Diplomarbeit

Erstellung eines Expertensystems zur kategoriebasierten Auswahl von Performancetest-werkzeugen in verteilten Umgebungen

Martin Grohs
Fachhochschule Brandenburg

DLR

Fachhochschule Brandenburg
Informatik und Medien
Digitale Medien
Prof. Dr. Gabriele Schmidt

Diplomarbeit

„Erstellung eines Expertensystems zur kategoriebasierten Auswahl von Performancetestwerkzeugen in verteilten Umgebungen“

„Creation of an Expert-System for a Category-Based Selection of Performance-Testwerktools in distributed Environments”

Autor: Martin Grohs
Matrikelnummer: 20022066
Studiengang: Digitale Medien

Referent: Frau Professor Doktor Gabriele Schmidt
Co Referent: Herr Dipl.-Phys. Frank Dannemann

Vorgelegt im: Wintersemester 2007/08
Am: 20.12.2007

Vorwort und Danksagung

Die vorliegende Arbeit entstand aus der Idee Performancemessung für eine bestimmte Software durchzuführen. Durch die Suche nach einem geeigneten Testwerkzeug und der Schwierigkeit sich mit unbekannter zu testender Software auseinander zu setzen, ergab sich die Idee, eine Heuristik für die Auswahl von Performancetestwerkzeugen zu schreiben. Diese Idee wurde von Woche zu Woche verfeinert, bis eine Diplomarbeit mit dem Thema: „Erstellung eines Expertensystems zur kategoriebasierten Auswahl von Performancetestwerkzeugen in verteilten Umgebungen“ entstand. Diese Arbeit soll Softwaretestern eine Möglichkeit bieten, auf einem schnellen Weg, ein passendes Performancetestwerkzeug für die zu testende Software oder das zu testende System zu finden.
Diese Arbeit wurde im Rahmen eines Vertrages als studentische Hilfskraft im Forschungszentrum der Bundesrepublik Deutschland für Luft- und Raumfahrt, für die Einrichtung Simulations- und Softwaretechnik (SC) und deren Abteilungen verteilte Komponenten und Qualitätssicherung, geschrieben.
Meinen Dank möchte ich an dieser Stelle besonders Frau Prof. Dr. Gabriele Schmidt, sowie Herrn Frank Dannemann aussprechen, die mich während der Diplomarbeit von Hochschul- und Einrichtungsseite sehr gut betreuten. Weiterhin möchte ich mich bei dem Entwicklerteam von D3Web der Universität Würzburg und beim Herrn Eldar Sultanov von der Berliner Producto AG für die gute Zusammenarbeit bedanken. Bedanken möchte ich mich auch bei den Mitarbeitern der Einrichtung Simulations- und Softwaretechnik für die Ratschläge, Diskussionen und Bereitstellung aller notwendigen Mittel um diese Arbeit möglich zu machen. Natürlich bedanke ich mich auch bei allen Professoren und Lehrbeauftragten der Fachhochschule Brandenburg für die Ausbildung und Betreuung während des Studiums.
Weiterhin möchte ich mich bei meinen Kommilitonen Christoph, Basti, Marcus, Jan und vielen Anderen für die schöne Studienzeit bedanken.
Mein größter Dank jedoch gilt meinen Eltern, welche mich bis zum heutigen Zeitpunkt in allen Ideen und Vorhaben unterstützt haben und ohne sie die Ausbildung, von der Grundschule bis zum Abschluss des Studiums, nicht möglich gewesen wäre.

Erklärung

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Diplomarbeit selbständig und ohne unzulässige fremde Hilfe angefertigt habe. Die verwendeten Quellen und Hilfsmittel sind vollständig zitiert.

Brandenburg, den 20. Dezember 2007

Martin Grohs
Huttenstrasse 11
38114 Braunschweig

Inhaltsverzeichnis

KAPITEL 1

Einleitung

1.1 Motivation

Um die Qualität einer Software zu testen, gibt es verschiedene Verfahren. Die Qualitätssicherung beschäftigt sich mit diesen Verfahren und stellt sicher, dass die entwickelte Software dem geforderten Qualitätsniveau entspricht. Im ISO 9000 – Ansatz steht, dass die Qualitätssicherung dazu dient, eine festgelegte Qualität zu erfüllen und nicht um diese zu optimieren (vgl. [Ba98] S.327 ff). Die Testverfahren die verwendet werden um die Qualität zu sichern sind Modultests, Komponententests und Systemtests. Der Performancetest ist eine Testmethode des Systemtests und dient dazu die Anforderungen an die Leistung eines zu testenden Systems, unter bestimmten simulierten Gegebenheiten, zu sichern (vgl. [Ba98] S.539 ff). „Allgemein dient der Leistungstest der Überprüfung des in der Produktdefinition festgelegten Leistungsverhaltens.“ [Ba98] S.539

1.2 Problemstellung

Das Ziel dieser Diplomarbeit ist die Erstellung einer zeitlichinvarianten Methodik zur Kategorisierung von Performancewerkzeugen in verteilten Umgebungen. Weiter beinhaltet die Arbeit eine Beschreibung von Performancetestverfahren und deren Einordnung in den Software Engineering Prozess. Diese Arbeit soll den Prozess der Performancetest-werkzeugvalidierung für ein Projekt mit verteilten Szenarien vereinfachen und möglichst für jede Software ein passendes Performancetestwerkzeug anbieten können. Das Expertensystem soll zeitlich unabhängig sein.

1.3 Zusammenfassung

Diese Arbeit setzt im Softwareentwicklungsprozess nach den Funktionstests an. Das heißt, eine zu entwickelte Software wurde funktional gestestet und hat die geforderten funktionalen Anforderungen erfüllt.
Als erstes wurden Grundlagen über Prozessmodelle, Testverfahren und der Performancetest selbst erläutert. Diese Grundlagen dienten als Basis der Arbeit. Das dritte Kapitel beschreibt sowohl Gedankengänge und Wege zur Auswahl von Testwerkzeugen und zur Kategorisierung als auch den Weg des Entscheidungsvorgangs. Abgeschlossen wird dieses Kapitel mit der Vorstellung des Diagnosewerkzeugs D3Web. Das entwickelte Expertensystem wird im vierten Kapitel beschrieben. Es wird auf das Anwendungsgebiet, den Aufbau, die Ergebnisdarstellung und dem nachträglichen Einfügen von Testwerkzeugen eingegangen. Die Nutzung oder Anwendung des Expertensystems und dessen Webinterfaces werden in Kapitel fünf erläutert. Außerdem wird dem Nutzer ein Leitfaden für das Erstellen von Testfällen vorgeschlagen.

KAPITEL 2

Grundlagen und Stand der Technik

2.1 Einordnung in Prozessmodelle

Im Folgenden wird kurz, an Hand von ausgewählten Prozessmodellen, die Einordnung des Performancetests in den Prozessablauf beschrieben.
Die Zeit für den Tester aktiv zu werden hängt in den Projekten von der Wahl des Prozessmodells ab. Es gibt verschiedene Herangehensweisen an eine zu entwickelnde Software. Das Prozessmodell bestimmt die Phasen in welchen getestet wird oder besser gesagt den Zeitpunkt, wobei die Reihenfolge der Tests gleich bleibt.
Der Performance- oder Leistungstest kann also in jedem Prozessmodell integriert werden. (vgl. [Ba98] S.538 ff)

2.1.1 Wasserfall

Das Wasserfallmodell besitzt eine starre Ablaufreihenfolge. Die einzelnen Entwicklungsphasen werden überschaubar unterteilt und am Ende verifiziert. Trotz der Möglichkeit in vorhergehende Phasen zurückzuspringen wird das älteste Prozessmodell jedoch als zu undynamisch befunden. Im Wasserfallmodell gibt es eine allgemeine Testphase vor der in Betriebnahme, in welcher auch der Performancetest angesiedelt ist.
(vgl. [Th02] S.21 f)

2.1.2 V-Modell

Zu der Verifikation des Wasserfallmodells kommt im V – Modell noch die Validation hinzu. Es werden nicht nur die Produkte der einzelnen Phasen gegen die Spezifikation, sondern auch die Richtigkeit in Bezug auf den Einsatzzweck getestet. (vgl. [Ba98] S.101 f)

Durch ein gewolltes Abgrenzen der Phasen voneinander ist das Anwenden des V-Modells für Produkte im Bereich Embedded Software sinnvoll. Eine der letzten Phasen ist die Systemtestintegrationsphase, in welcher die Performancetests gefahren werden. (vgl. [Th02] S.23)

2.1.3 Sashimi Modell

Bei dem Sashimi Modell sind die einzelnen Phasen nicht mehr voneinander getrennt. Durch das Überlappen der Phasen kann es zu unnötiger Doppelarbeit kommen. Performancetests sind durch die Überschneidung der Phasen auch Parallel zur Kodierung und den Unittests möglich. (vgl. [Th02] S.24)

2.1.4 Incremental Delivery

Eine eigenständig funktionierende Teilmenge des gesamten Projekts wird nach einer gewissen Zeit an den Kunden ausgeliefert. Diese Version ist lauffähig und kann vom Kunden genutzt werden. Die restlichen geforderten Module werden beim Incremental Delivery Modell in den folgenden Versionen ausgeliefert. Im Idealfall werden die Performancetests also vor jeder Auslieferung zusammen mit der vorhergehenden Version erneut ausgeführt. Der Testaufwand ist demnach größer als beim Wasserfall oder dem V - Modell, da er für gleiche Komponenten wiederholt werden muss.
(vgl. [Th02] S.26 f)

2.2 Testverfahren

Um Performancetests effektiv in einem Projekt einzugliedern ist es sinnvoll vorher die Funktionalität der zu testenden Komponenten zu gewährleisten. Der Performancetest ist ein möglicher Teil des Systemtests und ist nach der Integrationstestphase zu finden. Im Folgenden werden die Grundtestverfahren kurz vorgestellt.

2.2.1 Modultest

Modul- oder Unittests sollen kleine Bausteine der späteren Software prüfen. Der Test findet auf Quellcodeebene statt und wird normalerweise vom Entwickler durchgeführt. Wenn die Grundlage annähernd fehlerfrei ist, gibt es später auch weniger Folgefehler. (vgl. [Th02] S.122 f) Da der Quellcode und damit der Aufbau der Module den Entwicklern bekannt sind, nennt man diese Methode des Testens auch White Box Test. (vgl. [Th02] S.59 f)
Mit White Box Tests prüft man die interne Struktur der Software. Testdaten werden an Hand des Wissens über die Programmlogik erstellt. (vgl. [My95] S.5-9)
Ein funktionierendes System ist für den Performancetest natürlich eine Voraussetzung.

2.2.2 Integrationstest

Wenn sichergestellt ist, dass die einzelnen Komponenten überprüft worden sind und jedes Modul für sich funktioniert, kann das Zusammenspiel der Komponenten getestet werden. Dabei gibt es verschiedene Integrationsstrategien. Abhängig von der gewählten Strategie kann der Performancetest früher oder später beginnen und fertige Teilsysteme testen. Generell sollte aber nach dieser Phase sichergestellt sein, dass die einzelnen Komponenten funktionieren und zusammenarbeiten. (vgl. [Ba98] S.505 ff)

2.2.3 Systemtest

Die Systemtests sollen sicherstellen, dass die bei der Produktdefinition erstellten Forderungen auch eingehalten worden sind. Es gibt dabei unterschiedliche Testverfahren welche z.B. die Benutzbarkeit, die Zuverlässigkeit, die Vollständigkeit, die Konfiguration und noch viele mehr testen. Ein Beispiel wäre auch der Performancetest. Systemtests prüfen also das System als Ganzes. (vgl. [Ba98] S.537 ff)
Bei optimalen Vorbedingungen dienen die Ergebnisse eines Systemtests als Grundlage für Optimierungsaktionen. Optimale Vorbedingungen wären der erfolgreiche Abschluss der Modul- und Integrationstests.

Bei einem Systemtest prüft ein, vom Entwicklerteam unabhängiges, Testteam die Testfälle oder Szenarien. Das geschieht um der vorbelasteten Sichtweise der Entwickler entgegenzuwirken. Diesen Ansatz nennt man Black Box Test. Der Tester braucht kein Wissen über den internen Teil der Applikation zu haben. Lediglich die Eingabe und das Ergebnis interessieren ihn. (vgl. [Th02] S.81 ff) Der Black Box Test kann auf allen Teststufen angewandt werden.[1]

2.3 Performance

Der Mensch nutzt nur ca. 10% seines Gehirns. Das Gehirn arbeitet korrekt und liefert Ergebnisse. Mit Hilfe von Übungen kann dieser Prozentsatz aber erhöht werden und der Mensch ist zu mehr Leistung fähig.

Ähnlich ist es bei einer Software. Nach den Modul- und Integrationstests ist sichergestellt, dass die entwickelte Software funktioniert und die geforderte Arbeit erledigt. Es gibt aber keine Aussage darüber, ob die Arbeit schnellstmöglich gemacht wird oder ob sie gemacht wird wenn Grenzwerte überschritten werden. Um die Software auf solche Dinge zu testen, gibt es die Performancetests und deren Unterarten. Das Ziel ist es Schwachstellen oder Engpässe zu lokalisieren.

Eine Methode dieses zu tun ist das „Monitoring “. Mit Hilfe des Monitorings kann zu einem bestimmten Zeitpunkt eine Aussage über den Systemzustand gemacht werden. Abhängig vom Testwerkzeug ist, ob diese Aussagen auch parallel zur Laufzeit gemacht werden können. Neben dem Antwortzeitverhalten und dem Speicher-verbrauch wird noch das Monitoring von Datendurchsätzen als gängiges graphisches Ergebnis dargestellt.

Sollten Performanceprobleme herausgefunden werden, gilt es diese zu lokalisieren. Beispielsweise steht fest, dass irgendwo Zeit verloren gegangen ist. Ein Profilingwerkzeug analysiert die gesamte Zeitspanne in dem es die Antwortzeit in kleine Stücke zerlegt. Der Engpass in der Applikation kann nur mit einer Ursachenforschung gefunden werden. Kaum ein System ist perfekt und es wird immer Engpässe geben. Wenn aber die vorher definierten Bedingungen erreicht sind, sollte auch nicht weiter nach Engpässen gesucht werden. Oft kostet das Suchen nach möglichen Engpässen, nach Erfüllung der Anforderung, zu viel Zeit und Ressourcen. Engpässe werden auch immer langsamer gefunden, je mehr repariert worden sind.

[1] Fraunhofer IESE: Verfahren – Black-Box test http://www.software-kompetenz.de

Grundsätzlich gilt aber, dass es besser ist erkannte Probleme mit Hilfe von Performance- und Profilingtests zu beseitigen, als ständig die Hardware des Systems zu verbessern um so die Leistung zu steigern.[2]
Das Ergebnis eines Performancetest kann beispielsweise die Aussage sein, dass eine Profilinganalyse gemacht werden muss.

2.3.1 Definition

In der Informatik werden der Verbrauch von Ressourcen und die Ausgabe der Qualität von Software und Hardware als Performance (dt. Leistung) beschrieben. Kenngrößen sind Zeit und Speicherplatz.
Das Ziel eines Performancetests ist es also zu zeigen, dass die Anforderungen eines Programms in Bezug auf die Leistung nicht erfüllt werden. (vgl. [My95] S.114)
In verschiedenen DIN Normen steht über die Effizienz, also der möglichen Leistung / Performance eines Softwareproduktes, dass in dieser Hinsicht nichts grundlegendes gefordert ist und ein Programm die nur funktionalen Anforderungen erfüllen sollte. Es sei denn die Produktbeschreibung beinhaltet Angaben zur Effizienz der Software.
Für die nachfolgende Methodik in Kapitel 3, ist diese Basisaussage aber weniger zutreffend, da Performanceanalysen nur von Nutzer gemacht werden, welche die Effizienz ihrer Software in Bezug auf die Leistung und den zeitlichen Aspekt untersuchen wollen.

[2] Computerwoche: Wege zum Performance – Test http://www.computerwoche.de

2.3.1.1 Lasttest

Das zu testende System wird im Bereich der erlaubten Grenzen getestet. Zusätzlich kann es Testfälle geben in denen unvorhergesehene Ereignisse auftreten. Es kann zum Beispiel die Reaktion der Software auf den Ausfall von verschiedenen Komponenten getestet werden. (vgl. [Ba98] S.540)

In vielen Fällen sind die Grenzen der Belastbarkeit einer Software auch nicht bekannt und können durch den Last- oder Volumentest gefunden werden. Normalerweise sind die Grenzen aber so gelegt, dass diese vom Endnutzer nicht erreicht oder überschritten werden können. (vgl. [Th02] S.124 f)

Sinnvoll ist es die Standardgrenzwerte und den Mittelwert der verschiedenen Daten oder Nutzer zu prüfen (lower bound, average value, upper bound).

Dem Lasttest werden der Massentest, der Zeittest und der Mehrbenutzertest untergeordnet.

2.3.1.2 Stresstest

Beim Stresstest werden die definierten Grenzen bewusst überschritten um die Stabilität des Programms zu testen. Geprüft werden soll das System in Ausnahmesituationen. (vgl. [Ba98] S.540) Die Reaktionen der Software auf solche Ausnahmesituationen sind ohne Testen nicht bekannt und können demnach auch häufig nicht vom Entwickler vorhergesehen und abgefangen werden. Das Überschreiten der Grenzen kann beispielsweise durch das Verändern von Parametern, wie Speichereduzierung oder umfangreicheren Datenmengen erreicht werden.

KAPITEL 3

Methodik zum Performancetest

3.1 Voraussetzung für einen Performancetests

Bevor sich entschieden wird ein Performancetest zu starten, ist es wichtig klar zu stellen, dass viele Projekte in der Anforderungsanalyse keine Grenzwerte für zeitliche Abläufe oder die Größe der Daten definiert haben.

„… Sind Grenzwerte nicht absolut angebbar - sondern hängen zum Beispiel von der Art der Nutzung oder von Daten ab - so sind die Beschränkungen zu benennen. …“ [DIN 66285 (3.1)]

Wenn dieses Szenario gegeben ist, dann gilt es zuerst zu prüfen, wo diese Grenzwerte liegen und mit welchen Parametern diese überschritten werden. Die Entwickler können damit eine Routine zur Fehlbehandlung in das System einbauen um möglichen Ausnahmezuständen vorzubeugen.

Zum Beispiel sollte die Verarbeitungsbestätigung einer Auftragsanfrage in einer vorher definierten Zeit erfolgen. Sollte dies nicht geschehen, kann es zu Fehlern kommen, da eventuell der allocierte Speicher, in der Zeit die nun mehr zur Verfügung steht, zu voll geschrieben wird.

Da nun Grenzwerte definiert sind, kann der eigentliche Performancetest beginnen. Einerseits gilt es herauszufinden bei welcher Bandbreite, mit welcher Datenmenge in einer noch akzeptablen Zeit Daten transportiert werden können. Wie verhält sich die Software außerdem, wenn mehrere Benutzer gleichzeitig auf das zu testende System zugreifen und Anfragen stellen? [siehe Lasttest] Dabei ist Datenmenge nicht gleich Datenmenge. Zum Beispiel benötigen mehrere Dateien von der gleichen Größe einer einzigen Datei mehr Zeit zum Transport, da mehrere Dateien ein Protokoll overhead produziert wird. Anderseits gilt es die eingebaute Routine zur Fehlerbehandlung zu testen, in dem das zu testende System mit Parametern oder Werten außerhalb der Grenzwerte konfrontiert wird. [siehe Stresstest]

Jemand, der also einen Performancetest durchführen will, muss genau wissen was er testen will und welche Anforderungen getestet werden sollen. Ohne diese festgelegten Informationen ist es nicht möglich ein passendes Testwerkzeug zu entwickeln oder auszuwählen.

3.2 Testwerkzeug

3.2.1 Gedanken eines Testers

Performanceuntersuchungen sind manuell schwierig oder nicht durchführbar. Es muss also ein passendes Testwerkzeug für das jeweilige Bedürfnis gefunden werden. Die Suche nach einem geeigneten Werkzeug ist sehr zeitaufwendig und ist von Projekt zu Projekt unterschiedlich sein. Es ist also zu überlegen, je nach Größe des Projekts, ob die Erstellung eines, genau auf diese Software, zugeschnittenes Werkzeug sinnvoll ist. Die Programmierung und das Testen dieses Werkzeugs kostet jedoch erneut viel Zeit und Geld.
In vielen Unternehmen gibt es auch die Lösung, eine eigene Abteilung oder Nebenfirma mit dieser Problematik aufzustellen. Diese Abteilung oder Nebenfirma entwickelt ständig parallel zu den Versionen der eigentlichen Software ihre Bausteine. Die Nutzer des entstandenen Testwerkzeugs können jederzeit Wünsche zur Verbesserung des Testwerkzeugs äußern und diese in der nächsten Version in dem Testwerkzeug wiederfinden. Die Testwerkzeuge sind oft für verschiedene Projekte nutzbar und werden nur auf die Bedürfnisse zugeschnitten oder erweitert. Das Modell des Incremental Delivery [siehe 2.1.4] wird dort also je nach Projektgröße, oft aber in kleinem Maßstab, verwendet.

3.2.2 Kategorisierung in zwei Ansätzen

Testwerkzeuge passen sich ständig der aktuellen Technologie an und werden weiterentwickelt. Wird heutzutage ein, anscheinend sehr gutes, Testwerkzeug benutzt, ist dieses in zehn Jahren bei gleich bleibender Weiterentwicklung der Technik wahrscheinlich veraltet und kann mit den dann aktuellen Werkzeugen nicht mehr mithalten.

1. Um eine Methodik zu entwickeln, welche noch in zehn Jahren gültig sein könnte, ist es notwendig Testwerkzeugs zu kategorisieren. Das heißt, dass bestimmte Eigenschaften, welche ein Testwerkzeug haben kann, aufgelistet und gruppiert werden. Beispielsweise ist es von Vorteil, dass das Testwerkzeug die geforderte Plattform der zu testenden Software nutzen kann. Ein gutes weiteres Beispiel ist das Ergebnisformat. Während viele Firmen einfache Grafiken bevorzugen, gibt es auch Unternehmen, welche die gewonnenen Ergebnisse sofort weiterverarbeiten und von einem weiteren Automatismus einbinden lassen. Es würden nur Werkzeuge in Frage kommen, welche Ergebnisformate wie XML oder HMTL unterstützen. Hier wäre eine mögliche Kategorisierungseigenschaft nach Ergebnisformatierung gegeben. Die einzuordnenden Testwerkzeuge können auch mehreren Kategorien angehören, wenn sie die jeweilig geforderten Eigenschaften erfüllen. In diesem Fall sind diese Werkzeuge sehr allgemein gehalten, was Vorteile, aber auch Nachteile haben kann. Der Nutzer der Kategorisierung soll mit Hilfe dieser Einordnung auch Testwerkzeugs zuordnen können, die in dieser Studie nicht aufgezählt sind. Dies soll durch die Kategorisierungseigenschaften unterstützt werden.

2. Ein weiterer Ansatz wäre, die einzelnen Eigenschaften der Universaleigenschaftsmenge, welche aus den Testwerkzeugeigenschaften gewonnen wird, mit den Anforderungen des zu testenden Systems zu vergleichen. Die daraus gewonnen Übereinstimmungen sind dann sicherlich nicht mehr klar den oben beschriebenen Kategorien unterzuordnen, aber zur genauen Testwerkzeugbestimmung ist dieser Ansatz wahrscheinlich gut geeignet.

Die zwei beschriebenen richtungweisenden Ansätze sind das Herausfinden der Eigenschaftsschnittmenge der Testwerkzeugs und der zu testenden Software, um diese zu vergleichen und Schlüsse zu ziehen und die Kategorisierung und Einordnung von Testwerkzeugeigenschaften. Diese beiden Ansätze gilt es zu untersuchen. Das Ergebnis ist besonders nützlich für unabhängige und nicht vorbelastete Tester, da sie dadurch automatisch familiär mit der zu testenden Software werden. Vorstellbar wäre auch, beide Ansätze zu nutzen um ein optimales Testwerkzeug zu finden. Neben dem gewonnen Testwerkzeug wird auch deutlich wie vollständig oder unvollständig die Anforderungsanalyse der zu testenden Software ist. Die Anforderungen an die Performance werden schon untersucht bevor die eigentliche Software einbezogen wird.

3.2.3 Auswahl von Testwerkzeugen

Um eine Kategorisierung zu erstellen, müssen zuerst Testwerkzeugs gefunden werden. Diese Testwerkzeuge sollten einerseits alle verteilte Lasttests durchführen können und andererseits nicht alle die gleichen Eigenschaften aufweisen. Testwerkzeuge zu vergleichen und auszuwählen, welche alle in ein bis zwei Kategorien einzuordnen sind hat keinen großen Nutzen. Bevor das eigentliche Nachdenken über die Kategorien beginnt, müssen also zuerst in Frage kommende Testwerkzeugs betrachtet werden. Daraus ergibt sich schon die nächste Frage.

„W*elche der zahlreichen Testwerkzeugs sollen ausgewählt werden?“*

Die Recherche ergab eine sehr große Menge an Werkzeugen. Viele Werkzeuge haben ähnliche oder gleiche Eigenschaften. Um einen Überblick zu bekommen, was ein Performancetestwerkzeug im Einzelnen leisten kann, ist es notwendig die Eigenschaften der Testwerkzeuge herauszufinden. Auf den Webseiten der Werkzeuganbieter befinden sich oft Stichwörter wie „Systemanforderung“ oder „Datenblatt“. Mit Hilfe dieser Informationen und dem Ausprobieren von Demonstrationen und Präsentationen war es möglich, einen Einblick - je nach Dokumentation - in das jeweilige Testwerkzeug zu bekommen.

So lange kein Standard auf diesem Gebiet besteht, ist dies die einzige Möglichkeit ohne Mitwirkung der Entwickler oder Veröffentlichungen in Zeitschriften Informationen zu einem Testwerkzeug zu erhalten.
Die Eigenschaften der angebotenen Werkzeuge wurden ohne Wertung und Zuordnung zu deren Testwerkzeugs aufgeschrieben. Aus den dreiundvierzig Eigenschaften ergibt sich die Universaleigenschaftsmenge. Eine gebildete Universaleigenschaftsmenge ist die Grundlage um die Anzahl der Werkzeuge einzuschränken und wird im folgenden Abschnitt beschrieben.
Durch Befragung von Mitarbeitern und eigener Erfahrung in der Validierung von Testwerkzeugen ergaben sich erste Kategorien, welche beispielsweise die Eigenschaften der Usability oder der Kommunikation vereinten. Die Basis zur Auswahl der Testwerkzeuge ergibt jedoch eine Kategorie, welche sich mit der Plattform - auf dem das Werkzeug betrieben werden kann - und der Finanzierung beschäftigt. Diese Kategorie werde ich im folgenden Basiskategorie nennen. Zur Basiskategorie gehören die Eigenschaften: Windows (NTFS), Linux / Unix, plattformunabhängig, open source und kommerziell. Diese Eigenschaften sind meiner Meinung nach die Eigenschaften, nach denen häufig zu erst gesucht wird, wenn ein Testwerkzeug gebraucht wird. Wenn eine gewünschte Eigenschaft dieser Kategorie eines Testwerkzeugs nicht zu trifft, kann das Testwerkzeug oder die Kategorie, nicht als Endergebnis auftreten. Wobei auf Eigenschaften wie beispielsweise das Monitoring von verschiedenen Werten eher verzichtet werden kann als auf Eigenschaften der Basiskategorie. Die Wichtigkeit ist Anwender bezogen und hat keinen Einfluss auf die Heuristik. Das heißt, sie ist in jedem Fall unterschiedlich. Die bestimmte Basiskategorie wird als erstes im Entscheidungsbaum abgefragt.
Mit der Bestimmung einer Basiskategorie gibt es eine Möglichkeit, um die Anzahl der auszuwählenden Testwerkzeuge sinnvoll einzugrenzen.

Da diese Methodik den Anspruch hat, zeitlich- und projektunabhängig zu sein, macht es wenig Sinn eine große Anzahl von jetzt aktuellen Testwerkzeugen auszuwählen.

Es ist aber wichtig, dass alle Möglichkeiten der hoch gewichteten Basiskategorie vorhanden sind. Deshalb sollten plattformunabhängige und plattformabhängige Testwerkzeuge, sowie open source und kommerzielle Testwerkzeugs ausgewählt werden. Im Laufe der Recherchen ergaben sich Favoriten, welche öfter genannt worden sind. Im open source Bereich gibt es einen Überblick über Performancetestwerkzeuge. Dort wird angezeigt, welches Performancetestwerkzeug wie oft von den Seiten des Anbieters heruntergeladen wurde.

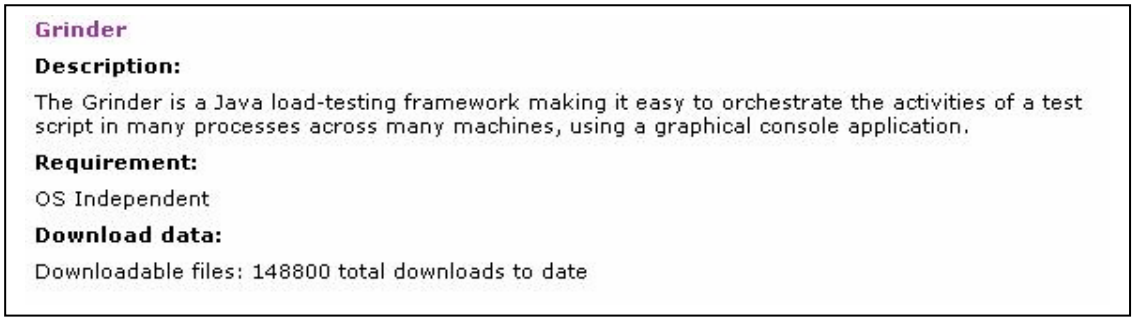


Abbildung 1: Beispiel für Werkzeugüberblick mit Downloadzähler [3]

An Hand der Downloadhäufigkeit sind vier open source Testwerkzeuge ausgewählt worden. Drei weitere sind kommerzielle Werkzeuge. Davon sind zwei Werkzeuge für Windows (NTFS), wobei es ein Werkzeug auch mit einem open source Paket gibt. Um die Linux / Unix - Sparte abzudecken wurde dort auch ein Testwerkzeug bestimmt. Dieses Testwerkzeug wird zwar als plattformunabhängig beschrieben, ist aber nur unter Linux / Unix Systemen getestet worden. Die restlichen vier Werkzeuge sind plattformunabhängig. Damit sind sieben Testwerkzeugs in die nähere Auswahl gekommen. Die Auswahl dieser Werkzeuge ist nur für diese Arbeit und in der näheren Zukunft relevant. Die Werkzeuge dienen hauptsächlich zur Eigenschaftsforschung. Auf der folgenden Grafik ist die Verteilung zwischen den Eigenschaften und der Werkzeuge zu sehen.

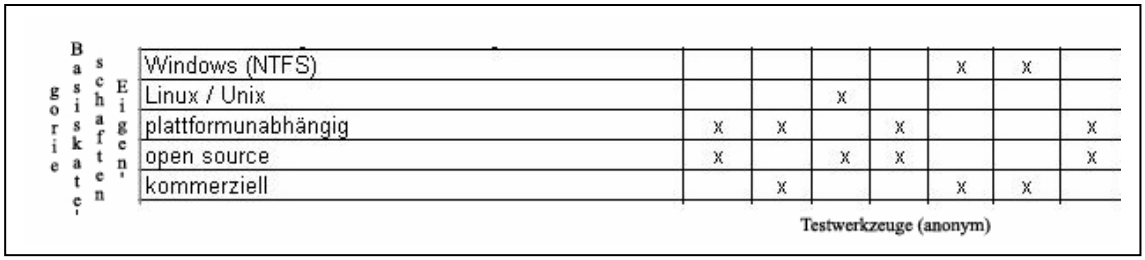


Abbildung 2: Überblick über Zusammenstellung der ausgewählten Testwerkzeuge

[3] Mark Aberdour: Performance test tools http://www.opensourcetesting.org

3.2.4 Kategorisierung der Testwerkzeugeigenschaften

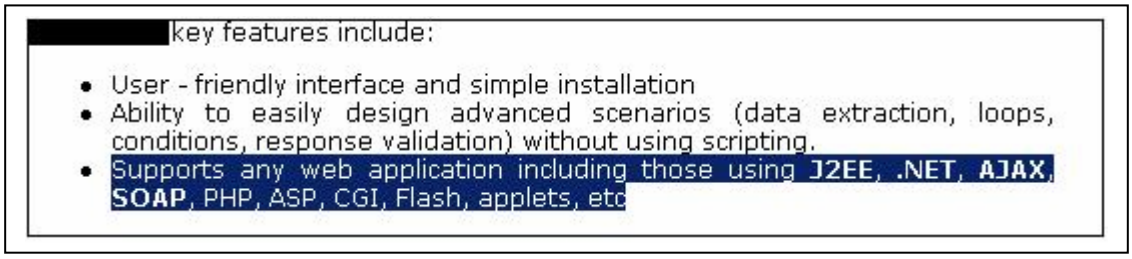
Die erhaltenen Testwerkzeuge müssen der Universalmenge an Werkzeugeigenschaften zugeordnet werden, im Folgenden wird diese Universalmenge immer Universaleigenschaftsmenge genannt. Durch diese Zuordnung können Unterschiede herausgearbeitet und die Testwerkzeuge verglichen werden.

Universaleigenschaftsmenge								
	Windows (NTFS)					x	x	
	Linux / Unix			x				
	plattformunabhängig	x	x		x			x
	open source	x		x	x			x
	kommerziell		x			x	x	
	Parallele Nutzeranzahl bei Ø Serverhardware < 1600		x					x
	Parallele Nutzeranzahl bei Ø Serverhardware > 1600					x	x	
	graphische Oberfläche Steuerungsinstanz vorhanden	x	x		x		x	x
	einfache Bedienung	x	x		x			
	komplizierte Bedienung						x	
	Steuerung der Clients von einem Rechner	x	x		x	x	x	x
	manuelle Steuerung der Clients							
	einfache Installation	x	x		x			
	komplizierte Installation						x	
	Scripting	x			x	x	x	x
	Nutzung/Erstellung von Testskripten möglich	x	x	x	x	x		x
	Monitoring von Datendurchsatz	x		x	x		x	x
	Monitoring von Antwortzeiten	x	x	x	x		x	x
	Monitoring von Speicherverbrauch			x	x		x	x
	Darstellung der Grafiken zur Laufzeit				x		x	
	Ergebnissformat *.csv, *.html, *.xml oder *.pdf	x			x			x
	Ergebnissformat *.log	x						
	Auswertung von Ergebnissen includiert							
	Kommunikation über TCP/UDP	x	x	x	x			
	Corba MW						x	
	RMI MW	x					x	
	SOAP MW		x				x	
	andere MW	x						
	POP	x					x	
	SMTP	x					x	
	FTP	x					x	x
	HTTP/HTTPS	x	x		x	x	x	x
	parallele Nutzertests	x		x				x
	keine parallelen Nutzertests				x			
	mehrere Lastclients (verteilt)	x		x	x	x	x	x
	nicht verteilt							
	Blackbox Sicht		x					
	Whitebox Sicht	x			x			x
	Cookies unterstützt	x	x					
	Cookies nicht unterstützt				x			
	Datenbanktests möglich		x				x	
	Mainframetests						x	
	gängige CRM Systeme				x	x	x	
					Testwerkzeuge (anonym)			

Abbildung 3: Universalmenge der Eigenschaften von Performancetestwerkzeugen

Die Testwerkzeuge auf der X-Achse sind nicht genannt, da diese - wie schon beschrieben - keinen Einfluss auf die Methodik haben sollten. . Felder mit einem „x“ sagen aus, dass das jeweilige Testwerkzeug, in einer Spalte, die Eigenschaft der jeweiligen Zeile besitzt. In späteren Abbildungen tauchen auch Felder auf, welche mit „egal“ gefüllt sind. Die jeweilige Eigenschaft die ein „egal“ in ihrer Zeile vorweißt, ist dann umgangssprachlich „Nice To Have“.

Leere Felder heißen nicht immer, dass diese Eigenschaften von dem Testwerkzeug nicht unterstützt werden. Oft sind die Eigenschaftsbeschreibungen der Anbieter nicht eindeutig – Abbildung 4 – und sollten deshalb auch nicht als wichtiges Entscheidungskriterium gehandelt werden. Die unterstütze Middleware wird nur selten auf den Seiten der Anbieter genannt. Deshalb enthalten die Basiskategorien auch nicht die Frage nach der jeweilig unterstützten Middleware. Würde diese Frage in den Basisentscheidungen auftauchen, wäre die spätere Auswahl, die dadurch getroffen wird, nicht repräsentativ für das Expertensystem. Eine zu große Menge an Kategorien würde nicht als Lösung in Frage kommen, da der Großteil der Produktbeschreibungen nicht explizit auf eine bestimmte Middleware hinweißt. Dennoch kann es sein, dass die Middleware unterstützt wird, nur wird sie nicht genannt. In vereinzelten Fällen in fortgeschrittenen Zweigen des Entscheidungsbaums treten aber Vergleiche zur Middleware auf.


[redacted] key features include:

- User - friendly interface and simple installation
- Ability to easily design advanced scenarios (data extraction, loops, conditions, response validation) without using scripting.
- Supports any web application including those using **J2EE**, **.NET**, **AJAX**, **SOAP**, PHP, ASP, CGI, Flash, applets, etc

Abbildung 4: Beispiel: nicht eindeutige Produktbeschreibung („etc“)

3.2.5 Der Weg zum Entscheidungsvorgang

Es gibt unterschiedliche Arten von Entscheidungsbäumen: Diagramme, Matrizen, Graphen oder zum Beispiel „if – else“ Abfragen. Viele verschiedene Wege wurden theoretisch für logisch und gut empfunden, aber bei der Realisierung traten immer wieder unzufriedenstellende Ergebnisse auf. Oft konnte Datenredundanz nicht ausgeschlossen werden oder die Eingaben der Nutzer waren zu aufwändig.

→ Es wurde versucht die Eigenschaften der AUT und eines Testwerkzeugs zu vergleichen. Bei vielen Übereinstimmungen wurde angenommen, dass dieses Werkzeug den Bedürfnissen der AUT entspricht und eine gute Wahl als Testwerkzeug wäre. Dieser Ansatz trifft stellenweise zu, aber das ist nur zufällig so geschehen. Es gibt beispielsweise Eigenschaften die ein zu testendes System nicht hat, ein Testwerkzeug aber besitzt und anders herum. Dadurch sind manche universelle Testwerkzeuge nicht in die engere Auswahl gekommen. Es wurde eine Lösung gefunden, aber diese Lösung war nicht immer die bestmögliche. Beispielsweise entfiel die Möglichkeit neu entwickelte kommerzielle Software mit open source Werkzeugen zu testen, da die Eigenschaften ungleich sind.

Eigenschaft „kommerziell“ != Eigenschaft „open source“

Es gab aber auch völlig falsche Ansätze, welche vom eigentlichen Ziel abgewichen sind und sich stellenweise in einer ganz anderen Thematik verloren hatten.

→ Es wurden Testerwerkzeuge ausgewählt, welche Schnittstellen besaßen um mit Hilfe von Scriptsprachen nicht gegebene, aber gewünschte Funktionalitäten hinzuzufügen. Zuerst wurde ein bestmögliches Testwerkzeug, aus einer Menge von Testwerkzeugen die Scriptschnittstellen haben, gesucht. Die Idee war mehrere Komponenten zu haben, welche universell den Testwerkzeugen angegliedert werden. Der Programmieraufwand wäre groß und die schon genannte Anforderung der zeitlichen Invarianz wäre nicht gegeben gewesen, da die Komponenten speziell für manche

Schnittstellen entwickeltet worden sind. Viele Testwerkzeuge bieten eine Schnittstelle über die objektorientierte und auf Java basierende Sprache Python an.[4]

Eine weitere Anforderung an das EXPS ist, dass dem Nutzer soviel Arbeit wie möglich abgenommen wird. Der Entscheidungsvorgang soll für den Nutzer einfach und schnell zu Ziel führen.

→ Beispielsweise wurde versucht mit Hilfe von Eigenschafts- und Anforderungsschnittmengen zwischen der „**A**pplikation **U**nter **T**est“ (AUT) und der Testwerkzeugs eine Menge von Gemeinsamkeiten mit Hilfe von Matrizen und später von dreidimensionalen Graphen zu definieren. Bei dieser Idee wäre aber der Aufwand des Nutzers sehr groß gewesen. Außerdem wurde ein bestimmter Wissensstand des Anwenders über die AUT vorausgesetzt. Zusätzlich sind die Informationen die ein Tester über eine ihm unbekannte Software hat, oftmals nur auf Blackbox - Sicht begrenzt.

Aus dieser, zuletzt genannten, speziellen Idee entstand dann ein Gedanke der Diagnosesystemen glich. Diagnosesysteme werden hauptsächlich in der Medizin[5] verwendet um mit Hilfe von Symptomen auf Krankheitsbilder zu schließen, es gibt sie in der Automobilindustrie[6] oder Betriebswirtschaft. Gesucht wurde also eine Software, welche Regeln zur Verfügung stellte, um sich sein eigenes Diagnosesystem zu schaffen. D3Web[7] war für diese Zwecke sehr gut geeignet und wurde zur Realisierung des Entscheidungsvorgangs genutzt.

[4] jython contributors: The Jython Project http://www.jython.org

[5] Hitachi: Diagnoseprodukt http://hitachi.de/products

[6] ServiceXpert: Diagnose-Center http://www.servicexpert.de

[7] Universität Würzburg / iisy / denkbar: Diagnosesystem D3Web http://www.d3web.de

3.3 D3Web

„D3Web ist der Java-basierte Nachfolger von D3, welches beispielsweise als Web-basiertes Front-End für ein Diagnosesystem benutzt werden kann…“[8]

Warum ist D3Web gut geeignet?

D3Web kann genutzt werden um wiederholbare Entscheidungsprozesse mit Hilfe von Entscheidungsbäumen, heuristischen, fallbasierten und überdeckendem Wissen zu erarbeiten. Mit D3Web ist es möglich durch ein Grundgerüst von Regeln ein browserbasiertes Expertensystem zu entwickeln. Dieses ist eine Art Entscheidungsbaum und wichtet je nach Inhalt von importierten Dokumenten möglich Schlussfolgerungen.

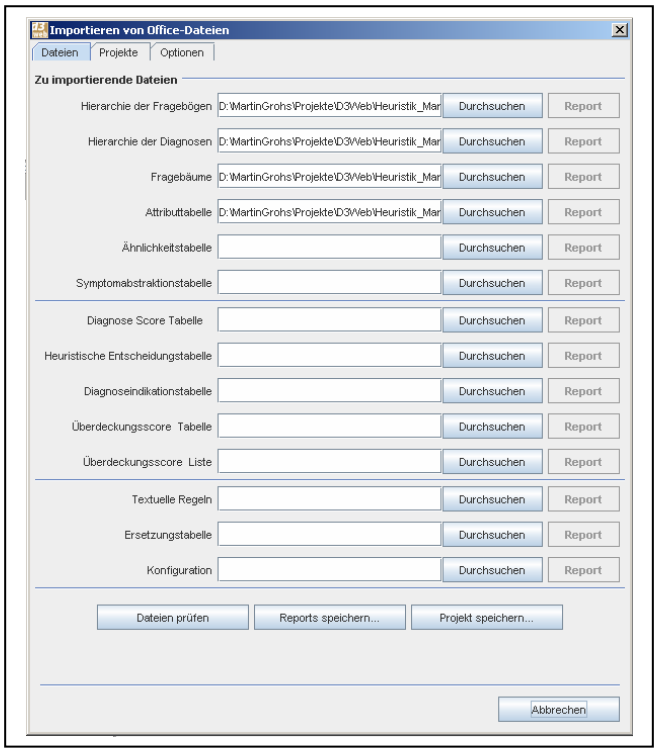


Abbildung 5: Screenshot - Importieren von Dateien D3Web GUI

Diese Dokumente sind sowohl Excel- als auch normale Textdateien, dies ist für das Expertensystem von Vorteil, weil D3Web so leicht zu nutzen ist und keine weiteren Vorkenntnisse über andere Sprachen voraussetzt. In den Dateien müssen verschiedene Syntaxregeln angewendet werden. Es ist möglich - je nach Bedarf - vierzehn verschiedene Dateien zu erstellen und zu importieren. Diese Dateien dienen als Input für das EXPS. Für die Erstellung eines Expertensystems zur kategoriebasierten Auswahl von Testwerkzeugen wurden lediglich vier Dateien benötigt.

[8] Universität Würzburg / iisy / denkbar: Produktbeschreibung D3Web http://d3web.informatik.uni-wuerzburg.de/product/index_dt.htm

Es hätten auch mehr Dateien verwendet werden können, aber davon wären einige Eingaben doppelt gewesen und mit der Anzahl der zu überblickenden Dateien wachsen auch die Unübersichtlichkeit und die Abhängigkeiten an.

3.3.1 Benötigte Dateien und Interaktionen

In dem folgenden Punkt 3.3.1 werden die vom Expertensystem genutzten Dateien beschrieben. Dabei geht es nicht um den Inhalt, sondern dem Leser soll die generelle Funktionsweise nahegelegt werden. Die Funktionsweise dient als Grundlage für den im Kapitel 4 beschriebenen inhaltlichen Aufbau der Dateien. Ohne das Verständnis des Grundgerüsts der benötigten Dateien, ist es schwierig den Aufbau des Expertensystems in Kapitel 4 zu verstehen. Es ist unter anderem schwierig, weil auf Wissen über die Abhängigkeiten der einzelnen Dateien zu einander in Kapitel 4 nicht weiter eingegangen wird.
Die vier verwendeten Dateien setzen sich aus drei Textdateien und einer Exceldatei zusammen. Die Textdateien sind als einfache *.txt Dateien realisiert. Da dieses die drei wichtigsten Textdateien zur Erstellung einer Wissensbasis mit Hilfe von D3Web sind, werden diese neben der Exceldatei folgend allgemein beschrieben. Screenshots zur Syntax werden an einer späteren Stelle – in Kapitel 4 - eingefügt, wenn spezieller das Erstellen des Expertensystems mit D3Web behandelt wird.

In Abbildung 6 wird grob veranschaulicht wie die vier D3Web Dateien miteinander interagieren. Der Entscheidungsbaum wird mit Informationen der Ergebnis- und Fragegruppendatei gespeist. Tabellenblatt 1 der Attributtabelle liefert den Startfragenkatalog mit den möglichen Antworten. Eine Auswahl oder Entscheidung liefert ein Ergebnis, welches der Entscheidungsbaum an das Tabellenblatt 2 der Attributtabelle weitergibt. Das Tabellenblatt 2 der Attributtabelle beinhaltet die Verlinkungen zu der D3Web externen Webseite. Auf der Webseite sind den - an Tabellenblatt 2 gelieferten – Ergebniskategorien, die Testwerkzeuge untergeordnet.

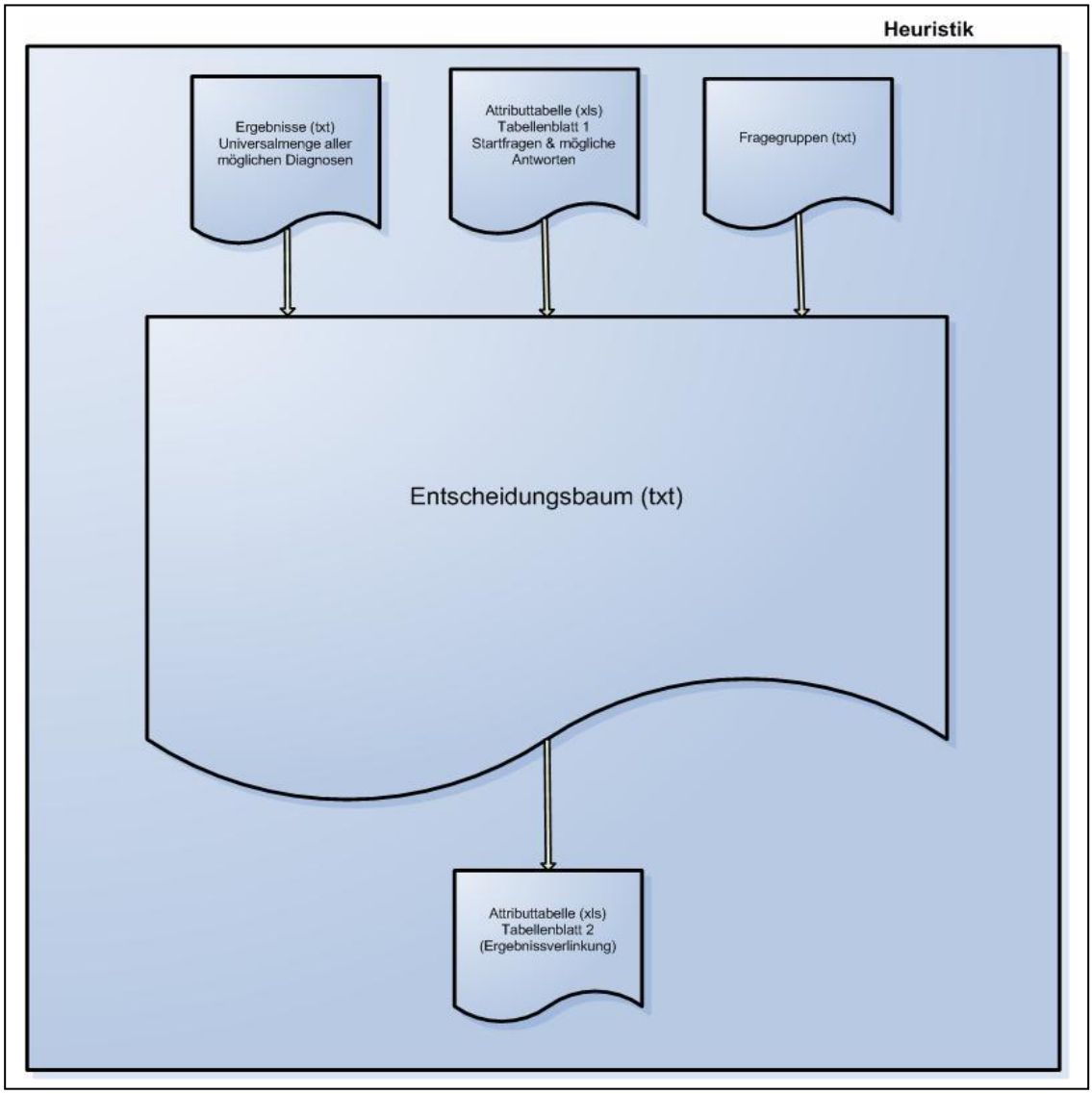


Abbildung 6: Interaktion der Dateien

3.3.1.1 Fragegruppen (txt)

Die inhaltlich kleinste Datei. Sie besteht aus einer Anordnung von Fragegruppen ohne die Fragen selbst. Ähnlich vorstellbar wie bei einer abstrakten Klasse welche keinen Methodenrumpf enthält. So wie die Frageklassen in diesem Dokument angeordnet sind, müssen die Fragen später auch im Entscheidungsbaum angeordnet sein. Es werden die Fragegruppen / klassen in diesem Dokument bekannt gemacht, also im weitesten Sinne deklariert und im Entscheidungsbaum definiert. Diese Datei dient allgemein zu Strukturierung auf oberster Ebene und wird von D3Web als „Hierarchie der Fragebögen“ betitelt.

Abbildung 7: Importanweisung für die Fragegruppen

3.3.1.2 Entscheidungsbaum (txt)

Diese Datei enthält wie der Name schon sagt den Entscheidungsbaum. Sie ist das Herzstück einer D3Web Anwendung. Hier werden Fragen und Folgefragen gestellt und die Antworten gewertet. Folgefragen können bei vorher definierten Antworten gestellt werden. Nach den Basisstartfragen werden die in der Fragegruppen – Datei deklarierten Klassen jetzt definiert. Die Basisstartfragen eignen sich zur groben Einordnung und ersten Deutung möglicher Ergebnisse. Das Ergebnis der Basisstartfragen sind mögliche Frageklassen. Zu diesen wird, ähnlich wie bei dem, aus der Programmierwelt bekanntem, „goto“ gesprungen. Um Hierarchien zu gewährleisten wird mit Spiegelstrichen gearbeitet. Vorangestellt ist ein Minus „-“. Je mehr eingerückt und je mehr Frageklasse aufgerufen worden sind, desto tiefer befindet sich der Nutzer in einem Zweig des Entscheidungsbaums. Die Aussage über ein mögliches Testwerkzeug wird dadurch immer spezieller. Die Ergebnisse und Teilergebnisse können gewichtet werden.

Die Gewichtung dient später zum Vergleichen von mehreren Lösungen. Die Fragen mit höherer Priorität erhalten dabei eine höhere Wichtung. Mit Hilfe von P(1-7) und N(1-7) kann positiv oder negativ gewertet werden. P1 wäre dabei eine niedrige positive Wichtung.

Abbildung 8: Importanweisung für den Entscheidungsbaum

3.3.1.3 Ergebnisse (txt)

In dieser Datei sind alle möglichen Ergebnisse genannt. Die unterschiedlichen Zweige sind in Gruppen eingeteilt. Es kann auch mehrere Lösungen in einer Gruppe geben. Um diese zu spezialisieren, ist es möglich wie beim Entscheidungsbaum mit einem oder mehreren vorangestellten „-“ Ergebnisse einzurücken. Auch hier gilt: je mehr eingerückt worden ist, desto zutreffender ist die Deutung auf ein mögliches Ergebnis.

Abbildung 9: Importanweisung für die Diagnosedatei

3.3.1.4 Attributtabelle (xls)

Diese Exceldatei besteht inhaltlich aus zwei Tabellenblättern. **Tabelle 1** besteht aus den Fragen und Antworten des Startfragenkataloges. Anders als bei den Frageklassen, welche wie oben beschrieben im Entscheidungsbaum definiert sind, wird hier die Definition explizit vorgenommen. Dies gestaltet die Basiseinordnung beim Aufruf der Startfragen übersichtlicher. Der Aufbau der ersten Tabelle ist recht einfach: Spalte A beinhaltet freiwählbare Variablen. Diese Variablen werden in Spalte C, auf gleicher Zeilenhöhe, einer Frage oder Aussage zugeordnet. Eine Zeile tiefer, in Spalte B, können Antworten bereitgestellt werden. Jede Antwort muss in einer eigenen Zeile stehen. D3Web stellt beispielsweise Antworttypen wie multiple - choice (mc), one - choice (oc) oder yes - no (yn; jn) bereit, welche im Entscheidungsbaum angewendet werden. Hier sind diese Antworttypen nur von Wichtigkeit, weil dadurch die Anzahl der Antworten vorgegeben wird.

Tabelle 2 beinhaltet ein interessantes Feature, dort ist es möglich die Ergebnisse mit einer Adresse zu versehen. Die Ergebnisse liefert der Entscheidungsbaum. Dieses Feature wird in der Ergebnisdarstellung eine Rolle spielen. In der Präsentation von D3Web waren beispielsweise die Homepages der Ergebnisse oder der Deutungen eingetragen. In Spalte A werden wieder Variablen bestimmt. In Spalte C wird eine Adresse oder ein Link zugeordnet.

KAPITEL 4

Das Expertensystem

In diesem Kapitel wird das erstellte Expertensystem ausführlich vorgestellt. Der im dritten Kapitel beschriebene Weg zur Auswahl von Testwerkzeugen und die Kategorisierung der Testwerkzeugeigenschaften fließen maßgeblich in das Expertensystem mit ein und werden in den kommenden Punkten auch immer wieder auftauchen. Das Expertensystem besteht aus der Heuristik und einer Webseite zur Ergebnisdarstellung.
Dem Leser wird verdeutlicht, wann das Expertensystem zum Einsatz kommen sollte und wie es prinzipiell und praktisch aufgebaut ist. Nach der Erläuterung des Aufbaus wird die Darstellung des Ergebnisses erklärt. Der letzte Punkt dieses Kapitels zeigt wie die Anforderung der zeitlichen Invarianz erfüllt worden ist.

4.1 Anwendungsgebiet

Um die folgenden Punkte besser zu verstehen und einen Bezug „im Hinterkopf“ zu haben, sollte das Anwendungsgebiet des Expertensystems, im Folgenden mit EXPS abgekürzt, kurz geklärt sein.
Viele Einrichtungen verbringen Monate damit nach Software zu suchen, welche ihren Ansprüchen gerecht wird. Es werden Präsentationen genutzt und Testlizenzen gekauft. Testwerkzeuge sind in der Anwendung oft sehr komplex und setzen eine hohe Einarbeitungszeit voraus. Das EXPS kann dem Nutzer diese Einarbeitungszeit nicht abnehmen, aber der Auswahlprozess wird erheblich erleichtert. Das EXPS zielt auf Softwareentwickler oder Tester ab, die ihre AUT auf Performance testen wollen, und nicht mehrere Wochen damit verbringen können einen Mitarbeiter abzustellen um ein mögliches Testwerkzeug zu suchen und zu validieren. Das Ergebnis des EXPS soll eine Entscheidungshilfe für die Auswahl von Performancetestwerkzeugs in verteilten Umgebungen sein. Theoretisch ist also jede verteilte Software, welche Optimierung im Bereich der Schnelligkeit, Datenverarbeitung und der Speicherverwaltung bedarf, eine mögliche AUT dieser Heuristik.

4.2 Aufbau

Wie schon im dritten Kapitel beschrieben, wurde D3Web als Gerüst genutzt um das EXPS zu entwickeln. Die vier erforderlichen Dateien wurden allgemein beschrieben und werden in den kommenden Punkten speziell im Zusammenhang mit dem EXPS vorgestellt.

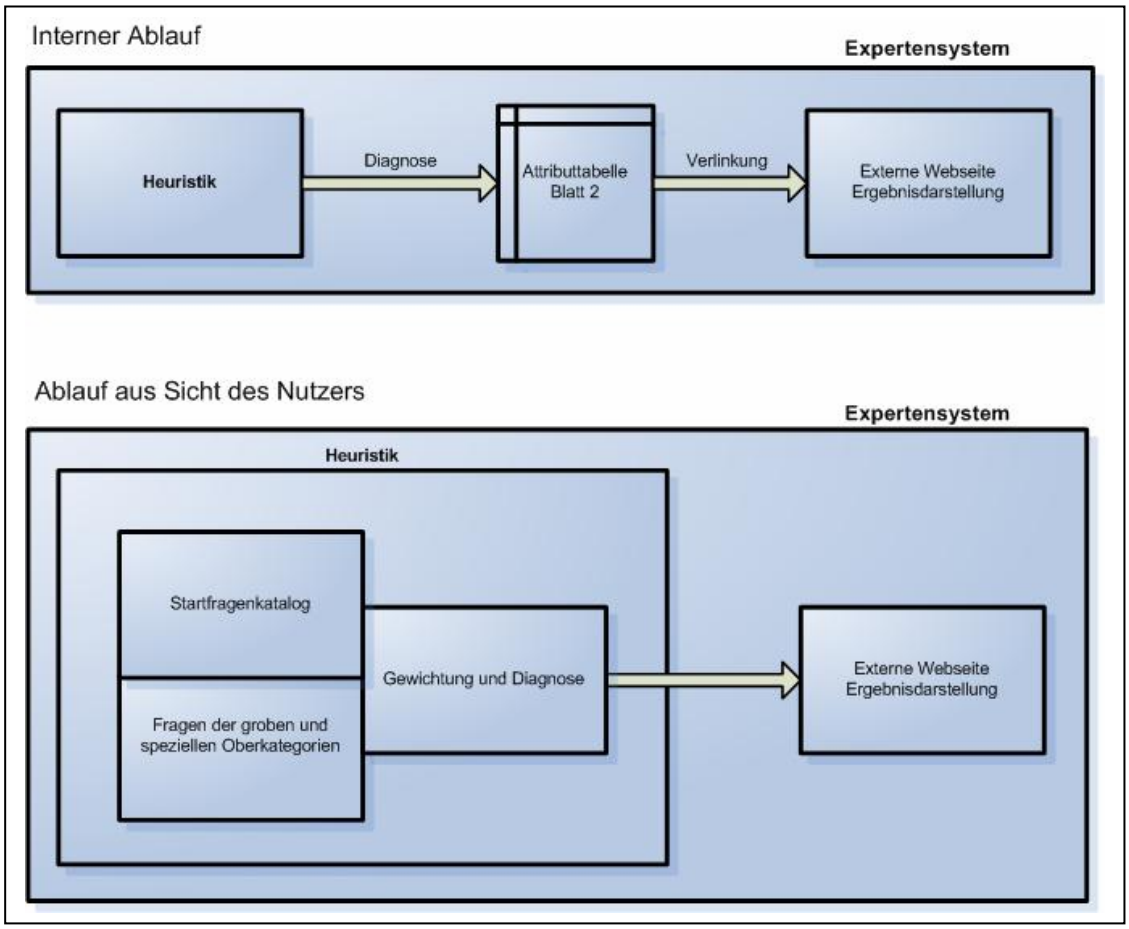


Abbildung 10: Überblick Expertensystem

Abbildung 10 zeigt einerseits den internen Ablauf des Expertensystems und anderseits den Ablauf, wie ihn der Nutzer wahrnimmt. Der interne Ablauf soll zeigen, dass die Heuristik auf Blatt 2 der Attributtabelle verweist und diese das Ergebnis weiter an eine externe Webseite liefert. Der Nutzer sieht nur, dass er nach dem Beantworten von Startfragen und Fragen zu den Oberkategorien eine Gewichtung von einem oder mehreren Ergebnissen erhält. Diese Ergebnisse sind die Ergebniskategorien a1-a27 und beinhalten eine Verlinkung auf eine externe Webseite, wo den Ergebniskategorien die Testwerkzeuge untergeordnet sind.

4.2.1 Die Fragegruppen

→ Kategorien_Testwerkzeug-Fragegruppen.txt

Den besten Überblick über den linearen Ablauf der Fragen erhält man durch die Fragegruppendatei. In dem entwickelten EXPS wird mit den Startfragen begonnen. Aus den Startfragen heraus ergeben sich dann Sprünge zu den Oberkategorien: OK_1, OK_2, …, OK_6. So wie die Oberkategorien hier deklariert worden sind, so sind diese auch im Entscheidungsdokument definiert zu finden. Die „[10]“ hinter Startfragen ist in diesem Fall ein Defaultwert. Bei einer nichtlinearen Aufzählung kann der Wert in der eckigen Klammer die Reihenfolge angeben. Da hier nur die Startfragen eine Zahl zugewiesen bekommen haben, werden sie auch zuerst aufgerufen. Eine „[9]“ würde folgerichtig vorangestellt werden. Die Startfragen befinden sich in einem Exceldokument. Ein Beispiel, wie der Zuordnungswert verwendet werden kann, ist in der Abbildung 12 zu sehen. Das Frageklassendokument ist wie ein Inhaltsverzeichnis, welches im Laufe der Erarbeitung des Entscheidungsbaumes anwächst.

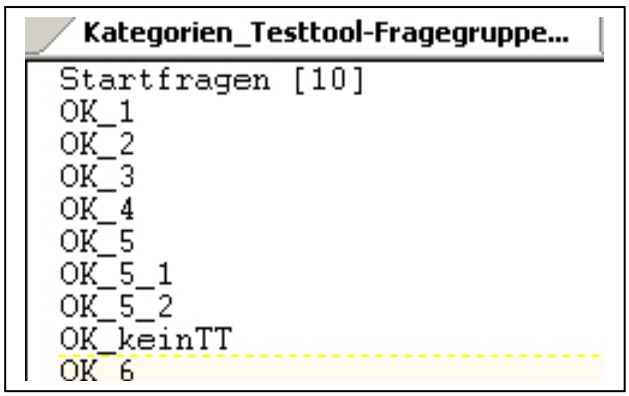


Abbildung 11: Screenshot: Kategorien_Testwerkzeug-Fragegruppen.txt

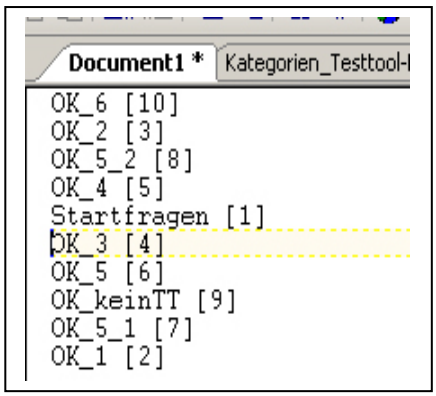


Abbildung 12: Beispiel zur Nutzung des Zuordnungs wertes

4.2.2 Die Attributtabellen

→ Kategorien_Testwerkzeug-Attributtabellen.xls

Dieses Dokument ist, wie schon beschrieben, in zwei Tabellenblätter unterteilt. Jedes Tabellenblatt hat eine andere Aufgabe. Im Tabellenblatt 1 sind die Startfragen und deren mögliche Antworten aufgeführt.

Die folgenden Aussagen beziehen sich auf Abbildung 13. In Feld C2 ist ein Text geschrieben, welcher nur eine Antwort, Feld B3, besitzt. Dieser Text dient als Information für den Nutzer und ist das Erste was später im Webinterface zu finden ist. Der Text ist der Variable A0 zu geordnet. Da die Startfragen zur ersten groben Einordnung der Kategorien gehören, sollten sie auch Fragen nach wichtigen Eigenschaften enthalten. Die drei Startfragen, zusätzlich zur Information A0, klären die Frage nach der Plattform und ob das Testwerkzeug frei verfügbar sein sollte. Die Antworten für die Variable B1 sind hier „ja“, „nein“ oder „egal“. Speziell in diesem Fragefall ist die Antwort „nein“ notwendig. Die Antwort „ja“ beinhaltet alle open source – Tools. Die Antwort „egal“ beinhaltet die Universalmenge der Testwerkzeuge. Aus versichungstechnischen Gründen sind aber kommerzielle Testwerkzeuge bei vielen Firmen oder Forschungseinrichtungen gefragt, da diese größtenteils zertifiziert sind. Bei einer Nichteinhaltung eines Zeitplans durch Fehler in einem Testwerkzeug und damit Entstehung von Mehrkosten, sind die Entwickler der Testwerkzeuge ohne Zertifizierung nicht haftbar. Die Antworten auf die Startfragen A und B2 sollten selbsterklärend sein.

	A	B	C
1			**Fragetext**
2	**A0**		Wie schon in der Dokumentation beschrieben ist eine detailierte Unters
3		Information zur Kenntnis genommen.	
4	**A**		Muss das Testtool in der Lage sein, auf jeder Plattform zu laufen ?
5		Ja, eine Unabhängigkeit ist Vorrausetzung.	
6		Nein, es wird nur ein bestimmtes Betriebssystem verwendet.	
7	**B2**		Welche Plattform bevorzugen Sie für das Testen Ihrer Applikation ?
8		Win	
9		Linux / Unix	
10	**B1**		Soll das auszuwählende Testtool Opensource sein ?
11		ja	
12		egal	
13		nein	

Abbildung 13: Tabellenblatt 1 Kategorien_Testwerkzeug-Attributtabellen.xls

4.2.3 Der Entscheidungsbaum

→ Kategorien_Testwerkzeug-Entscheidungsbaum.txt

Im Entscheidungsbaum werden alle anderen Dateien genutzt. Die Datei besteht aus circa 200 Lines of Code (LOC) und ist damit vom Umfang und vom Inhalt das größte Dokument. Um den Aufbau strukturiert zu erklären, wird als erstes in prinzipiellen und praktischen Aufbau unterschieden.

4.2.3.1 Prinzipieller Aufbau

Die Darstellung des prinzipiellen Aufbaus soll einen Überblick über den Aufbau der Datei geben. Die Abarbeitung beginnt immer mit den Startfragen, durch die gewählten Antworten ergibt sich eine Oberkategorie. Diese Oberkategorie verweißt auf eine Fragegruppe, welche dann ausgeführt wird. Die Fragegruppe enthält wiederum Fragen und vorgegebene Antworten. Das Ergebnis dieser Fragen können einerseits die Testwerkzeugkategorien und andererseits abermals andere Fragegruppen mit neuen Fragen sein. Die Fragegruppen sind in diesem Dokument definiert und wurden schon vorher - wie beschrieben – in der Datei Kategorien_Testwerkzeug-Fragegruppen.txt deklariert. Im Entscheidungsbaum befinden sich für die Startfragen nur die Variablen. Diese wurden bereits in Punkt 4.2.2 beschrieben.

Startfragen

    Oberkategorie_speziell_x1 (selektiert durch Startfragen)

        Fragegruppe_f1 (beschreibt selektierte Oberkategorie)

    Oberkategorie_speziell_x2

        Fragegruppe_f2

    …

        …

    Oberkategorie_speziell_xn

        Fragegruppe_fn

Fragegruppe_f1

    Kategorie_a1 (selektiert durch Fragegruppe_f1)

Fragegruppe_f2

    Kategorie_a2

…

    …

Fragegruppe_fn

    Kategorie_an

Kategorie_a1 = http://www.sistec.dlr.de/heuristik/#a1

Kategorie_a2 = http://www.sistec.dlr.de/heuristik/#a2

…

Kategorie_an = http://www.sistec.dlr.de/heuristik/#an

Oben ist der tatsächliche Ablauf der Entscheidungsbaums dargestellt. Das Endergebnis sind die Kategorien. Hinter diesen steht ein Link zu einer Webseite, wo null (0), eins (1) oder viele (*) Testwerkzeuge den Kategorien zugeordnet sind. Der Weg zur Findung eines möglichen Testwerkzeugs ist nebenstehend vereinfacht dargestellt und dient zur schnellen Übersicht des oben abgebildeten theoretischen Aufbaus.

Startfragen
Oberkategorie
Fragegruppe
Kategorie

4.2.3.2 Praktischer Aufbau

In diesem Unterpunkt wird der beschriebene prinzipielle Aufbau des erstellten Expertensystems mit den tatsächlichen Inhalten des fertigen Entscheidungsbaums gefüllt und erläutert. Dabei wird nicht jede Zeile erklärt werden, sondern nur auszugsweise Abschnitte ausgewählt, die zum besseren Verständnis beitragen.

Wie in Kapitel 3 schon beschrieben wird je nach Tiefe der Verzweigung mit Hilfe eines oder mehrerer Spiegelstriche eingerückt. Die Startfragensequenz sieht demnach wie folgt aus.

Kategorien_Testtool-Entscheidung...

```
1  Startfragen
2  - A0 [oc]
3  -- Information zur Kenntnis genommen.
4  --- A [oc]
5  ---- Ja, eine Unabhängigkeit ist Vorrausetzung.
6  ----- B1 [oc]
7  ------ ja
8  ------- Oberkategorie_Plattformunabhängig_Opensource (P7)
9  -------- OK_1
10 ------ egal
11 ------- Oberkategorie_Plattformunabhängig_egal (P7)
12 -------- OK_5
13 ------ nein
14 ------- a27 (P7)
15 ---- Nein, es wird nur ein bestimmtes Betriebssystem verwendet.
16 ----- B2 [oc]
17 ------ Win
18 ------- B1 [oc]
19 -------- ja
20 --------- Oberkategorie_Win_Opensource (P7)
21 ---------- OK_1
22 -------- egal
23 --------- Oberkategorie_Win_egal (P7)
24 ---------- OK_2
25 -------- nein
26 --------- Oberkategorie_Win_Kommerziell (P7)
27 ---------- OK_6
28 ------ Linux / Unix
29 ------- B1 [oc]
30 -------- ja
31 --------- Oberkategorie_Linux_Opensource (P7)
32 ---------- OK_3
33 -------- egal
34 --------- Oberkategorie_Linux_egal (P7)
35 ---------- OK_4
36 -------- nein
37 --------- Oberkategorie_Linux_Kommerziell (P7)
38 ---------- OK_keinTT
```

Abbildung 14: Auszug(nur Startfragen) aus Kategorie_Testwerkzeug_Entscheidungsbaum.txt

In **Zeile 2** wird die Variable A0 aufgerufen, an der eckigen Klammer ist der Typ der Variablen erkennbar. In diesem Fall eine Variable vom Typ oc = one choice. Das bedeutet, es kann aus einer Menge von möglichen Antworten nur eine selektiert werden. Im Dokument Kategorien_Testwerkzeug-Attributtabellen.xls wird der Variable der Inhalt zu gewiesen. Dort kann man sehen, dass es nur eine mögliche Antwort gibt. Die Frage, die hinter der Variable A0 steht dient, also lediglich zur Information für den Nutzer und kann nur mit der Antwort *„Information zur Kenntnis genommen.“* oder mit der Defaultantwort *„unbekannt“* beendet werden. Bei Nutzung der Defaultantwort wird die Heuristik ohne Wertung beendet und kann erneut gestartet werden.

Abbildung 15: Startfrage A0 bei der Ausführung im Browser

In der Startfragensequenz werden also mit Hilfe von Fragen die groben Oberkategorien gebildet. Die groben Oberkategorien setzen sich aus zwei Parametern zusammen. Beide Parameter können drei Werte annehmen.

Oberkategorie_x_y

x ∈ {plattformunabhängig, Win, Linux}

y ∈ {open source, kommerziell, egal}

Oberkategorie_Win_Opensource
Oberkategorie_Win_Kommerziell
Oberkategorie_Win_egal
Oberkategorie_Linux_Opensource
Oberkategorie_Linux_Kommerziell
Oberkategorie_Linux_egal
Oberkategorie_Plattformunabhängig_Opensource
Oberkategorie_Plattformunabhängig_Kommerziell
Oberkategorie_Plattformunabhängig_egal

Daraus ergeben sich $3^2 = 9$ mögliche grobe Oberkategorien. Davon können acht Kategorien in den Startfragen gefunden werden. Die Kategorie „Oberkategorie_Plattformunabhängig_Kommerziell“ wurde hier vernachlässigt und gleich die spezielle Kategorie a27 gesetzt. Die Recherchen ergaben, dass es wenig kommerzielle Testwerkzeuge gibt die plattformunabhängig sind, da im kommerziellen Bereich gezielt Werkzeuge für eine bestimmte Problematik entwickelt werden. Dagegen versuchen open source Werkzeuge eine große Anzahl von Nutzern zu erreichen und fächern daher die Funktionen des Tools. Das heißt keinesfalls, dass die open source Angebote deshalb besser sind.

Wenn eine grobe Oberkategorie feststeht, wird in der kommenden Zeile eine spezielle Oberkategorie (OK) aufgeführt. Diese OK's wurden beabsichtigt nicht Unterkategorie genannt, da der Name den Charakter dieser speziellen Kategorie nicht gerecht geworden wäre. In Abbildung 16 wird die Frage B1 mit „ja" beantwortet. Das gesuchte Testwerkzeug soll unter Windows laufen und open source sein. In OK_1 befinden sich Testwerkzeugkategorien welche diese Anforderungen erfüllen. Die spezielle Oberkategorie OK_1 erfüllt auch Anforderungen anderer Kategorien, deshalb konnte **Zeile 20** nicht gleich als Frageklasse genutzt werden. Beispielsweise ruft OK_2 → OK_1 auf. In OK_2 befinden sich Testwerkzeugkategorien welche Windows fordern und kommerziell und open source sein können. Demnach kann OK_1 Teilmenge von OK_2 sein und ist eine eigenständige Frageklasse. Durch die OK's soll zusätzlich Datenredundanz verhindert und der Quelltext übersichtlicher werden.

```
18 ------- B1 [oc]
19 -------- ja
20 ---------- Oberkategorie_Win_Opensource (P7)
21 ---------- OK_1
```

Abbildung 16: Auszug Startsequenz

Zeile 21 kann als eine Sprunganweisung betrachtet werden. Das Dokument wird jetzt nach der speziellen Oberkategorie OK_1 durchsucht. Die weitere Bearbeitung und Eingrenzung der Testwerkzeugkategorien erfolgt in der Fragegruppe OK_1.

```
39 OK_1
40 - Soll neben dem gegebenen verteilten Aspekt auch parallele Nutzertests mö
41 -- "Ja, das wäre schon wichtig."
42 --- Sollen auch POP, SMTP und FTP Funktionen auf Performance getestet werd
43 ---- "Ja, das wäre auch wichtig"
44 ----- a1 (P7)
45 ---- "Brauchen tue ich es nicht unbedingt, aber wenns dabei ist wäre es ok
46 ----- a2 (P7)
47 -- "Nein, das brauche ich nicht unbedingt."
48 --- Sollen auch SMTP,  POP und FTP Funktionen auf Performance getestet wer
49 ---- "ja"
50 ----- a3 (P7)
51 ---- "nein"
52 ----- a4 (P7)
```

Abbildung 17: Auszug (nur OK_1) aus Kategorie_Testwerkzeug_Entscheidungsbaum.txt

In OK_1 werden vier Ergebniskategorien unterschieden. In Zeile 44, 46, 50 und 52 sind die Kategorien a1, a2, a3 und a4 jeweils am Ende eines Zweiges zu finden. In Abbildung 17, in Kategorie a1 befinden sich beispielsweise Testwerkzeuge, welche unter Windows laufen, open source sind, bei denen parallele Nutzertest möglich sind und außerdem können auch POP, SMTP und FTP Funktionen auf Performance getestet werden.

4.2.4 Die möglichen Ergebnisse

→ Kategorie_Testwerkzeug_Ergebnisse.txt

In dieser Datei, sind alle möglichen Lösungen definiert. Sie sind strukturiert und wieder mit Spiegelstrichen eingerückt. Die speziellsten Lösungen des EXPS sind mit zwei Spiegelstrichen eingerückt. Es gibt siebenundzwanzig von diesen Lösungen und eine ergebnislose Defaultkategorie. Sechsundzwanzig der siebenundzwanzig Lösungskategorien beinhaltet zum Stand der Abgabe der Diplomarbeit mindestens ein Testwerkzeug. Die Kategorie a27 ist die einzig leere Lösungskategorie. Die Kategorie a27 soll später plattformunabhängige kommerzielle Testwerkzeuge beinhalten. Die Recherchen zu Performancetest-werkzeugen mit diesen Eigenschaften waren erfolglos oder nicht sehr vielversprechend. Es gibt Anbieter die ausschreiben kommerzielle plattformunabhängige Werkzeuge haben, aber getestet sind die Testwerkzeuge immer nur auf einer Plattform.

Kategorien_Testtool-Ergebnisse.txt

```
Oberkategorie_Plattformunabhängig_egal
- OK_5
-- a21
- OK_5_1
-- a15
-- a16
-- a17
- OK_5_2
-- a18
-- a19
-- a20
- noch kein TT
Oberkategorie_Plattformunabhängig_Opensource
Oberkategorie_Win_Opensource
- OK_1
-- a1
-- a2
-- a3
-- a4
Oberkategorie_Win_egal
- OK_2
-- a5
-- a6
-- a7
Oberkategorie_Linux_Opensource
- OK_3
-- a8
-- a9
-- a10
-- a11
-- a12
Oberkategorie_Linux_egal
- OK_4
-- a13
-- a14
Oberkategorie_Linux_Kommerziell
- OK_keinTT
-- kein Testtool gefunden
Oberkategorie_Win_Kommerziell
- OK_6
-- a22
-- a23
-- a24
-- a25
-- a26
```

Abbildung 18: Auszug Kategorie_Testwerkzeug_Ergebnisse.txt

Beispiel: „The software has only been extensively tested on Unix. “ [9]

[9] Netlogger: Software-Requirements http://www.dsd.lml.gov/Netlogger

Für eine praktische Anwendung sollten solche Testwerkzeugs nicht herangezogen werden und sind deshalb auch nur in der jeweilig getesteten Plattform eingeordnet.
Die Anzahl der Ergebniskategorien wird gleich bleiben, aber die Anzahl der zugeordneten Testwerkzeuge wird durch ständig neu entwickelte Testsoftware anwachsen. Der Punkt 4.4 „Nachträgliches Einfügen von Testwerkzeugen“ wird sich mit dieser Thematik noch genauer auseinandersetzen.

4.3 Ergebnisdarstellung / Webseite

Das Ergebnis der Heuristik ist eine der Kategorien a1- a27. Diese haben nebenstehend – wie schon im Punkt **4.2.3.1** angedeutet – einen Link. Dieser Link, ein Button mit blauem Internet-Explorersymbol – Abbildung 19 – führt auf die Webseite https://wiki.sistec.dlr.de/MartinGrohs/Heuristik/, dort sind allen 27 Ergebniskategorien Testwerkzeuge zugeordnet. In vielen Fällen kann es sein, dass das EXPS mehrere Ergebniskategorien als mögliche Lösungen auflistet. Die Ergebnisse sind aber immer nebenstehend gewichtet - Abbildung 19 -. Umso höher der Wert der Wichtung, desto mehr Eigenschaften der Universaleigenschaftsmenge alle Testwerkzeugs treffen auf die Anforderung der AUT zu. Der Wert „999“ ist default gesetzt wenn es keine Ergebniskategorie gibt, gegen die verglichen werden muss. Zu Informationszwecken ist neben der Ergebniskategorie noch die grobe Oberkategorie aufgelistet.

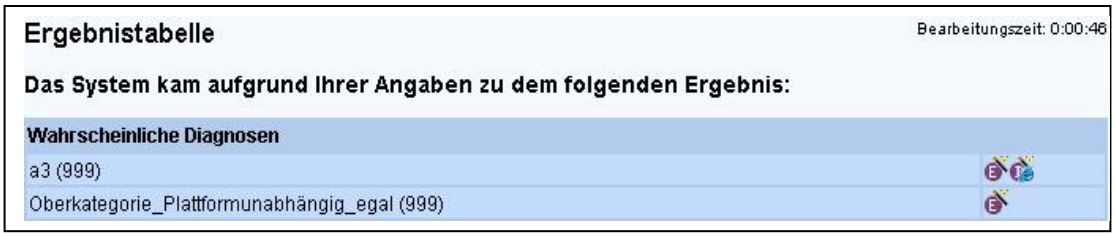


Abbildung 19: Auszug Ergebnistabelle

„Warum extra Ergebniskategorien und nicht gleich mehrere Testwerkzeuge als Lösungen in der Ergebnistabelle anzeigen lassen?“

Das Expertensystem hat den Anspruch zeitlich unabhängig zu sein und das nicht in dem es alle zwölf Monate umgeschrieben wird. Das EXPS deckt mit den Ergebniskategorien eine große Anzahl von zusammengestellten Eigenschaften ab und wird auch nicht mehr geändert. Auf der, von dem EXPS abgegrenzten, alleinstehenden

Webseite – Abbildung 20 – werden nur Namen von Testwerkzeugen den zutreffenden Ergebniskategorien zugeordnet. Die Eigenschaften von noch nicht eingeordneten Performancetestwerkzeugen werden der Universaleigenschaftsmenge zugeordnet. Daraus können Ergebniskategorien abgelesen werden.

Der Punkt 4.4 beschäftigt sich mit dem nachträglichen Einfügen von Testwerkzeugs. Durch die Möglichkeit nachträglich Testwerkzeuge einfügen zu können, wird das EXPS zeitlich unabhängig.

Abbildung 20: Auszug Testwerkzeugzuordnung auf Webseite

4.4 Nachträgliches Einfügen von Testwerkzeugen

Im Laufe der Zeit werden neue Performancetestwerkzeugs entwickelt werden. Diese Werkzeuge sind dem dann aktuellen Technikstand angepasst. Alte Versionen können erneuert werden und bieten vorteilhafte Features. Die Entwicklung von Testwerkzeugen ist ein ständiger Lernprozess, welcher auf Erfahrungen mit bereits genutzten Testwerkzeugen basiert.

Das Expertensystem bietet die Möglichkeit, mit wenig Aufwand, neu entstandene Performancetestwerkzeuge in den Abfragemechanismus zu integrieren. Dabei muss weder der Entscheidungsbaum, noch eine andere Datei der Heuristik geändert oder bearbeitet werden. Die Heuristik, so wie sie zum Zeitpunkt der Abgabe dieser Diplomarbeit vorhanden ist, muss nicht mehr „angefasst“ werden. Um dieses zu gewährleisten wurde - wie im Punkt 4.3 beschrieben - eine vom EXPS unabhängige Webseite als Endergebnisformat gewählt. Der Output der Heuristik sind die

Kategorien a1 – a27. Diese sind gleichzeitig der benötigte Input der Webseite. Die Heuristik und die Webseite zusammen ergeben das Expertensystem.

Jede Ergebniskategorie der Heuristik hat eine eigene Charakteristik. Bestimmte Eigenschaften der Universaleigenschaftsmenge bilden dabei eine Charakteristik.

3	Eigenschaften / Kategorien	
4	Windows (NTFS)	x
6	plattformunabhängig	x
7	open source	x
8	kommerziell	x
10	Parallele Nutzeranzahl bei Ø Serverhardware > 1600	x
12	einfache Bedienung	egal
13	komplizierte Bedienung	egal
16	einfache Installation	egal
17	komplizierte Installation	egal
44	Datenbanktests möglich	x

Abbildung 21: Beispieleigenschaften einer Ergebniskategorie

Die, in Abbildung 21 dargestellte, zufällig erwählte Ergebniskategorie beinhaltet alle Testwerkzeuge die unter Windows laufen oder plattformunabhängig sind. Ausgegrenzt sind somit die Werkezuge, die nur unter Linux / Unix laufen. In **Zeile 7 & 8** wird gezeigt, dass es egal ist, ob das Testwerkzeug open source oder kommerziell ist. Zwischen kommerziell oder open source wurde nicht unterschieden, d.h. das alle Testwerkzeuge, welche die Eigenschaften der **Zeile 4 & 6** erfüllen, noch mögliche Testwerkzeuge der dargestellten Ergebniskategorie sein können. In den folgenden Zeilen der Abbildung 21 werden weitere Eigenschaften der Kategorie aufgezeigt.

Neue Testwerkzeuge oder neuere Versionen eines bekannten Testwerkzeugs, werden über ihre Eigenschaften einer Ergebniskategorie zugeordnet. Dabei werden die spezifischen Eigenschaften des Werkzeugs in die Universaleigenschaftsmenge

- Abbildung 3 - eingeordnet. Danach kann abgelesen werden zu welcher Kategorie das neue Testwerkzeug gehört. Natürlich ist es auch möglich, dass ein Testwerkzeug in mehreren Kategorien auftaucht. Diese Werkzeuge sind entweder - wie schon im Punkt 4.2.3.2 angesprochen - weit gefächerte open source Testwerkzeuge oder Werkzeuge welche wenig, aber dafür allgemeine Basiseigenschaften beinhalten.

Im Ausblick werden, im Zusammenhang mit dem Hinzufügen von Performancetestwerkzeugen, noch Ideen erläutert um einen Automatismus einzubauen. Der Sinn dieses Automatismus wäre eine Aufwandsminimierung.

4.4.1 Beispielszenario

Das Beispielszenario soll zeigen wie ein beliebiges Performancetestwerkzeug den Ergebniskategorien zugeordnet werden kann. In diesem Szenario wird vorausgesetzt das die Eigenschaften des zu zuordnenden Werkzeuges bekannt sind. Diese Eigenschaften werden der Matrix der Universaleigenschaftsmenge hinzugefügt. Dadurch kann das Testwerkzeug mit den existierenden Ergebniskategorien verglichen werden.

Das fiktive Bespieltestwerkzeug ist für Datenbankperformancemessungen entwickelt worden. Es kann unter jedem Betriebssystem genutzt werden und ist frei erhältlich. Es können mehr als 1600 parallel Benutzer simuliert werden. Außerdem ist die Installation und die Bedienung leicht zu verstehen und bedarf keiner weiteren Schulung. Ein, für eine kleine Firma, optimales Testwerkzeug. Es sind keine großen Einarbeitungszeiten von Nöten sind und das Paket ist kostenlos.

			a1	a2	a3	a4	a5	a6	a7
2	Eigenschaften / Kategorien	Eigenschaften eines Performancetesttools							
3	Windows (NTFS)						x	x	x
4	Linux / Unix								
5	plattformunabhängig	x	x	x	x	x	x	x	x
6	open source	x	x	x	x	x	x	x	x
7	kommerziell						x	x	x
8	Parallele Nutzeranzahl bei Ø Serverhardware < 1600								
9	Parallele Nutzeranzahl bei Ø Serverhardware > 1600	x					x	x	x
10	graphische Oberfläche Steuerungsinstanz vorhanden								
11	einfache Bedienung	x					x	egal	egal
12	komplizierte Bedienung							egal	egal
13	Steuerung der Clients von einem Rechner								
14	manuelle Steuerung der Clients								
15	einfache Installation	x						egal	egal
16	komplizierte Installation							egal	egal
17	Scripting								
18	Nutzung/Erstellung von Testskripten möglich								
19	Monitoring von Datendurchsatz								
20	Monitoring von Antwortzeiten								
21	Monitoring von Speicherverbrauch								
22	Darstellung der Grafiken zur Laufzeit								
23	Ergebnissformat *.csv, *.html, *.xml oder *.pdf								
24	Ergebnissformat *.log								
25	Auswertung von Ergebnissen includiert								
26	Kommunikation über TCP/UDP								
27	Corba MW								
28	RMI MW								
29	SOAP MW								
30	andere MW								
31	POP		x	egal	x				
32	SMTP		x	egal	x				
33	FTP		x	egal	x				
34	HTTP/HTTPS								
35	parallele Nutzertests		x	x	egal	egal			
36	keine parallelen Nutzertests				egal	egal			
37	mehrere Lastclients (verteilt)								
38	nicht verteilt								
39	Blackbox Sicht								
40	Whitebox Sicht								
41	Cookies unterstützt								
42	Cookies nicht unterstützt								
43	Datenbanktests möglich	x						x	egal
44	Mainframetests								
45	gängige CRM Systeme								

Abbildung 22: fiktive Eigenschaften eines Testwerkzeugs

Der Auszug in Abbildung 22 zeigt die Einordnung der Eigenschaften in die Eigenschaftsmatrix.

Der rechte Teil stellt sieben von siebenundzwanzig Ergebniskategorien dar. Abgelesen werden kann, dass das fiktive Testwerkzeug Eigenschaften der Kategorien a6 und a7 aufweißt. Das Testwerkzeug kann 100% untergeordnet werden, da alle sechs Eigenschaften in einer oder mehreren Kategorien auftreten. Dennoch erfüllt das Testwerkzeug nicht zu 100% die Charakteristik einer, der dargestellten, Ergebniskategorien. Die Ergebniskategorien beinhalten mehr Eigenschaften als das fiktive Testwerkzeug. Es ist somit möglich, dass wenn die Kategorien a6 oder a7 Ergebnisse der Heuristik sind, das fiktive Testwerkzeug nicht 100% die Anforderungen der Benutzer erfüllt. Das ist ein Beispiel warum es schwierig ist, ein Testwerkzeug zu finden welches wirklich alle Anforderungen erfüllt. Durch eine

ständig anwachsende Zahl von Testwerkzeugen, wird die Chance jedoch größer ein immer mehr zutreffendes Performancetestwerkzeug zu finden. Neben der Aktualität des EXPS, ist die dadurch wachsende Auswahl an Testwerkzeugen der Hauptgrund für die Berechtigung des nachträglichen Einfügens von Performancetestwerkzeugs.

KAPITEL 5

Anwendung

Dieses Kapitel beschreibt die Anwendung des Expertensystems, zeigt die Formatvorlage für die Nutzer des EXPS und stellt eine Orientierungshilfe für einen Testplan bereit.

5.1 Starten des Webinterfaces

Es gibt zwei Möglichkeiten das EXPS zu starten, einmal können über D3Web Quellcodedateien importiert und dann die Webanwendung gestartet werden. Diese Möglichkeit soll aber für die Benutzern nicht angeboten werden, da sie sonst zugriff auf die Quellcodedateien hätten. Zusätzlich setzt dieser Weg ein gewisses Grundwissen über D3Web und vor allem die Software D3Web an sich voraus.

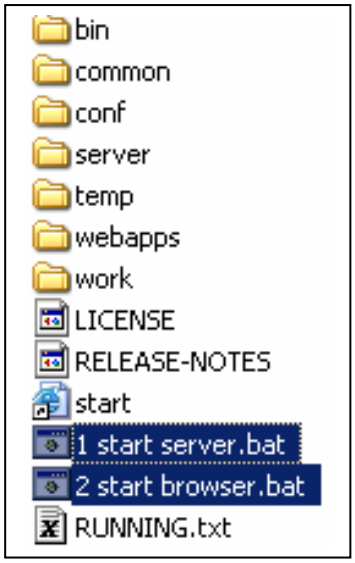


Abbildung 23: exportiertes EXPS

Genauso wie Dateien importiert werden können, gibt es auch den Weg einen Dialogserver zu exportieren. Das Ergebnis des Exportvorgangs ist ein circa neun Megabyte großer Ordner. Dieser Ordner enthält die übersetzen Dateien und die dazugehörige nummerierte Server- und Browseranwendung. Vorteilhaft ist, dass D3Web bei diesem Weg nicht nötig ist um das Expertensystem zu starten. Nach dem Starten der 

kann das Webinterface mit der 

aufgerufen werden.

5.1.1 Voraussetzungen

Um den Server zu starten zu können muss Java auf dem Anwendungssystem vorhanden sein und das System muss wissen wo Java zu finden ist. Das heißt die Umgebungsvariable JAVA_HOME muss gesetzt sein. Getestet wurde mit der Java Version 1.5.

Als Vorrausetzung für die Nutzung eines Browsers wird in der Dokumentation von D3Web der Internet Explorer Version 5.5 oder höher genannt. Ich habe aber mit Opera oder Mozilla keine schlechteren Erfahrungen gemacht. Lediglich die Darstellung ist etwas unterschiedlich und die Popupeinstellungen müssen generell angepasst sein. Das EXPS wird automatisch mit dem festgelegten Standardbrowser des Systems geöffnet.

5.2 Nutzen des Webinterfaces

In diesem Abschnitt möchte ich nicht genau auf die einzelnen sichtbaren Funktionalitäten von D3Web im Webinterface eingehen, sondern nur die für den Nutzer interessanten Funktionen ansprechen. Nach dem Starten des EXPS öffnet sich ein Webinterface. Dieses Webinterface wird in vier Zonen eingeteilt werden um dieses besser zu beschreiben.

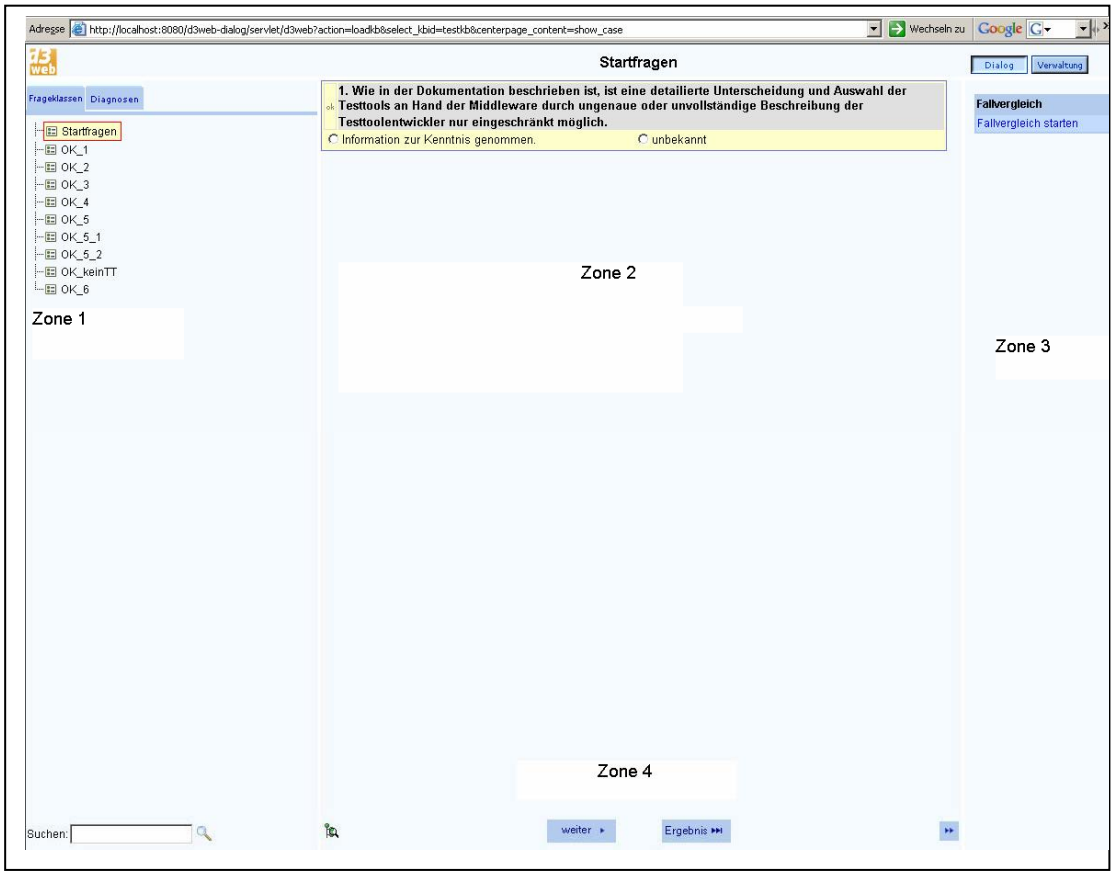


Abbildung 24: Webinterface Start

5.2.1 Zone 1

Zone 1 ähnelt einer Ordnerstruktur und gibt dem Nutzer einen Überblick, in welcher Kategorie er sich, auf Grund seiner Antworten, aktuell befindet. Im Reiter 

findet man die möglichen Ergebnisse. Die groben und speziellen Oberkategorie, sowie die Ergebniskategorien sind dort zu finden. Diese Zone, kann also im Augenwinkel mit verfolgt werden und dient als Information.

5.2.2 Zone 2

Zone 2 ist der Hauptteil der Seite. Hier werden dem Nutzer Fragen gestellt und verschiedene Antworten vorgegeben. Als erstes werden die Startfragen abgearbeitet. Zu Beginn der Startfragen dient eine Information als Eröffnung des Webinterfaces.

Abbildung 25: Beginn der Startfragen

Die Auswahlmöglichkeit 

ist eine Defaultantwort und beendet den Entscheidungsvorgang. Auf Grund der bis dahin gesammelten Infos wird eine Diagnose erstellt. In diesem Falle würde keine passende Ergebniskategorie gefunden werden, da die Universalmenge noch nicht eingegrenzt worden ist.

5.2.3 Zone 3

In Zone 3 dient der Reiter Verwaltung dazu in Zone 1 ein neues Menü zu schaffen, indem es möglich ist den aktuellen Entscheidungsfall zu speichern, zu laden, verschiedene Einstellungen vorzunehmen oder Wissensbasen zu laden. Wissensbasen sind in dem EXPS nicht Importiert, sie können beispielsweise Beschreibungen und Informationen über die Heuristiken enthalten. Interessant für den Nutzer ist in diesem Menü damit nur der Reiter „Fälle“.

Abbildung 26: Verwaltung Fälle Zone 1

5.2.4 Zone 4

In Zone 4 gibt es mehrere Möglichkeiten die in Zone 2 selektierte Antwort zu verwalten.

Der Button weiter bestätigt die Selektion und ruft die nächste Frage auf. Die jeweils nächste Frage bestimmt dabei der Entscheidungsalgorithmus, welcher in Kapitel 3 & 4 beschrieben wurde. Von Frage zu Frage füllt sich die Zone 2 mit schon beantworteten Fragen, diese haben einen weißen Hintergrund. Die aktuell zutreffende Entscheidung ist gelb hinterlegt. Sobald in eine andere spezielle Oberkategorie gesprungen wird löscht sich Zone 2 mit der Fragenhistorie und stellt nur noch die Fragen und Antworten der nun neu aktiven Kategorie dar.

Abbildung 27: Hintergrundfarbgebung am Beispiel der Startfragenhistorie

Der Button 

bricht den Entscheidungsvorgang ab und springt zur Diagnose. Die Funktion ist äquivalent zur Auswahlmöglichkeit 

. Es kann jedoch sein, dass noch keine Eingrenzung der Universalmenge getroffen worden ist und dadurch keine Kategorie als Ergebnis empfohlen werden kann.

5.3 Formatvorlage und Testplan für Nutzer des Expertensystems

Die Formatvorlage dient einerseits als Feedback und als Plattform für Verbesserungsvorschläge des entwickelten Expertensystems und anderseits auch als Zusammenfassung der AUT.
Die Zusammenfassung der AUT hilft auch dem Tester noch mal einen Überblick über die zu testende Software zu bekommen und sich die Anforderungen an das Testwerkzeug vor Augen zu führen. Dies soll mit dem Ausfüllen der Eigenschaftsmatrix unterstützt werden. Weiterhin wird ein Default-Testplan zur Erstellung von Testfällen bereitgestellt der dem weniger erfahrenen Tester als Leitfaden oder Orientierungshilfe dienen kann.

5.3.1 Eigenschaftsmatrix – AUT

Die folgend aufgezeigten Eigenschaften sind mögliche Entscheidungsfragen welche im erstellten EXPS als Entscheidungskriterium auftreten können. Der Tester sollte diese Matrix nutzen um die Eigenschaften seiner AUT einzutragen. Verschiedene Teilmengen dieser Eigenschaften ergeben die jeweiligen speziellen Test-tooloberkategorien des EXPS's. Dadurch kann nach der Nutzung des EXPS's verglichen werden ob ein Zusammenhang zwischen den Eigenschaften der AUT und den Anforderungen an ein Testwerkzeug besteht. Weitere Untersuchungen oder Forschungen auf diesem Gebiet werden durch dieses Feedback möglich gemacht. Ein nützlicher Nebeneffekt ist außerdem, dass der Tester, welcher ja nicht immer der Entwickler der zu testenden Software ist, einen Überblick über die Anwendung bekommt.

Windows (NTFS)	
Linux	
Plattformunabhängigkeit	
Opensource	
Kommerziell	
Parallele Nutzeranzahl bei Ø Serverhardware < 1600	
Parallele Nutzeranzahl bei Ø Serverhardware > 1600	
graphische Oberfläche Steuerungsinstanz vorhanden	
einfache Bedienung	
komplizierte Bedienung	
Steuerung der Clients von einem Rechner	
manuelle Steuerung der Clients	
einfache Installation	
komplizierte Installation	
Kommunikation über TCP/UDP	
Corba MW	
RMI MW	
SOAP MW	
andere MW	
POP	
SMTP	
FTP	
HTTP/HTTPS	

Abbildung 28: Eigenschaftsmatrix für Anwender

5.3.2 Testplan – eine Orientierungshilfe

Wenn ein Testwerkzeug gefunden und Kenntnisse darüber erworben sind, sollte sich der oder die Tester Gedanken um sinnvolle Testfälle machen. Diese sollten neben dem einheitlichen Format einen aussagekräftigen Namen und wieder erkennbare Ordnerstrukturen besitzen.

5.3.2.1 Testobjekte

Hier sollte eine kurze Beschreibung des Teils oder der Komponente der AUT stehen. Dabei sollte diese Beschreibung so kurz, aber speziell wie möglich ausfallen um einen Überblick zu gewinnen was in diesem Testfall getestet wird.

5.3.2.2 zu testende Merkmale

1. ob mit dem System noch zu arbeiten ist
2. ob sich das Antwortzeitverhalten ändert
3. kommt es zu undefinierten Verhalten
4. …

5.3.2.3 nicht zu testende Merkmale

Um diesen Testfall von ähnlichen Tests abzugrenzen kann hier beschrieben werden welche Merkmale evtl. testbar wären, aber in diesem Zusammenhang nicht gestestet werden.

5.3.2.4 Testvorgehen (formal)

1. Ausgangszustand festlegen
2. Test beschreiben
3. Testeingaben
4. erwartetes Ergebnis
5. tatsächliches Ergebnis

5.3.2.5 Testvorgehen (praktisch)

Das Testvorgehen ist vom Testwerkzeug abhängig, aber als erstes sollte für das Testwerkzeug und zu testende Software ein wiederholbarer Ausgangszustand festgelegt werden.

5.3.2.6 Dokumente des Testprozesses

- Testfall → TF_xx.pdf (xx = numerischer Wert)
- automatisierter Testfall der Testsuite (des genutzten Testwerkzeugs)
- Reportdatei

5.3.2.7 Testszenario

Ein Testszenario beinhaltet oft mehrere Testfälle. Ein Anwendungsbeispiel wird getestet, dabei werden mehrere Funktionen oder sogar Komponenten genutzt. Hier kann beschrieben werden, ob der Testfall einem Testszenario unterzuordnen ist.

5.3.2.8 Testumgebung

Eine knappe Beschreibung der Testumgebung. Dazu gehören auch Version des Testwerkzeugs und Systeminformationen des genutzten Testsystems. Bei bestimmten Testsystemen ist auch der Standort von Wichtigkeit.

KAPITEL 6

6.1 Erkenntnisse

Das gesamte Thema Performancetest, ist sehr interessant und bietet viele Schnittstellen zu benachbarten Themengebieten, wie beispielsweise der Profiling- und Grenzwertanalyse. Ein kategoriebasiertes Expertensystem für Performancetestwerkzeuge bietet eine Zeiteinsparung bei der Suche nach einen geeigneten Testwerkzeug für eine bestimmte Software. Solange neue Werkzeuge regelmäßig den bestehenden Kategorien untergeordnet werden, hat diese Arbeit eine Zukunft und kann noch in mehreren Jahren genutzt werden.

6.2 Ausblick

Um das Einpflegen von neuen Testwerkzeugen einfacher zu machen, ist es von Vorteil wenn die Entwickler eines neuen Performancetestwerkzeugs ein standardisiertes Dokument vorliegen haben. Dort könnten sie die Eigenschaften ihres Testwerkzeugs eintragen. Für die Entwickler ist das eine Aufgabe die in kurzer Zeit erledigt werden kann. Für Leute die dieses neue Testwerkzeug nicht kennen, bedarf es viel Zeit an Recherche. Dieses Standarddokument könnte dann von einem Automatismus eingelesen und ausgewertet werden. Ein weiterer Automatismus, welcher Schreibrechte auf der Ergebnis darstellenden Webseite hat, ordnet das Performancetestwerkzeug den Kategorien zu.

Die Idee des Standarddokuments liegt nicht sehr fern. Das Performancetestplugin TPTP von Eclipse wurde von Herrn Eldar Sultanov in einer IX Ausgabe[10] vorgestellt. Auf Anfrage füllte Herr Sultanov, welcher in dem Falle Experte über ein neues Performancetestwerkzeug ist, ein Eigenschaftsdokument aus. Mit Hilfe dieses Dokumentes konnte TPTP leicht und in kurzer Zeit den Kategorien untergeordnet werden. Ein Weg könnte demnach sein, entwickelnde Firmen das Expertensystem also Werbeplattform für ihr Testwerkzeug anzubieten. Im Gegenzug erhält das Expertensystem die zugeordneten Eigenschaften des neuen Performancetestwerkzeugs direkt von den Entwicklern.

Vorstellbar wäre auch, diese Arbeit als Teilprodukt eines generellen Testexpertensystems zu betrachten. Ein Expertensystem, welches die Tester mit Testwerkzeugen für jedes Testverfahren versorgt. Mehrere Heuristiken zu den Themen Modeltest, Komponententest und Systemtest könnten erstell werden und in einer Heuristik auf einer höheren Ebene verwaltet werden. Dazu könnten Testpläne und Auswertungen von Anforderungsdokumenten gehören.

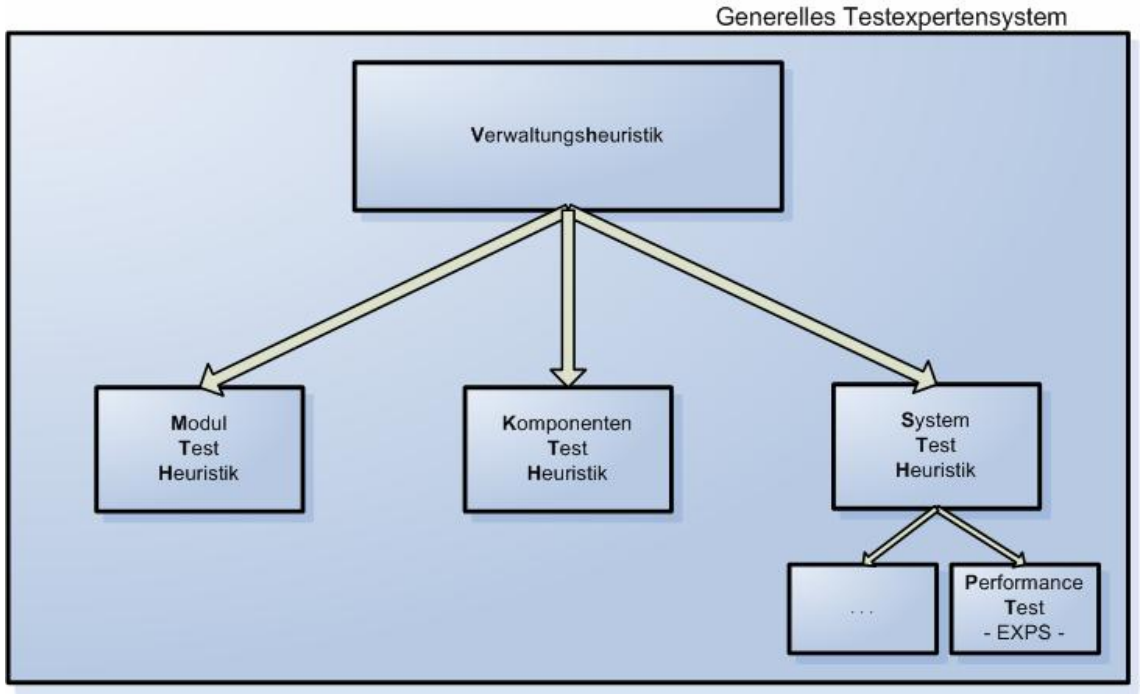


Abbildung 29: generelles Testexpertensystem

[10] ix Ausgabe 9 - September 2007 - Seite 82ff - http://www.heise.de/ix

Literaturverzeichnis

Bücher

[My95] Myers, G.J.: Methodisches Testen von Programmen
(5.Auflage)

[Th02] Thaller, G.E.: Software-Test
(Verifikation und Validation)

[Ba98] Balzert, H.: Lehrbuch der Software-Technik
(Software-Management, Software-Qualitätssicherung,
Unternehmensmodellierung)

[FG99] Fewster, M. & Dorothy, G.: Software Test Automation
(Effective use of test execution tools)

[Fo05] Fowler, M.: Refactoring
(oder: wie Sie das Design vorhandener Software verbessern)

[SW95] Simon, A.R. & Wheeler, T.: Open Client/Server
(Computing and Middleware)

[As07] Ascheron, C.: Die Kunst des wissenschaftlichen Präsentierens und
Publizierens

Internetquellen

[1] Fraunhofer IESE: Verfahren – Black-Box test
http://www.software-kompetenz.de/?10142
15.05.2007 / 13:20

[2] Computerwoche: Wege zum Performance-Test
http://www.computerwoche.de/produkte_technik/software/591754/index3.html
24.05.2007 / 13:35

[3] Mark Aberdour: Performance test tools
http://www.opensourcetesting.org/performance.php
16.08.2007 / 13:00

[4] jython contributors: The Jython Project
http://www.jython.org/Project/index.html
20.11.2007 / 16:02

[5] Hitachi: Diagnoseprodukt
http://hitachi.de/products/business.jsp?
sectionid=12&catid=136&productid=1292&marker=2
Beispiel für ein medizinisches Diagnosesystem.
19.11.2007 / 13:12

[6] ServiceXpert: Diagnose-Center
http://www.servicexpert.de/index.php?id=27
Beispiel für ein Diagnosesystem in Automobilen
19.11.2007 / 12:40

[7] Universität Würzburg / iisy / denkbar: Diagnosesystem D3Web
http://d3.informatik.uni-wuerzburg.de/
24.09.2007 / 12:23

[8] Universität Würzburg / iisy / denkbar: Produktbeschreibung D3Web
http://d3web.informatik.uni-wuerzburg.de/product/index_dt.htm
24.09.2007 / 13:26

[9] Netlogger: Software-Requirements
dsd.lbl.gov/NetLogger/documentation/netlogger-manual.pdf
www-didc.lbl.gov/DPSS/logging/documentation/netlogger-manual/Software-Requirements.html
25.09.2007 / 14:01

[10] ix: Magazin für professionelle Informationstechnik.
Artikel über TPTP
http://www.heise.de/kiosk/archiv/ix/2007/9/82_Bessere-Software-mit-Eclipse-TPTP
5.12.2007 / 13:01

Abbildungsverzeichnis

ANHANG

A Entscheidungsbaum – Datei (D3Web)

Startfragen

- A0 [oc]

-- Information zur Kenntnis genommen.

--- A [oc]

---- Ja, eine Unabhängigkeit ist Vorrausetzung.

----- B1 [oc]

------ ja

------- Oberkategorie_Plattformunabhängig_Opensource (P7)

-------- OK_1

------ egal

------- Oberkategorie_Plattformunabhängig_egal (P7)

-------- OK_5

------ nein

------- a27 (P7)

---- Nein, es wird nur ein bestimmtes Betriebssystem verwendet.

----- B2 [oc]

------ Win

------- B1 [oc]

-------- ja

--------- Oberkategorie_Win_Opensource (P7)

---------- OK_1

-------- egal

--------- Oberkategorie_Win_egal (P7)

---------- OK_2

-------- nein

--------- Oberkategorie_Win_Kommerziell (P7)

---------- OK_6
------ Linux / Unix
------- B1 [oc]
-------- ja
--------- Oberkategorie_Linux_Opensource (P7)
---------- OK_3
-------- egal
--------- Oberkategorie_Linux_egal (P7)
---------- OK_4
-------- nein
--------- Oberkategorie_Linux_Kommerziell (P7)
---------- OK_keinTT
OK_1
- Soll neben dem gegebenen verteilten Aspekt auch parallele Nutzertests möglich sein ? [oc]
-- "Ja, das wäre schon wichtig."
--- Sollen auch POP, SMTP und FTP Funktionen auf Performance getestet werden ? [oc]
---- "Ja, das wäre auch wichtig"
----- a1 (P7)
---- "Brauchen tue ich es nicht unbedingt, aber wenns dabei ist wäre es ok."
----- a2 (P7)
-- "Nein, das brauche ich nicht unbedingt."
--- Sollen auch SMTP, POP und FTP Funktionen auf Performance getestet werden ? [oc]
---- "ja"
----- a3 (P7)
---- "nein"
----- a4 (P7)
OK_2
- Sollten mehr als 1600 virtuelle parallele Nutzer möglich sein ? [oc]
-- "Ja"
--- Ist Ihnen eine einfache Bedienung und leichte Installation des Testtools wichtig oder würde Sie auch eine Schulung zur Bedienung in Kauf nehmen ? [oc]
---- "Einfache Bedienung und leichte Installation wäre wichtig."
----- a5 (P7)
---- "Eine Schulung zur Bedienung und Installation des Testtools wäre kein Problem."

----- Falls Sie eine Anbindung an eine Datenbank haben, wollen Sie diese Performance auch Prüfen ? [oc]
------ "Ja das sollte das Testtool können!"
------- a6 (P7)
------ "Das wäre ein gutes Feature, aber brauchen würde ich es nicht unbedingt."
------- a7 (P7)
-- "Nein"
--- OK_1
OK_3
- Sollte neben dem gegebenen verteilten Aspekt auch parallele Nutzertests möglich sein ? [oc]
-- "Ja das wäre schon wichtig."
--- Sollen auch PoP, SMTP und FTP Funktionen auf Performance getestet werden ? [oc]
---- "Ja, das wäre auch wichtig"
----- a8 (P7)
---- "Brauchen tue ich es nicht unbedingt, aber wenns dabei ist wäre es ok."
----- a9 (P7)
-- "Nein, das brauche ich nicht unbedingt."
--- Sollen auch PoP, SMTP und FTP Funktionen auf Performance getestet werden ? [jn]
---- "Ja"
----- a10 (P7)
---- "Nein"
----- Spielt für Sie neben dem Monitoring von Datendurchsatz und Antwortzeit zusätzlich das Monitoring vom Speicherverbrauch eine Rolle ? [jn]
------ "Ja"
------- a11 (P7)
------ "Nein"
------- a12 (P7)
OK_4
- Sollte Ihr Testtool eine grafische Bedienoberfläche haben ? [oc]
-- "Ja, das ist zwingend notwendig."
--- OK_1
-- "Nein, das wäre nicht zwingend notwendig."
--- Wollen Sie zu einem beliebigen Zeitpunkt eine Aussage über den Systemzustand Ihrer Software haben ? [oc]

---- "Das wäre von Vorteil, ist aber kein Auswahlkriterium für uns."

----- Zusätzlich zu den Schnittstellen für Scriptsprachen, ist auch die Aufnahme von Testfällen von Notwendigkeit. [jn]

------ "ja"

------- a13 (P7)

-------- OK_1

------ "Nein"

------- a14 (P7)

---- "Das wäre von Vorteil."

----- OK_3

OK_5_1

----- Um zu jedem Zeitpunkt über Ihr System Bescheid zu wissen, ist auch das Monitoring zur Laufzeit möglich. Brauchen Sie diese Informationen ? [oc]

------ "ja"

------- a15 (P7)

------ "nein"

------- Nutzen Sie das File Transfer Protokoll in Ihrer Anwendung ? [oc]

-------- "ja"

---------a16 (P7)

-------- "nein"

--------- a17 (P7)

OK_5_2

- Um zu jedem Zeitpunkt über Ihr System bescheid zu wissen, ist auch das Monitoring zur Laufzeit möglich. Brauchen Sie zusätzlich diese Informationen ? [oc]

-- "ja"

--- a18 (P7)

-- "wäre schön zu haben, aber kein Entscheidungsgrund"

--- Nutzen Sie das File Transfer Protokoll in Ihrer Software ? [oc]

---- "ja"

----- a19 (P7)

---- "nein"

----- a20 (P7)

OK_5

- Das Monitoring von Antwortzeiten unterstützen alle Testtools dieser Kategorie. Zusätzlich können Sie noch Datendurchsatz und Speicherverbauch angezeigt bekommen. [oc]
-- "zusätzlich Datendurchsatz und Speicherverbauch"
--- OK_5_1
-- "zusätzlich Speicherverbrauch"
--- OK_5_1
-- "zusätzlich Datendurchsatz"
--- OK_5_2
-- "Das Monitoring der Antwortzeit reicht mir"
--- OK_1
--- Mit Ihrem Testtool ist das Erstellen von Testszenarien / Testfällen möglich. Benötigen Sie zusätzlich noch eine Schnittstelle für Scriptsprachen ? [oc]
---- "ja"
----- OK_1
---- "nein"
----- Für Sourcecode-Fremde ist es einfacher die zu testende Software nur aus Blackbox-Sicht betrachten zu müssen. Bevorzugen Sie ein Testtool was auf Blackboxebene testet ? [oc]
------ "Das ist mir egal, ich habe genaue Kenntnisse über den Aufbau der Software."
------- OK_1
------ "Ja, das wäre von Vorteil."
------- a21 (P7)

OK_6
- Ein paar Testtools können bei durchschnittlicher Systemhardware, Stand 2006, mehr als 1600 virtuelle Benutzer parallel simulieren. [oc]
-- "Das wäre für mein Projekt von Vorteil."
--- Die Bedienung und Installation von Performancetesttools, gerade im kommerziellen Bereich, ist oftmals ohne Schulung kaum möglich und wird in bestimmten Fällen explizit von Testtoolanbietern empfohlen. [oc]
---- "Ein Testtool welches eine extra Schulung benötigt kommt nicht in Frage."
----- a23 (P7)
---- "Eine Schulung wäre kein Problem. Dafür sind genügend Ressourcen vorhanden."
----- Neben der Steuerung alle Clients von einem Rechner, muss es auch eine GUI geben. [oc]
------ "Ja, eine GUI ist ein Pluspunkt."

------- a22 (P7)
------ "Eine GUI ist kein Entscheidungskriterium."
------- Meine Software beschäftigt sich u.a. mit dem Verwalten von Kunden. Das Testtool sollte also neben Datenbanktests auch CRM Systeme, CustomerRelationshipModel, unterstützen. [oc]
-------- "Ja Datenbankperformancetests reichen nicht aus."
--------- a23 (P7)
-------- "Datenbanktests reichen zu diesem Thema vollkommen aus."
--------- a24 (P3)
-- "So viele Benutzer wird meine Software nie gleichzeitig haben."
--- a25 (P7)

OK_keinTT
- Leider haben wir kein Testtool für Ihre Anforderungen gefunden. [oc]
-- " "

B Diagnose – Datei (D3Web)

a27

Oberkategorie_Plattformunabhängig_egal

- OK_5

-- a21

- OK_5_1

-- a15

-- a16

-- a17

- OK_5_2

-- a18

-- a19

-- a20

- noch kein TT

Oberkategorie_Plattformunabhängig_Opensource

Oberkategorie_Win_Opensource

- OK_1

-- a1

-- a2

-- a3

-- a4

Oberkategorie_Win_egal

- OK_2

-- a5

-- a6

-- a7

Oberkategorie_Linux_Opensource

- OK_3

-- a8

-- a9

-- a10

-- a11

-- a12

Oberkategorie_Linux_egal

- OK_4

-- a13

-- a14

Oberkategorie_Linux_Kommerziell

- OK_keinTT

-- kein Testtool gefunden

Oberkategorie_Win_Kommerziell

- OK_6

-- a22

-- a23

-- a24

-- a25

-- a26

C Fragegruppen – Datei (D3Web)

Startfragen [10]

OK_1

OK_2

OK_3

OK_4

OK_5

OK_5_1

OK_5_2

OK_keinTT

OK_6

D CD mit Expertensystem