Fraunhofer Institut
Experimentelles
Software Engineering

Research Lab Rheinland-Pfalz
Testen und Testautomatisierung: Anforderungen an Testwerkzeuge und Marktstudie

Autoren:
Samir Amiry
Ove Armbrust
Julia Berger
Janine Klinck
Konstantin Luttenberger

Infrastruktur für eine Forschungs- und Transferplattform am Fraunhofer IESE in Kaiserslautern für regionale, software-entwickelnde KMUs (KMU Forschungslaborplattform)

Gefördert mit Mitteln des Europäischen Fonds für Regionalentwicklung (EFRE) und des Landes Rheinland-Pfalz (Ministerium für Wirtschaft, Verkehr, Landwirtschaft und Weinbau Rheinland-Pfalz: MWVLW RLP).

Förderkennzeichen: MWVLW;
Az.: 8315 38 51 04 IESE;
Kapitel 0877 Titel 892 02.

IESE-Report Nr. 131.05/D
Version 1.0
Dezember 2005

Eine Publikation des Fraunhofer IESE

Das Fraunhofer IESE ist ein Institut der Fraunhofer-Gesellschaft.
Das Institut transferiert innovative Software-Entwicklungstechniken, -Methoden und -Werkzeuge in die industrielle Praxis. Es hilft Unternehmen, bedarfsgerechte Software-Kompetenzen aufzubauen und eine wettbewerbsfähige Marktposition zu erlangen.

Das Fraunhofer IESE steht unter der Leitung von
Prof. Dr. Dieter Rombach (geschäftsführend)
Prof. Dr. Peter Liggesmeyer
Fraunhofer-Platz 1
67663 Kaiserslautern

Abstract

Software-Systeme werden immer komplexer. Durch diese Komplexitätssteigerung wird auch die Durchführung von Software-Tests zu einer größeren Herausforderung. Um die Zeit bis ein Produkt auf den Markt kommt (Time-to-Market) zu verkürzen, werden immer effizientere Automatisierungstechniken gefordert. Ein Ansatz hierfür ist der Einsatz von automatisierenden Testwerkzeugen. In der Tat existieren eine Vielzahl von Automatisierungsansätze und Implementierungen in Wissenschaft und Industrie. Aufgrund der Tatsache, dass Software-Firmen unterschiedliche Anforderungen bzgl. des Testens haben, wird die Auswahl eines Werkzeuges, das die verschiedenen automatischen Testnotwendigkeiten unterstützt, schwierig.

In diesem Bericht werden typische Anforderungen, die ein Testautomatisierungswerkzeug erfüllen sollte, identifiziert. Gleichzeitig werden im Rahmen einer Fallstudie spezifische Anforderungen der Projektpartner WIKON und market maker festgehalten. In einem zweiten Teil präsentiert dieser Bericht die Ergebnisse einer Marktstudie von Testautomatisierungstools, die dann genutzt werden können, um die Akquise von solchen Tools zu unterstützen. Die Ergebnisse der Marktstudie werden sodann benutzt, um Werkzeuge für die Projektpartner auszuwählen.

Danksagung

Wir danken den folgenden Unternehmen und Personen für die Bereitschaft zu Interviews:

- market maker: Sebastian Scheffler
- WIKON: Clemens Schitter

Schlagworte

Software testing, automated testing, testing tools, acquisition

Inhaltsverzeichnis

1 Einleitung

Software-Produzenten sind daran interessiert, ihre Produkte schnell und effizient zu testen. Um diese Aufgabe zu lösen, sollten die Software-Unternehmen durch Werkzeuge zur Testautomatisierung unterstützt werden, denn bei der Testdurchführung müssen oft sehr viele Testfälle betrachtet werden. Daher ist eine manuelle Durchführung der Testfälle schon aus Zeit- und damit verbundenen Kostengründen oft gar nicht möglich. Spezialisierte Werkzeuge können einige Arbeitsschritte ohne menschliches Zutun durchführen und damit Aufwand einsparen. Hauptsächlich sind dies momentan mit der Testausführung beschäftigte Schritte, seltener auch die Testfallerstellung oder Testdatengenerierung.

1.1 Projektkontext

Dieser Bericht ist ein Produkt des Projektes „Research Lab Rheinland-Pfalz: Testen und Testautomatisierung“. Wie schon aus dem Namen ersichtlich, behandelt das Projekt unterschiedliche Bereiche der Testautomatisierung, wobei der Fokus auf dem Einsatz entsprechender Konzepte in kleinen und mittelständischen Unternehmen (KMU) liegt.

Das Projekt ist in acht verschiedene Arbeitspakete (AP) unterteilt. Im ersten Arbeitspaket werden Grundlagen bzgl. Testen und Testautomatisierung geschildert, insbesondere werden dabei der aktuelle Stand der Forschung und der Stand der Praxis untersucht. Der Ist-Zustand der Partner bzgl. Testautomatisierung wird im AP2 dargestellt. Ziele und Kennzahlen werden in AP3 behandelt. Die Ergebnisse der Arbeitspakete vier und fünf werden in diesem Bericht beschrieben. AP4 beschreibt dabei Anforderungen an Automatisierungstools, die teilweise zusammen mit den Partnern market maker und WIKON aufgestellt wurden. Eine Marktanalyse der bereits auf dem Markt vorhandenen Testautomatisierungstools findet im AP5 statt, ebenso die Auswahl passender Tools für die Projektpartner WIKON und market maker.

AP6 beschäftigt sich anschließend mit der Anpassung bzw. Implementierung der Testtools. In AP7 werden die Integration der Tools in die Unternehmensinfrastruktur und die Schulung der Mitarbeiter der Projektpartner bzgl. der Testtools vorgenommen. Im letzten Arbeitspaket (AP8) werden schließlich die Ergebnisse evaluiert.

Dieser Bericht zeigt einen verständlichen und nachvollziehbaren Weg auf, Testtools basierend auf spezifischen Anforderungen auszuwählen. Das dabei entwickelte Verfahren wurde angewandt, um die Aufgaben der Arbeitspakete vier und fünf durchzuführen. Die Ergebnisse sind ebenfalls Teil dieses Reports.

1.2 Vorüberlegungen

Software-Tests spielen im täglichen Leben eines Entwicklers eine große Rolle. In der Praxis finden große Teile der Qualitätssicherung hauptsächlich in Form von Tests statt, die deshalb mitunter sehr aufwendig werden können.

Um den Aufwand für Tests zu senken, können Teile automatisiert werden. Hierbei wird die Software nicht manuell in einer bestimmten Reihenfolge mit bestimmten Eingabedaten konfrontiert und ihre Reaktion überprüft, sondern (teil-)automatisch mittels Werkzeugen. Die Forschung hält hierfür einige Ansätze bereit, um auch kompliziertere Fälle automatisch handhaben zu können [AOS04], in der industriellen Praxis haben sich bisher jedoch nur weit weniger aufwendige Verfahren etablieren können, sofern überhaupt automatisiert getestet wird [AOS04-2].

Ein Werkzeug alleine sorgt jedoch nicht für die Automatisierung der Tests. Zuerst einmal kann nur automatisiert getestet werden, was überhaupt getestet werden kann. Dies mag trivial erscheinen, ist jedoch in der Praxis der Software-Entwicklung eine echte Herausforderung. Viele Systeme sind über Jahre hinweg gewachsen und stark verflochten, so dass einzelne Tests für die kleinsten Einheiten des Systems (Units) zwar noch möglich sind, schrittweise Integrationstests, bei denen einzelne Units zu größeren Einheiten zusammengesetzt werden, jedoch oft wegfallen, da es unmöglich ist, Komponenten unabhängig voneinander zu testen. Häufig schließt sich daher an die Unit-Testphase direkt der Systemtest an, der dann zwangsläufig Fehler findet, die in Komponententests gefunden worden wären.

Damit ergibt sich die Notwendigkeit, Systeme so zu gestalten und zu entwickeln, dass sie überhaupt erst testbar sind. Bei Neuentwicklungen ist dies von Anfang an möglich und auch ausreichend durch kommerzielle Werkzeuge unterstützt. Gewachsene Systeme müssen schrittweise in einen besser testbaren Zustand überführt werden. Für die gewachsenen Systeme kommt meist eine Sammlung (teilweise) selbst entwickelter Testwerkzeuge zum Einsatz, die auf die Systemstruktur angepasst sind. Vor diesem Hintergrund sind auch viele Anforderungen an Automatisierungstools zu sehen.

1.3 Aufbau des Reports

Der vorliegende Bericht ist wie folgt gegliedert. In Kapitel 2 werden Konzepte und Begrifflichkeiten des Software-Testens eingeführt. Dies beinhaltet die Grundlagen des Software-Testens anhand allgemeiner Informationen zum Testen (Kapitel 2.1), der Testaktivitäten (Kapitel 2.2), Testphasen (Kapitel 2.3) und eine Einordnung von Testtechniken (Kapitel 2.4).

Kapitel 3 beschreibt das Bewertungsverfahren für ein Testtool anhand eines Beispiels. In Kapitel 4 wird das Arbeitspaket 4 bearbeitet. Hierbei werden allgemeine (Kapitel 4.1) und spezifische Anforderungen (Kapitel 4.2) von Unternehmensseite für die Auswahl eines Testtools erarbeitet. Anschließend

wird der Prozess von der Definition der Anforderungen bis zur Auswahl eines geeigneten Automatisierungstools beispielhaft für die Projektpartner market maker und WIKON durchgeführt.

Die Definition der spezifischen Anforderungen der Projektpartner findet sich in Kapitel 5. Kapitel 6 behandelt anschließend das fünfte Arbeitspaket des Projekts, in dem der Markt existierender Testautomatisierungstools analysiert wird. Dabei werden die Bewertungskriterien einzelner Testtools erklärt (Kapitel 6.1), die Bewertung der Automatisierungstools (Kapitel 6.2) vorgenommen und jedes einzelne Tool detailliert beschrieben (Kapitel 6.3).

In Kapitel 7 wird anschließend eine Auswahl von Testtools für die Projektpartner WIKON (Kapitel 7.1) und market maker (Kapitel 7.2) anhand ihrer Anforderungen durchgeführt. Kapitel 8 bildet den Abschluss in Form einer Zusammenfassung.

2 Software-Testen

In diesem Kapitel werden die Grundlagen des Software-Testens vorgestellt. Dazu werden u.a. die wichtigsten Begriffe der verschiedenen Testaktivitäten, Testtechniken und Testphasen, die im weiteren Bericht Verwendung finden, eingeführt und kurz erläutert.

2.1 Testen und Testautomatisierung

Heutige Software-Systeme sind wegen ihrer zunehmenden Größe und Komplexität nicht nur schwerer testbar, sondern erfordern auch eine höhere Anzahl an Testfällen als früher, um möglichst viele Fehler entdecken zu können. Tests benötigen ebenso wie die Entwicklung der Software selbst einen strukturierten Ablauf, der es erlaubt, Tests sicher und nachvollziehbar durchzuführen. Ein solcher Testprozess ist nicht nur für die Entwicklung von Ein-Rechner-Systemen sinnvoll, sondern gewinnt wegen weiter Verbreitung verteilter Systeme, wie etwa Client/Server-Systeme, auch für diese eine immer höhere Bedeutung. Zusätzlich können viele Unternehmen ihre Testaufgaben meist nicht mehr manuell bewältigen, so dass die Automatisierung dieser Aufgaben ebenfalls weiter in den Vordergrund rückt.

2.1.1 Terminologie

Generell kann unterschieden werden zwischen Fehlverhalten (failure), Fehlern (fault) und Fehlerursachen (error). Unter einer Fehlerursache versteht man dabei eine menschliche Handlung, aufgrund derer ein Fehler entsteht. Ein Beispiel für eine Fehlerursache wäre ein Denkfehler eines Programmierers, der zu einem Fehler im Programm in Form einer falschen Anweisung/Codezeile führt. Dieser Fehler ist von außen nicht sichtbar, selbst mit diesem Fehler kann sich das Programm noch korrekt verhalten, etwa falls die fehlerhafte Zeile nicht zur Ausführung kommt. Wird sie ausgeführt, so resultiert daraus ein Fehlverhalten des Programms, welches von außen beobachtet werden kann.

Hieraus folgt, dass Fehler nicht zwangsläufig zu Fehlverhalten und damit zu zusätzlichen Kosten führen müssen. Insbesondere weist ein aufgetretenes Fehlverhalten immer auf die Anwesenheit von Fehlern hin, die Anwesenheit von Fehlern jedoch führt nicht zwingend zu Fehlverhalten. Das hat ganz praktische Auswirkungen für den Test: Wenn im Test kein Fehlverhalten der Software beobachtet wird, dann können trotzdem Fehler enthalten sein, die einfach kein beobachtbares Fehlverhalten der Software provoziert haben.

Im weiteren Verlauf dieses Reports wird der Begriff „Fehler“ synonym für die hier aufgeführten Begriffe „Fehler“ und „Fehlverhalten“ benutzt, da eine weitere Unterscheidung in diesem Kontext nicht notwendig ist.

2.1.2 Testautomatisierung

Unter Automatisierung versteht man im Zusammenhang mit Software-Tests, dass einige oder alle der Aktivitäten eines Testprozesses nicht mehr von Menschen ausgeführt werden, sondern ohne menschliches Zutun von Maschinen. Sind an bestimmten Stellen Eingriffe von Menschen notwendig, so spricht man von einer Teil-Testautomatisierung.

Sollen Tests manuell durchgeführt werden, so ist eine möglichst kleine Anzahl von Testfällen anzustreben, wohingegen die erwünschte hohe Abdeckung möglicher Fehlerfälle oft eine große Anzahl von Testfällen oder zumindest komplexe Testfälle provoziert. Die Forderung nach wenigen, einfachen Testfällen lässt sich durch Testautomatisierung abschwächen. Dies begründet sich dadurch, dass

a) Maschinenzeit meist deutlich kostengünstiger ist als menschliche Arbeitszeit und

b) Tests ohne erforderliches menschliches Zutun deutlich schneller ablaufen können als manuelle oder teilautomatisierte Tests.

2.2 Testaktivitäten

Der gesamte Testprozess [@SWKompetenz] kann in mehrere Aktivitäten aufgeteilt werden. Dieser Report unterscheidet vier Hauptaktivitäten:

- Testplanung: Die erste Aktivität im Testprozess ist die Testplanung. Dabei soll eine detaillierte Planung in Form eines Testplans erfolgen. Im Testplan finden sich Beschreibungen der Ressourcenplanung, der Teststrategie, der Testpriorisierung sowie der geplanten Werkzeugunterstützung. Die Ressourcenplanung umfasst dabei den Aufwand an Mitarbeitern samt von ihnen voraussichtlich benötigter Arbeitszeit, eingesetzter Werkzeuge und weiterer Hilfsmittel. Hierbei wird zusätzlich ein Zeitplan für den Einsatz der Projektarbeiter festgehalten. Die Teststrategie legt fest, welche Teile des Systems in welcher Intensität getestet werden sollen, da ein ganzheitlicher Test meist aus Zeitgründen nicht durchzuführen ist. Um die Auswahl und Testintensität möglichst sinnvoll zu gestalten, wird im Allgemeinen eine Risikoabschätzung durchgeführt. Die Testpriorisierung hat zum Ziel, möglichst kritische Systemteile auch zuerst zu testen, da es aus Zeitdruck sein kann, dass nicht alle festgelegten Tests durchgeführt werden können. Die geplante Werkzeugunterstützung enthält die Analyse der Einsetzbarkeit vorhandener Tools sowie evtl. nötiger Neuanschaffungen.

- Testfallerstellung: Testfälle können auf unterschiedliche Weise erstellt werden. Testfälle können etwa ad-hoc definiert werden. Das bedeutet, die Tester oder Entwickler entscheiden eher spontan darüber einen bestimmten Bereich zu testen. Daraufhin wird ohne weitere Planung getestet. Eine andere Möglichkeit ist die Erstellung der Testfälle aus der Systemspezifikation. Dabei werden anhand der spezifizierten funktionalen und nicht-funktionalen Eigenschaften des zu entwickelnden Systems Testfälle abgeleitet.

- Testdurchführung: Nach der Erstellung des Testplans und der Testfälle kann mit der Durchführung der Tests begonnen werden. Voraussetzung dafür ist natürlich, dass die zu testenden Systemteile verfügbar, d.h. implementiert und an den Tester übergeben worden sind und eine Testumgebung vorhanden ist. Anschließend werden die Testfälle der im Testplan festgelegten Strategie folgend abgearbeitet.

- Testauswertung: Diese Aktivität beschreibt die Auswertung der Tests. Die Ergebnisse und eventuelle Fehler werden in Tabellen, Fehlerausgaben oder textuellen Dokumenten an die zuständige Person weitergeleitet. Wichtig bei der Testauswertung ist die exakte Angabe und Beschreibung des Fehlers, da nur sie es ermöglicht den Fehler im Programmcode zu finden bzw. nachzuweisen, dass der Fehler durch einen anderen Faktor verursacht wurde. Solche Faktoren sind beispielsweise eine ungenaue oder fehlerhafte Testspezifikation, eine fehlerhafte Testinfrastruktur oder auch fehlerhafte Testfälle.

Eng mit Tests verbunden ist immer die Aktivität des Fehlermanagements. Beim Fehlermanagement beschäftigt man sich mit der Nachverfolgung von identifizierten Programmfehlern. Dies beinhaltet das Aufzeichnen und Nachprüfen sowie der Reproduzierbarkeit der Fehler, das Aufzeichnen der benötigten Korrekturmaßnahmen und beispielsweise die Entscheidung, ob und wann der Fehler korrigiert wird. Darüber hinaus müssen einzelne Tests häufig nach Behebung des Fehlers erneut durchgeführt werden, da die Korrektur des Fehlers neue Fehler in das System eingebracht haben kann. Ebenso müssen evtl. neue Testfälle eingeführt oder existierende angepasst werden, wenn es sich um tief greifende Änderungen gehandelt hat. Ein systematisches Fehlermanagement ermöglicht es nachzuvollziehen, was mit den identifizierten Fehlern geschah, warum und wie das System geändert wurde.

2.3 Testphasen

Software wird üblicherweise in mehreren Phasen getestet. Begonnen wird bei sehr feingranularen Testobjekten, anschließend werden schrittweise die getesteten Objekte vergrößert, bis letztendlich das gesamte System und seine Akzeptanz durch die Nutzer getestet werden. Die einzelnen Testphasen können nach der Granularität der Testobjekte geordnet werden:

- *Unit-Phase*: Eine Unit ist die kleinste Einheit der zu testenden Software, also beispielsweise eine einzelne Klasse. Häufig werden Unit-Tests direkt von den Entwicklern durchgeführt, so dass das dafür erforderliche Wissen über die Interna der Einheiten vorhanden ist.

- *Integrationsphase*: Das korrekte Funktionieren jeder einzelnen Unit garantiert nicht, dass die Units auch im Zusammenspiel wie vorgesehen funktionieren. Dies wird mit Hilfe von Integrationstests überprüft. Hierzu werden mehrere Units zusammengesetzt und ihre Interaktion getestet. Während einem Integrationstest wird schrittweise das gesamte System zusammengesetzt, während gleichzeitig Tests ausgeführt werden, die beispielsweise Fehler in den Schnittstellen der schon erfolgreich einzeln getesteten Units aufdecken.

- *Systemphase*: Das Ziel eines Systemtests ist es, eine gewisse Sicherheit herzustellen in Bezug auf das spezifikationsgemäße Funktionieren des Gesamtsystems. Insofern stellt der Systemtest eine Vorbereitung auf den Akzeptanztest durch den Benutzer dar, da die vom Benutzer spezifizierte Leistung (sowohl funktional als auch nichtfunktional) im Ganzen getestet wird. Bei Tests auf Systemebene herrschen Black-Box-Techniken (siehe Kapitel 2.4) als Testart vor, da das Hauptaugenmerk auf der Erfüllung der Anforderungen liegt, nicht auf der Implementierung.

- *Installationsphase:* Der Begriff des Installationstests wird in der Literatur nicht einheitlich verwendet, insofern ist es nicht möglich, eine allgemeingültige Definition festzulegen. Allgemein können Installationstest auf zwei Arten verstanden werden [CBC01].

    1. Ein Installationstest kann als ein Test der Installationsumgebung verstanden, bevor bzw. während das System tatsächlich auf den entsprechenden Computer übertragen wird. Dabei steht die aktuelle Computerkonfiguration im Vordergrund. Dieser auch als „Configuration Checking“ bezeichnete Prozess prüft dabei u. a., ob benötigte Dateien und Bibliotheken, die nicht mit dem Produkt ausgeliefert werden, in der Installationsumgebung vorhanden sind. Außerdem wird dabei überprüft, ob Programme, mit denen das System kooperieren soll, auf dem Installationsrechner installiert wurden und ob eine gültige Hardware-Konfiguration vorliegt.

    2. Ein Installationstest kann aber auch als eine Funktionalitätsprüfung nach der Übertragung auf den Computer definiert werden. Ein so interpretierter Installationstest wird dabei häufig als Teil des Akzeptanztests verstanden, da er dem Endnutzer des Systems zeigt, dass die funktionalen Systeman-

forderungen auch in seiner konkreten Umgebung erfüllt werden.

Im Folgenden sollen unter Tests in der Installationsphase alle Tests verstanden werden, die mindestens einer der obigen Definitionen genügen.

- *Akzeptanzphase*: Bei diesen Tests überprüfen typischerweise Repräsentanten des endgültigen Benutzerkreises die Funktionsweise des Systems, d.h. sie führen am laufenden System Arbeiten aus, wie sie typisch für die spätere Benutzung sind. Dies stellt den letzten Test vor der formalen Auslieferung und Inbetriebnahme beim Kunden dar.

2.4 Einordnung von Testtechniken

Grundsätzlich unterscheidet man zwei Klassen von Testtechniken, deren Anwendung davon abhängt, welches Wissen man über die zu testende Software besitzt.

- Unter *Black-Box-Tests* [SL03] versteht man alle Testtechniken, die zur Herleitung oder Auswahl von Testfällen keine Informationen über die innere Struktur des Testobjekts benötigen.
- Unter *White-Box-Tests* [SL03] versteht man alle Testtechniken, die zur Herleitung oder Auswahl von Testfällen Informationen über die innere Struktur des Testobjekts benötigen.

Beim Black-Box-Testen werden üblicherweise die spezifizierten funktionalen Anforderungen an die zu testende Software herangezogen, um Testfälle abzuleiten. Jedoch ist anzumerken, dass es auch Black-Box-Techniken gibt, die nicht nach diesem Muster vorgehen [Ligg02]. Beim White-Box-Testen analysiert man die Kontrollstruktur des zu testenden Programms, um Testfälle abzuleiten, die bestimmte Kriterien der Abdeckung erfüllen. Hierbei gilt wiederum, dass dies nicht notwendigerweise so durchgeführt werden muss [Ligg02].

Die Abdeckungskriterien beim White-Box-Testen fokussieren auf verschiedene Eigenschaften der internen Struktur der zu testenden Software:

- *Anweisungsabdeckung:* Hierbei werden Testfälle so angelegt, dass jede Anweisung mindestens einmal ausgeführt wird. Dies führt zu mäßigem Aufwand, da analysiert werden muss, wie welche Entscheidungen im Programmablauf zu fällen sind, um alle Anweisungen zu erreichen.
- *Pfadabdeckung:* Hierbei werden Testfälle so angelegt, dass jeder Pfad, der im Kontrollfluss der Software möglich ist, mindestens einmal durchlaufen wird. Dies ist meist recht aufwendig, da oft viele Pfade in einem Programm existieren.

Beim Black-Box-Testen unterscheidet man ebenfalls mehrere Arten, wie die Anforderungen verarbeitet werden:

- *Äquivalenzklassen:* Die Unterteilung in Äquivalenzklassen geht von der Annahme aus, das sich die Inputs und Outputs eines Programms oder einer Methode in Klassen einteilen lassen, sodass sich das Programm für alle Elemente einer Klasse als Input gleichartig verhält. Daraufhin werden ein oder mehrere Repräsentanten der Klasse getestet und davon auf das Verhalten der ganzen Klasse geschlossen. Die Schwierigkeit liegt hierbei darin, die Klassen zu bestimmen und repräsentative Werte zu wählen, die getestet werden.

- *Grenzwertanalyse:* Die Grenzwertanalyse kann als ein Spezialfall der Äquivalenzklassentests gesehen werden. Die zugrunde liegende Annahme der Klassen ist dieselbe, anstatt jedoch Repräsentanten aus der Klasse zu wählen, konzentriert man sich bei der Grenzwertanalyse auf Werte direkt auf den Grenzen bzw. dicht auf beiden Seiten der Grenzen der einzelnen Klassen.

- *Zufallstesten:* Bei dieser Testmethode werden die Testdaten zufällig erzeugt und das zu testende Programm damit beschickt. Die Testergebnisse werden aufgezeichnet und können dann inspiziert werden.

- *Entscheidungstabellen:* Entscheidungstabellen werden häufig für die Erstellung von Testfällen eingesetzt [Ligg02]. Dabei werden Eingangswerte/Bedingungen mit ihren möglichen Belegungen gelistet, sowie die jeweils erwartete Ausgabe. Diese Testtechnik garantiert eine gewisse Testvollständigkeit, besitzt aber den Nachteil, dass der Umfang der Darstellung, die häufig in Form einer Tabelle vorliegt, exponentiell mit der Anzahl der Bedingungen steigt. Beispielsweise könnte die Möglichkeit eine Online-Bestellung per Rechnung zu bezahlen davon abhängen, ob es sich um einen Erstkunden, einen Privatkunden oder eine Bestellung mit einem Wert von über 1000€ handelt. Es gibt dabei 8 (2^3) mögliche Belegungen dieser drei Bedingungen, da jede Bedingung unabhängig von den anderen wahr oder falsch sein kann. Allgemein gibt es bei n Bedingungen 2^n mögliche Belegungen, was bei 10 Bedingungen schon zu 2^{10} = 1024 möglichen Belegungen führt, die alle einzeln überprüft werden müssen. Entscheidungstabellen können direkt aus der Spezifikation erstellt oder indirekt ermittelt werden, beispielsweise aus Ursache-Wirkungs-Graphen.

Beide Testarten Black-Box und White-Box schließen sich nicht gegenseitig aus, sie sind als komplementär zu verstehen und ergänzen sich, da sie unterschiedliche Arten von Fehlern aufdecken.

Eine von der eben beschriebenen Klassifikation relativ unabhängige Menge von Testtechniken ist das so genannte Built-in oder Self Testing (BIT). Das

Konzept bezieht sich auf den Test von Komponenten bzw. Units. Solche Systemkomponenten werden bei BIT mit Testfunktionalität ausgestattet, so dass sie sich in zwei Modi befinden können. Entweder laufen die Komponenten im Laufzeitmodus, in dem sie kein spezielles Verhalten zeigen, oder befinden sich im Testmodus, in dem die Funktionalität der Komponente getestet werden kann. Built-in-Tests werden oft bei der Integration mehrerer großer Komponenten eingesetzt, da es sich gerade in großen Systemen schwierig gestaltet, das korrekte Funktionieren der Komponenten in ihrem Zusammenspiel zu überprüfen. Grundsätzlich kann man zwei Arten von BIT unterscheiden [Comp+01]:

- *Built-in Contract Testing:* Diese Art des BIT bezeichnet die Möglichkeit einer Komponente, ihre korrekte Installation in die Laufzeitumgebung zu überprüfen. Der Begriff Laufzeitumgebung schließt die zugrunde liegende Plattform als auch das Zusammenspiel mit anderen Komponenten ein.
- *Built-in Quality of Service Testing:* Diese Art des BIT bezieht sich auf die Ausstattung einer Systemkomponente mit Testfunktionalität, um die Qualität ihrer Aufgabenerfüllung testen zu können. Ein Beispiel für einen derartigen Qualitätstest wäre etwa das Senden einer Nachricht innerhalb eines vorgegebenen Zeitintervalls.

3 Nutzwertanalyse

Als Grundlage zur Auswahl eines passenden Tools wird eine Nutzwertanalyse [@UniFlensburg] herangezogen. Diese wird zunächst anhand eines kleinen Beispiels beschrieben.

Steht eine Entscheidung an, so sollten zunächst alle wichtigen Kriterien, die für diese benötigt werden aufgelistet werden. So können z.B. beim Kauf einer bestimmten Software bestimmte Eigenschaften als Kriterien herangezogen werden. Dazu gehören in diesem Beispiel: Erweiterbarkeit, Performance, Benutzerfreundlichkeit, Wartbarkeit, Verfügbarkeit und Funktionalität. Diese werden zur Beschreibung der Nutzwertanalyse genügen.

Nachdem die relevanten Kriterien ausgewählt sind, wird jedes Kriterium nach Wichtigkeit der Erfüllung bewertet. Die entsprechende Bewertungsskala soll im Folgenden von 0…2 reichen. Hierbei bedeutet 0 unwichtig, 1 teilweise wichtig und 2 wichtig.

Anschließend werden in Frage kommende Tools bzgl. ihres Erfüllungsgrades der zuvor definierten Kriterien betrachtet. Hier wird ebenfalls eine Skala von 0…2 angesetzt. Hier bedeutet 0 nicht erfüllt, 1 teilweise erfüllt und 2 vollständig erfüllt.

Ist sowohl die Gewichtung der Kriterien als auch die Bewertung der Produkte anhand der Kriterien erfolgt, kann der Nutzwert berechnet werden. Dazu wird zunächst jeweils die Wichtigkeit eines Kriteriums mit seinem Erfüllungsgrad multipliziert und die Ergebnisse aufsummiert. Das Ergebnis bildet den realen Wert eines Werkzeugs ab. Außerdem wird ein Idealwert errechnet, indem für jedes Kriterium die Wichtigkeit der Erfüllung mit dem maximal möglichen Erfüllungsgrad (in diesem Falle also 2 = vollständig erfüllt) multipliziert wird. Auch diese Werte werden aufsummiert. Schließlich wird der reale zum Idealwert in Relation gesetzt. Der Quotient beschreibt den Nutzwert des jeweiligen Werkzeugs.

Tabelle 1 verdeutlicht dies. Die erste Spalte listet die als relevant erachteten Kriterien, die zweite Spalte die Einschätzung ihrer Wichtigkeit. Spalte 3 enthält den Erfüllungsgrad jedes Kriteriums, Spalte 4 das Produkt aus der Wichtigkeit und dem Erfüllungsgrad für das erste Werkzeug. Die Spalten 5 und 6 enthalten die jeweiligen Werte für ein zweites Werkzeug. Spalte 7 schließlich enthält für jedes Kriterium den Idealwert. Dieser ist für alle Werkzeuge gleich. Die vorletzte Zeile enthält mit den Summen der Produkte aus Wichtigkeit und Erfüllungsgrad die realen Werte jedes Werkzeugs. Die letzte Zeile schließlich beschreibt mit dem Quotienten aus realem und idealem Wert den

Nutzwert jedes Werkzeugs, nämlich 12/16=75% für Werkzeug 1 und 9/16 ≈ 56,3% für Werkzeug 2.

Aufgrund dieses Ergebnisses würde die Entscheidung auf Tool 1 fallen, da es die Anforderungen zu einem höheren Prozentsatz erfüllt, d.h. den größeren Nutzwert besitzt.

Kriterium	Wichtigkeit (W)	Tool 1 (T1)	W * T1	Tool 2 (T2)	W * T2	W * T_{max}
Erweiterbarkeit	2	2	4	1	2	4
Performance	1	1	1	2	2	2
Benutzerfreundlichkeit	2	2	4	1	2	4
Wartbarkeit	1	1	1	1	1	2
Verfügbarkeit	0	0	0	2	0	0
Funktionalität	2	1	2	1	2	4
Summe			**12**		**9**	**16**
Nutzwert		**75,0%**		**56,3%**		

Tabelle 1: Beispiel Nutzwertanalyse

4 Allgemeine Anforderungen an Automatisierungstools

Dieses Kapitel beschreibt verschiedene Anforderungen, die an Testtools gestellt werden können. Dazu werden sowohl allgemeine als auch spezielle Anforderungen aufgestellt. Durch die allgemeinen Anforderungen kann bestimmt werden, wie wichtig einem Unternehmen einzelne Testaktivitäten, Testphasen und Testtechniken sind bzw. wie wichtig deren Automatisierbarkeit ist. Bei den speziellen Anforderungen handelt es sich um Anforderungen, die speziell auf die Auswahl eines Testtools ausgerichtet sind.

4.1 Allgemeine Anforderungen

Im Folgenden werden die allgemeinen Anforderungen, die später als Vergleichsgrundlage der Marktstudie dienen, erläutert. Zu solchen Anforderungen kann man die Wichtigkeit bzw. Automatisierbarkeit von Testaktivitäten, Testtechniken und Testphasen zählen. Zu den einzelnen Testaktivitäten gehören Testplanung, Testfallerstellung, Testdurchführung Testauswertung, und im Anschluss an Tests das Fehlermanagement. Die Testtechniken werden in White-Box-Tests, Black-Box-Tests und BIT unterteilt. Für die Testphasen ist ein Bottom-Up-Vorgehen ausgehend von kleinen Systemeinheiten bis zum Test des Gesamtsystems üblich. Prinzipiell ist auch ein Top-Down-Vorgehen denkbar. Jedoch würden so auch die meisten „einfachen“ Implementierungsfehler beim Systemtest entdeckt, so dass der Aufwand zur Identifizierung der Fehlerquellen und zur Behebung der Fehler höher als beim Bottom-Up-Ansatz wäre. Daraus ergibt sich folgende Durchführungsreihenfolge: Zuerst Unit-Tests, dann Integrationstests, danach System- und Installationstests und schließlich die Akzeptanztests.

4.1.1 Testaktivitäten

Eine detaillierte Beschreibung der einzelnen Testaktivitäten erfolgte bereits in Kapitel 2.2. Die erste Aktivität ist die Testplanung und beinhaltet die Erstellung der Ressourcenplanung, der Teststrategie sowie der Testpriorisierung. Um den gewünschten Unterstützungsgrad der Testplanung feststellen zu können, stellt sich folgende Frage:

- Wie wichtig ist eine gute Unterstützung bei der Testplanung?

Bei den folgenden Testaktivitäten werden zusätzlich auch die Automatisierungsmöglichkeiten betrachtet.

Die zweite Aktivität ist die Testfallerstellung, während der die einzelnen Testfälle entwickelt werden. Hier ist es wichtig zu wissen:

- Wie wichtig ist eine gute Unterstützung bei der Testfallerstellung?
- Wie wichtig ist eine Automatisierung der Testfallerstellung?

Anschließend sollte im Testprozess die Testdurchführung erfolgen. Diese ist durch den Namen eigentlich selbsterklärend und besteht wie bereits in Kapitel 2.2 beschrieben aus mehreren Teilen. Auch hier stellen sich wiederum zwei Fragen zur Einstufung:

- Wie wichtig ist eine gute Unterstützung bei der Testdurchführung?
- Wie wichtig ist eine Automatisierung der Testdurchführung?

Die letzte Aktivität ist die Testauswertung.

- Wie wichtig ist eine gute Unterstützung bei der Testauswertung?

Das Fehlermanagement beschäftigt sich mit der Behandlung von Programmfehlern. Folgende Fragen werden bzgl. des Fehlermanagements formuliert:

- Wie wichtig ist eine gute Unterstützung des Fehlermanagements?
- Wie wichtig ist eine Automatisierung des Fehlermanagements?

4.1.2 Einordnung von Testtechniken

Testtechniken können in White-Box-Tests und Black-Box-Tests unterteilt werden. Da die Unterstützung von Black-Box-Tests allerdings eine Mindestanforderung an ein Testtool darstellt, wird hier ausschließlich die Unterstützung von White-Box-Tests betrachtet. Hier soll nun untersucht bzw. beurteilt werden inwieweit solche Testtechniken unterstützt werden sollen. Daher wird folgende Frage gestellt:

- Wie wichtig ist eine gute Unterstützung beim White-Box-Testen?

Um die Anforderungen bzgl. BIT zu betrachten, wird folgende Frage gestellt:

- Wie wichtig ist eine gute Unterstützung beim BIT?

4.1.3 Testphasen

Die verschiedenen Testphasen wurden bereits in Kapitel 2.3 im Detail beschrieben. Ähnlich wie bei den Testaktivitäten werden auch hier zwei verschiedene Aspekte betrachtet: Wie gut die Tests in der jeweiligen Phase unterstützt werden sollen, und ob die entsprechende Phase automatisiert werden soll. Zunächst wird die Testphase mit den kleinsten Einheiten, den Units, betrachtet:

- Wie wichtig ist eine gute Unterstützung bei Unit-Tests?
- Wie wichtig ist eine Automatisierung der Unit-Tests?

Die Integrationstests überprüfen das Zusammenspiel der einzelnen Units. Hierbei erfolgt eine Beurteilung bzw. Einstufung wie gut Integrationstests unterstützt und automatisiert werden sollen.

- Wie wichtig ist eine gute Unterstützung der Integrations-Tests?
- Wie wichtig ist eine Automatisierung der Integrations-Tests?

Der Systemtest soll erste Sicherheit in Bezug auf das spezifikationsgemäße Funktionieren des Gesamtsystems geben und bildet die Vorstufe zum Akzeptanztest. Es stellen sich folgende Fragen.

- Wie wichtig ist eine gute Unterstützung der System-Tests?
- Wie wichtig ist eine Automatisierung der System-Tests?

Um den gewünschten Grad der Unterstützung bzw. Automatisierung von Installationstests abschätzen zu können, werden folgende Fragen gestellt.

- Wie wichtig ist eine gute Unterstützung der Installations-Tests?
- Wie wichtig ist eine Automatisierung der Installations-Tests?

Die Akzeptanztests werden am laufenden System durch die Repräsentanten des endgültigen Benutzerkreises durchgeführt.

- Wie wichtig ist eine gute Unterstützung der Akzeptanz-Tests?

4.2 Spezielle Anforderungen

Im Folgenden werden die speziellen Anforderungen, die später als Vergleichsgrundlage der Marktstudie dienen, erläutert. Zu diesen Anforderungen gehören technische Anforderungen, die Unterstützung von Testtypen und weitere Anforderungen, die nicht den anderen Kategorien zuzuordnen waren.

Zu den technischen Anforderungen sind verwendete Betriebssysteme und Sprachen, Datenimport und Datenexport sowie die Zusammenarbeit mit anderen Testtools zu zählen. Unter einer Sprache wird im Folgenden nicht nur eine einzelne Programmiersprache verstanden, sondern auch ganze Programmierkonzepte, wie z.B. .NET oder J2EE. Außerdem werden ganze Klassen von Programmierkonzepten unter dem Begriff Sprache zusammengefasst. Ein Beispiel hierzu wäre der Einsatz web-basierter Sprachen. Darunter fallen HTML und XML, die über HTTP ausgetauscht werden können. Ebenso gehören zu web-basierten Sprachen auch spezielle XML-Formate wie etwa SOAP für Web Services. Der Begriff web-basierte Sprache wird verwendet, sofern mehrere solche Formate in einer Anforderung auftreten.

Weiterhin wird auf die Unterstützung von Testtypen eingegangen. Dies bedeutet, dass es in die Betrachtung miteinbezogen werden sollte, in welcher

Umgebung getestet wird und welche Testtypen für die jeweilige Applikation benötigt werden. Unterschieden werden: funktionale Tests, Web-Tests und Performance-Tests. Die Erklärung dieser Begriffe erfolgt weiter unten.

Abschließend werden weitere wichtige Anforderungen beschrieben, die sich nicht unter einem übergeordneten Sammelbegriff zusammenfassen lassen. Zu diesen zählen Erweiterbarkeit, Konfigurierbarkeit, Benutzeranforderungen, Nutzerexpertise, Dokumentation, Support und Kosten.

Die einzelnen Anforderungen werden im Folgenden näher erläutert.

4.2.1 Technische Anforderungen

Verwendete Betriebssysteme

Häufig ist ein Unternehmen auf die Verwendung bestimmter Betriebssysteme festgelegt, d.h. aber auch, dass ausgewählte Werkzeuge auf diesen Betriebsystemen lauffähig sein müssen. Dieser Sachverhalt verdeutlicht, dass es sich bei der Art der unterstützten Betriebssysteme um eine wichtige Anforderung an ein Testtool handelt.

- Welche Betriebssysteme sollen unterstützt werden?

Verwendete Sprachen

Bezogen auf die vorgestellte Definition des Begriffs Sprache, ist es für ein Unternehmen von großem Interesse, welche Sprachen von einem Tool unterstützt werden. Wird beispielsweise die vom Unternehmen eingesetzte Programmiersprache von einem Tool für Unit-Tests nicht unterstützt, so ist das Tool automatisch ungeeignet für das Unternehmen. Grund dafür ist, dass Unit-Tests die kleinsten Einheiten des Systems und damit im Allgemeinen Einheiten der Programmiersprache testen. Dementsprechend ergibt sich folgende Frage, die in Form einer Aufzählung beantwortet wird.

- Welche Sprachen werden verwendet?

Datenimport, Datenexport

Unter Datenimport bzw. Datenexport werden die Datenformate, z.B. HTML oder Text, verstanden, die zur Definition, Beschreibung oder Dokumentation der Tests eingesetzt werden. Beispielsweise könnten während der Testfallerstellung Daten in Form von Diagrammen importiert und dann als Programmcode exportiert werden. Um diese Tooleigenschaften in die Bewertung mit einfließen zu lassen, werden die folgenden Fragen gestellt und durch eine Auflistung der unterstützten Datenformate beantwortet:

- Welche Dateiformate sollen importiert werden können?
- Welche Dateiformate sollen exportiert werden können?

Zusammenarbeit mit anderen Testtools

Sind in den Unternehmen bereits eigene oder eingekaufte Testtools in Gebrauch, so kann es sein, dass ein weiteres Testtool mit diesen kooperieren muss.

- Mit welchen anderen (Test-)Tools soll das Tool zusammenarbeiten?

4.2.2 Testtypen

In diesem Abschnitt werden Testtechniken unter verschiedenen Überbegriffen (ähnlich wie in Kapitel 2.4) in Testtypen zusammengefasst. Dabei kann z.B. zwischen funktionalen Tests, GUI-Tests, Web-Tests und Performance-Tests unterschieden werden. Die Performance-Tests werden weiter in Lasttests, Stresstests und Tests für Antwortzeiten unterteilt.

Ein funktionaler Test [Ligg02] soll die funktionalen Anforderungen, die an ein System gestellt werden, kontrollieren, z.B. indem die Ergebnisse einer Funktion auf Korrektheit überprüft werden. Es ist dabei darauf zu achten, dass ein funktionaler Test stets ein Black-Box-Test ist. Jedoch gilt der Umkehrschluss nicht immer, da es auch Black-Box-Techniken gibt, die nicht so vorgehen.

Beim GUI-Testing wird die Benutzerschnittstelle evtl. inklusive der dahinterliegenden Funktionalität überprüft. In einigen Fällen werden solche Tests teilautomatisiert durchgeführt, wie folgendes Beispiel zeigt: Bei der Testfallerstellung könnte eine Mausspur aufgezeichnet werden, die die Benutzer-System-Interaktionen beschreibt. Anschließend könnte der so aufgezeichnete Testfall automatisch durch ein Tool abgespielt werden. Werkzeuge, die beim GUI-Testing so vorgehen werden als Capture-and-Replay-Tools bezeichnet. Nachteilig wirkt sich bei der ausschließlichen Aufzeichnung von Mausoperationen die Abhängigkeit von der Bildschirmauflösung, der Fensterposition der Anwendung, etc. aus. Daher verwenden einige Tools Anwendungswissen, wie etwa, dass es sich um eine Windows-Anwendung handelt, um die Interaktionen auf andere Art und Weise wiederholbar aufzuzeichnen.

Mit Web-Tests können Web-Applikationen getestet werden. Je nachdem, ob eher Benutzer-System-Interaktionen oder der korrekte Aufbau von Nachrichten überprüft werden soll, sind sie mit den Testtypen funktionales Testen bzw. GUI-Testing kombinierbar.

Die Performance-Tests [Ligg02] werden hier in drei untergeordnete Testtypen gegliedert: Stresstests, Lasttests (Load-Tests) und die Antwortzeiten. Ein Last- bzw. Leistungstest bringt das Software-System an Grenzbereiche heran, jedoch nicht darüber hinaus. Dazu werden Lasten erzeugt sowie Zeiten und Auslastungen gemessen. Beim Stresstest wird das Software-System unter Wegnahme von Ressourcen überlastet. Der Unterschied zu Lasttests ist unter anderem durch die erzeugte Last gegeben. Mit Hilfe von Stresstest soll beantwortet werden, wie sich das Verhalten bei Überlast ändert und ob

es beim Rückgang der Last wieder in den Normalbetrieb zurückkehrt. Schließlich sind auch noch Tests bzgl. der Antwortzeiten für die Bewertung der Geschwindigkeit relevant.

- Welche Testtypen sollen unterstützt werden (funktionale Tests, Web-Tests, Performance-Tests, etc.)?

4.2.3 Sonstige Anforderungen

Hier werden die sonstigen Anforderungen, die nicht in eine der obigen Kategorien passen, erläutert. Dazu zählen: Erweiterbarkeit, Konfigurierbarkeit, Benutzeranforderungen, Nutzerexpertise, Dokumentation, Support und Kosten.

Erweiterungen/Konfiguration

Teilweise passen die Tooleigenschaften nicht exakt zu den Anforderungen eines Unternehmens. Um solchen speziellen Anforderungen genügen zu können, muss ein geeignetes Tool so gestaltet sein, dass Erweiterungen der Funktionalität leicht durchgeführt werden können. Dies kann beispielsweise über einen Plugin-Mechanismus realisiert werden.

- Wie wichtig ist eine gute Erweiterbarkeit des Tools?

Für jedes Unternehmen mit unterschiedlichen Systemen müssen Testtools oft unterschiedlich konfiguriert werden. Ein Testtool sollte daher Konfigurationsmechanismen anbieten, so dass verschiedene Einstellmöglichkeiten übersichtlich und nachvollziehbar durchgeführt werden können. Der Unterschied zur Erweiterbarkeit eines Tools besteht nun darin, dass keine Funktionalitäten hinzugefügt, sondern nur vorhandene aktiviert bzw. eingestellt werden.

- Wie wichtig ist eine gute Konfigurierbarkeit des Tools?

Benutzeranforderungen

Zusätzlich sollte beachtet werden, ob die Bedienung des Tools selbsterklärend und damit einfach zu erlernen ist. Daraus ergibt sich folgende Frage bzgl. der Bedienbarkeit:

- Wie wichtig ist eine leichte Bedienbarkeit des Tools?

Nutzerexpertise

Wird ein Tool für Nutzer mit relativ geringer Erfahrung im Bereich des Testens gewählt, so sollte die Grundfunktionalität des Testtools einfach zugänglich sein. Beispielsweise bieten Capture-Replay-Tools häufig einen solchen Direkteinstieg an, so dass nur wenige Klicks zum Aufzeichnen bzw. Abspielen eines Testfalles nötig sind. Demgegenüber stehen etwa Tools, die zur Definition eines Testfalles das Erlernen einer Skriptsprache benötigen.

Ein grafisches Frontend kann die Benutzung von Testtools um einiges erleichtern, da dadurch Eingaben leichter durchzuführen sind. Daher muss jedes Unternehmen entscheiden, ob ein grafisches Frontend relevant ist oder nicht.

- Wie wichtig ist die Existenz eines grafischen Frontends für das Tool?

Je kürzer eine Einarbeitungszeit ist, desto schneller kann der Toolnutzer produktiv arbeiten. Dementsprechend sollte die Einarbeitungszeit in die Bewertung mit eingehen.

- Wie wichtig ist eine kurze Einarbeitungszeit für das Tool?

Dokumentation

Eine gute Dokumentation kann den Umgang mit dem Tool erheblich erleichtern und sollte deshalb als Auswahlkriterium einbezogen werden.

- Wie wichtig ist das Angebot einer Dokumentation zu dem Tool?

Support

Auch durch den Support kann der Umgang mit einem Testtool erleichtert werden. Darunter können sowohl Schulungen, telefonischer Support als auch Community-Support gezählt werden. Werden Schulungen angeboten, so sollten sie möglichst kostengünstig sein. Wichtig sind zudem die Dauer und der Inhalt der Schulungen.

- Wie wichtig ist das Angebot von Schulungen?
- Wie wichtig ist das Angebot eines telefonischen Supports?

Bei einigen Tools finden sich Benutzergruppen zu Gemeinschaften (sog. Communities) zusammen, die sich beispielsweise in Form eines Online-Forums gegenseitig bei der Lösung von Problemen im Umgang mit dem entsprechenden Tool unterstützen. Teilweise werden derartige Foren vom Hersteller initiiert bzw. moderiert. Da ein solcher Community-Support die Einführung des Tools erleichtert bzw. Unterstützung bei Problemen bietet, kann folgende Frage formuliert werden.

- Wie wichtig ist die Existenz eines Community-Supports?

Kosten

Die Kosten eines Tools sind teilweise maßgeblich für dessen Auswahl. Dies liegt insbesondere daran, dass gerade kleine und mittelständische Unternehmen oft nur kleine Budgets für (Neu-)Beschaffungen zur Verfügung haben. Die hier angegebenen Werte beziehen sich auf durchschnittliche Kosten eines Tools. Eine genaue Aussage lässt sich für ein Testtool nur schwer vornehmen, da die Kosten oft direkt mit dem Kunden ausgehandelt werden oder stark von der gewünschten Lizenzart abhängen.

- Wie viel darf das Testtool kosten (durchschnittliche Kosten, um das Tool im Unternehmen einsetzen zu können)?

5 Spezifische Anforderungen der Projektpartner

In den folgenden Unterkapiteln sind die speziellen Anforderungen an Testtools der Projektpartner WIKON und market maker zu finden. Zunächst erfolgt eine Vorstellung der beiden Projektpartner mit einer Beschreibung ihrer Anforderungen. Anschließend werden anhand vorgefertigter Tabellen die Anforderungen dargestellt. In diesen Tabellen setzen die Projektpartner Prioritäten für die einzelnen Anforderungen. Die entsprechenden Fragen zu den Tabellen wurden in Kapitel 4 näher erläutert.

5.1 Anforderungen WIKON

Der Projektpartner WIKON entwickelt, produziert und vertreibt Produkte auf dem Gebiet der technischen Kommunikation. Darunter zählt die Fernüberwachung gebäudetechnischer Systeme, wie z. B. Heizungsanlagen, Telemetering für die Energiewirtschaft (Fern-Zählerauslese) oder die Schachtüberwachung. Das System von WIKON besteht aus zwei Subsystemen. Das erste bildet die Geräte-Software, die über SMS[1] Nachrichten in festgelegten Zeitintervallen an einen Server schickt. Der Server, der das zweite Subsystem darstellt, nimmt die Mitteilungen entgegen und verarbeitet sie so, dass die entsprechenden Daten über ein Web-Frontend für den Nutzer zugreifbar werden. Für den Test der Geräte-Software werden Skripte eingesetzt, jedoch lassen sich derartige Aufgaben wegen ihrer speziellen Art nur schwer weiter automatisieren. Dies liegt unter anderem daran, dass für einzelne Testfälle Hardware-Aktionen nötig wären, wie etwa das Drücken eines Knopfes. Daher wird der Fokus im Kontext dieses Reports auf den Test der Server-Software gelegt. Wichtig ist hierbei der korrekte Aufbau der Nachrichten, die zwischen Server und dem Browser eines Nutzers ausgetauscht werden. Eine Nachricht ist in diesem Sinne korrekt aufgebaut, wenn die in ihr enthaltenen Benutzerinformationen korrekt sind. Im Folgenden werden Tools gerade für diesen Bereich betrachtet, so dass eine Unterstützung von Systemtests, womit auch Subsysteme gemeint sein können, im Vordergrund steht.

Die Anforderungen von WIKON beziehen sich auf ein ideales Tool und finden sich in Tabelle 3 wieder. Wie man erkennt, sind dem Projektpartner viele Dinge sehr wichtig. Hier soll jedoch der Test auf Korrektheit der Nachrichten im Vordergrund stehen. Besonders hervorzuheben ist die von WIKON gewünschte Konfigurierbarkeit und Erweiterbarkeit des Testautomatisierungstools. Dies ist dadurch zu begründen, dass die Server-Software Nachrichten in vielen Formaten, z.B. HTML und Email, an den Nutzer senden kann.

[1] Short Message Service

Weitere wichtige Aspekte sind der Kostenfaktor und die Benutzerfreundlichkeit des Testautomatisierungstools, da WIKON keine eigene Testabteilung besitzt, sondern Testaufgaben direkt durch die Entwickler mit übernommen werden. Eine lange Einarbeitungszeit kann somit nicht akzeptiert werden.

Die Bewertung der einzelnen Anforderungen erfolgt nach folgendem Schema:

✓	wichtig
o	teilweise wichtig
×	unwichtig

Tabelle 2: Bewertungsschema Anforderungen

WIKON			Bewertung
Allgemeine Anforderungen	Testaktivitäten	Testplanung	×
		Testfallerstellung	o
		Testfallerstellung automatisierbar	o
		Testdurchführung	✓
		Testdurchführung automatisierbar	✓
		Testauswertung	✓
		Testauswertung automatisierbar	✓
		Fehlermanagement	o
		Fehlermanagement automatisierbar	o
	Test-technik	White-Box-Test	✓
		BIT	×
	Testphasen	Unittest	o
		Unittests automatisierbar	✓
		Integrationstest	✓
		Integrationstest Automatisierbar	✓
		Systemtest	✓
		Systemtest automatisierbar	✓
		Installationstest	×
		Installationstest automatisierbar	×
		Akzeptanztest	×
Spezielle Anforderungen	Technische Anforderungen (Freitext)	Betriebssysteme	Linux
		Sprachen	C, C++, SQL, PHP, Bash
		Datenimport	XML
		Datenexport	HTML, Datenbankeintrag
		Zusammenarbeit mit anderen Tools	
	Testtypen (Freitext)		Web-Tests, funktionale Tests
	Weitere Anforderungen	Erweiterbarkeit	✓
		Konfigurierbarkeit	✓
		Bedienung des Tools	✓
		Grafisches Frontend	o
		Einarbeitungszeit	✓
		Dokumentation	✓
		Schulung	o
		Telephonischer Support	o
		Community Support	o
		Kosten	500$

Tabelle 3: Anforderungen des Projektpartners WIKON

5.2 Anforderungen market maker

Ein weiterer Projektpartner, aufgrund dessen Anforderungen ein geeignetes Testtool ausgewählt werden soll, ist market maker. Market maker entwickelt Software für Finanzdienstleister und private Anwender. Die Software-Lösungen eignen sich für Portfoliomanagement und Wertpapieranalyse, daher gehören zu den Kunden von market maker Vermögensverwalter, institutionelle Anleger, Berater in Banken und Sparkassen sowie private Investoren. Die Produktpalette reicht von Börseninformationen bis zur Software für private und institutionelle Anwender. Da market maker hauptsächlich mit Banken und Sparkassen zusammen arbeitet, sind die Produkte oft strengen Richtlinien unterworfen. Eine weitere Herausforderung sind Gesetzesänderungen, die gerade in diesem Bereich häufig rasch umgesetzt werden müssen. Die eben genannten Dinge spiegeln sich auch in den Anforderungen an ein Testtool wieder.

Tabelle 4 zeigt die Anforderungen an ein Tool für market maker. Einige Anforderungen werden schon durch Verwendung bestimmter Testkonzepte bzw. Testtools zumindest teilweise abgedeckt, deren Automatisierungsgrad aber eher gering ist. So werden beispielsweise Unit-Tests durch die Entwickler u.a. mit Hilfe von DUnit durchgeführt, bei dem es sich um ein Testframework für Delphi handelt.

Problematisch sind vorwiegend Integrations- und Installationstests. Integrationstests besitzen für market maker eine große Bedeutung, da ein fehlerhaftes Zusammenspiel der einzelnen Komponenten, die in diesem Fall eher als Subsysteme zu bezeichnen sind, häufig erst bei Tests auf Systemebene entdeckt werden. Installationstests sind für market maker ebenso sehr wichtig, wobei sowohl Tests vor, als auch nach der Übertragung auf die Kundensysteme eine Rolle spielen. Vor der eigentlichen Installation auf das jeweilige System soll überprüft werden, ob gewisse Bedingungen an die Systemumgebung erfüllt werden. Ein Beispiel wäre in diesem Bereich ein Test auf die Existenz gewisser Bibliotheken in bestimmten Versionen, ohne die die Software nicht korrekt ablaufen kann. Zum anderen soll die Funktionalität des Systems nach Installation überprüft werden. Für einen derartigen Installationstest, der das Funktionieren des Systems nachweist, ist ein BIT-Konzept wünschenswert, so dass der Tester/Benutzer nur das System in einem speziellen Modus starten muss und dieses anschließend selbstständig überprüft, ob die Bedingungen für ein korrektes Verhalten erfüllt sind. Beispielsweise könnte hierbei überprüft werden, ob eine Datenbankanbindung tatsächlich vorhanden und ansprechbar ist. Die Notwendigkeit für derartige Tests ergibt sich aus dem Kundenkreis von market maker, der u.a. auch Banken beinhaltet. Für solche Kunden muss nachgewiesen werden, dass das System in ihrer Umgebung zur Installationszeit korrekt funktioniert.

Im weiteren Verlauf dieses Reports werden insbesondere die Anforderungen im Hinblick auf Installationstests betrachtet, da solche bei market maker wegen der oben genannten Gründe von hoher Priorität sind.

Die Bewertung erfolgt nach dem bereits bekannten Schema.

MARKET MAKER			Bewertung
Allgemeine Anforderungen	Testaktivitäten	Testplanung	×
		Testfallerstellung	o
		Testfallerstellung automatisierbar	×
		Testdurchführung	✓
		Testdurchführung automatisierbar	✓
		Testauswertung	✓
		Fehlermanagement	o
		Fehlermanagement automatisierbar	×
	Test-technik	White-Box-Test	×
		BIT	✓
	Testphasen	Unittest	o
		Unittests automatisierbar	✓
		Integrationstest	✓
		Integrationstest Automatisierbar	o
		Systemtest	o
		Systemtest automatisierbar	o
		Installationstest	✓
		Installationstest automatisierbar	✓
		Akzeptanztest	×
Spezielle Anforderungen	Technische Anforderungen (Freitext)	Betriebssysteme	Windows NT, 2000, XP, 2003, Vista
		Sprachen	haupts. Delphi (auch C und .Net/C#)
		Datenimport	Prüfpunkte, Konfiguration
		Datenexport	Prüfprotokoll, Systemdiagnose, archivierte Umgebung, um Fehler weiter untersuchen zu können
		Zusammenarbeit mit anderen Tools	
	Testtypen (Freitext)		funktionale Tests
	Weitere Anforderungen	Erweiterbarkeit	o
		Konfigurierbarkeit	✓
		Bedienung des Tools	✓
		Grafisches Frontend	×
		Einarbeitungszeit	✓
		Dokumentation	×
		Schulung	×
		Telephonischer Support	×
		Community Support	×
		Kosten	3500$

Tabelle 4: Anforderungen des Projektpartners market maker

6 Marktstudie

Das Arbeitspaket 5 beinhaltet sowohl eine Marktanalyse der verschiedenen Testtools, als auch die Auswahl geeigneter Testtools für die Projektpartner.

In diesem Kapitel werden die Eigenschaften von Testtools verglichen. Dabei werden die Eigenschaften der Tools ähnlich abgefragt, wie die Anforderungen von Unternehmensseite und werden dann in einer entsprechenden Tabelle bewertet. Im ersten Unterkapitel werden die Bewertungskriterien detailliert beschrieben, d.h. welche Bewertungsmaßstäbe für die verschiedenen Kriterien zugrunde gelegt werden. Dabei werden die Anforderungen an Testautomatisierungstools auf die entsprechenden Eigenschaften eines Tools abgebildet. Die einzelnen Kriterien, die zur Bewertung der Tools herangezogen werden, korrelieren stark mit den in Kapitel 4 vorgestellten Anforderungen, z.B. kann die Toolunterstützung der einzelnen Testaktivitäten bewertet werden. Dieser Sachverhalt gilt gleichermaßen für alle diskutierten Anforderungen an Testautomatisierungstools, jedoch ist darauf zu achten, dass die Anforderungen von Unternehmen formuliert werden, die eigene Maßstäbe zugrunde legen. Ein direkter Bezug zu den hier verwendeten Bewertungsmaßstäben ist somit zwar wünschenswert jedoch nicht zu garantieren.

6.1 Erklärung der Bewertungskriterien

Bei der Bewertung der Tools anhand ihrer Eigenschaften wird für jedes Kriterium eine Unterscheidung bzgl. der Wichtigkeit und des Unterstützungsgrades vorgenommen. Da diese stets eine subjektive Sicht beinhalten, werden in diesem Unterkapitel die zugrunde gelegten Bewertungsmaßstäbe konkretisiert. Die Bewertung erfolgt aus praktischen Gründen nicht durch eine Evaluierung der Tools, sondern beruht hauptsächlich auf den öffentlich zugänglichen Informationen der jeweiligen Hersteller. Für jede Tooleigenschaft, bei der es sich nicht um ein Freitextfeld handelt, wird eine Unterteilung in drei Stufen analog zur Bewertung der Anforderungen vorgenommen.

Die Eigenschaften, die die Unterstützung einzelner Testphasen und deren Automatisierungsgrad behandeln, werden folgendermaßen bewertet:

Das Tool unterstützt eine Testaktivität sehr gut, sofern sie in den erhältlichen Informationen direkt erwähnt wird bzw. offensichtlich anhand der Dokumentation abzuleiten ist. Eine Testaktivität wird teilweise unterstützt, wenn sich die Definitionen aus Kapitel 2.2 nicht komplett auf das Tool, sondern nur zum Teil abbilden lassen. In allen anderen Fällen wird von einer fehlenden Unterstützung ausgegangen. Eine Ausnahme von dieser Regel ist die Bewertung des Unterstützungsgrades der Testauswertung, die als sehr gut un-

terstützt definiert wird, sofern eine Vielzahl von statistischen Auswertungsmöglichkeiten zur Verfügung steht. Eine mittlere Unterstützung wird angenommen, wenn eine einfache Auswertung beispielsweise in Form eines Kuchendiagramms unter Angabe der fehlgeschlagenen Testfälle existiert. Die Testauswertung wird schlecht bis gar nicht unterstützt, wenn die Abarbeitung der Testfälle beim ersten Auftreten eines Fehlers abgebrochen wird oder die Dokumentation der Fehler sich darauf beschränkt, dass ein Fehler aufgetreten ist, jedoch nicht nachvollziehbar dargestellt wird, innerhalb welchen Testfalles er entstanden ist.

Der Automatisierungsgrad einer Testaktivität ist „sehr hoch“, wenn der Nutzer keine Interaktion mit dem Tool durchführen muss. Beispielsweise ist die Erstellung von Testfällen vollautomatisch, wenn für White-Box-Tests der Programmcode analysiert wird und Testfälle entsprechend dem gewünschten Überdeckungsgrad daraus hergeleitet werden. Dabei können etwa Belegungen von Eingabewerten oder Eigenschaften automatisch generiert werden. Eine Testaktivität ist teilautomatisiert, wenn der Benutzer eine Interaktion mit dem System eingehen muss, aber dennoch ein Teil der Aktivität automatisch abgearbeitet werden kann. Ein Beispiel ist etwa eine teilautomatische Testdurchführung, bei der der Nutzer jeden Testfall per Hand anstoßen muss, diese jedoch für sich automatisch ablaufen. Eine Testaktivität ist überhaupt nicht automatisiert, wenn der Nutzer (mit der möglichen Ausnahme sehr kleiner Prozessteile) die Hauptverantwortung trägt. Im Falle der Testfallerstellung wäre hier als nicht automatisiert das Schreiben von Skripten per Hand zu verstehen.

Die Unterstützung von White-Box-Tests und BIT, sowie der Testphasen werden analog zur Unterstützung der Testaktivitäten anhand ihrer Definition und der erhältlichen Toolinformation bewertet. Etwas schwierig gestaltet sich hierbei die Unterstützung von Akzeptanztests, da diese ja durch einen späteren Benutzer durchgeführt werden und damit von diesem abhängen. Hier wird wieder auf die Dokumentation Bezug nehmend bewertet. Wird die Unterstützung von Akzeptanztests in der Dokumentation hervorgehoben, so erfolgt eine „sehr gute“ Bewertung. Wird sie jedoch nur am Rande erwähnt oder ist implizit enthalten, so wird die Unterstützung als „mittel“ angenommen. Beispielsweise wird bei einigen Tools eine Benutzer-System-Interaktion aufgezeichnet, um funktionale bzw. GUI-Tests automatisch durchzuführen. Einige Tools speichern dabei die Testfälle in einer nicht von Menschen lesbaren Form, z.B. in einem Binärformat. Werden die Testfälle jedoch in einem lesbaren Format abgespeichert und können sie darüber hinaus manuell durchgeführt werden, so kann dies als Definition eines Testfalles für den Akzeptanztest verstanden werden, da ein solcher durch einen Benutzer abgearbeitet und bzgl. bestimmter Kriterien bewertet werden kann.

Ein Tool ist im hier verwendeten Sinne sehr gut erweiterbar, wenn beispielsweise ein Plugin-Konzept vorhanden ist, mit dem neue Funktionalität ergänzt werden kann. Eine mittlere Erweiterbarkeit besteht, sofern der Funk-

tionsumfang des Tools größtenteils feststeht, aber spezielle Teile erweitert werden können. Wird beispielsweise eine API angeboten, mit der Testfälle mit zusätzlicher Kontrolllogik versehen werden können, ist die Erweiterbarkeit des Testautomatisierungstools als mäßig einzustufen, da sie nur beschränkt vorhanden ist. In allen anderen Fällen ist eine Erweiterbarkeit kaum bis gar nicht vorhanden und wird entsprechend schlecht bewertet.

Die Konfigurierbarkeit eines Tools wird im weiteren Verlauf als hoch angenommen, wenn sowohl die Möglichkeit als auch eine ausreichende Dokumentation zur Konfigurierung vorhanden sind. Ein Tool ist etwa sehr gut konfigurierbar, sofern dokumentierte Konfigurationsdateien der Software vorhanden sind. Die Konfigurierbarkeit des Tools befindet sich auf mittlerer Ebene, sofern die Software zumindest auf gewisse Testtypen oder Sprachen abstimmbar ist. Sonst wird die Software als nicht konfigurierbar eingestuft.

Ein weiterer wichtiger Aspekt ist die Bedienbarkeit des Testtools. Hierbei wird eine rein subjektive Bewertung vorgenommen, was sich in diesem Bereich auch kaum vermeiden lässt. Ein Tool gilt hier als sehr benutzerfreundlich, wenn zum einen die Benutzeroberfläche (graphisch oder textbasiert) einfach gehalten ist und ausreichende Hilfemöglichkeiten vorhanden sind. Zusätzlich spielt die Modellierbarkeit der Produkte, wie etwa der Testfälle, eine große Rolle. Es ist beispielsweise nicht benutzerfreundlich einen Testfall nur durch Schreiben kryptischer Dateien zu erstellen. Hingegen kann in diesem Kontext eine sehr gute Bedienbarkeit in Form einer Zusammenstellung eines Testfalls basierend auf graphischen Elementen erreicht werden.

Mit der Bedienbarkeit stehen ebenfalls die Eigenschaften „graphisches Frontend“ und „Einarbeitungszeit“ in Verbindung. Ein sehr gutes grahisches Frontend ist eine Benutzeroberfläche, die die Funktionalitäten in einer offensichtlichen Art und Weise widerspiegelt und den Benutzer in seinen Aufgaben graphisch unterstützt. Ein graphisches Frontend ist nur mittelmäßig, sofern dies nicht der Fall ist. Wie man sieht, ist hierbei wiederum ein gewisses Maß Subjektivität vorhanden. Die Einarbeitungszeit korreliert negativ sowohl mit der Benutzbarkeit als auch der vorhandenen Dokumentation. Zusätzlich spielt noch der Automatisierungsgrad einzelner Testaktivitäten, Testtechniken und Testphasen eine Rolle. Muss etwa erst eine proprietäre Skriptsprache zur Definition von Testfällen erlernt werden, steigt die Einarbeitungszeit in erheblichem Maße im Vergleich mit einem Tool, das eine vollautomatische Testfallerstellung unterstützt, und wird entsprechend nur maximal als durchschnittlich bewertet. Wohingegen die Unterstützung einer allgemeinen Sprache, wie etwa Visual Basic, für eine kürzere Einarbeitungszeit spricht. In einem solchen Fall sind jedoch auch die anderen oben genannten Faktoren mit einzubeziehen.

Eine weitere Eigenschaft, anhand derer ein Tool bewertet werden kann, ist seine Dokumentation. Das Tool verfügt über ausgezeichnete Dokumentation, wenn neben einem Benutzer-Handbuch auch Dokumente zum Schnell-

einstieg, wie etwa Tutorials, und eine integrierte Hilfeoption vom Hersteller angeboten werden. Sie ist nur durchschnittlich, wenn eine oder mehre der eben genannten Dokumentationsarten gar nicht oder nur in geringem Umfang vorhanden sind.

Der Support-Bereich ist in die Eigenschaften Schulung, telefonischer Support und Community-Support gegliedert. Schulungen werden in großem Umfang unterstützt, wenn der Hersteller nicht nur grundlegende Tooleinführungen anbietet, sondern auch spezielle Anforderungen der Benutzer unterstützt. Werden nur wenige, schlecht abgestimmte Schulungen angeboten, so wird die Eigenschaft nur als gut bewertet. Findet sich in den Herstellerinformationen kein Schulungsangebot, so wird die Kategorie entsprechend als nicht unterstützt festgelegt.

Da die Güte des telefonischen Supports relativ schwierig im Vorhinein zu bestimmen ist, wird hier folgender Bewertungsmaßstab zugrunde gelegt. Wird explizit in den Herstellerinformationen auf telefonischen Support hingewiesen, so wird die Kategorie als sehr gut unterstützt angesetzt. Findet sich in den entsprechenden Informationen ein Verweis auf einen telefonischen Ansprechpartner für die Toolnutzung und wird sie nicht als eigene Support-Art angeboten, so erhält das Tool nur eine mittlere Bewertung.

Mit Community-Support wird die Existenz zusätzlicher Hilfeleistungen durch Zusammenschließungen von Nutzern und/oder Angehörigen des Herstellers bezeichnet, die frei in Anspruch genommen werden können. Beispielsweise finden sich bei weit verbreiteten Tools Nutzer in Online-Foren zusammen, die sich gegenseitig bei der Lösung im Umgang mit dem Testtool unterstützen. Für sehr guten Community-Support spricht die Existenz eines vom Hersteller unterstützten Forums, da dieser wohl die größte Problemlösungskompetenz besitzt. Die Existenz allein reicht jedoch nicht für eine überdurchschnittliche Bewertung aus, sondern wird abhängig von der Anzahl der Einträge und zugehöriger Lösungsvorschläge vergeben. Sind die Einträge in den existierenden Foren veraltet oder werden nicht zeitnah beantwortet, so wird diese Eigenschaft höchstens als durchschnittlich bewertet. Finden sich im gesamten Internet nur vereinzelt Bemerkungen zu dem Tool, so wird ein Support durch eine Nutzergemeinschaft als nicht existent angenommen.

6.2 Automatisierungstools

Hier werden die untersuchten Tools anhand einer Überblickstabelle nach den Kriterien aus Kapitel 4 beschrieben. Tabelle 6 ist ähnlich wie Tabelle 3 aufgebaut. In den Zeilen befinden sich die Bewertungen der einzelnen Eigenschaften und in den Spalten die untersuchten Tools.

Hierbei erfolgt die Bewertung nach dem Schema aus Kapitel 6.1. In Tabelle 5 sind zusätzlich die in der Bewertungstabelle verwendeten Zeichen erklärt.

✓	erfüllt	sehr gut	enthalten	vollauto-matisierbar
o	teilweise erfüllt	mittelmäßig	teilweise enthalten	teilauto-matisierbar
×	nicht erfüllt	schlecht	nicht enthalten	nicht auto-matisierbar

Tabelle 5: Bewertungsschema Testtools

Software Tools			Automated Test Designer	AutoTester One	Functional Tester
Allgemeine Eigenschaften	Testaktivitäten	Testplanung	o	×	×
		Testfallerstellung	o	o	✓
		Testfallerstellung automatisierbar	o	o	o
		Testdurchführung	o	✓	✓
		Testdurchführung automatisierbar	o	o	✓
		Testauswertung	×	✓	✓
		Fehlermanagement	×	×	o
		Fehlermanagement automatisierbar	×	×	×
	Test-technik	White-Box-Test	×	×	o
		BIT	×	×	×
	Testphasen	Unittest	×	×	o
		Unittests automatisierbar	×	×	o
		Integrationstest	×	×	×
		Integrationstest automatisierbar	×	×	×
		Systemtest	o	✓	o
		Systemtest automatisierbar	o	o	o
		Installationstest	×	×	×
		Installationstest automatisierbar	×	×	×
		Akzeptanztest	o	o	o
Spezielle Eigenschaften	allg. Info	Anbieter	AtYour Side	Autotester	Rational IBM
	Technische Eigen-schaften (Freitext)	Betriebssysteme	Windows	Windows	Windows, Linux
		Sprachen	beliebig	Windows, web-basierte Sprachen, Sprachen für ERP Systeme	Java, .NET, web-basierte Sprachen
		Datenimport	eigenes Format	eigenes Format	eigenes Format
		Datenexport	eigenes Format	eigenes Format	eigenes Format
		Zusammenarbeit mit anderen Tools	Keine	TestDirector	verschiedene Rational-Produkte, Visual Studio .NET, Eclipse, etc.
	Testtypen (Freitext)		unabhängig	GUI-Tests, Web-Tests	Funktionale Tests, GUI-Tests, Web-Tests
	Weitere Eigenschaften	Erweiterbarkeit	×	×	×
		Konfigurierbarkeit	×	×	×
		Bedienung des Tools	o	o	✓
		Grafisches Frontend	o	o	✓
		Einarbeitungszeit	o	o	o
		Dokumentation	o	o	✓
		Schulung	×	×	✓
		Telefonischer Support	o	o	✓
		Community Support	×	×	✓
		Kosten	6000$	auf Anfrage	4100$

Software Tools			GUITAR	LoadRunner	Netvantage Functional Tester
Allgemeine Eigenschaften	Testaktivitäten	Testplanung	o	×	×
		Testfallerstellung	✓	✓	✓
		Testfallerstellung automatisierbar	✓	o	o
		Testdurchführung	✓	✓	o
		Testdurchführung automatisierbar	✓	o	o
		Testauswertung	✓	✓	o
		Fehlermanagement	o	o	×
		Fehlermanagement automatisierbar	o	o	×
	Test-technik	White-Box-Test	✓	o	×
		BIT	×	×	×
	Testphasen	Unittest	o	✓	×
		Unittests automatisierbar	o	o	×
		Integrationstest	o	o	×
		Integrationstest automatisierbar	o	o	×
		Systemtest	o	✓	o
		Systemtest automatisierbar	o	o	o
		Installationstest	×	×	×
		Installationstest automatisierbar	×	×	×
		Akzeptanztest	×	×	×
Spezielle Eigenschaften	allg. Info	Anbieter	University of Maryland	Mercury	Netvantage Technologies
	Technische Eigen-schaften (Freitext)	Betriebssysteme	unabhängig	Windows,Linux, HP Unix, Solaris	Windows
		Sprachen	Java	web-basierte Sprachen, J2EE, .NET, XML, ERP/CRM	web-basierte Sprachen
		Datenimport	eigenes Format	XML	eigenes Format
		Datenexport	eigenes Format	XML	eigenes Format
		Zusammenarbeit mit anderen Tools	Keine	Keine	Keine
	Testtypen (Freitext)		GUI-Tests, Web-Tests	Performance-Tests	funktionale Web-Tests, GUI-Tests (Web)
	Weitere Eigenschaften	Erweiterbarkeit	✓	×	×
		Konfigurierbarkeit	✓	×	×
		Bedienung des Tools	✓	✓	o
		Grafisches Frontend	✓	✓	o
		Einarbeitungszeit	✓	o	o
		Dokumentation	o	✓	o
		Schulung	×	✓	×
		Telefonischer Support	×	o	×
		Community Support	×	✓	×
		Kosten	Kostenlos	Auf Anfrage	Auf Anfrage

Software Tools			PETA	QA Wizard	Rational Robot
Allgemeine Eigenschaften	Testaktivitäten	Testplanung	×	×	×
		Testfallerstellung	✓	✓	✓
		Testfallerstellung automatisierbar	×	o	o
		Testdurchführung	✓	✓	✓
		Testdurchführung automatisierbar	✓	✓	✓
		Testauswertung	✓	✓	o
		Fehlermanagement	o	×	o
		Fehlermanagement automatisierbar	×	×	o
	Test-technik	White-Box-Test	×	×	×
		BIT	×	×	×
	Testphasen	Unittest	×	×	×
		Unittests automatisierbar	×	×	×
		Integrationstest	✓	×	×
		Integrationstest automatisierbar	✓	×	×
		Systemtest	✓	✓	✓
		Systemtest automatisierbar	✓	✓	✓
		Installationstest	×	×	×
		Installationstest automatisierbar	×	×	×
		Akzeptanztest	×	×	×
Spezielle Eigenschaften	allg. Info	Anbieter	Verit	Seapine	IBM Rational
	Technische Eigen-schaften (Freitext)	Betriebssysteme	unabhängig	Windows	Windows
		Sprachen	web-, nachrichten-basierte Sprachen	.NET, Java, PHP, ASP, C++, Visual Basic, web-basierte Sprachen	.NET, Java, web-basierte Sprachen, Windows
		Datenimport	XML	eigenes Format	eigenes Format
		Datenexport	XML	XML	eigenes Format
		Zusammenarbeit mit anderen Tools	Eclipse	TestTrack Pro, Surround SCM, Visual Source Safe	verschiedene Rational-Produkte
	Testtypen (Freitext)		funktionale Tests (Nachrichten), Web-Tests	funktionale Tests, Web-Tests, GUI-Tests	Web-Tests, GUI-Tests, funktionale Tests, Performance-Tests
	Weitere Eigenschaften	Erweiterbarkeit	o	o	✓
		Konfigurierbarkeit	✓	×	o
		Bedienung des Tools	✓	✓	✓
		Grafisches Frontend	✓	✓	✓
		Einarbeitungszeit	✓	×	✓
		Dokumentation	✓	✓	o
		Schulung	o	✓	o
		Telefonischer Support	✓	o	×
		Community Support	×	×	o
		Kosten	auf Anfrage	3000$	4000$

Software Tools			TestComplete	TestSmith	Vermont HighTest Plus
Allgemeine Eigenschaften	Testaktivitäten	Testplanung	o	×	×
		Testfallerstellung	✓	✓	o
		Testfallerstellung automatisierbar	o	o	o
		Testdurchführung	✓	✓	✓
		Testdurchführung automatisierbar	✓	o	✓
		Testauswertung	✓	✓	o
		Fehlermanagement	o	×	o
		Fehlermanagement automatisierbar	o	×	o
	Test-technik	White-Box-Test	o	×	×
		BIT	×	×	×
	Testebenen	Unittest	✓	×	×
		Unittests automatisierbar	✓	×	×
		Integrationstest	o	×	×
		Integrationstest automatisierbar	o	×	×
		Systemtest	✓	✓	✓
		Systemtest automatisierbar	✓	o	✓
		Installationstest	×	×	×
		Installationstest automatisierbar	×	×	×
		Akzeptanztest	×	×	o
Spezielle Eigenschaften	allg. Info	Anbieter	AutomatedQA	Quality Forge	Vermont Creative Software
	Technische Eigen-schaften (Freitext)	Betriebssysteme	Windows	Windows	Windows
		Sprachen	.NET, Visual Basic, Delphi, C++ Builder, Java	web-basierte Sprachen	web-basierte Sprachen, Windows
		Datenimport	XML	eigenes Format	eigenes Format
		Datenexport	XML	Text, HTML	eigenes Format
		Zusammenarbeit mit anderen Tools	Plugins für ADO, etc.	Keine	Keine
	Testtypen (Freitext)		funktionale Tests, GUI-Tests	Web-Tests	Web-Tests, GUI-Tests
	Weitere Eigenschaften	Erweiterbarkeit	o	×	×
		Konfigurierbarkeit	o	o	×
		Bedienung des Tools	✓	o	o
		Grafisches Frontend	✓	o	o
		Einarbeitungszeit	o	✓	✓
		Dokumentation	✓	✓	o
		Schulung	✓	×	o
		Telefonischer Support	✓	o	o
		Community Support	×	×	o
		Kosten	500$	1500$	300$

Software Tools			WebInject	WebKing	WinRunner
Allgemeine Eigenschaften	Testaktivitäten	Testplanung	×	×	×
		Testfallerstellung	✓	o	✓
		Testfallerstellung automatisierbar	o	o	o
		Testdurchführung	✓	o	✓
		Testdurchführung automatisierbar	✓	✓	✓
		Testauswertung	✓	✓	✓
		Fehlermanagement	×	×	×
		Fehlermanagement automatisierbar	×	×	×
	Test-technik	White-Box-Test	×	×	×
		BIT	×	×	×
	Testphasen	Unittest	×	×	×
		Unittests automatisierbar	×	×	×
		Integrationstest	×	×	×
		Integrationstest automatisierbar	×	×	×
		Systemtest	✓	o	✓
		Systemtest automatisierbar	✓	✓	✓
		Installationstest	×	×	×
		Installationstest automatisierbar	×	×	×
		Akzeptanztest	×	×	o
Spezielle Eigenschaften	allg. Info	Anbieter	Webinject	parasoft	Mercury
	Technische Eigenschaften (Freitext)	Betriebssysteme	Windows, Unix	Windows, Linux, Solaris	Windows
		Sprachen	web-basierte Sprachen	Java Script, web-basierte Sprachen	.Net, web-basierte Sprachen, Windows
		Datenimport	XML	eigenes Format	eigenes Format
		Datenexport	XML, HTML	eigenes Format	eigenes Format, HTML
		Zusammenarbeit mit anderen Tools	Keine	Eclipse	Quick Test Professional
	Testtypen (Freitext)		Web-Tests, Performance-Tests	funktionale Web-Tests, Load-Tests	Web-Tests, funktionale Tests
	Weitere Eigenschaften	Erweiterbarkeit	o	×	o
		Konfigurierbarkeit	✓	×	✓
		Bedienung des Tools	o	o	✓
		Grafisches Frontend	o	✓	✓
		Einarbeitungszeit	✓	o	o
		Dokumentation	o	o	o
		Schulung	×	×	✓
		Telefonischer Support	×	o	o
		Community Support	o	×	✓
		Kosten	Kostenlos	4000$	6000$

Tabelle 6: Übersicht Testtools

6.3 Detaillierte Beschreibung der Tools

Dieses Unterkapitel stellt nun die einzelnen Testtools vor. Für jedes Tool aus Tabelle 6 existiert ein eigener Abschnitt, in dem seine Besonderheiten erläutert werden.

Einige Hersteller bieten ihre Produkte in zwei Formen an: Entweder als individuelle Tools oder in so genannten Tool-Suiten. Ein individuelles Tool ist hierbei ein einzelnes Software-Produkt, das als eine Einheit verkauft und benutzt werden kann. Es kann aber sein, dass ein individuelles Tool aus mehreren Komponenten besteht, die jedoch nicht einzeln angeboten werden. Im Kontext dieses Berichts wird der Begriff Tool mit einem individuellem Tool gleichgesetzt. Andererseits kann ein Software-Produkt für das Testen auch als Sammlung individueller Tools angeboten werden. Diese so genannten Tool-Suiten haben neben ökonomischen Gesichtspunkten das Ziel, eine geeignete Zusammenstellung darzustellen, um einen wohldefinierten Prozess oder Teilprozess zu unterstützen. Ein Beispiel wäre eine Tool-Suite für Software-Testen, die jede Testaktivität unterstützt. Da viele Unternehmen eher den Fokus auf geeignete Toolunterstützung in der Testfallerstellung und Testdurchführung wünschen, wird im Folgenden das Hauptaugenmerk auf Tools und Produkte gelegt, die zur automatischen Behandlung dieser Aktivitäten geeignet sind.

6.3.1 Automated Test Designer (ATD)

Der Automated Test Designer [@ATD] von AtYourSide dient der automatischen Erstellung von Testfällen aus Anforderungen heraus. Die in natürlicher Sprache definierten funktionalen Systemanforderungen werden in Ursache-Wirkungs-Bäume transformiert und können in dieser Form erneut begutachtet und verändert werden. Wurden die Ursache-Wirkungs-Bäume in eine eindeutige und deterministische Form gebracht, so erstellt der ATD anhand dieser eine minimale Anzahl von Testfällen, die die Anforderungen vollkommen überdecken. Bei der Erzeugung der Testfälle kommt hier ein Optimierungsalgorithmus für neuronale Netze zum Einsatz.

Die Testfälle können manuell von einem Benutzer durchgeführt bzw. in ein anderes Tool zur automatischen Abarbeitung importiert werden. Somit bietet der ATD eine gute Unterstützung von Akzeptanz- und Systemtests, jedoch nicht bei der Durchführung, sondern nur bei der Erstellung.

Ein Vorzug des ATD ist seine Auslegung auf Teams in Form einer dreischichtigen Client/Server-Architektur. Dadurch ist es möglich, funktionale Anforderungen und damit einhergehend direkt die zu erstellenden Testfälle für das Projekt zu definieren. Ebenso wird die Konsistenz von Anforderungen und Testfällen automatisch gewahrt, so dass bei Änderungen einzelner Anforderungen nur ein Minimum an Anpassungen nötig ist.

6.3.2 AutoTester One

Bei AutoTester One [@AutoTester] handelt es sich um ein Testautomatisierungstool für die Durchführung funktionaler Tests von Windows-, Client/Server-, Host/Legacy- und Web-Software. Die Idee dabei ist, die Funktionalitäten der Software über die GUI, bzw. ein textbasiertes Frontend zu testen.

In Tabelle 6 wird ein für Testautomatisierungstools zur funktionalen Überprüfung über eine GUI typische Bewertung ersichtlich. Derartige Tools unterstützen den Nutzer gerade in den Aktivitäten der Testfallerstellung und Testdurchführung. Dabei wird der Automatisierung durch einmalige Aufzeichnung einer Benutzer-System-Interaktion Rechnung getragen, die anschließend beliebig oft wiederholt werden kann. Die Wiederholbarkeit der Testfälle ist jedoch meist nur garantiert, wenn kleinere Änderungen an der Benutzeroberfläche zwischen Testdurchführungen vorgenommen wurden. Ein weiterer Aspekt dieser Toolklasse ist die hauptsächliche Unterstützung von Systemtests, da im Allgemeinen davon auszugehen ist, dass die korrekte Interaktion erst nach Erstellung des Gesamtsystems sinnvoll testbar ist. Prinzipiell sind aber auch Integrations- und Akzeptanztests denkbar. Letztere Tests könnten etwa durch manuelle Abarbeitung erstellter Skripte erzeugt werden.

Auch AutoTester One bietet diese Aufnahme- und Abspielmöglichkeiten an. Die Benutzer-System-Interaktion wird in Form eines Skriptes in proprietärem Format aufgezeichnet und kann anschließend um Kontrollflussanweisungen erweitert werden. Das Produkt bietet darüber hinaus eine Wiederherstellungsfunktion, so dass versucht wird innerhalb eines Testfalls auftretende Fehler zu umgehen. AutoTester One ist nicht (wie viele andere Testtools dieser Art) nur für Anwendungen mit Benutzeroberfläche einsetzbar, sondern kann auch mit zeichenbasierten Anwendungen verwendet werden.

Weiterhin können mit AutoTester One Last- und Stresstests durchgeführt werden, um die Performance eines Systems unter Extremsituationen zu testen.

6.3.3 Functional Tester

Der Functional Tester [@FunctionalTester] von IBM Rational wird zum automatischen, funktionalen Testen von Java-, VB.NET und Web-Anwendungen eingesetzt. Testfälle können in Form von Benutzer-System-Interaktionen aufgezeichnet werden, sofern die Anwendung über eine Benutzeroberfläche verfügt. Die so entstehenden Skripte basieren auf Java bzw. Visual Basic. Damit ist es möglich, die Skripte mit Kontrollflussinformationen, z.B. für wiederholte Durchführung, zu versehen ohne eine proprietäre Skriptsprache zu erlernen. Trotz der Verwendung von Java bzw. Visual Ba-

sic muss man zu Beginn die mitgelieferte komplexe API erlernen, um die komplette Funktionalität des Functional Tester verwenden zu können.

Ein herausragendes Konzept des Functional Tester ist der Ansatz des datengetriebenen Testens, bei dem eine große Menge Daten definiert und für einen oder gleich mehrere Testfälle als Datenbasis dienen kann. Die Daten werden in sog. Data Pools organisiert. Das Auslesen der Daten wird in den Skripten selbst durchgeführt, in denen nur Variablen statt den zur Laufzeit eingesetzten Werten stehen.

Weiterhin verwendet der Functional Tester eine Methode, um robust gegen kleinere Änderungen einer Benutzeroberfläche zu sein. Dieses so genannte ScriptAssure-Konzept basiert auf Techniken der Musterübereinstimmung (Pattern-Matching).

6.3.4 GUITAR

GUITAR [@GUITAR] wurde an der Universität von Maryland entwickelt. Es handelt sich dabei um ein erweiterbares Framework zum Test von Applikationen mit Benutzeroberfläche. Für die Testfallerstellung und die Testdurchführung wird ein selbst definierter Prozess eingesetzt.

GUITAR unterstützt zurzeit nur Applikationen, die in Java oder für Windows geschrieben wurden. Das Testtool besteht in der Version, die direkt von der Universität Maryland angeboten wird, aus vier Modulen:

1. GUI-Ripper: Durch dieses Modul wird die Benutzeroberfläche in eine Baumstruktur von Darstellungskomponenten zerlegt
2. Ereignis-Flussgraph-Generator: Dieses Modul erzeugt bzw. verändert den teilautomatisch erstellten Ereignisfluss der Benutzeroberfläche
3. Testfallgenerator: GUITAR bietet drei Arten der Testfallerstellung: zufällig, strukturiert und manuell. Dabei werden bei den beiden erstgenannten Arten die Testfälle zumindest teilautomatisiert erzeugt.
4. Testdurchführungskomponente: Dieses Modul führt die Testfälle aus und gibt die Ergebnisse zurück. Dabei kann die Überdeckung des Programmcodes anhand der in Kapitel 2.4 dargestellten Arten von White-Box-Testing gemessen werden.

Das Produkt verfügt über ein Plugin-Konzept, mit dem der gesamte Testprozess auf ein Unternehmen angepasst werden kann. Ebenso werden verschiedene Testtypen unterstützt. Andererseits kann das Tool somit um neue Funktionalitäten erweitert werden.

Auch wenn GUITAR aus dem universitären Bereich stammt, so vereint es durchaus sinnvolle und vor allem anwendbare Automatisierungstechniken in

sich. Das oben erwähnte Plugin-Konzet ermöglicht neben der freien Erhältlichkeit des Programmcodes überdurchschnittliche Adaptivität und Funktionserweiterung.

6.3.5 LoadRunner

LoadRunner [@LoadRunner] von Mercury Interactive ist eines der marktführenden Tools im Bereich des automatisierten Performance-Testens. Mit dem Produkt kann ein Eindruck über so genannte Flaschenhälse des Systems, also Engpässe bei hoher Last, gewonnen werden. Dabei steht insbesondere die Früherkennung von lastkritischen Systemteilen während eines Entwicklungszyklus im Vordergrund.

Zur Durchführung von Lasttests kann das Produkt beliebig viele virtuelle Nutzer simulieren. Eine Verteilung der virtuellen Nutzer auf physische Ressourcen kann ebenfalls vorgenommen werden.

Nach der Durchführung der Lasttest wird die Analyseeinheit von LoadRunner benutzt, um einfache Sichten auf Endnutzer-, System- und Code-Level-Performance zu erzeugen. Letztere ist nur aufgrund der Integrationsmöglichkeit mit J2EE, VB.NET und Web Services möglich und kann nicht für beliebige Systeme eingesetzt werden.

LoadRunner bietet eine API an, um Skripte in Visual Basic oder Java zu erstellen. Testfälle können aber auch durch Aufzeichnen von beispielhaften Benutzer-System-Interaktionen erzeugt werden.

Durch die herausragende Marktstellung des Produktes kann davon ausgegangen werden, dass nachträgliche Ergänzungen um weitere bzw. Integration mit anderen Produkten des Herstellers leicht durchzuführen sind. LoadRunner kann beispielsweise neben WinRunner (siehe Kapitel 6.3.15) eingesetzt werden. Mercury garantiert hier eine einfache Austauschbarkeit der Skripte.

6.3.6 Netvantage Functional Tester

Bei diesem Programm der Firma Netvantage Technologies [@Netvantage] handelt es sich um ein einfaches Tool zum automatischen Test von Web-Applikationen. Dabei kann die Navigation über mehrere Web-Seiten korrekt und automatisch aufgezeichnet werden, die Verifikationsmöglichkeiten beschränken sich jedoch auf sehr einfache Anfragen, z.B. ob der HTML-Code einer Webseite eine bestimmte Zeichenkette beinhaltet. Zusätzlich lässt sich die Verwendung mehrerer Werte einer Variablen nur schwerlich automatisieren. Nur eine zufallsbedingte automatische Auswahl aus einer Liste möglicher Variablenwerte wird unterstützt. Nachteilig wirkt sich auch das Skriptformat aus, in dem Kontrollinformationen nur durch die vorgegebenen Konstrukte Verzweigung und Schleife unterstützt werden. Für die Testauswer-

tung wird ein proprietäres Ausgabeformat eingesetzt, das sich nur durch ein entsprechendes Teilprogramm lesen lässt.

Insgesamt lässt sich für das Produkt festhalten, dass es sich hauptsächlich für den Test statischer Web-Applikationen eignet. Die fehlende automatische Unterstützung von wiederholten Abläufen mit mehreren Variablenwerten wäre im Falle einer Anwendung, die dynamische Inhalte produziert, problematisch, da unterschiedliche Werte die Erzeugung neuer Testfälle nach sich ziehen würde.

6.3.7 PETA

Bei dem Produkt PETA [@PETA] der Firma verit Informationssysteme GmbH handelt es sich um ein Werkzeug zum automatischen Test von nachrichtenbasierten Anwendungen, z.B. Client/Server-Systeme, die über HTTP (Hypertext Transfer Protokoll) miteinander kommunizieren.

Mit PETA, das vollständig in die Entwicklungsumgebung Eclipse [@Eclipse] integriert ist, ist es möglich Testfälle für Nachrichten zu erstellen. Dabei können je nach Bedarf beliebig viele beteiligte Systeme durch PETA simuliert werden. Die Erstellung der Testfälle erfolgt hier nicht automatisch oder wird aus beispielhaften Interaktionen abgeleitet, sondern ist vom Benutzer durchzuführen. Das Format, in dem sowohl die Testfälle als auch die Nachrichtentypen vorliegen, ist XML (Extensible Markup Language) [@XML]. Durch die Verwendung einer speziellen XML-Sprache können entsprechende Konvertierungsmechanismen verwendet werden, um konkrete Nachrichten im gewünschten Format, wie z.B. HTML, zu erzeugen. Somit ist eine hohe Konfigurierbarkeit an die Bedürfnisse eines Nutzers möglich. Die Erstellung der Testfälle besteht nicht im Schreiben von XML-Dateien, sondern wird durch ein graphisches Frontend, mit dem die Testfälle in Form von Sequenzdiagrammen spezifiziert werden können, erheblich vereinfacht. PETA versteckt somit einen Großteil der Komplexität, die beim manuellen Schreiben von Testfällen in einer Skriptsprache entstehen würde, vor dem Nutzer. Zusätzlich unterstützt PETA auch das Fehlermanagement, indem die Abbildung von Änderungsanfragen (nach Auftreten eines Fehlers) und den zugehörigen Testfällen möglich ist.

Da PETA auch für service-orientierte Softwarearchitekturen (SOAs) eingesetzt werden kann, wird die Unterstützung von Integrationstests als sehr gut gewertet. Solche Systeme bestehen aus einzelnen Web Services [@W3C], die die kleinsten Einheiten einer SOA darstellen und zu komplexeren Services kombiniert werden können. Da eine Kommunikation zwischen Services durch Nachrichten in bestimmten Formaten realisiert wird, kann PETA somit Integrationstests in SOAs unterstützen.

Wegen der Verwendung der offenen Eclipse-Plattform als Entwicklungsumgebung wird zusätzlich eine hohe Erweiterbarkeit ermöglicht. Das von Ec-

lipse verwendete Plugin-Konzept kann mit einigen Einschränkungen auf PETA angewendet werden. Ebenso vorteilhaft ist der Einsatz von Eclipse-Hilfekonzepten, so dass die in überdurchschnittlichem Maße vorhandene Dokumentation leicht zugreifbar gemacht wird. Hiermit verbunden erklärt sich auch die hohe Wertung bei der Einarbeitungszeit, da ein Tutorial für den Einstieg in PETA inklusive Erweiterungskonzepten bereitgestellt wird.

6.3.8 QAWizard

Bei Seapines QAWizard [@QAWizard] handelt es sich um ein Testtool zur Erstellung, Durchführung und Auswertung von funktionalen Web- und Regressionstests. Das Programm kann eine typische Benutzer-System-Interaktion automatisch aufzeichnen. Die hierbei eingesetzte eigene Skriptsprache, deren Konstrukte über die Benutzeroberfläche zugreifbar sind, erlaubt es dem Nutzer anschließend in intuitiver Weise Logik in die Testdurchführung zu integrieren.

QAWizard versucht einen der Hauptnachteile anderer Web-Testtools ähnlicher Bauart bei der Aufzeichnung zu überwinden. Andere Tools identifizieren die Interaktionen meist anhand der Position oder des Aussehens des Elements auf der (Web-)Oberfläche. Anders geht QAWizard bei der Aufzeichnung vor, indem es die Elemente durch ihre Attribute, wie sie im HTML-Code zu finden sind, und durch das Vaterelement identifiziert. Nachteilig ist jedoch, dass sich solche Bindungen nicht automatisch erzeugen lassen, sondern per Hand spezifiziert werden müssen.

Das Tool eignet sich nicht nur für den Test von Web-Applikationen, sondern auch von Windows-Anwendungen, die über eine GUI verfügen.

Da der Datenexport in XML erfolgt können andere Tools (evtl. nach einer Transformation), die XML als Eingabeformat nutzen, die Testergebnisse, mit einbeziehen.

6.3.9 Rational Robot

Die Test-Software Robot [@RationalRobot] von IBM Rational dient dem funktionalen Testen von Anwendungen verschiedenster Art. So werden etwa Java- und Web-Anwendungen genauso unterstützt wie Anwendungen in Visual Basic.

Rational Robot besitzt große Vorteile gegenüber anderen Testautomatisierungstools beim Sprachumfang der eingesetzten Skriptsprache, der Testauswertung und durch eine gute Strukturierung der Benutzeroberfläche.

Die in Rational Robot eingesetzte Skriptsprache (SQABasic) ist eine leichte Einschränkung von Visual Basic, so dass es für Tester mit entsprechender Erfahrung nicht nötig ist, eine weitere proprietäre Programmiersprache zu er-

lernen. Zusätzlich lassen sich auch sehr komplexe Testfälle schreiben. Den meisten anderen Testtools fehlt eine derart umfangreiche Skriptsprache.

Die Testauswertung wird durch Rational Robot ebenfalls sehr gut unterstützt, da für die Testergebnisse verschiedenste statistische Auswertungen durchgeführt werden können. Eine Vergleichsmöglichkeit mit vorhergehenden Testergebnissen findet sich bei wenigen Konkurrenzherstellern.

Ein integriertes Plugin-Konzept sorgt für eine ausgezeichnete Erweiterbarkeit des Testtools. Natürlich erfordert eine solche Flexibilität zusätzliche Einarbeitungszeit. Rational Robot ist darüber hinaus wegen der Unterstützung der verschiedensten Sprachen überdurchschnittlich gut konfigurierbar.

Als nachteilig kann bei dieser Software neben dem Preis die sehr hohe Gesamtkomplexität verstanden werden, die eine lange Einarbeitungszeit notwendig macht.

6.3.10 TestComplete

TestComplete [@TestComplete] ist ein Testautomatisierungstool von Automated QA, das für funktionale Unit- und GUI-Tests eingesetzt werden kann. Solche Tests können unter gewissen Voraussetzungen auch mit White-Box-Tests kombiniert werden.

Interessant an diesem Produkt sind die verschiedenen Skriptsprachen, die an unterschiedliche Programmiersprachen angelehnt sind. Somit ist es Nutzern z.B. möglich ihre Java-Applikationen mit einer Delphi-Skriptsprache zu testen. Demnach ist die Einarbeitungszeit nicht so hoch wie bei anderen skriptbasierten Testtools, die ein proprietäres Format einsetzen.

Mit der TestComplete können Applikationen, die beispielsweise .NET von Microsoft einsetzen, mit White-Box-Tests versehen werden. Nachteilig ist jedoch, dass bei Anwendungen in einer Programmiersprache wie C++ viele Konstrukte nicht zugelassen sind. Dies liegt in der Verwendung einer Ausführungseinheit begründet, die nur bestimmte Datentypen zulässt, die mit dem OLE-Standard konform sind. Darüber hinaus können bei Anwendungen, die in Delphi geschrieben sind, nur Klassen- und keine Objektmethoden getestet werden.

Zusätzlich bietet TestComplete ein Plugin-Konzept an. Hieraus kann jedoch nicht auf eine sehr gute Erweiterbarkeit geschlossen werden, da Automated QA das Plugin-Konzept nicht komplett offen legt und damit eigene Funktionalitätserweiterungen kaum möglich sind. Nichtsdestotrotz ist Erweiterbarkeit und Konfigurierbarkeit in dieser Hinsicht in mittlerem Maße vorhanden, da die Verwendung von Plugins die Ausrichtung auf eine bestimmte Sprache ermöglicht.

6.3.11 TestSmith

Bei TestSmith [@TestSmith] von QualityForge handelt es sich um ein Testtool für Windows- und Web-Anwendungen. TestSmith bietet Funktionalitäten, wie etwa automatisches Aufzeichnen von Benutzerinteraktionen. Das Produkt verwendet hierbei jedoch ein proprietäres Format. Um komplexere Testfälle zu erstellen, als es über die Oberfläche von TestSmith möglich ist, existieren APIs für Java, C++ und C. Dabei wird jedoch weiterhin nativer Windows-Code eingesetzt, so dass eine Unterstützung von z.B. Linux-Anwendungen nicht möglich ist. Durch Verwendung der APIs können Variablen besser kontrolliert und komplexere Abläufe durchgeführt werden.

Eines der grundlegenden Konzepte von TestSmith ist das so genannte Fuzzy Matching. Dabei werden Elemente einer Web-Seite bei Durchführung eines Testfalls nicht auf genaue Übereinstimmung gestestet, sondern nur „unscharf“ miteinander verglichen. Mit einem „unscharfen“ Vergleich ist hierbei gemeint, dass nur Teile eines Elements für die Identifizierung herangezogen werden. Fuzzy Matching kann gerade bei dynamischen Inhalten gewinnbringend eingesetzt werden, gleichzeitig reicht aber eine solche „unscharfe“ Übereinstimmung nicht immer für eine erfolgreiche Testdurchführung aus. Da das eben beschriebene Konzept Nachteile bei der Testdurchführung und der Automatisierung von Systemtests besitzt, wurde das Tool in den entsprechenden Rubriken nur durchschnittlich bewertet.

Als kritisch erweist sich, dass Benutzerinteraktionen abhängig von der Bildschirmauflösung aufgezeichnet werden. Die Wiederverwendbarkeit von Testskripten auf verschiedenen Rechnern ist somit stark eingeschränkt.

6.3.12 Vermont HighTest Plus

HighTest Plus [@HighTestPlus] ist ein Tool von Vermont Creative Software, um Windows- und Web-Applikationen automatisch testen zu können. Letzteres ist durch die Integration mit dem Internet Explorer von Microsoft möglich.

HighTest Plus bietet zwei Möglichkeiten, Testfälle zu erstellen. Zum einen durch Aufzeichnen der Maus- und Tastaturoperationen, die dann in eine proprietäre Skriptsprache übersetzt werden, und zum anderen durch Schreiben der Skripte von Hand. Es ist möglich vorher erstellte Testfälle, die im Binärformat von HighTest Plus erstellt wurden als Textdateien, die den Skriptcode in lesbarer Form enthalten, zu im- bzw. exportieren. Damit können nachträgliche Änderungen an den Testfällen durchgeführt werden.

Nachteilig erweist sich die Struktur der Skriptsprache beim Wiederabspielen der Testfälle, da Mausoperationen anhand von relativen Bildschirmpositionen gespeichert werden. Zwar existiert ein Mechanismus für die Abbildung von Mausaktionen, wie etwa ein Klick auf eine Check-Box, jedoch funktioniert er nicht in jedem Fall.

Ein interessantes Feature des Produktes ist die Möglichkeit, Bereiche eines Fensters von der Überprüfung auszunehmen. Somit können sich ändernde Daten wie etwa Datumsinformationen von einer Überprüfung auf genaue Übereinstimmung ausgenommen werden. Dies zeigt zum anderen einen großen Nachteil des Programms auf. Stark dynamische Testfälle wie sie etwa beim Test einer dynamischen HTML-Seite auftreten sind nicht sinnvoll durchzuführen.

Zusätzlich bietet HightTest Plus einen sehr einfachen Breakpoint-Mechanismus bzw. eine schrittweise Abarbeitung der Testfälle.

6.3.13 WebInject

WebInject [@WebInject] ist ein frei erhältliches Web- und Performance-Testtool, das unter der GNU Public License (GPL) steht. Produkte unter dieser Lizenz sind nicht nur frei zugängig, sondern ihr Quellcode ist ebenfalls frei zugänglich und darf unter gewissen Voraussetzungen geändert werden. Somit sind maximale Konfigurierbarkeit und Erweiterbarkeit für WebInject generell gewährleistet.

In WebInject werden Testfälle in XML formuliert. Die XML-Dateien müssen per Hand geschrieben werden, so dass eine Automatisierung der Testfallerstellung nicht möglich ist. Es können jedoch ganze Testsuiten definiert werden, deren Testfälle automatisch nacheinander ausgeführt werden. Die Testmöglichkeiten sind in WebInject etwas eingeschränkt. Für Verifikationen können reguläre Ausdrücke eingesetzt werden, so dass insbesondere HTTP-Sessions und sichere Verbindungen relativ einfach einbezogen werden können. Nachteilig ist die fehlende Unterstützung von mehreren Werten für eine Variable. Will ein Nutzer beispielsweise mehrere Benutzernamen-Passwort-Paare bei der Systemanmeldung testen, so ist der Ablauf des Testskripts dabei bis auf die Eingabe von Benutzername und Passwort identisch. In WebInject werden zwar Variablen in Testskripten unterstützt, jedoch müssen sie per Hand geändert werden. Alternativ kann man durch ein selbst geschriebenes Programm automatisch entsprechende Skripte erzeugen, allerdings müssten solche Programme für verschiedene Testfälle wiederum angepasst werden. Daher ist die Unterstützung automatischer Systemtests als eher gering einzuschätzen.

WebInject wurde nicht nur für die automatische Durchführung von Testfällen erstellt, sondern kann auch für Monitoring-Aufgaben in Zusammenarbeit mit anderen frei erhältlichen Tools eingesetzt werden. Zwar ist die Zusammenarbeit mit anderen Testtools von Haus aus nur mäßig, kann aber mit etwas Aufwand, da XML als Datenimport- und Datenexportformat verwendet wird, selbst realisiert werden.

Prinzipiell bietet WebInject einen guten Einstiegspunkt, sofern genügend Zeit und Aufwand für die Anpassung zur Verfügung steht.

6.3.14 WebKing

WebKing [@WebKing] von Parasoft ist ein Web Testing-Produkt, das funktionale, Sicherheits- und Performance-Tests unterstützt. Wie bei den meisten der hier vorgestellten Web-Testtools kann auch in WebKing eine typische Benutzer-System-Interaktion aufgezeichnet werden. Hierbei werden auch Sprachkonstrukte in Java, JavaScript und Python in den Web-Seiten erkannt. WebKing verwendet zur Aufzeichnung von Interaktionen mit dem System ein eigenes Format und bietet keine dazugehörige Skriptsprache an. Die Dateiformate zur Ausgabe der Testergebnisse umfassen HTML, XML und einfachen Text. Durch die Spezifikation von Variablen und zugehörigen Variablenwerten kann eine Vielzahl von Interaktionen automatisiert getestet werden.

Zwar unterstützt WebKing wie eben erwähnt funktionale Tests, jedoch ist es eher für Performance- und Sicherheitstests, sowie die statische Analyse von Webseiten geeignet. Gerade bei Load-Tests werden die Vorzüge von WebKing offensichtlich, da es die Resultate in einer eingängigen aber trotzdem detaillierten Weise darstellt. Nutzern ist es möglich einen typischen Lastverlauf ihres Systems anzugeben, der dann von WebKing bei der Simulation der virtuellen Clients berücksichtigt wird.

6.3.15 WinRunner

WinRunner [@WinRunner] von Mercury besitzt einen hohen Markanteil bei automatischen Funktions- und Regressionstests für Client/Server-Systeme, was mitunter durch die herausragende Stellung von Mercury als Marktführer im gesamten Bereich der Testtools zu erklären ist. Die Möglichkeit der Integration von WinRunner in andere Produkte von Mercury wie etwa Quick Test Professional hat zusätzlich die Verbreitung von WinRunner erhöht.

Als herausstechendes Merkmal von WinRunner kann die Anpassbarkeit der automatisch aufgezeichneten Skripte, die in einem eigenen Format vorliegen, bezeichnet werden. Durch die Verwendung eines proprietären Formats wird leider auch die Bindung an andere Mercury-Produkte weiter erhöht.

Aufgrund der weiten Verbreitung existiert eine große Nutzerbasis (Community), die sich in entsprechenden Foren austauscht und somit Hilfestellung bei Problemen geben kann.

Wegen der hohen Komplexität des Programms, die durch den großen Funktionsumfang entsteht, und der damit verbundenen hohen Einarbeitungszeit in die Konstrukte der Skriptsprache muss viel Zeit bei der Einführung des Tools zur Verfügung stehen. Dieser Sachverhalt erklärt die schlechte Wertung der entsprechenden Eigenschaften.

Bei der Testauswertung handelt es sich um ein weiteres Kernkonzept von WinRunner, da Testergebnisse in HTML exportiert oder durch den Einsatz

von Quick Test Professional in das Fehlermanagement integriert werden können.

7 Auswahl eines Testtools für die Projektpartner

In diesem Kapitel sollen Testtools für die Projektpartner ausgewählt werden. Dazu wird die Nutzwertanalyse herangezogen, die bereits in Kapitel 3 erläutert wurde. Es ist im Folgenden darauf zu achten, dass nur Tools, wie sie zu Beginn von Kapitel 6.3 definiert wurden, in den Bewertungsprozess miteinbezogen werden. Es ist wenig hilfreich, individuelle Tools mit ganzen Tool-Suiten zu vergleichen. Das Problem dabei wäre die im Vergleich hohe Bewertung von Tool-Suiten, da sie meist mehr Testaktivitäten und Testphasen als individuelle Tools abdecken. Prinzipiell lässt sich der im weiteren Verlauf vorgestellte Bewertungsprozess auch für den Vergleich von Tool-Suiten untereinander anwenden. Um dies auch in einer geeigneten Weise durchführen zu können, müssen evtl. weitere Anforderungskriterien hinzugefügt werden, damit die Unterschiede zwischen Tool-Suiten besser sichtbar werden.

In den Tabellen in Kapitel 5 haben die Projektpartner bereits eine Bewertung der Wichtigkeit ihrer Anforderungen anhand der einzelnen Fragen aus Kapitel 4 vorgenommen. Durch diese Einstufung erhält man eine Gewichtung für die Nutzwertanalyse. Verschiedene Testtools wurden in Tabelle 6 nach ihren Eigenschaften, die eine starke Beziehung zu den Anforderungen aufweisen, beurteilt. Diese beiden Bewertungen erfolgen zunächst durch die bekannten Zeichen bis die Zahlenwerte (0 = unwichtig, 1 = teilweise wichtig, 2 = wichtig) für die Nutzwertanalyse eingesetzt werden. Die beiden Gewichtungen werden pro Frage bzw. Kriterium miteinander multipliziert und schließlich aufsummiert. Letztlich werden sie in Relation mit dem Höchstwert bei völliger Erfüllung aller Anforderungen gesetzt und man erhält den prozentualen Erfüllungsgrad des Tools bzgl. der Anforderungen.

Allerdings stellen die so erhaltenen Nutzwerte nur eine Hilfestellung dar. Nicht außer Acht zu lassen sind die Freitextfelder, die meist Ausschlusskriterien darstellen. Daher sollten erst diese für eine Vorauswahl berücksichtigt werden. Anschließend können die verbleibenden Tools mittels der Nutzwertanalyse ausgewertet werden. Im Allgemeinen stellt das Tool mit der höchsten so erhaltenen Bewertung das am besten geeignete für das Unternehmen dar.

7.1 WIKON

In diesem Kapitel wird die Auswahl eines Testautomatisierungstools für den Projektpartner WIKON beschrieben. Die Anforderungen an ein ideales Testtool aus der Sicht von WIKON finden sich in Kapitel 5.1.

Bei der Auswahl eines geeigneten Testtools kommt den Freitextfeldern im Allgemeinen eine sehr große Bedeutung zu, beispielsweise kommen Tools

nicht für einen Kauf in Frage, sofern sie nicht für die in den Anforderungen definierten Testtypen geeignet sind. Meist stellen die in Freitextfeldern angegebenen Tooleigenschaften Ausschlusskriterien für Tools dar und sollten daher vor der eigentlichen Nutzwertanalyse untersucht werden.

Im hier beschriebenen Fall wird folgendermaßen für die Toolauswahl vorgegangen:

1. Wähle anhand der Anforderungen, die in Freitext formuliert sind, die Tools aus, die eine möglichst große Überdeckung mit den Anforderungen zeigen.
2. Führe die Nutzwertanalyse der ausgewählten Tools nach dem Verfahren aus Kapitel 3 durch.
3. Wähle von den Tools das mit dem höchsten Nutzwert aus. Im Falle einer großen Anzahl bewerteter Tools, bietet sich ein weiterer Vergleich der Tools mit den höchsten Nutzwerten anhand der detaillierten Toolbeschreibungen an. Dieser Arbeitsschritt kann als abschließende Vergewisserung verstanden werden.

Im ersten Schritt fallen für eine Vorauswahl alle Tools heraus, die weder funktionale noch Web-Tests unterstützen. Da der Fokus bei LoadRunner auf Perfomance-Tests liegt, wird es genauso wie GUITAR, das für GUI-Testing aber nicht ohne weiteres für Web-Testing geeignet ist, von einer weiteren Betrachtung ausgenommen. Zusätzlich wird auch TestComplete, das nicht unmittelbar für Web-Tests verwendet werden kann, nicht weiter betrachtet. Es ist zu beachten, dass diese Tools einen sehr hohen Nutzwert für WIKON hätten. Dies liegt daran, dass beispielsweise GUITAR, wegen seiner allgemeineren Ausrichtung auf verschiedene Testaktivitäten eine höhere Bewertung erzielt, als es reinen Web-Testtools möglich ist. Für Tools, bei denen kein konkreter Testtyp angegeben ist, muss anhand der detaillierten Beschreibung überprüft werden, ob es sich für das Unternehmen eignet. In diesem Teilschritt wird der Automated Test Designer ausgeschlossen, da er nur für die automatische Generierung von Testfällen eingesetzt werden kann.

Die Menge potentiell geeigneter Tools kann nach dem gleichen Verfahren durch Analyse weiterer Freitextfelder weiter eingeschränkt werden, was im Folgenden jedoch nicht gemacht wird. Hier werden stattdessen die sechs verbleibenden Tools Functional Tester, PETA, QAWizard, Rational Robot, WinRunner sowie WebInject im Detail zur Erklärung der weiteren Prozessschritte analysiert. Die Nutzwerte dieser Tools finden sich in Tabelle 7. Die angegebenen Zahlenwerte sind Resultate der Durchführung der Nutzwertanalyse. Im Folgenden wird der Rechenweg genauer beschrieben.

Tools	Nutzwert
FunctionalTester	58,3%
PETA	76,4%
QA Wizard	59,7%
Rational Robot	61,1%
WebInject	54,2%
WinRunner	58,3%

Tabelle 7: Nutzwerte der Tools

Tabelle 8 zeigt die Nutzwertanalyse der eben genannten Tools im Detail. Die letzte Spalte der Tabelle gibt die Maximalbewertung eines Tools im Hinblick auf die von WIKON definierten Anforderungen wieder. Zu deren Berechnung wurden die Anforderungen von WIKON in Form von Zahlenwerten in der ersten Spalte vermerkt. Diese sind jeweils mit zwei (der Maximalbewertung einer Tooleigenschaft) zu multiplizieren und anschließend aufzusummieren. Damit erhält man eine Maximalbewertung eines Tools für WIKON, die in der letzten Spalte in der Zeile „Summe“ mit 72 angegeben ist. Dabei handelt es sich um den größten Wert, den ein Tool erzielen kann.

Für jedes der sechs untersuchten Tools existieren zwei Spalten, wobei die erste die Bewertung der Tooleigenschaften aus Kapitel 6.2 in Form von Zahlenwerten enthält. Die zweite Spalte eines Tools enthält das Produkt des Eigenschaftswertes (erste Spalte eines Tools) mit dem Anforderungswert von WIKON (erste Spalte insgesamt). Beispielsweise steht in der zweiten Spalte des Functional Tester in der Zeile „Testfallerstellung“ eine zwei. Diese ist das Produkt aus dem Anforderungswert von WIKON (eins) und dem Eigenschaftswert des Tools (zwei).

Für den prozentualen Erfüllungsgrad der Anforderungen muss schließlich die Summe der Produkte aus Anforderungs- und Eigenschaftswerten durch die Maximalbewertung dividiert werden. Im Falle des Functional Tester ergibt sich beispielsweise $41 / 72 \approx 56.9\%$.

Eigenschaften /Tools			Bewertungsfaktor Wikon	FunctionalTester	Auswertung	PETA	Auswertung	QA Wizard	Auswertung	Rational Robot	Auswertung	WebInject	Auswertung	WinRunner	Auswertung	Höchstpunkte
Allgemeine Eigenschaften	Testaktivitäten	Testplanung	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		Testfallerstellung	1	2	2	2	2	2	2	2	2	2	2	2	2	2
		Testfallerstellung automatisierbar	1	2	2	1	1	2	2	2	2	2	2	1	1	2
		Testdurchführung	2	2	4	2	4	2	4	2	4	2	4	2	4	4
		Testdurchführung automatisierbar	2	2	4	2	4	2	4	2	4	2	4	2	4	4
		Testauswertung	2	2	4	2	4	2	4	2	4	2	4	2	4	4
		Fehlermanagement	1	1	1	1	1	0	0	1	1	0	0	0	0	2
		Fehlermanagement automatisierbar	1	0	0	0	0	0	0	1	1	0	0	0	0	2
	Test-technik	White-Box-Test	2	1	2	0	0	0	0	0	0	0	0	0	0	4
		BIT	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Testphasen	Unittest	1	1	1	0	0	0	0	0	0	0	0	0	0	2
		Unittests automatisierbar	2	1	2	0	0	0	0	0	0	0	0	0	0	4
		Integrationstest	2	0	0	2	4	1	2	0	0	0	0	0	0	4
		Integrationstest automatisierbar	2	0	0	2	4	1	2	0	0	0	0	0	0	4
		Systemtest	2	1	2	2	4	2	4	2	4	2	4	2	4	4
		Systemtest automatisierbar	2	1	2	2	4	2	4	2	4	2	4	2	4	4
		Installationstest	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		Installationstest automatisierbar	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		Akzeptanztest	0	1	0	0	0	0	0	0	0	0	0	1	0	0
Spezielle Eigenschaften	Weitere Eigenschaften	Erweiterbarkeit	2	0	0	1	2	1	2	2	4	1	2	1	2	4
		Konfigurierbarkeit	2	0	0	2	4	0	0	2	4	2	4	2	4	4
		Bedienung des Tools	2	2	4	2	4	2	4	2	4	1	2	2	4	4
		Grafisches Frontend	1	2	2	2	2	2	2	2	2	1	1	2	2	2
		Einarbeitungszeit	2	1	2	2	4	0	0	0	0	2	4	1	2	4
		Dokumentation	2	2	4	2	4	2	4	1	2	1	2	1	2	4
		Schulung	1	2	2	1	1	2	2	2	2	0	0	2	2	2
		Telephonischer Support	1	2	2	2	2	1	1	0	0	0	0	1	1	2
		Community support	0	2	0	0	0	0	0	1	0	1	0	2	0	0
		Summe			42		55		43		44		39		42	72
		Nutzwert		**58,3%**		**76,4%**		**59,7%**		**61,1%**		**54,2%**		**58,3%**		

Tabelle 8: Nutzwertanalyse WIKON

Nun sollen die ausgewählten Tools bzgl. ihrer Eignung für WIKON mittels der detaillierten Toolbeschreibungen aus Kapitel 6.3 bewertet werden. Dieses Vorgehen entspricht dem dritten Schritt des Auswahlprozesses, der weiter oben beschrieben wurde.

WebInject hat einen relativ geringen Nutzwert, besitzt aber einige Eigenschaften, die für WIKON interessant sind. WebInject bietet neben der Definition von Testfällen in XML eine hohe Konfigurierbarkeit und Erweiterbarkeit, sowie ein sehr gutes Preis-Leistungsverhältnis, da es unter der GNU Public License steht. Jedoch ist der Automatisierungsgrad für die Durchführung und Erstellung von Systemtests schwach ausgeprägt, da der Benutzer Testfälle selbst schreiben muss. Die zur Definition von Testfällen von WebInject verwendete XML-Sprache ist für WIKON nicht mächtig genug, da das automati-

sche Testen vieler Testfälle mit nur leicht geänderten Werten nicht unterstützt wird. Somit kann WebInject als Kandidat ausgeschlossen werden.

Als Tool mit einer etwas bessere Bewertung erweist sich der Functional Tester von IBM, der ebenfalls mit 4120$ schon der gehobenen Preisklasse zuzuordnen ist. Zwar bietet das Produkt einige scheinbare Vorzüge für WIKON, wie etwa die Unterstützung von Unit-Tests. Es muss jedoch beachtet werden, dass der Projektpartner keine der vom Tool unterstützten Sprachen bei der Implementierung des Server-Systems eingesetzt hat. Dies alleine kann schon als Ausschlusskriterium verstanden werden.

Genauso wie der Functional Tester wurde Mercurys WinRunner bewertet. Das gute Abschneiden lässt sich neben der Unterstützung eines hohen Automatisierungsgrades während der Testdurchführung und Testfallerstellung mit der weiten Verbreitung und dem entsprechendem Community-Support als marktbeherrschendes Testautomatisierungstool begründen. Gegen eine Auswahl spricht alleine schon der sehr hohe Preis von ca. 6000$. Kleine- und mittelständische Unternehmen wollen bzw. können meist nicht so viel Geld für die Anschaffung eines Testtools aufbringen.

Noch besser wurde der QAWizard bewertet. Seine Art, Benutzer-System-Interaktionen auf „Objektbasis“ und nicht über Informationen wie Mauspositionen aufzuzeichnen ist einer der großen Vorteile des Tools. Nachteilig ist jedoch die Beschränkung auf HTML bzw. .NET, wobei letzteres nicht von WIKON eingesetzt wird und ersteres nicht das einzige Nachrichtenformat darstellt. Ebenso spielt der hohe Preis eine Rolle, so dass dieses Tool nicht für WIKON zu empfehlen ist.

Auf Platz zwei des Ratings findet sich Rational Robot wieder. Interessant ist der Einsatz von Rational Robot für HTML. Wie QAWizard ist er aber nicht einfach auf andere Nachrichtenformate anpassbar und besitzt ebenso den Nachteil eines sehr hohen Preises.

Das Testtool PETA der Firma verit Informationssysteme GmbH besitzt die höchste Bewertung der hier betrachteten Tools und wird als Testtool für WIKON vorgeschlagen. Der Grund für die hohe Bewertung von PETA liegt in den Bereichen der Testausführung, Erweiter- und Konfigurierbarkeit, sowie der kurzen Einarbeitungszeit. Letztere hängt auch mit dem graphischen Frontend und der guten Dokumentation zusammen. Auch wenn die Testfallerstellung mit PETA nicht vollautomatisch durchgeführt werden kann, bietet die graphische Darstellung des Nachrichtenaustauschs einen Vorteil gegenüber einfachen Aufnahme-Programmen. Ebenso trägt die Unterstützung von Integrationstests gerade für Web Services, die auch von WIKON eingesetzt werden, zur hohen Bewertung bei.

7.2 market maker

Da market maker sehr spezielle Anforderungen (Kapitel 5.2) an ein Testtool besitzt, gestaltet sich die Suche schwierig. Gerade die Anforderungen bzgl. der BIT und der Installationstests erweisen sich als von Tools kaum unterstützt.

Insbesondere die Toolunterstützung von BIT wäre für market maker sehr wichtig, weil es damit möglich wäre die Systemfunktionalität nach der Installation zu testen, indem das System beispielsweise mit einer bestimmten Option zur Aktivierung der Tests gestartet werden könnte. Ein Beispiel für einen derartigen Built-In-Test wäre die Überprüfung, ob die Datenbank korrekt angebunden und ansprechbar ist. Andererseits könnten solche Built-In-Testtools ebenso für Integrationstests eingesetzt werden.

Beim Installationstest kann man zwischen den Tests vor der Installation und den funktionalen Tests nach der Installation wie in Kapitel 2.2 beschrieben unterscheiden. Für die erste Gruppe der Tests finden sich einige Tools auf dem Markt, die jedoch nur einfache Checking-Aufgaben, wie etwa auf Existenz eines Verzeichnisses, übernehmen können. Sie wurden wegen ihres sehr beschränkten Funktionsumfanges nicht in Kapitel 6.2 miteinbezogen. Derartige Tools sind nicht alleine für market maker geeignet, da das Unternehmen zusätzliche, sehr spezielle Überprüfungen, wie etwa bzgl. geschriebener Registry-Einträge von Windows, durchführen muss. Die zweite Gruppe der Tests, also „Post-Installationstests“, hängen stark von der aktuellen Systemarchitektur der zu testenden Applikation ab und werden zurzeit von keinem Tool unterstützt. Unternehmen, die solche Überprüfungen durchführen wollen, greifen daher meist auf Eigenimplementierungen zurück.

Bei der Toolauswahl für market maker wird ähnlich wie für den Projektpartner WIKON vorgegangen. Hier bietet es sich jedoch für eine erste Vorauswahl an nicht die Testtypen sondern die unterstützen Sprachen zu betrachten. Da market maker ein Testtool benötigt, das u.a. für Delphi geeignet ist, fallen alle bewerteten Tools bis auf TestComplete heraus, da sie diese Anforderung nicht erfüllen können.

Im Folgenden wird wegen der Existenz nur eines potentiell geeigneten Tools für market maker auf eine Nutzwertanalyse verzichtet und direkt die Eignung anhand der detaillierten Beschreibung untersucht.

TestComplete kann nicht von market maker eingesetzt werden, da es Installations- und Integrationstests nicht ausreichend unterstützt. Ebenso negativ wirkt sich die fehlende Unterstützung von BIT aus.

Für market maker bietet sich somit eine Eigenentwicklung für die Installationstests und Built-In-Tests an, da ein derzeit auf dem Markt erhältliches Testtool die gestellten Anforderungen nicht erfüllen kann. Eine Eigenent-

wicklung hat darüber hinaus den Vorteil, dass sie gut in den Bereichen der Funktionalität und der Testauswertung kontrolliert werden kann.

8 Zusammenfassung

In diesem Report wurden zunächst allgemeine Begrifflichkeiten im Bereich des Software-Testens und der Testautomatisierung erläutert (Kapitel 2). In Kapitel 3 wurde anschließend eine Nutzwertanalyse vorgestellt, die im weiteren Verlauf zur Bewertung von Testautomatisierungstools im Hinblick auf spezifizierte Anforderungen (Kapitel 4) verwendet wurde. Daran anknüpfend wurden in Kapitel 5 die konkreten Anforderungen nach dem vorgestellten Schema von den Projektpartnern WIKON und market maker definiert. Anschließend wurden in Kapitel 6 die Ergebnisse einer Marktanalyse vorgestellt, in der verschiedene Tools nach ihren Eigenschaften bewertet wurden. Letztlich wurden in Kapitel 7 ausgehend von der Marktanalyse mit einem Bewertungsverfahren entsprechende Testtools für die Projektpartner ausgewählt. Hierbei konnte für WIKON ein geeignetes Tool bestimmt werden. Market maker hat im Gegensatz dazu zu spezielle Anforderungen, die zurzeit noch von keinem Tool unterstützt werden. Der Grund hierfür lag u.a. in der fehlenden Toolunterstützung von Built-In- und Installationstests. Da kein Tool für market maker ausgewählt werden konnte, wurde eine Eigenentwicklung empfohlen.

Der Report zeigte einen verständlichen und nachvollziehbaren Weg auf, um aufgrund von Anforderungen geeignete Testautomatisierungstools auszuwählen.

In diesem Report wurden bisher nur einzelne Tools betrachtet. Der Markt bietet jedoch eine Vielzahl ganzer Tool-Zusammenstellungen an, die ganzheitliche Lösungen für einen durchgängigen Testprozess im Unternehmen bieten. Solche Tool-Suiten könnten durchaus in die Bewertung mit einbezogen werden, allerdings sollten im Allgemeinen nur Tools mit Tools und Tool-Suiten mit Tool-Suiten verglichen werden.

Literatur

[AOS04] Ove Armbrust, Michael Ochs, Björn Snoek: Stand der Forschung von Software-Tests und deren Automatisierung (State-of-the-Art Report) URL:www.iese.fraunhofer.de/pdf_files/iese-068_04.pdf, IESE-Report Nr. 068.04/D, Kaiserslautern, 2004.

[AOS04-2] Ove Armbrust, Michael Ochs, Björn Snoek: Stand der Forschung von Software-Tests und deren Automatisierung – Interviews und Online-Umfrage (State-of-Practice Report) URL:http://publica.fhg.de/documents/N-24381.html, IESE-Report Nr. 093.04/D, Kaiserslautern, 2004.

[CBC01] Robert Culbertson, Chris Brown, Gary Cobb: Rapid Testing, Prentice Hall, 2001

[Comp+01] Component+, Built-In-Testing for Component-based Development: URL: http://www.component-plus.org, Project Technical Report, 2001

[Ligg02] Peter Liggesmeyer: Software Qualität: Testen, Analysieren und Verifizieren von Software, Spektrum Akademischer Verlag, Heidelberg, 2002

[SL03] Andread Spillner, Tilo Linz: Basiswissen Softwaretest, Dpunkt Verlag, Heidelberg, 2003

[@ATD] AtYourSide Automated Test Designer. http://www.atyoursideconsulting.com/our_products.htm, last visited: 06.01.2006

[@AutoTester] AutoTester ONE. http://www.autotester.com/content/soft_at1.htm, last visited: 06.01.2006

[@Eclipse] Eclipse Integrated Development Platform. http://www.eclipse.org/, last visited: 06.01.2006

[@FunctionalTester] FunctionalTester. http://www-306.ibm.com/software/awdtools/tester/functional/index.html, last visited: 03.01.2006

[@GUITAR] GUI Testing Framework,
http://www.cs.umd.edu/~atif/GUITARWeb/, last visited: 03.01.2006

[@LoadRunner] LoadRunner.
http://www.mercury.com/us/products/performance-center/, last visited: 06.01.2006

[@Netvantage] Netvantage Functional Tester, last visited: 06.01.2006
http://www.netvantagetech.com/FunctionalTester/

[@PETA] The PETA Product Platform.
http://www.verit.de/products/index.html, last visited: 06.01.2006

[@QAWizard] Seapine QA Wizard.
http://www.seapine.com/qawizard.html, last visited: 06.01.2006

[@RationalRobot] Rational Robot. http://www-306.ibm.com/software/awdtools/tester/robot/index.html, last visited: 06.01.2006

[@RLRh-Pf] Research Lab Rheinland-Pfalz: Testen und Testautomatisierung. http://www.kmu.rlp-labs.de/, last visited: 06.01.2006

[@TestComplete] TestComplete Standard Edition.
http://www.automatedqa.com/products/testcomplete/index.asp, last visited: 06.01.2006

[@TestSmith] QualityForge TestSmith.
http://qualityforge.com/testsmith/index.html, last visited: 06.01.2006

[@UniFlensburg] Universität Flensburg:Nutzwerte messen.
http://www.uniflensburg.de/mathe/zero/veranst/anwmod/kap1/Links/anw_kap1_lnk_6.html, last visited: 13.12.2005, Flensburg, 2001.

[@HighTestPlus] Vermont HighTest Plus.
http://www.vtsoft.com/vcsproducts/index.html, last visited: 06.01.2006

[@SWKompetenz] Virtuelles Software Engineering Kompetenzzentrum (Grundlegender Testprozess). http://www.software-kompetenz.de/?9725, last visited: 06.01.2006

[@W3C] W3C Web Services Activity. http://www.w3c.org/, last visited: 06.01.2006

[@WebInject] WebInject – Web/HTTP Test Tool.
http://www.webinject.org/, last visited: 03.01.2006

[@WebKing] Parasoft WebKing.
http://www.parasoft.com/jsp/products/home.jsp?product=WebKing&itemId=99, last visited: 03.01.2006

[@WinRunner] Mercury WinRunner.
http://www.mercury.com/us/products/quality-center/, last visited: 06.01.2006

[@XML] Extensible Markup Language. http://www.w3.org/XML/, last visited: 03.01.2006

Dokumentinformation

Titel:	Research Lab Rheinland-Pfalz Testen und Testautomatisierung: Anforderungen an Testwerkzeuge und Marktstudie
Datum:	31.12.2005
Report:	IESE-131.05/D
Status:	Final
Klassifikation:	Öffentlich