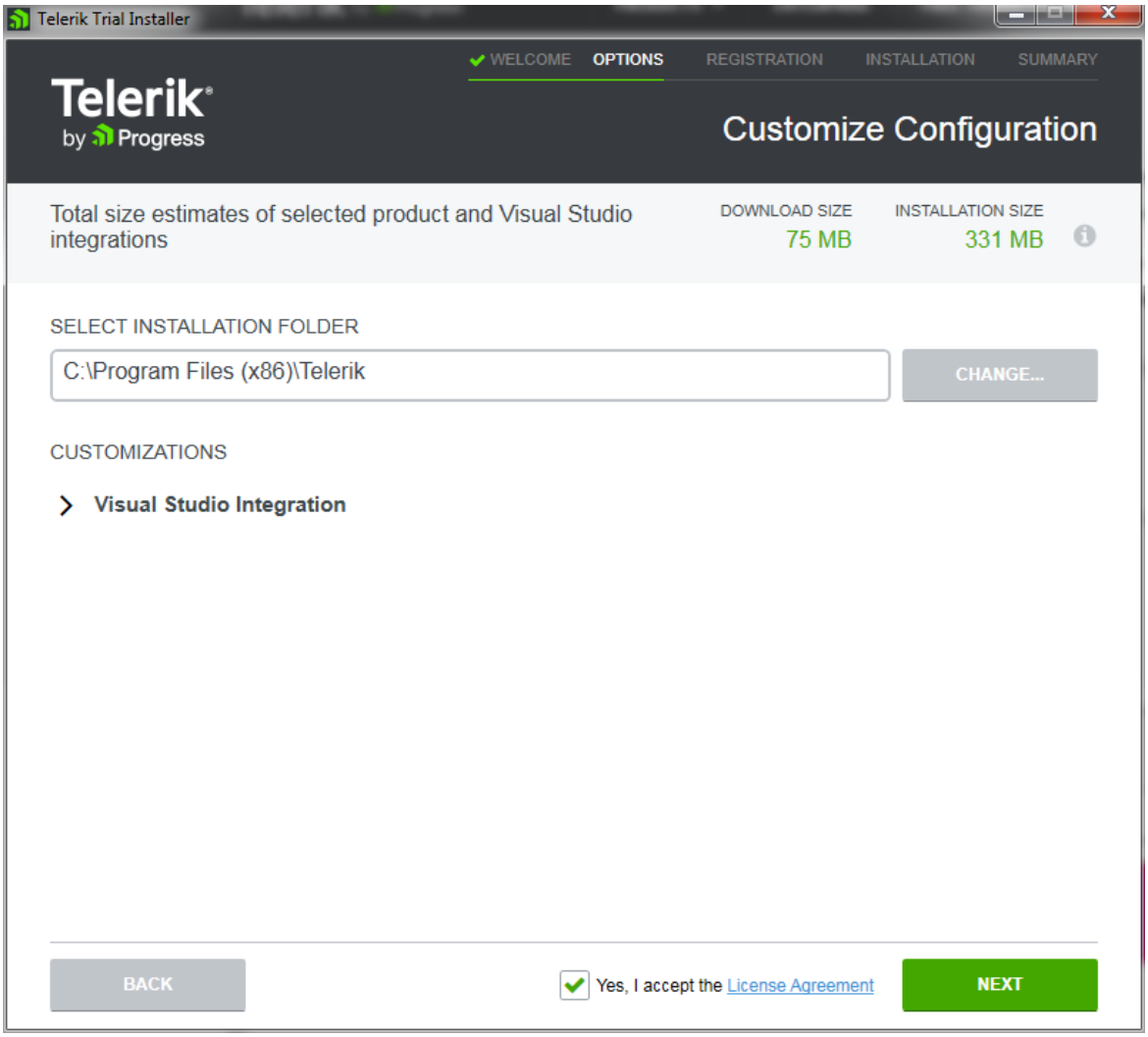

Abbildung B.13 Installation des Telerik Test Studios (2)

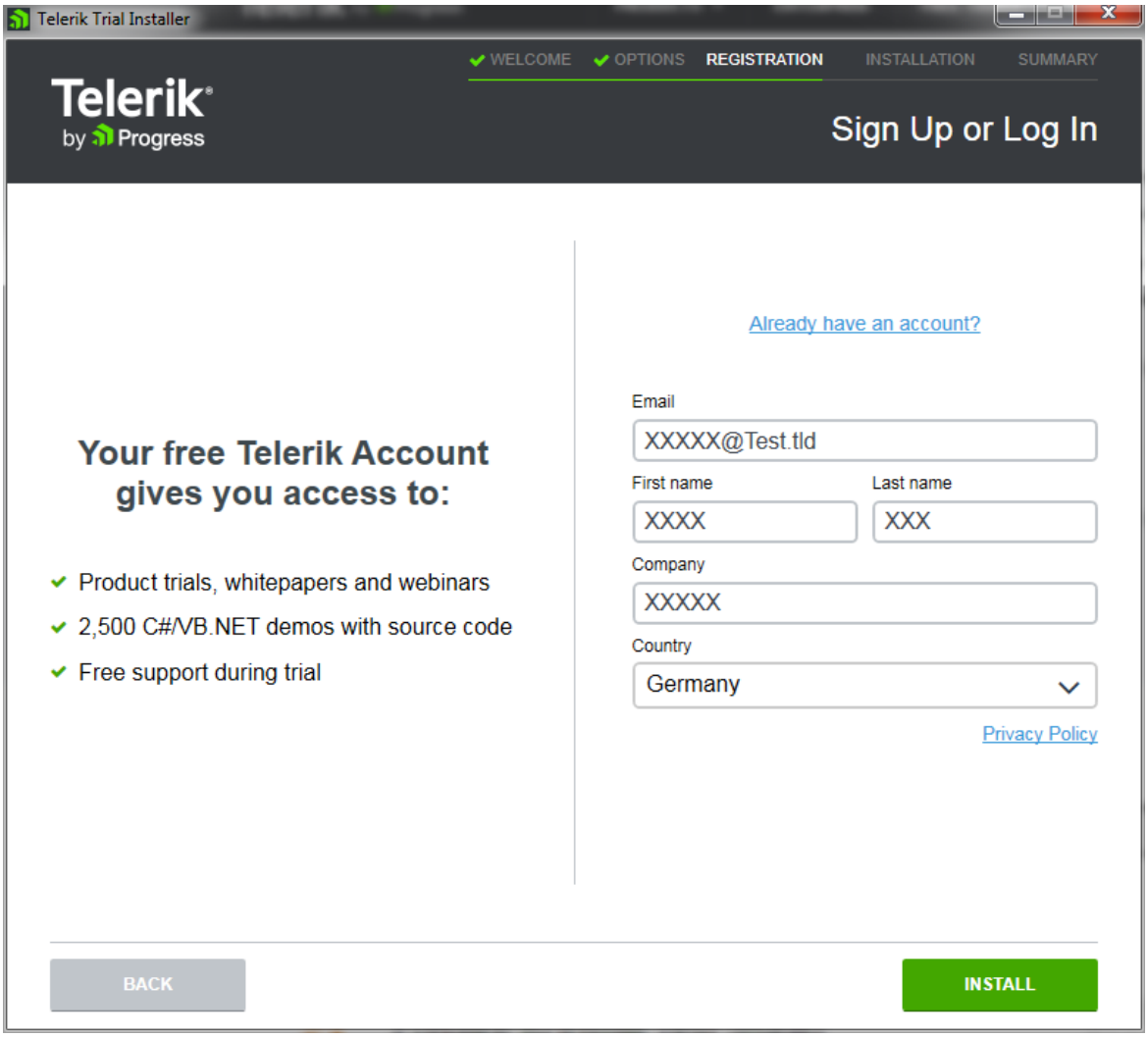

Abbildung B.14 Installation des Telerik Test Studios (3)

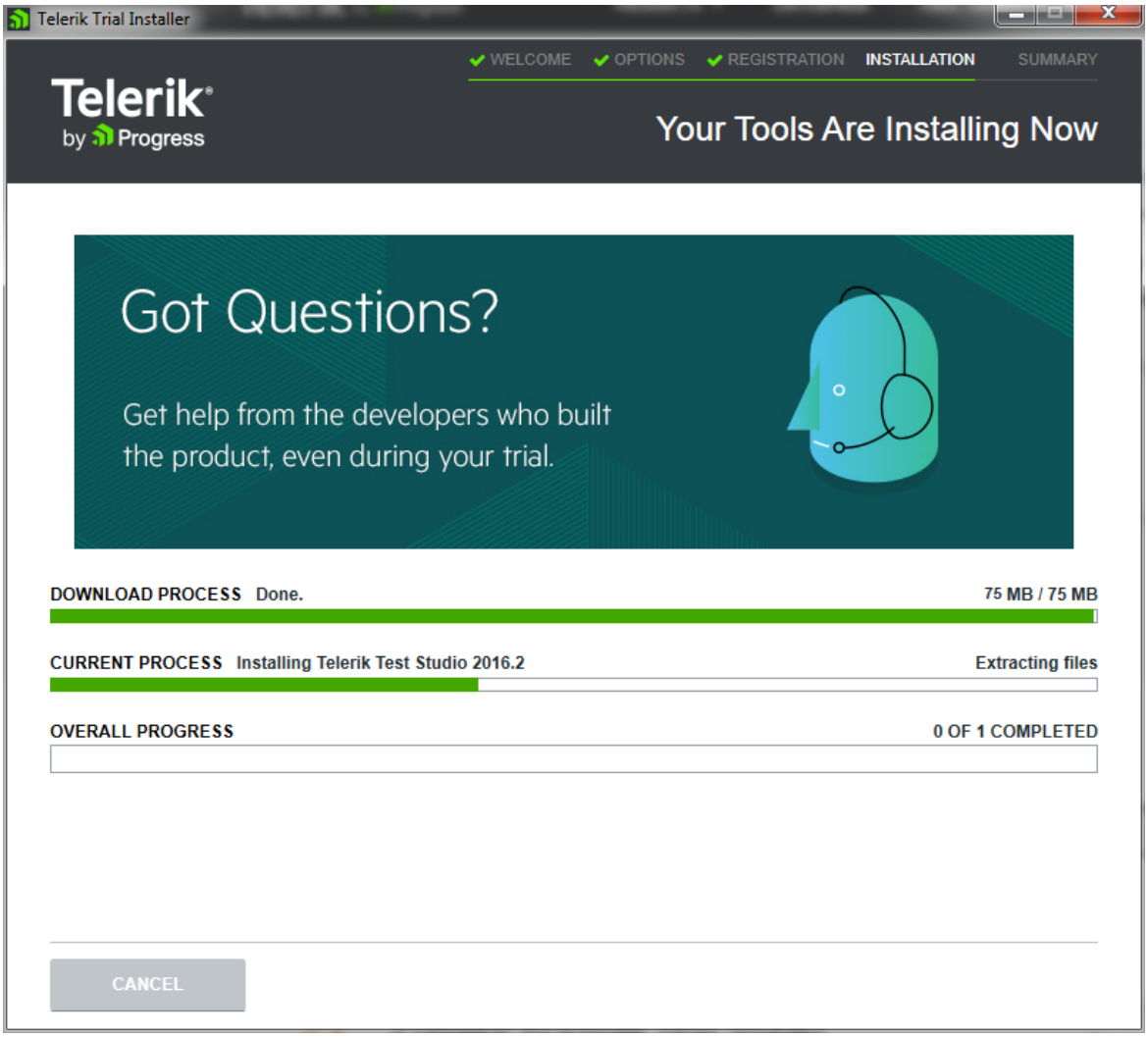

Abbildung B.15 Installation des Telerik Test Studios (4)

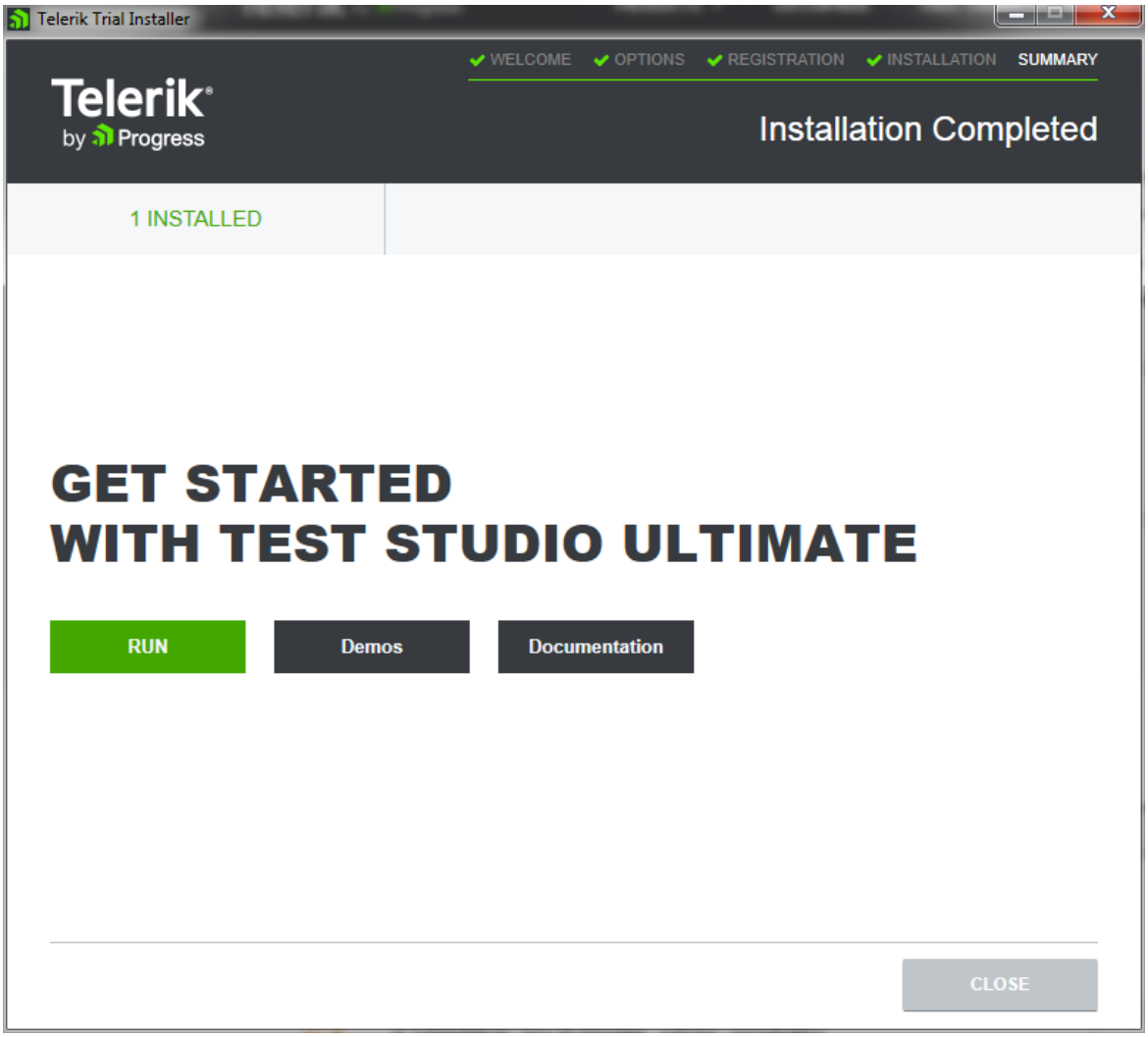

Abbildung B.16 Installation des Telerik Test Studios (5)

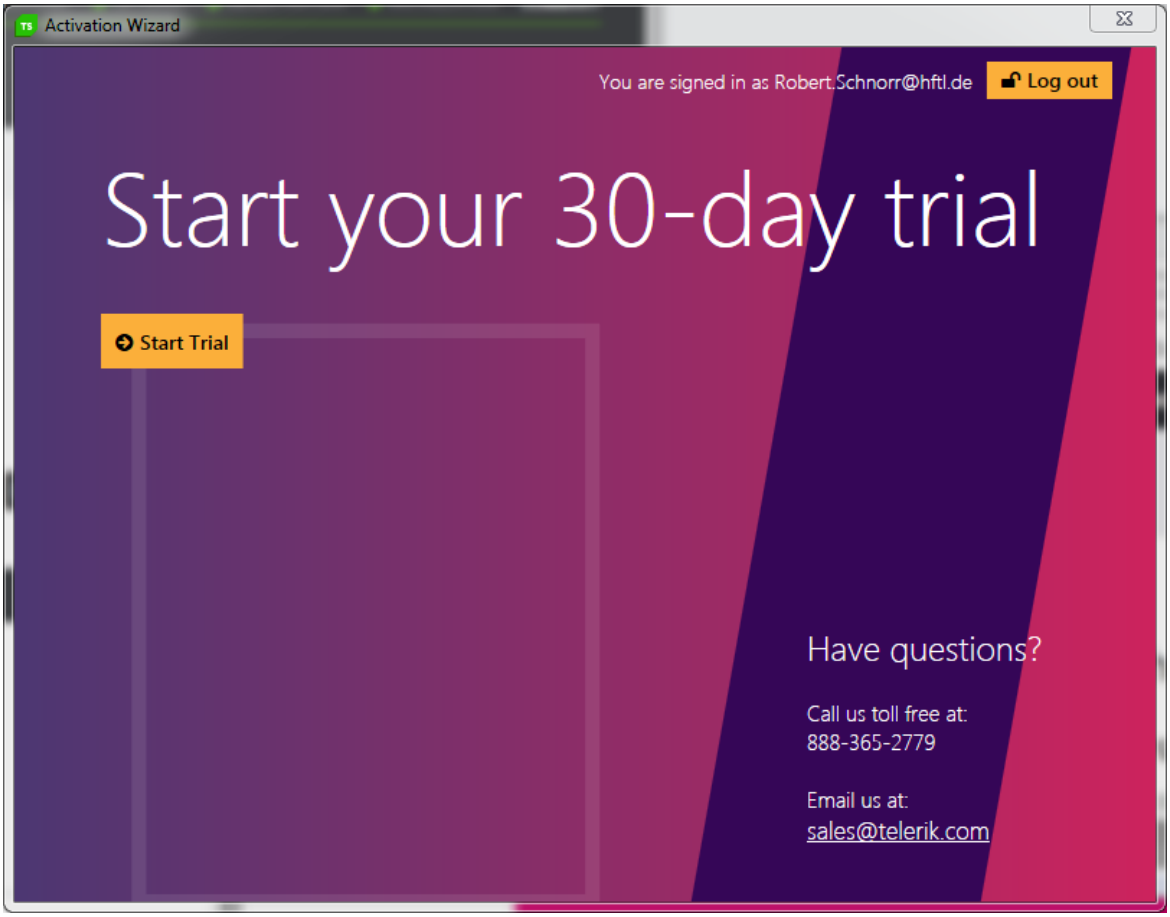

Abbildung B.17 Installation des Telerik Test Studios (6)

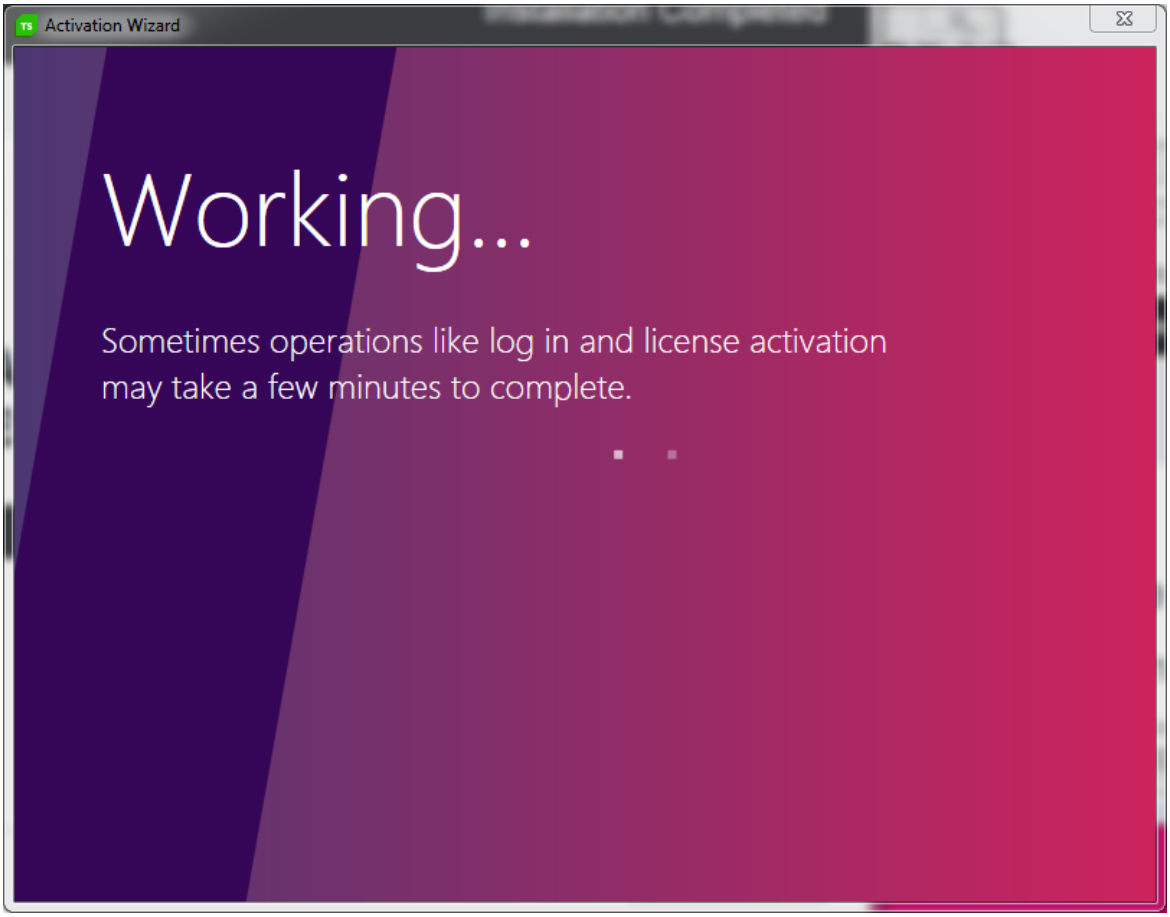

Abbildung B.18 Installation des Telerik Test Studios (7)

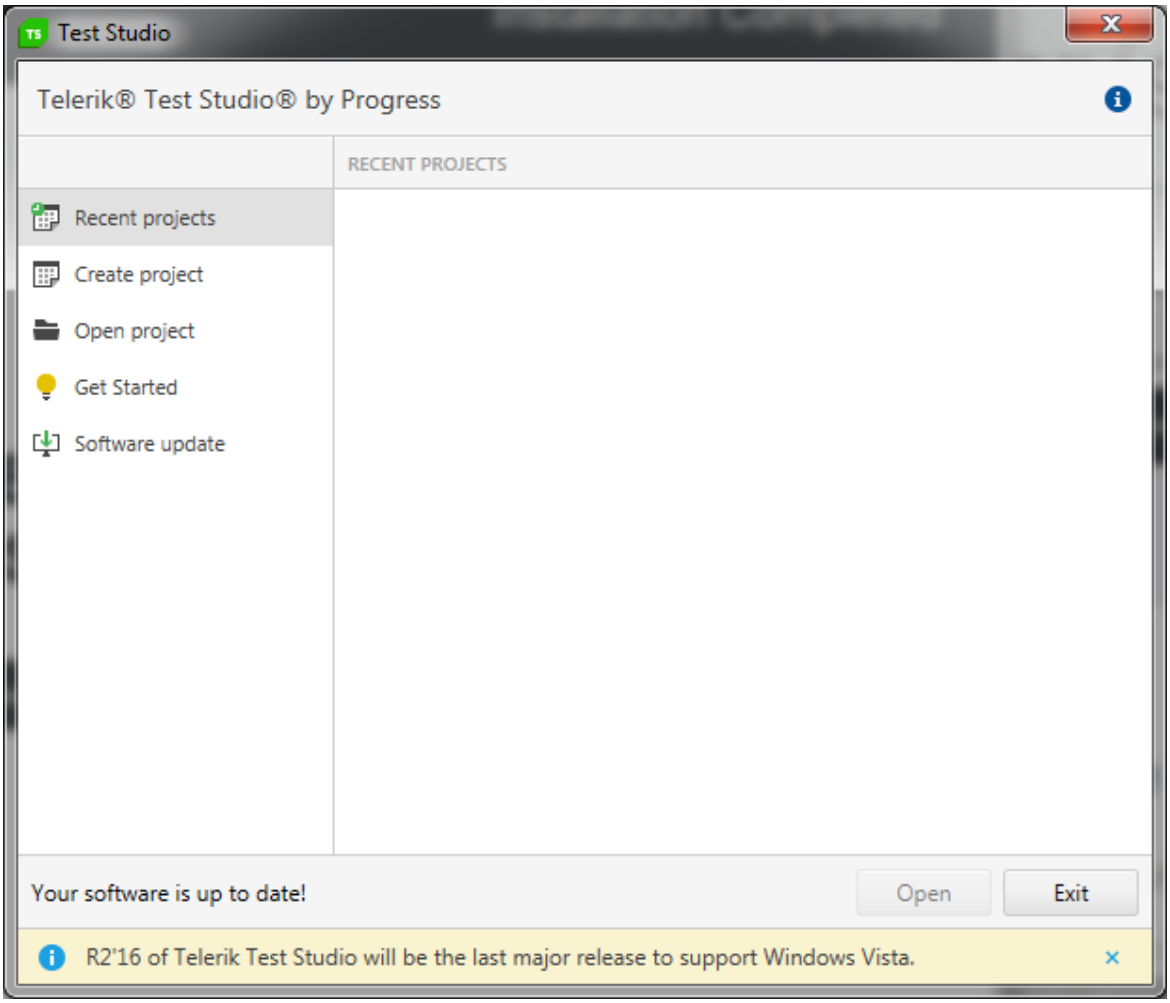

Abbildung B.19 Installation des Telerik Test Studios (8)

B.3 SikuliX

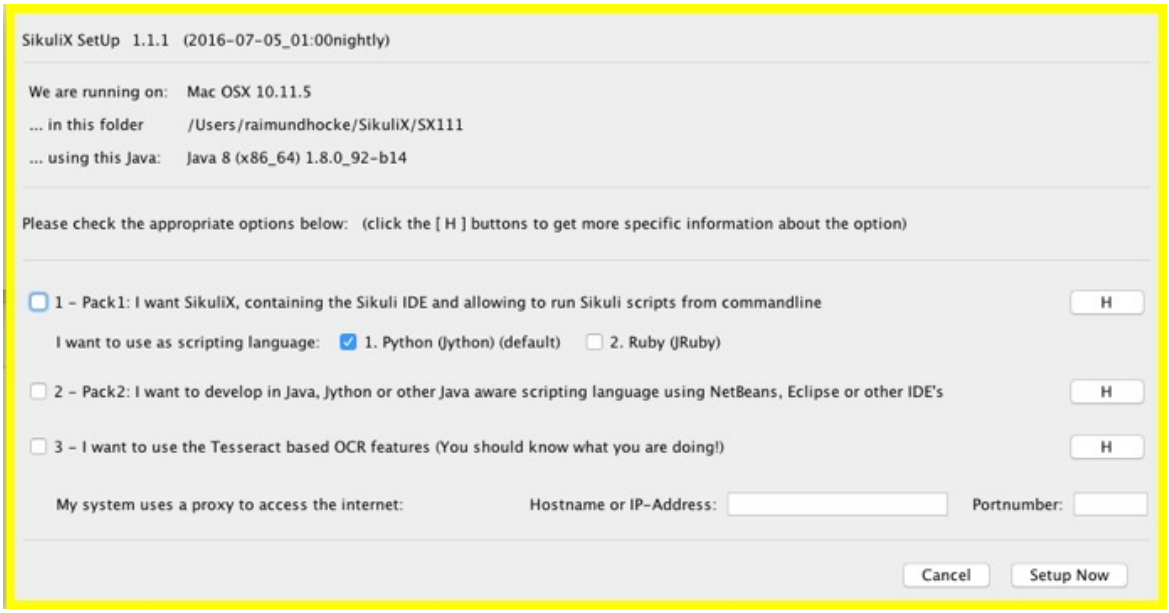

Abbildung B.20 Installation von SikuliX

B.4 HP Unified Functional Testing (UFT)

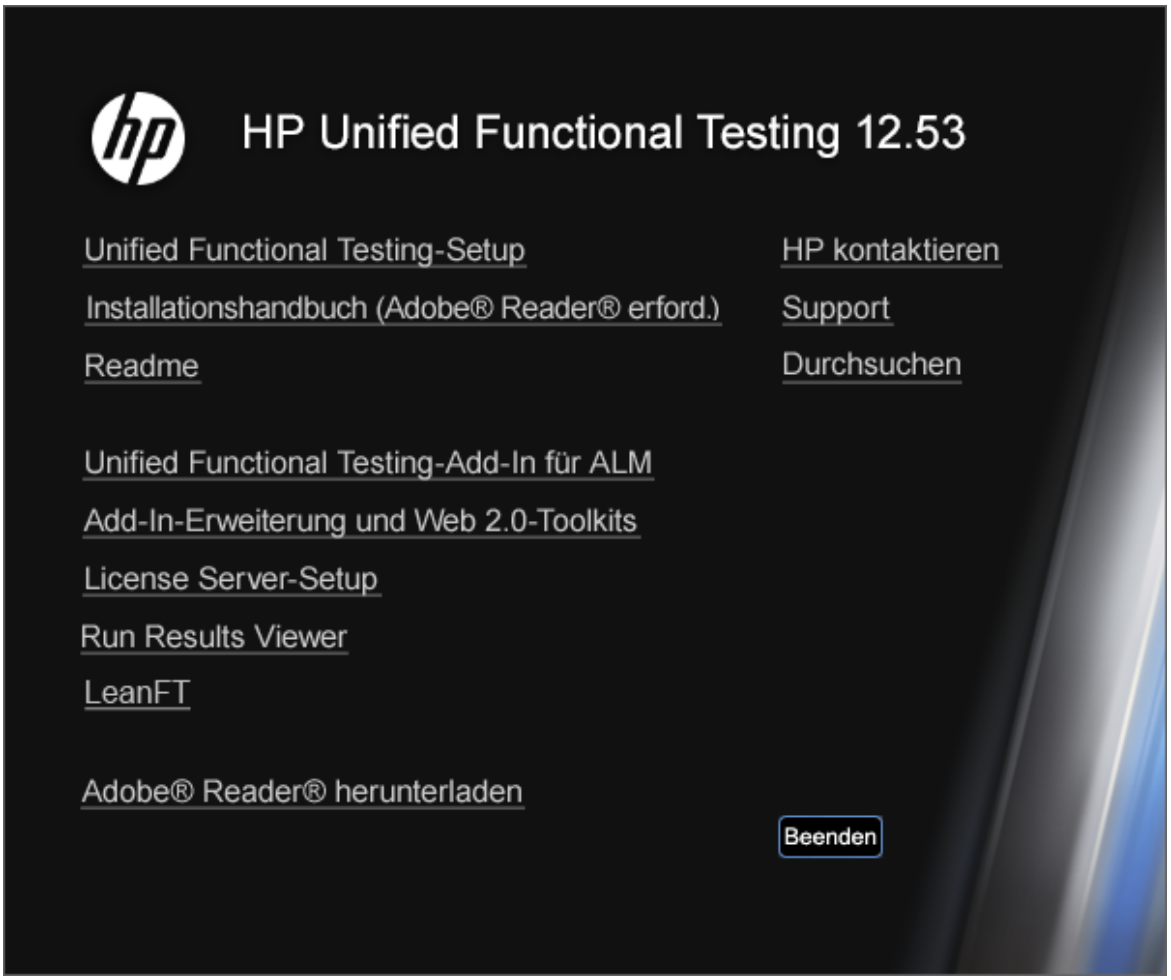

Abbildung B.21 Installation der HP UFT (1)

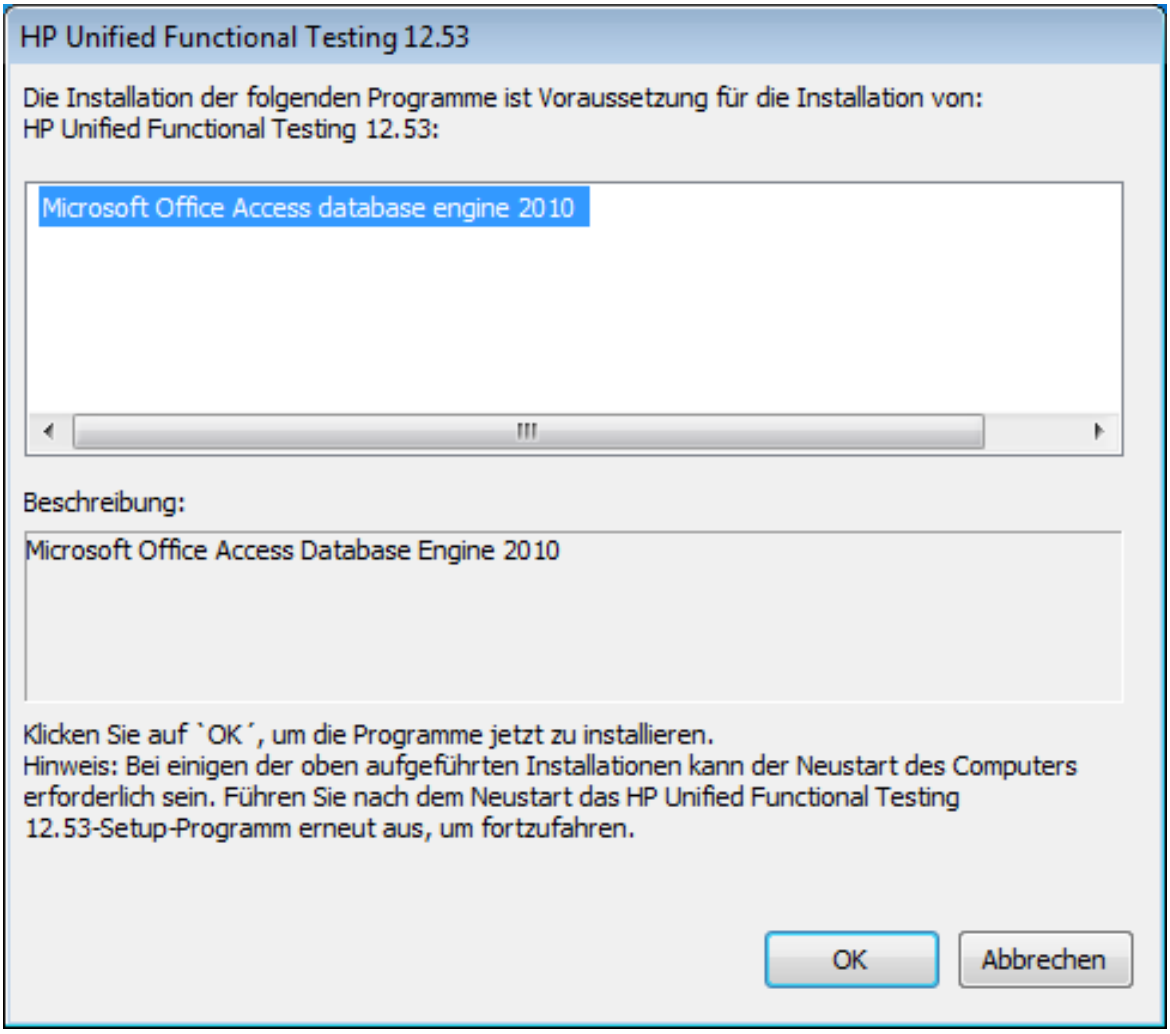

Abbildung B.22 Installation der HP UFT (2)

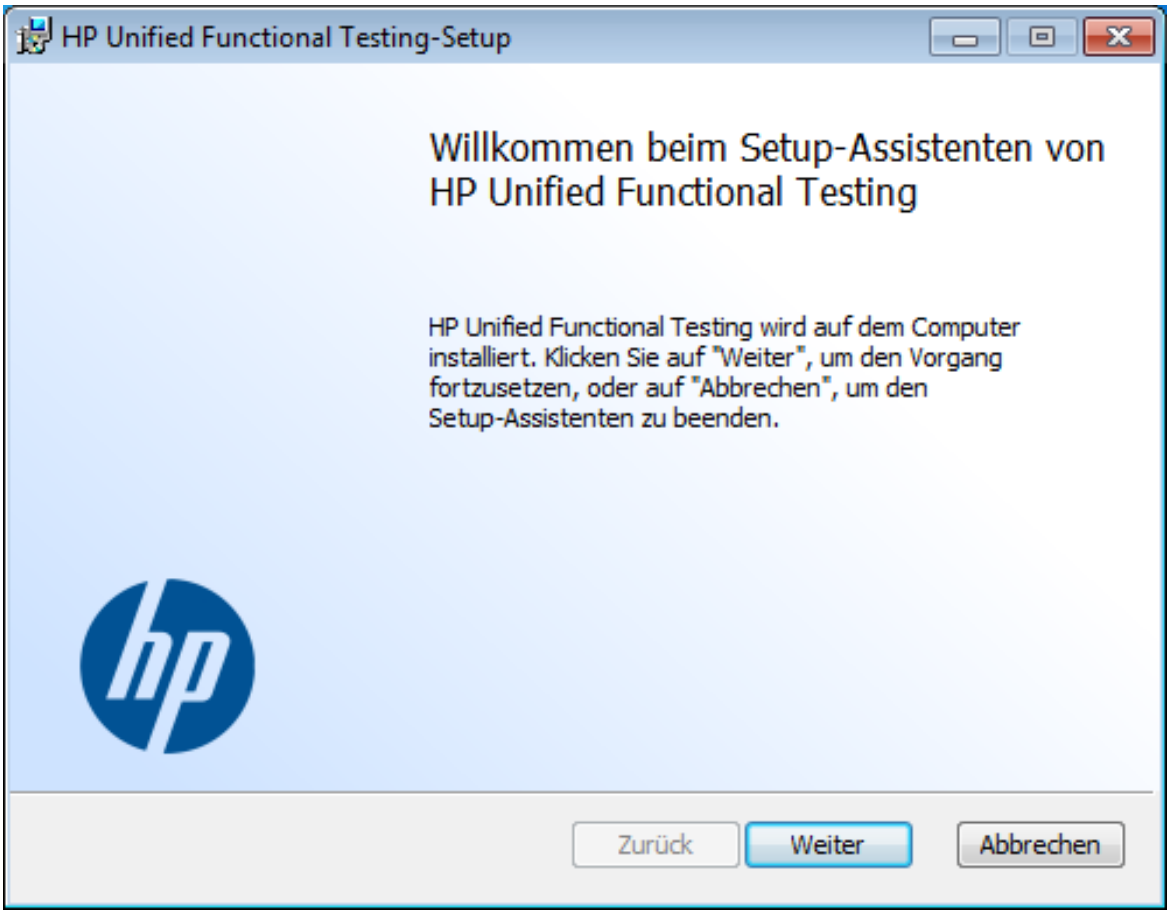

Abbildung B.23 Installation der HP UFT (3)

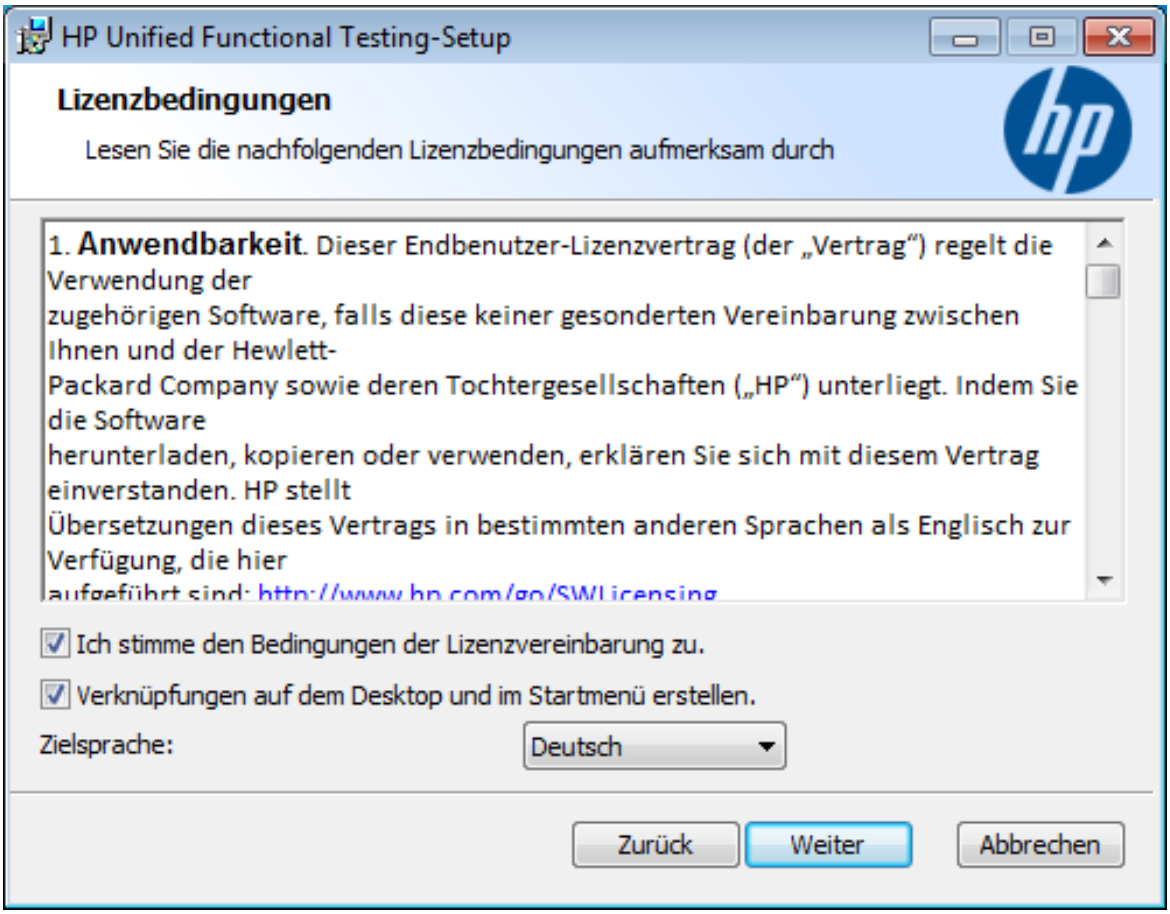

Abbildung B.24 Installation der HP UFT (4)

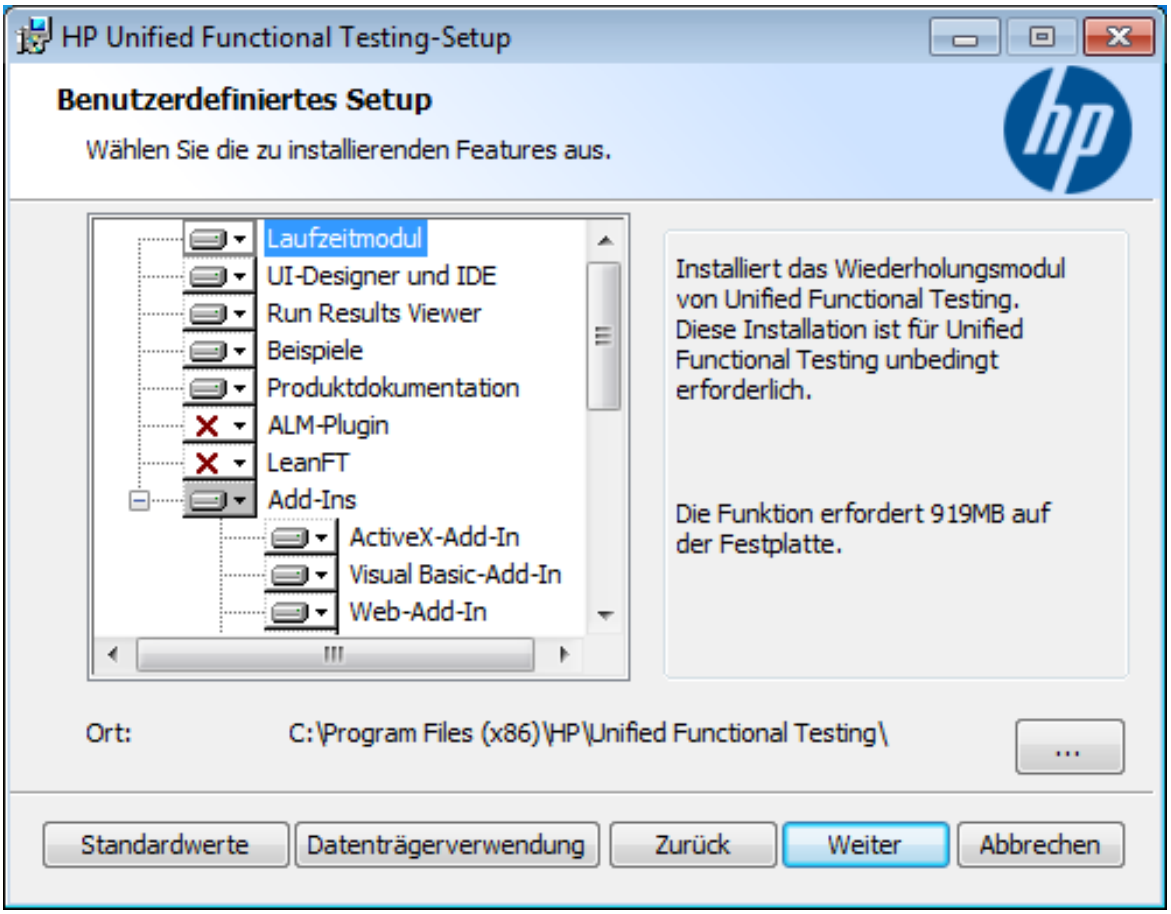

Abbildung B.25 Installation der HP UFT (5)

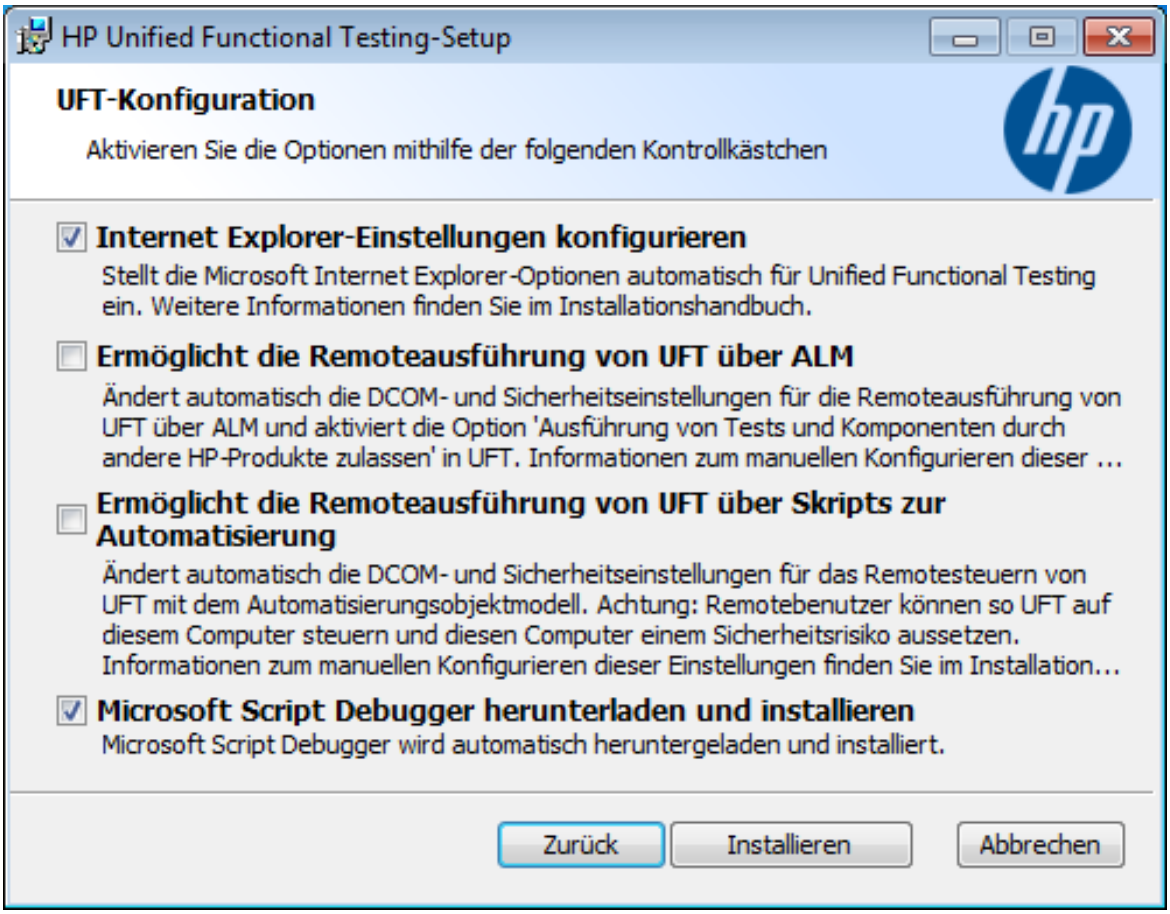

Abbildung B.26 Installation der HP UFT (6)

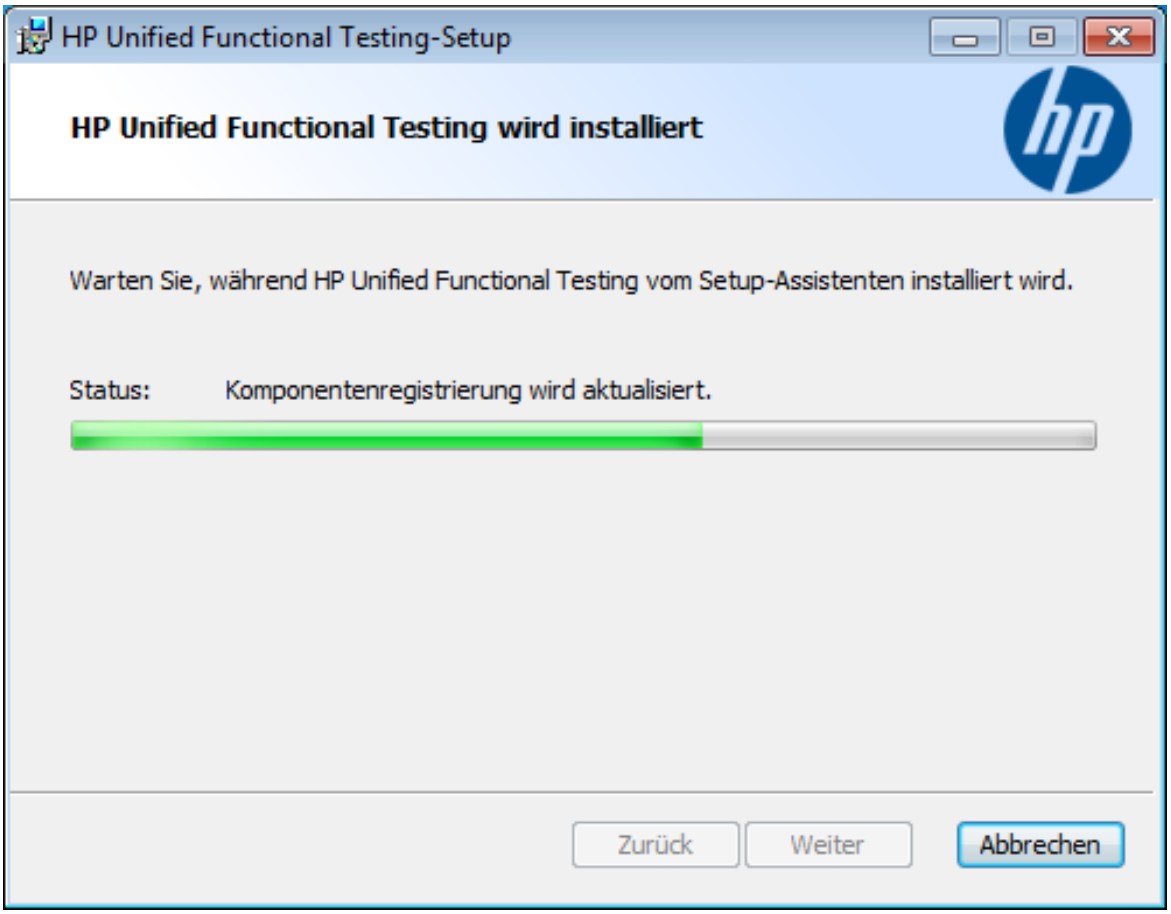

Abbildung B.27 Installation der HP UFT (7)

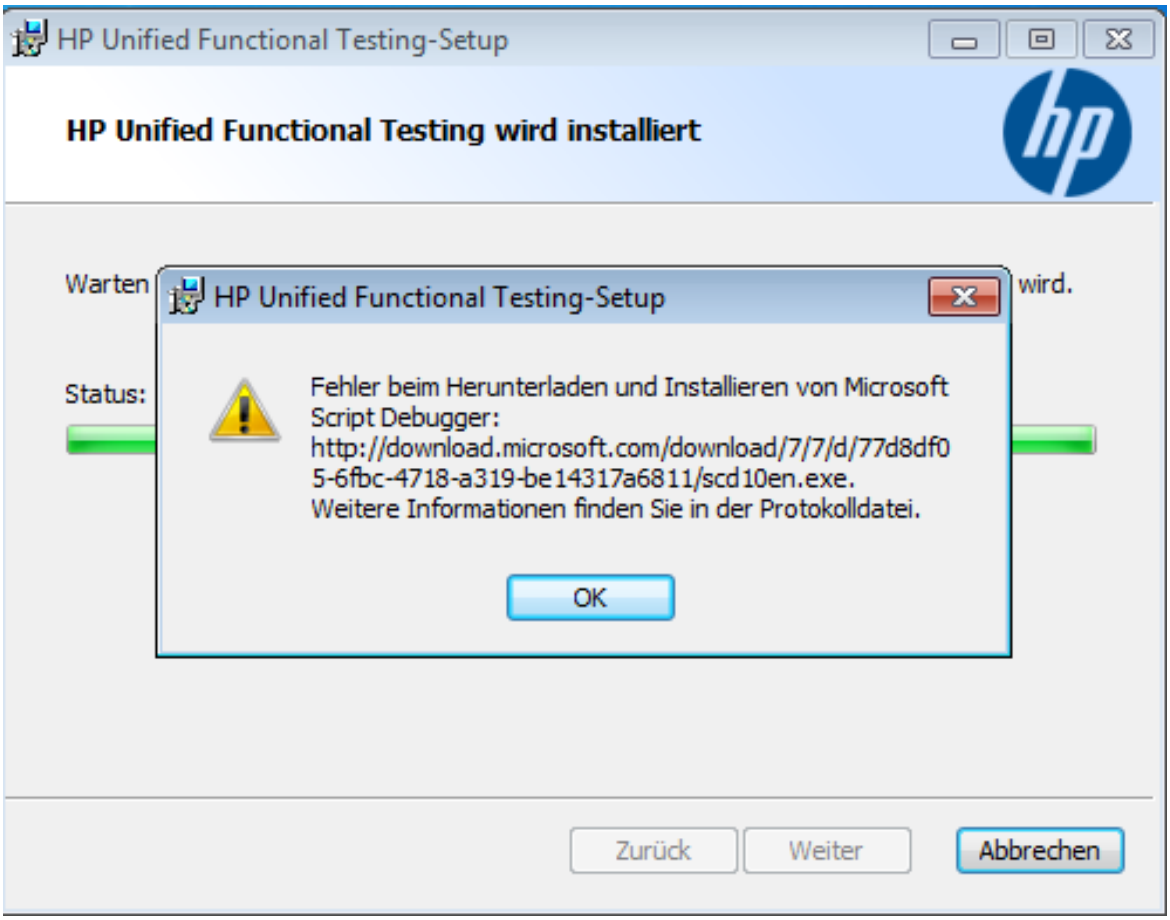

Abbildung B.28 Installation der HP UFT (8)

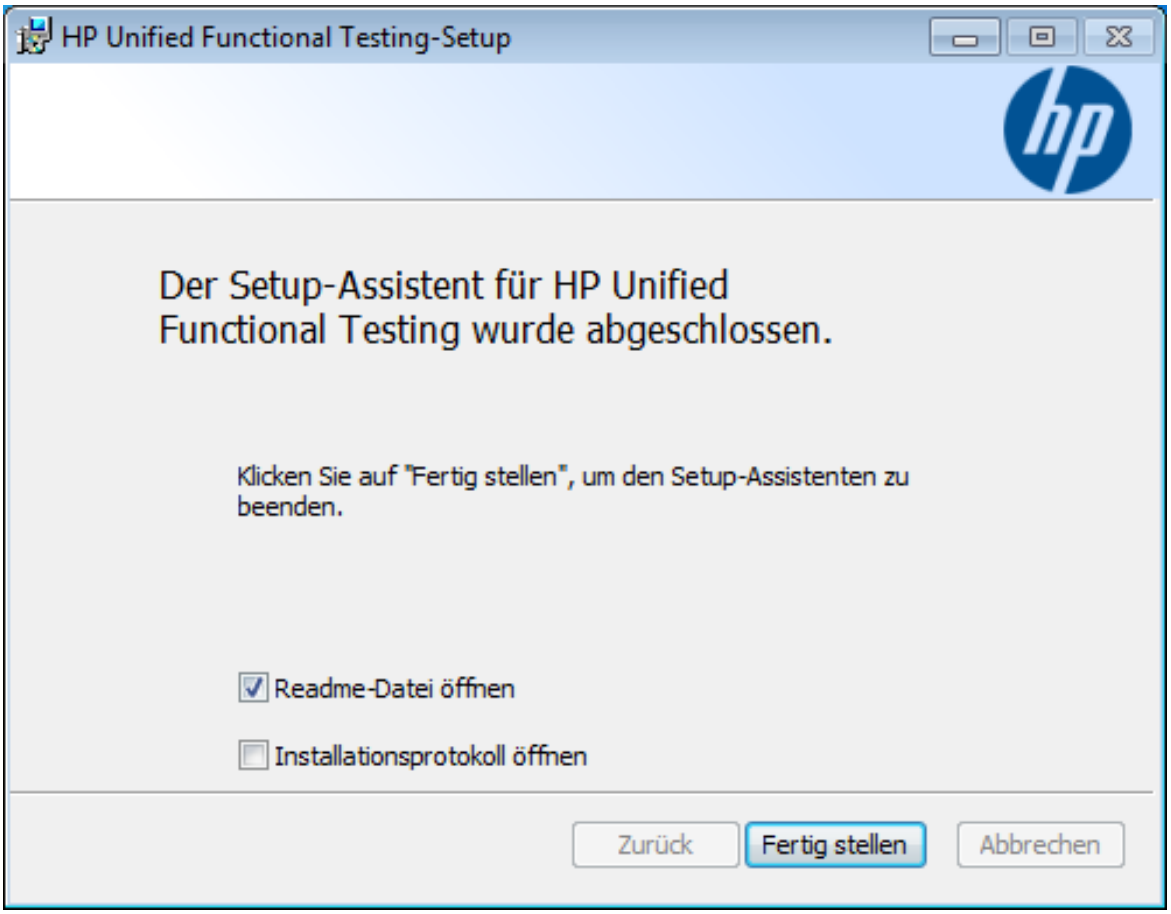

Abbildung B.29 Installation der HP UFT (9)

Anhang C

Kommerzielle Testsoftware

HP Unified Functional Testing (UFT)

Die Software vereint manuelle, automatisierte und Framework-basierte Tests in einer gemeinsamen Integrated Development Environment (IDE), alle Komponenten lassen sich dabei über eine gemeinsame Gui steuern und veranschaulichen. Getestet werden können sowohl Schnittstellen als auch Gui-Tests. Bereits existente manuelle Testfälle können in die Software importiert und automatisiert werden. Außerdem können einzelne Bestandteile existierender Testfälle per Drag & Drop zu neuen Testfällen zusammengestellt werden. Die Software ermöglicht eine Automatisierung von Testfällen für verschiedene Browser (Chrome, Firefox, Internet Explorer und Safari), Schnittstellen, mobile Plattformen (iOS, Android und Windows) und Enterprise-Resource-Planning-Anwendungen (zum Beispiel SAP). Um ein Testdurchlauf möglichst wirkbetriebsnah durchzuführen, ist UFT in der Lage, mehrere Instanzen der Testausführung zu starten, zum Beispiel auf mehreren Browsern oder sogar auf mehreren verteilten Systemen. Beim Testen einer Benutzeroberfläche (Gui) verwendet die Software eine Reihe von bildbasierten Text- und Objekterkennungen, die sogar in der Lage sein sollen, lernfähig neue Objekte in den Workflow einzubinden. Dank einer Schnittstelle zu HP Application Lifecycle Management (HP ALM) profitieren Testmanager von einer übersichtlichen Protokollierung der Testfälle. Zusätzlich verfügt die Software über einen erweiterten Expertenmodus, wodurch es möglich ist Testfälle in VBScript code abzubilden. Außerdem besteht die Möglichkeit, Tools zur kontinuierlichen Integration einzubinden.
basiert auf:
http://www8.hp.com/de/de/software-solutions/unified-functional-automated-testing/

VisualStudio Coded UI Tests

In VisualStudio Enterprise ist ein sogenannter CUIT-Test-Generator und -Editor enthalten, mit dessen Hilfe aus VisualStudio heraus ein automatischer Test der Benutzeroberfläche von in VisualStudio entwickelten Applikationen erfolgen kann. Getestet werden können sowohl Win32-GUIs als auch verschiedenste Browser durch ein Plugin (Internet Explorer inklusive Silverlight). Der CUIT-Test-Generator funktioniert dabei wie eine Art Record-&-Play-Tool, der Nutzer kann Interaktionen der Benutzeroberfläche aufzeichnen und in einer Art Testobjekt abspeichern. Der so erzeugte Programmcode kann beliebig editiert und durch die Play-Funktion ausgeführt werden, so dass die zuvor im Code festgelegten Interaktionen wiedergegeben werden. Durch eine Anbindung weiterer Tools aus dem Hause Microsoft erhält der Benutzer ebenfalls eine gute Auswertung der Testergebnisse zur optimalen Weiternutzung im Testmanagement.
basiert auf: https://msdn.microsoft.com/de-de/library/dd286726.aspx
https://visualstudiogallery.msdn.microsoft.com/11cfc881-f8c9-4f96-b303-a2780156628d/

Ranorex Automation Framework bzw. Studio

Laut Ranorex unterstützt die Software alle gängigen Arten von Software-Anwendungen auf unterschiedlichsten Umgebungen. Unterstützt werden dabei sowohl die verschiedensten Browser (Internet Explorer, Mozilla Firefox, Google Chrome, Safari) als auch Desktop und mobile Applikationen, welche auf den Technologien .NET, WPF, Win32-GUIs, Qt, Java, SAP, Delphi, HTML5, Flash, Flex, Silverlight, iOS und Android aufbauen. Der Hersteller garantiert außerdem eine nahtlose Integration in bestehende Umgebungen, um damit zum Beispiel Continuous-Integration-Prozesse und Testmanagement-Tools weiterhin nutzen zu können. Zur Erstellung von Testabläufen für diese Software bietet Ranorex den sogenannten Ranorex Test Recorder, mit ihm lassen sich Testschritte, Werte und Eigenschaften von Elementen einer Benutzeroberfläche während des manuellen Testens aufzeichnen. Des Weiteren gibt es eine Schnittstelle mit deren Hilfe in den Programmiersprachen VB.NET oder C# nachträglich Testfälle bearbeitet werden können. Selbstverständlich ist auch eine direkte Bearbeitung im Ranorex Studio möglich, hierbei kann der Nutzer mittels Drag & Drop Test-Aufzeichnungen bearbeiten. Außerdem ist es möglich, alte Testfälle, zum Beispiel Exeldateien, in den Testprozess von Ranorex Studio zu integrieren und sogar in Bestandteile zu zerlegen, um damit neue oder erweiterte Tests durchführen zu können.
basiert auf:
http://www.ranorex.de/wie-funktioniert-testautomatisierung.html
http://www.ranorex.com/support/user-guide-20.html Buch

Micro Focus SilkTest

Der Micro Focus SilkTest, neuerdings Borland SilkTest, ermöglicht es, Funktionalitäts- und Regressionstests zu automatisieren. Dabei unterstützt die Software verschiedene Technologien zum Testen einer Benutzeroberfläche auf Basis von Adobe AIR, Adobe Flex, Microsoft Silverlight (auch in Runtime-Umgebung), Java, Win32-Guis oder .Net GUIs. Des Weiteren werden durch die Software eine Reihe von Browser-Applikationen wie Google Chrome, Internet Explorer, Mozilla Firefox, Safari inklusive der neusten Browsertechnologien (Ajax, Xml, Html5 etc.) und ERP-Anwendungen wie zum Beispiel SAP unterstützt. Ein besonderes Merkmal der Software ist das rollenbasierte Testen, dabei können Tester und Entwickler gezielter einen Testfall aus verschiedenen Blickwinkeln bearbeiten. Es gibt die eigententwickelte Scriptsprache „4Test“ um Benutzern mit Programmiererfahrung die Möglichkeit zu bieten, Testklassen und -objekte zu verwalten. Andernfalls bietet die Software einen Modus zur Erkennung von Objekten, so lassen sich mittels der Record-Funktion Applikationen als Objektstruktur aufzeichnen. Das heißt, dass Silktest Fenster und andere Elemente einer Applikation erfasst und in ein Testscript abspeichert. Mittels der Run-Funktion kann dann ein Testscript automatisch abgearbeitet werden.
basiert auf:
http://www.borland.com/de-DE/Products/Software-Testing/Automated-Testing/Silk-Test

QFTest

Der QFTest wurde entwickelt, um Nutzeroberflächen aller möglichen in Java programmierten Applikationen automatisch zu testen, außerdem unterstüzt die Software Web-Applikationen unter Linux und Windows innerhalb der Browser Internet Explorer, Mozilla Firefox und Chrome jeweils inklusive Javascript. Zum Betrieb der Software benötigt der Benutzer keine Programmierkenntnisse, es besteht aber die Möglichkeit, über eine Schnittstelle selbstständig Erweiterungen zu programmieren oder andere Continuous Integration-Tools und auch Reporting-Tools zu integrieren. Für den einfachen Benutzer erfolgt die Erstellung der Testfälle dabei mit einem Aufzeichnungstool, welches die Testfälle anhand von manuellen Testdurchläufen aufzeichnet und bearbeitbar innerhalb einer Baumstruktur abspeichert. Durch die Wiedergabe-Funktionalität kann der so aufgezeichnete Testfall beliebig oft wiederholt werden.
basiert auf:
https://www.qfs.de/qf-test/testautomatisierung-mit-qf-test.html

Telerik Test Studio

Das Telerik Test Studio, neuerdings Test Studio by Progress, ermöglicht den automatischen Test verschiedener Technologien von Web, Desktop und mobilen Applikationen auf unterschiedlichen Umgebungen. Unterstützt werden dabei sowohl iOS und Android-Apps als auch die Browser Microsoft Internet Explorer, Mozilla Firefox, Google Chrome und Safari inklusive HTML5, Javascript, Silverlight und WPF. Es besteht die Möglichkeit der Integration von Continuous Integration-Tools und dem Testen von eigenen Schnittstellen via HTTP-Request. Testfälle können dabei auf der einen Seite durch Programmcode in C# und VB.NET in das Programm integriert werden oder auf der anderen Seite mittels des sogenannten Point-and-Click Test Recorder auf Basis von manuellen Testfällen übernommen werden. Des Weiteren bietet das Test Studio Möglichkeiten, Testfälle innerhalb einer Baumstruktur neu zu verknüpfen und zu bearbeiten wie zum Beispiel das Setzen von „intelligenten“ Werten mittels einer Art regulärem Ausdruck. Die Software ermöglicht auch die Ausführung mehrerer Test-Instanzen auf unterschiedlichen Systemen und eine umfangreiche Auswertung der Tests bzw. Analysewerkzeuge für das Testmanagement.
basiert auf:
http://www.telerik.com/teststudio

Die folgenden Applikationen wurden aus Zeitgründen nicht behandelt:

IBM Rational Functional Tester

eggPlant Functional

Tricentis TOSCA TestSuitec

expecco

TestComplete

RapidRep

TTworkbench

Parasoft Software Testing Tools

Certify