DHBW
Duale Hochschule
Baden-Württemberg
Stuttgart

KOS.content

01 | 2012

Ergebnisse der Untersuchungen des
Kompetenzzentrum Open Source der DHBW-Stuttgart

Sommer 2012
band.2

INHALT BAND.2

Inhalt __

Das Kompetenzzentrum Open Source (KOS)

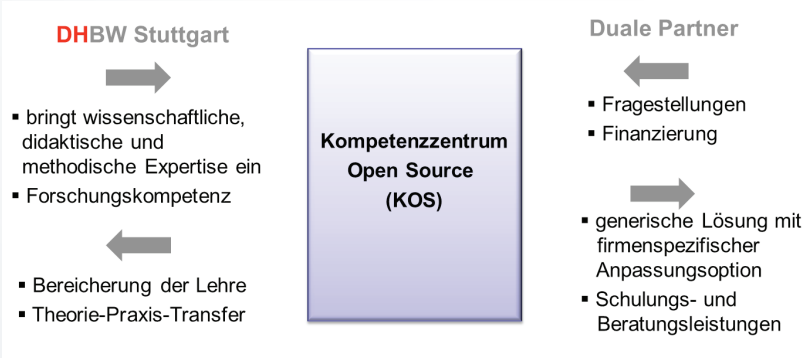
Ziel des Projektes

Das Projekt Kompetenzzentrum Open Source der DHBW Stuttgart wurde mit der Zielsetzung ins Leben gerufen, die Einsatzfelder für Open Source Software in Unternehmen zu identifizieren und durch den Einsatz quelloffener Produkte und deren kostengünstigen Einsatzmöglichkeiten Optimierungen in ausgewählten Geschäftsbereichen zu erzielen.

Dies bedeutet konkret, dass z.B. Open Source Software evaluiert wird, um Lizenzkosten zu reduzieren, bewertet wird, ob sie diverse Qualitätskriterien erfüllt und erfolgreich(er) und effizient(er) in Unternehmen genutzt werden kann. Das Ziel des Projektes ist es hierbei, allgemeingültige Lösungskonzepte für Problemstellungen zu erarbeiten, welche von den am Projekt beteiligten Unternehmen zu firmenspezifischen Lösungen weiterentwickelt werden können. Die beteiligten Unternehmen partizipieren so an den Ergebnissen des Projekts.

Zusammenarbeit mit den Dualen Partnern

Die Zusammenarbeit mit den Dualen Partnern gestaltet sich entlang deren Anforderungen und Bedürfnissen. Sie sind die Themengeber für betriebliche Fragestellungen, die im Rahmen des Projekts untersucht werden. Die DHBW steuert die wissenschaftliche, didaktische und methodische Expertise und Forschungskompetenz bei und untersucht die identifizierten Themenfelder.



Im Rahmen des Projektes steuert die DHBW Stuttgart die wissenschaftliche Expertise und Forschungskompetenz bei zur Bearbeitung der betrieblichen Fragestellungen der Dualen Partner. Es entstehen generische Lösungen, welche von den Partnern an Ihre Situation angepasst werden kann.
Im Rahmen der Arbeit entstehen (generische) Lösungen, an denen die Partner teilhaben können indem sie diese auf ihre spezifische Unternehmenssituation anpassen. Zudem fließen die Ergebnisse in die Arbeit der DHBW ein, sodass hier dem Anspruch an eine hohe Anwendungs- und Transferorientierung ganz im Sinne einer kooperativen Forschung Rechnung getragen wird.

An den Ergebnissen des Projekts partizipieren die Dualen Partner Allianz Deutschland AG, die Deutsche Rentenversicherung Baden-Württemberg und die HALLESCHE Krankenversicherung a.G.

Duale Hochschule Baden-Württemberg Stuttgart
Fakultät Wirtschaft

Projekt im Modul "Neuere Konzepte der Wirtschaftsinformatik" im 6. Semester

Fallstudie Versicherung

—

Evaluation von Eclipse-Plugins zur COBOL-Entwicklung

Autoren:	Ralf Hecktor, Daniel Hohmann, Felix Kugler und Stephen Said
Studiengang	Wirtschaftsinformatik – International Business Information Management
Aufgabenstellung/Betreuung:	Prof. Dr. Thomas Kessel
Datum:	5.7.2012

Inhaltsverzeichnis

Abkürzungsverzeichnis

AHP	Analytical Hierarchy Processes
CICS	Customer Information Control System
CVS	Concurrent Versions System
DHBW	Duale Hochschule Baden-Württemberg
GUI	Graphical User Interface
IDE	Integrated Development Environment
ISV	Independent Software Vendor
JDBC	Java Database Connectivity
MCLA	Metrixware Community License Agreement
ODBC	Open Database Connectivity
OSI	Open Source Initiative
OSI	Open Systems Interconnection
OSS	Open Source Software
PVCS	Polytron Version Control System
QA	Quality Assurance
RDz	Rational Developer for System z
SCM	Source-Code-Management
SQL	Structured Query Language
SSH	Secure Shell
SVN	Apache Subversion
SWT	Standard Widget Toolkit

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Motivation

Die Open Source-Plattform Eclipse stellt heutzutage im Umfeld der Programmiersprache Java eine der meistgenutzten integrierten Entwicklungsumgebungen (engl. Integrated Development Environment (IDE)) dar.[1] Im Zusammenhang mit neuen Konzepten der Software-Entwicklung (z.B. agiler Software-Entwicklung) bietet Eclipse viele Funktionalitäten, die den kompletten Entwicklungsprozess unterstützen. Dies vermeidet zum einen mögliche Schnittstellenproblematiken, zum anderen ermöglicht es eine einheitliche Arbeitsweise unter den Entwicklern. Vor allem für Java bietet das durch Plugins realisierte modulare Konzept der IDE viele Vorteile. Eben diese Plugins ermöglichen allerdings auch die Entwicklung in anderen Programmiersprachen.

Im Banken- und Versicherungssektor ist die Verbreitung der Eclipse-Plattform heute noch begrenzt. 95% dieser Unternehmen nutzen Mainframe-Großrechner für die Verarbeitung ihrer Daten.[2][3] Anwendungen werden größtenteils in der Programmiersprache COBOL programmiert, die als sehr alte Sprache nur wenige Aspekte moderner Programmiersprachen abbildet. Für die COBOL-Entwicklung existieren diverse eigenständige Entwicklungsumgebungen, die jedoch wenig mit der oben vorgestellten Eclipe-IDE gemein haben. So bieten sie im Vergleich relativ wenig Nutzerkomfort und kaum Unterstützungsfunktionen, die junge Programmierer im Rahmen ihrer Ausbildung kennengelernt haben. Interoperabilität mit anderen Anwendungen ist nicht immer gewährleistet.

Branchentypisch gestaltet sich die Software-Entwicklung auch bei der "Versicherung", einer Teilgesellschaft der Versicherung mit Hauptsitz in Stadt. Hier arbeiten XX Personen an der Entwicklung von COBOL-Applikationen. Lediglich XX Mitarbeiter sind mit der Anwendungsentwicklung in anderen Programmiersprachen betraut, welche bereits über die Eclipse-IDE stattfindet.

Im Rahmen eines Forschungsauftrages an die Duale Hochschule Baden-Württemberg (DHBW) soll evaluiert werden, inwieweit sich die COBOL-Entwicklung der Versicherung in Eclipse realisieren lässt. In der Vergangenheit wurde in einer Machbarkeitsstudie analysiert, inwieweit sich das kommerzielle Produkt Rational Developer for System z (RDz) in der Versicherung einsetzen lässt. Eine grundsätzliche Machbarkeit wurde nachgewiesen. Aufgrund finanzieller Aspekte wurde diese Software trotzdem nicht eingeführt. Vor diesem Hintergrund liegt der Fokus des Forschungsauftrags primär, aber nicht ausschließlich, auf Open Source Software (OSS). Im Speziellen sollen Plugins für Eclipse evaluiert werden, die die Programmierung in COBOL ermöglichen bzw. erleichtern. Vorrangiges Ziel ist es, von den oben genannten Vorteilen moderner IDEs profitieren zu können. Die Versicherung erhofft sich darüber hinaus eine bessere Interoperabilität mit anderen internen Entwicklungsprojekten, die beispielsweise in Java realisiert werden.

1.2 Zielsetzung

Wie zuvor aufgezeigt entspricht es dem Wunsch der Versicherung, die Entwicklungsumgebung Eclipse auch für die COBOL-Softwareentwicklung einzusetzen. Ziel dieser Arbeit ist es daher,

[1] Vgl. (2006)
[2] Vgl. XXX, U. et al. (2012)
[3] Vgl. (2012a)

Plugins für die COBOL-Entwicklung in Eclipse zu evaluieren und in diesem Zuge eine ausschnittsweise Marktübersicht zu erstellen. Hierbei liegt der Fokus auf nicht-kommerzieller OSS, da der finanzielle Initialaufwand einen zentralen Faktor in der unternehmensinternen Diskussion über eine mögliche Implementierung darstellt. Die Analyse beschränkt sich jedoch nicht ausschließlich auf Open Source Lösungen.

Für eine Analyse der Plugins ist eine Berücksichtigung der durch die Versicherung spezifizierten Anforderungen an eine Lösung essentiell. Ein untergeordnetes Ziel im Rahmen dieser Arbeit ist es aus diesem Grund, diese Anforderungen zu erfassen und zu priorisieren. Auf Basis dieser Priorisierung erfolgt die Plugin-Bewertung, deren Ergebnis eine Empfehlung einer spezifischen Applikation sein wird.

Vor dem Hintergrund des spezifischen Anforderungsprofils der Versicherung können die nachfolgenden Ausführungen nicht als generische, grundsätzliche Marktübersicht zu COBOL-Plugins für Eclipse angesehen werden. Die gesamte Arbeit stellt eine Fallstudie mit der Versicherung dar und orientiert sich damit stark an den Bedürfnissen dieses Unternehmens. Ein gewichtiger Teil besteht daher in einer Anforderungsanalyse. Anforderungen, die sich möglicherweise in anderen Unternehmenskontexten ergeben, sind nicht zwingend abgebildet.

1.3 Aufbau der Arbeit

Zu Beginn der Fallstudie werden die Termini IDE und Open Source definiert. Daran anschließend erfolgt eine theoretische Einführung in COBOL und Eclipse.

In zweiten Teil dieser Arbeit wird der betriebliche Kontext der Fallstudie erläutert, wobei eine Differenzierung zwischen dem Ist- und dem Sollzustand vorgenommen wird. Aus letzterem ergeben sich spezifische Anforderungen an die zu evaluierenden Plugins, die mitsamt einer Priorisierung dargestellt werden.

In einem dritten Schritt erfolgt, abgeleitet aus den priorisierten Anforderungen, eine begründete Vorauswahl zu evaluierender Plugins bzw. IDEs. Diese werden dann im Detail auf die Erfüllung der Anforderungen hin analysiert. Um eine Konfigurationsempfehlung aussprechen zu können, erfolgt eine Gegenüberstellung der Resultate. Die unterschiedlichen Lösungen werden zu Referenzzwecken auch noch einmal mit der bereits in der Vergangenheit analysierten Lösung RDz verglichen. Für die auf die Anforderungen am ehesten zugeschnittene Applikation wird schließlich eine Konfigurationsempfehlung ausgesprochen.

Abschließend werden limitierende Faktoren der Arbeit aufgezeigt, Schlüsselaspekte zusammengefasst und ein Ausblick auf weiterführende, an die Fallstudie anschließende Untersuchungsfelder gegeben.

2 Theoretische Grundlagen

2.1 Definitionen

2.1.1 Integrated Development Environment (IDE)

Eine integrierte Entwicklungsumgebung (IDE) ist eine in einer Applikation konsolidierte Sammlung von Anwendungsprogrammen, mit denen Aufgaben der Softwareentwicklung innerhalb einer Lösung bearbeitet werden können.[4] In einer IDE sind typischerweise ein Code-Editor, ein Compiler, eine Debugging-Lösung sowie eine Möglichkeit zum Erstellen von Benutzeroberflächen (Graphical User Interface (GUI)) enthalten.[5]

2.1.2 Open Source

Der Terminus Open Source beschreibt ein Konzept, nach dem Software mit ihrem Quellcode ausgeliefert wird.[6] Open Source Software ist eine der Erscheinungsformen sogenannter freier Software. Sie ist zumeist kostenfrei erhältlich und kann grundsätzlich unter bestimmten lizenzrechtlichen Voraussetzungen frei verbreitet werden.[7,8] Die Verwertung, Vervielfältigung und Bearbeitung ist demzufolge nicht vorbehaltlos gestattet.

Die Open Systems Interconnection (OSI) definiert zehn Kriterien, die OSS erfüllen muss.[9] So muss die Software z.B. unter einer von der OSI bestätigten Lizenz veröffentlicht werden. Dies beinhaltet die Idee der freien Verteilung der Quellen und des Binärcodes. Dabei sind Modifikationen am Code erlaubt und sogar erwünscht. Die Art und Weise der Modifikation kann jedoch innerhalb gewisser Rahmenbedingungen beschränkt werden. Zudem werden durch die OSI weitere Lizenzbedingungen, die Software als Open Source klassifizieren, spezifiziert.

2.2 Einführung COBOL

COBOL ist eine der ältesten heute noch verwendeten Programmiersprachen. Das Akronym COBOL steht für **CO**mmon **B**usiness-**O**riented **L**anguage. Bereits der Name spiegelt die Hauptanwendungsbereiche der Sprache wider: Geschäfts-, Finanz- und Verwaltungssoftware in Unternehmen und Behörden.[10]

Die erste COBOL-Spezifikation wurde 1959 durch ein Komitee von Forschern aus Wirtschaft, Forschung und öffentlicher Verwaltung definiert. Sie baute auf zu diesem Zeitpunkt gebräuchlichen Programmiersprachen auf (vor allem FLOW-MATIC und COMTRAN). Ziel war es, einen gemeinsamen Standard zu entwickeln, mit dem sich Geschäftsprozesse softwareseitig abbilden lassen.[11]

Die Spezifikation der Programmiersprache wurde im Verlauf der Jahre fortlaufend überarbeitet. Über die Versionen ANS COBOL 1968, COBOL 1974, COBOL 1985 und COBOL 2002 wurden

[4] Vgl. (2000) , S. 9
[5] Vgl. (2007)
[6] Vgl. (2012)
[7] Vgl. (2012)
[8] Diese Voraussetzungen sollen in diesem Rahmen nicht detaillierter ausgeführt werden.
[9] In Vollständigkeit einzusehen unter: `http://www.opensource.org/docs/osd`
[10] Vgl. (2000) , S. 1
[11] Vgl. (1978) , S. 124 ff.

die Standardisierung der Sprache vorangetrieben und zusätzliche Features hinzugefügt.[12] COBOL 2002 brachte zuletzt eine Reihe bedeutsamer Neuerungen mit sich. Diese aktuellste Spezifikation beinhaltet unter anderem:[13]

- Unterstützung von Objektorientierung
- Unterstützung nationaler Sprachen (u.a. durch Unicode)
- Unterstützung benutzerdefinierter Funktionen
- Unterstützung zusätzlicher primitiver Dateiformate (Bit und Boolean)
- Erzeugung und Parsing von XML
- Erhöhte Interkompatibilität mit anderen Progammiersprachen wie Java und C

Micro Focus, führender Anbieter von Software-Produkten rund um COBOL, beziffert die Verbreitung von COBOL im Jahr 2012 auf über 220 Milliarden Code-Zeilen. Das Unternehmen bezeichnet COBOL als „eine der wichtigsten Programmiersprachen“ mit weiter Verbreitung in transaktionsintensiven Anwendungen, wie jenen der Finanz, Versicherungs- und Touristikbranche.[14]

Um COBOL-Programme auf der jeweiligen Zielplattform auszuführen, bedarf es der Übersetzung des Codes in Maschinensprache (Kompilieren). Dieser Schritt wird von so genannten Compilern übernommen. Bereits 1960 erschienen die ersten beiden Compiler für COBOL, die die Ausführung ein und desselben Programms auf zwei unterschiedlichen Großrechnertypen (RCA Computer, Remington-Rand „Univac“) gestatteten. Heute gibt es eine Vielzahl von COBOL-Compilern für unterschiedliche Zielarchitekturen. Diese werden größtenteils von kommerziellen Anbietern bereitgestellt (z.B. IBM und Fujitsu).[15] Allerdings existieren auch Open Source-Compiler für COBOL (z.B. OpenCobol[16] und TinyCobol[17]).

Um der Nachfrage für die Nutzung moderner Software-Entwicklung gerecht zu werden, bieten unterschiedliche Hersteller integrierte Entwicklungsumgebungen für die Programmierung in COBOL an. Es existieren Produkte kommerzieller Hersteller (z.B. Micro Focus und IBM), aber auch Open Source-Lösungen.

Populär ist die Verwendung der Open Source-IDE Eclipse mit Hilfe ergänzender Plugins wie Visual COBOL, isCOBOL, cobolclipse, Elastic COBOL oder Cobos.

2.3 Einführung Eclipse

Die Eclipse-Plattform[18] ist eine quelloffene Entwicklungsumgebung, die primär zum Programmieren von Software eingesetzt wird und in Version 1.0 Ende 2001 veröffentlicht wurde.[19] Zum Zeitpunkt dieser Arbeit ist der aktuelle Entwicklungsstand in Version 4.2 (Versionsname "Juno")

[12] Vgl. (2000) , S. 3 f.
[13] Vgl. ISO – TBD
[14] Vgl. (2012a)
[15] Vgl. (2000) , S. 2 f.
[16] `www.opencobol.org`
[17] `www.tiny-cobol.sourceforge.net/`
[18] `www.eclipse.org`
[19] Vgl. (2004) , S. 1

enthalten.[20] Solche IDE genannten Lösungen unterstützen die Integration und Verknüpfung diverser Programmiersprachen. Diese gestaltet sich aufgrund der durch die Entwicklung moderner Applikationen entstehenden Komplexität zunehmend schwieriger.[21] Durch die Nutzung von IDEs wird Interoperabilität von vordergründig voneinander unabhängigen Tools gefördert.

Die Eclipse-Plattform als eine der populärsten IDEs[22] besitzt eine modular aufgebaute Architektur. Diese besteht, wie in Abbildung 1 dargestellt, außer dem "Platform Runtime" genannten Kernel, ausschließlich aus Plugins. Ein Plugin ist die kleinste Einheit der Plattform, die eigenständig programmiert und ausgeliefert werden kann.[23]

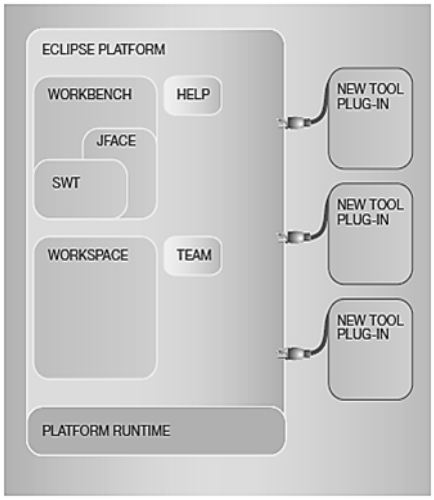


Abbildung 1: Eclipse Plattform Architektur[24]

Die als Standard Widget Toolkit (SWT) und „JFACE“ bezeichneten Komponenten sorgen für die grafische Darstellung der Entwicklungsumgebung, wohingegen die Workbench die generelle Struktur der Software repräsentiert. Der Workspace kennzeichnet die Sammlung der Projekte des jeweiligen Eclipse-Nutzers.

Die über Plugins realisierbaren Erweiterungen bieten zusätzlich zu den bereits aufgezeigten, vorhandenen generischen Möglichkeiten die eigentlichen Funktionalitäten: Somit kann Eclipse dazu genutzt werden, Anwendungen zu designen, mit Datenbanken zu interagieren, Testmanagement zu betreiben, sowie allgemein den gesamten Software-Entwicklungsprozess zu unterstützen.[25] Zudem kann die primär als Java-IDE genutzte Plattform zur Unterstützung anderer Programmiersprachen erweitert werden. Jedes Plugin besitzt eine „manifest“ genannte Datei, welche die Verbindung zu anderen Plugins beschreibt. Hierdurch lassen sich beim Starten der Software durch die Platform Runtime die verfügbaren Erweiterungen einlesen und die Interoperabilität sowohl verschiedener Programmiersprachen als auch diverser Funktionalitäten herstellen.

[20] `www.eclipse.org`
[21] Vgl. (2004) , S. 371
[22] Vgl. (2004) , S. 1
[23] Vgl. (2004) , S. 373
[24] Vgl. ebd., S. 374
[25] Vgl. (2004) , S. 1

Ein weiterer Vorteil der Plattform besteht in der Erstellung und Darstellung unterschiedlicher Perspektiven, wie z.B. der Java-, der Debug- oder der XML-Perspektive.[26] Diese lassen sich individuell mit Sichten und Editoren gestalten und bilden hiermit das Layout, welches dem Entwickler zur Verfügung steht. Ein Wechsel zwischen den Perspektiven ermöglicht ein zielorientiertes Arbeiten, da eine Anpassung an spezifische Aufgaben jederzeit möglich ist. Editoren und Sichten, die nicht im Eclipse-Standardumfang enthalten sind, lassen sich über Plugins hinzufügen. So können bspw. Syntax-Korrektur-Applikationen innerhalb von Editoren-Plugins ergänzt werden.

Einen die moderne Software-Entwicklung im Team unterstützenden Faktor stellt die mögliche Versionskontrolle und das Konfigurationsmanagement mit einem zugehörigen Repository dar.[27] Die Eclipse-Plattform bietet grundlegende Schnittstellen zur Interaktion mit Team-Repositories unterschiedlicher Hersteller. Hierbei können multiple Repositories wie das Concurrent Versions System (CVS), Apache Subversion (SVN) oder "Git" koexistieren.

Aufgrund der genannten Faktoren, vorrangig durch die nahezu unbegrenzte Erweiterbarkeit durch Plugins, stellt die Eclipse Plattform eine in vielen Unternehmen nicht nur zur Entwicklung von Java-basierten Applikationen genutzte Lösung dar.

[26] Vgl. (2004) , S. 379
[27] Vgl. (2004) , S. 153 ff.

3 Untersuchung

3.1 Vorgehensweise

Dieses Kapitel dient dazu, die methodische Vorgehensweise dieser Ausarbeitung darzustellen. Aus zeitlogischer Sicht folgt sie dem in nachfolgender Abbildung dargestellten Ablauf.

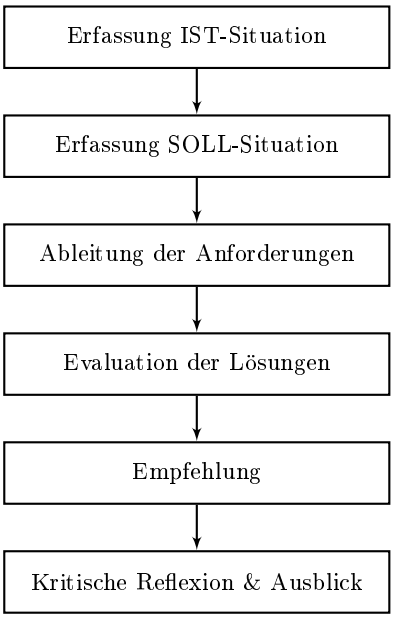


Abbildung 2: Methodisches Vorgehen

Im ersten Schritt wird eine Interviewrunde mit Experten aus den IT-Fachabteilungen der Versicherung geführt. [28] Ziel dieser Gespräche ist es, zu erfassen und zu dokumentieren, wie die Software- und insbesondere die COBOL-Entwicklung in der Versicherung zum Zeitpunkt der Entstehung dieser Fallstudie ausgestaltet ist. Dabei werden die unterschiedlichen Aufgabengebiete der Software-Entwicklung im Unternehmen vorgestellt. Es wird in diesem Zusammenhang auch erläutert, welche Programmiersprachen für welche Aufgaben zum Einsatz kommen. Hinsichtlich COBOL erfolgt eine Detaildarstellung der eingesetzten Werkzeuge sowie der Schnittstellen und Abhängigkeiten zwischen diesen. In diesem Kontext werden dann auch Optimierungspotenziale aufgezeigt, die sich in der COBOL-Entwicklung identifizieren lassen.

Im Rahmen der Interviews wird in einem zweiten Schritt erfasst, welche Zukunftsperspektive die IT-Leitung, auf Basis ihrer IT-Strategie, für das methodische Vorgehen im Umfeld der COBOL-Entwicklung hat. Hierzu wird gemeinsam mit der Versicherung eine Sammlung von konkreten Anforderungen (Requirements) aufgestellt, die eine mögliche zukünftige Orchestrierung von Werkzeugen und Plugins zu erfüllen hätte.

Im Rahmen einer nachgelagerten Interviewrunde werden in einem dritten Schritt die gesammelten und konsolidierten Anforderungen verifiziert und priorisiert. [29] Die Priorisierung dient der späteren Identifikation einer Lösung, die den zukünftigen Soll-Zustand am ehesten abbilden kann.

[28] Vgl. XXX, U. et al. (2012); XXX, K. (2012)
[29] Vgl. XXX, U. et al. (2012)

In einem vierten Schritt wird die Auswahl möglicher Tools und Plugins vorgestellt, die auf Basis der zuvor aufgestellten Anforderungen als grundsätzlich geeignet erachtet werden. Die jeweiligen Features werden hierbei knapp beschrieben und auf die Erfüllung der Anforderungen hin untersucht.

Abschließend wird der Grad der Erfüllung der verschiedenen Anforderungen für jedes Tool ermittelt und nach spezifischer Gewichtung jeder Anforderung in einer Gesamtkennzahl ausgedrückt. Die Kennzahlen aller Tools werden übergreifend in einer Vergleichsmatrix gegenübergestellt. Diese Matrix dient schließlich der Identifikation der am ehesten geeigneten Lösung.

3.2 Ist-Zustand

Im Regelfall teilen sich Anwendungen bei der Versicherung in zwei technologische Pakete auf. Ihr Frontend sind Java-basierte (Web-)Anwendungen und ihre Geschäftslogik COBOL-Anwendungen, die auf Großrechnern betrieben werden. Diese Aufteilung ist aus organisatorischer Sicht analog in der Entwicklungsabteilung vorhanden.

Die Frontend-Entwicklung erfolgt mit Hilfe der Open Source-IDE Eclipse. Die Betrachtung soll sich an dieser Stelle jedoch auf die Programmierung der Geschäftslogik beschränken.

Diese erfolgt mittels einer Reihe ineinandergreifender Tools, deren Funktion im Folgenden, anhand des allgemeinen Softwareentwicklungsprozesses der Versicherung, erläutert wird.[30]

- Die Fachabteilung formuliert eine Anforderung, die nach Freigabe an die Entwicklungsabteilung weitergeleitet wird.
- Die Entwicklungsabteilung erstellt für die Anforderung eine Arbeitskonfiguration mit diversen Metadaten (z.B. Auftragsname, Beschreibung, Angaben über zugehörigen Code) im Metadaten-Repository Rochade. Auf Rochade greifen die involvierten Entwickler verteilt von ihren jeweiligen Workstations unter Windows XP zu. Die Meta-Daten selbst liegen jedoch auf einem zentralen Repository-Server.
- Die Entwicklungsabteilung entwickelt die Anforderung. Für die Codierung werden die Editoren UltraEdit und Animator verwendet. Der Code wird mit Hilfe des Code-Repositorys Serena Polytron Version Control System (PVCS) verwaltet.
- Die Binaries werden in einer Testumgebung unter zOS installiert und funktionalen Tests unterzogen. Dies erfolgt ohne die Zuhilfenahme spezieller Tools durch Abgleich der dokumentierten Anforderungen mit der Funktionsweise des entwickelten Programms.[31]
- Sobald das Testing erfolgreich abgeschlossen ist, werden die Anwendungen auf den Produktiv-Instanzen der Versicherung installiert. Auf diesen ist ebenfalls zOS als Betriebssystem installiert.

Abbildung 3 illustriert die oben beschriebene Unterteilung von Entwicklung, Testing und Produktivsystemen.

[30] Vgl. XXX, K. (2012)
[31] Vgl. XXX, A. (2012)

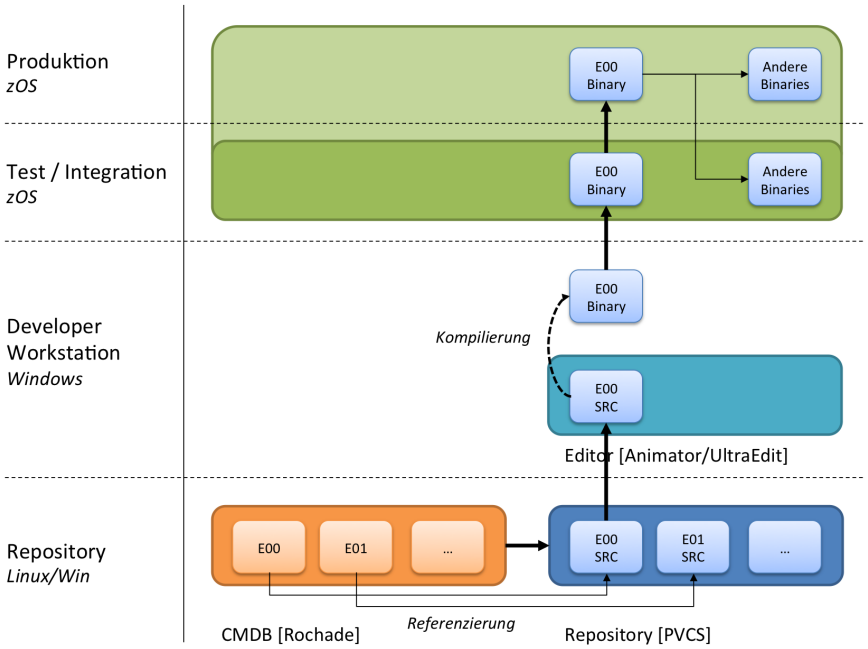


Abbildung 3: Logische Schritte, Systeme und Tools der COBOL-Entwicklung

3.3 Soll-Zustand

An dieser Stelle wird beschrieben, welchen Soll-Zustand die Versicherung mit der Einführung einer neuen softwareseitigen Lösung für die COBOL-Programmierung zu erreichen gedenkt.[32]

Eine Anforderung der Versicherung ist es, die Java- und COBOL-Programmierung in eine technologisch einheitliche Entwicklungsumgebung zu heben, Eclipse also auch für die COBOL-Entwicklung zu verwenden.

Hiervon verspricht sich die Versicherung Möglichkeiten zur verbesserten Kommunikation zwischen den Entwicklern des Frontends und der Geschäftslogik. Zudem sieht sie hierin das Potential, moderne Ansätze der Software-Entwicklung auch auf die COBOL-Entwicklung anzuwenden.

Des Weiteren erhofft sich die Versicherung, die Anzahl der Medienbrüche zwischen unterschiedlichen Tools in der COBOL-Entwicklung (von der Formulierung der Geschäftsanforderung bis zum Produktivgang der technischen Lösung) zu reduzieren.

Ein weiterer Wunsch besteht darin, das formale Testing weg von den Test-Serverinstanzen hin zu den Workstations der Entwickler zu verlagern. So müssten beispielsweise Syntax-Checks und die Kontrolle der Validität von Bezügen innerhalb des Codes nicht auf dem Mainframe getestet werden. Die Reduktion des Testaufwandes auf den Servern selbst würde eine Verringerung der Serverlast und damit Kosteneinsparungen mit sich bringen.[33]

[32] Vgl. XXX, U. et al. (2012); XXX, K. (2012)
[33] Die Versicherung zahlt an einen externen Dienstleister je nach Auslastung der Server unterschiedlich hohe Nutzungsgebühren. Die zu erwartenden Kosteneinsparungen wurden von der Versicherung nicht quantifiziert.

3.4 Abgeleitete Anforderungen an eine neue Lösung

Die Kriterien, unter denen die untersuchten Plugins und Entwicklungsumgebungen verglichen werden, ergeben sich zum einen aus den konkreten arbeitsorganisatorischen- und technologischen Anforderungen der Versicherung.[34] Zum anderen wird der generelle aktuelle technische Stand von Entwicklungsumgebungen, z.B. auch in anderen Programmierplattformen wie Java oder .NET, gewürdigt.

Hinsichtlich nicht-funktionaler Anforderungen[35] liegt bei den reinen Plugins ein Augenmerk auf der Entkopplung der Abhängigkeit von der verwendeten Eclipse-Version. Wie im vorherigen Kapitel geschildert wurde, wird Eclipse bei der Versicherung bereits im Bereich Frontend- und Middleware-Entwicklung eingesetzt. Hierbei wird die IDE mit für diesen Anwendungsfall spezialisierten Plugins genutzt. Sofern Eclipse nun auch bei der Entwicklung von COBOL-Anwendungen genutzt werden würde, gilt es aus Sicht der Versicherung sicherzustellen, dass die Plugins für die beiden Anwendungsfälle keine kollidierenden Anforderungen an die Eclipse-Version haben. Andernfalls wäre die Folge daraus, dass zwei unterschiedliche Eclipse-Version, mit u.U. unterschiedlichen Abhängigkeiten zu weiteren Plugins oder Features, betrieben werden würden. Dies würde bedeuten, dass separate Pflegeaktivitäten (Updates, Fixes) für beide Anwendungsfälle von Eclipse anfallen würden. Letztendlich wäre mit erhöhten IT-Aufwänden zu rechnen.

Aus dieser Problematik resultiert auch die Anforderung, dass vorrangig ein unabhängiges Plugin und keine angepasste Eclipse-Version für die COBOL-Entwicklung gesucht wird. Im Fall, dass die jeweilige COBOL-Entwicklungsumgebung als eigenständige Eclipse-Version ausgeliefert wird, wäre diese Flexibilität nicht gegeben, da der Hersteller der Entwicklungsumgebung bindende Vorgaben über die Eclipse-Version machen könnte.

Aufgrund von Überlegungen des Risikomanagements ergibt sich aus den Aspekten der Zuverlässigkeit und Nachhaltigkeit die Anforderung, dass für ein Plugin bzw. eine IDE Beratungsdienstleistungen verfügbar sein müssen. Hierbei spielt es eine untergeordnete Rolle, ob diese durch den Hersteller der Software oder ein externes Beratungsunternehmen angeboten werden. Die gewünschten Leistungen betreffen u.a. Aspekte wie die Unterstützung bei der Inbetriebnahme der Entwicklungsumgebung sowie der Implementierung von Individualisierungsmaßnahmen. Zusätzlich ist explizit die Verfügbarkeit von Supportdienstleistungen, z.B. bei unvorhergesehen Kompatiblitätsproblemen, gefordert.

Im Bereich der IDE-Funktionalitäten ergibt sich aus den Gegebenheiten der Versicherung, dass spezielle Anforderungen hinsichtlich des Code-Editors existieren. Diese betreffen die Funktionen Autovervollständigung, Code-Highlighting und Fehlererkennung.

Unter Autovervollständigung versteht man die automatische Ergänzung von Programmierbefehlen bzw. das Vorschlagen von Befehlen, die Teile des getippten Fragments beinhalten. Code-Highlighting bezeichnet die optische Hervorhebung von bestimmten Schlüsselwörtern. So werden beispielsweise Fallunterscheidungen, wie etwa if-else-Anweisungen, mit einer anderen

[34] Vgl. XXX, U. et al. (2012); XXX, K. (2012)

[35] Nicht-funktionale Anforderungen beschreiben die Begleitumstände, unter denen eine bestimmte Funktionalität erbracht werden soll.

Textfarbe als der Aufruf einer Prozedur dargestellt. Eine Fehlerkennung markiert automatisch die Stellen, an denen Verletzungen der Programmiersyntax vorliegen.

Neben diesen Standardfeatures ist aus Sicht der Versicherung in besonderem Maße die Unterstützung von Gastsprachen zu würdigen. Über diese werden in COBOL Routineaktivitäten, wie z.B. der Zugriff auf externe Datenbanken, umgesetzt. Innerhalb des Buildvorgangs einer COBOL-Anwendung werden Code-Fragmente, die in einer eingebetteten Gastsprachen codiert sind, durch Pre-Prozessoren vor Ausführung des COBOL-Compilers in COBOL interpretiert. Der Code-Editor muss diesen Vorgang nachvollziehen können, um die Gastsprachen nicht als fehlerhaften COBOL-Code zu identifizieren.

Ebenfalls von Bedeutung ist die Existenz einer Debuggingfunktionalität. Eine Vorsichtung des Marktes an COBOL-Entwicklungsumgebungen und des Vergleiches mit Microsoft Visual Studio sowie der Java-Entwicklung in Eclipse ergibt, dass die Unterstützung von grafischem Schritt-für-Schritt Debugging sowie entferntem Debugging relevant sind. Unter grafischem Debugging versteht man die manuelle Steuerung der Programmausführung über ein GUI der Entwicklungsumgebung. Dazu gehört z.B. der Sprung zu einer bestimmten, im Quellcode zur Laufzeit gewählten Code-Zeile. Entferntes Debugging bedeutet das Debugging einer Anwendung, die auf einem entfernten System, z.B. einem Mainframe, ausgeführt wird. Die Entwicklungsumgebung des Softwareentwicklers erhält hierbei die Kontrolle über Teile der Ausführungsumgebung der Anwendung.

Von Seiten der Versicherung ist es ferner wünschenswert, dass aus der COBOL-Entwicklungsumgebung ein direkter Zugriff auf einen Großrechner, d.h. z.B. IBM System z, möglich ist. Hierzu gehört neben einer manuellen Kommandozeilenschnittstelle auch ein grafischer Zugriff auf das Dateisystem. Unter letzterem wird beispielsweise die Darstellung des Dateisystems des Großrechners in Form eines Explorers verstanden. Ebenfalls ist es vorteilhaft, wenn die IDE einen eigenständigen Zugriff auf Datenbanken (Open Database Connectivity (ODBC), Java Database Connectivity (JDBC), etc.) ermöglicht. Hierzu gehört die Verwaltung mehrerer Accounts inklusive Zugriffsdaten (Benutzer und Passwort), sowie die Ausführungsmöglichkeit und Anzeige von Datenbankabfragen.

Die nachfolgende Tabelle auf Seite 12 (Tabelle 1) bietet eine zusammenfassende Darstellung der oben genannten Kriterien.

Allgemeines	
Hersteller	
Softwaretyp	IDE/Plugin/Cloud
Quellcode	Offen/Geschlossen
Kosten/Lizenzierung	
Systemanforderungen	CPU/RAM/HDD
Funktionale Kriterien	
Editorfunktionen	
Autovervollständigung	Ja/Nein
Code-Highlighting	Ja/Nein
Fehlererkennung (Syntax-Check)	Ja/Nein
Unterstützung von Gastsprachen	Ja/Nein
Debuging	
Grafisches Schritt-für-Schritt Debuging	Ja/Nein
Remote Debugging	Ja/Nein
Anbindung von externen Systemen	
SSH	Ja/Nein
Datenbanken	Ja/Nein
Nicht-Funktionale Kriterien	
Beratungsdienstleistungen	Ja/Nein
Support	Ja/Nein
Eclipse-Version	

Tabelle 1: Vergleichskriterien

3.5 Produktvergleich

Der ursprüngliche Anspruch an diese Fallstudie war es, aus dem gesamten Spektrum an verfügbaren Produkten vorrangig jene herauszufiltern, die als OSS-Plugin-Lösung verfübar sind. Nach einer umfangreichen Literatur- und Internetrecherche wurden zunächst alle Tools zusammengetragen, die auf Eclipse basieren. Hieraus wurde die Erkenntnis gewonnen, dass die Verfügbarkeit reiner OSS-Applikationen zum Zeitpunkt der Entstehung dieser Arbeit marginal ist. Deren Evaluation widerum war aufgrund unterschiedlicher Faktoren (veraltete Referenzen auf den Quellcode, technische Aspekte, mangelnde Kommunikationsbereitschaft der Entwickler) nicht möglich. Daher besteht die zu analysierende Stichprobe, entgegen dem ursprünglichen Ansatz, primär aus kommerziellen Lösungen. Um die Stichprobenlänge zu erhöhen, wurden zudem Eclipse-Bundles miteinbezogen, statt ausschließlich Plugins zu betrachten.

Von dieser Auswahl an Produkten wurden letztlich die Tools getestet, die zum Zeitpunkt der Durchführung dieser Untersuchung der DHBW Stuttgart verfügbar waren. Eben genannte Einschränkungen und Überlegungen sind der Grund dafür, dass von initial 13 identifizierten Produkten nur 5 untersucht werden konnten. Zusätzlich wurde der IBM RDz als Referenz für die Entwicklung auf der Plattform System z, wie es bei der Versicherung der Fall ist, miteinbezogen.

3.5.1 Visual COBOL IDE

Die Entwicklungsumgebung Visual COBOL[36] IDE von Micro Focus ist ein kommerziell vertriebenes Werkzeug, dessen Quellcode geschlossen gehalten wird.[37] Der Hersteller bietet zu Visual COBOL zwei Erweiterungen an. Dazu zählt „Database Connectors“, das zur Anbindung von Datenbanken dient, und „XDBC“, welches über ODBC/JDBC eine generische Anbindungsmöglichkeit von externen Anwendungen bereitstellt.[38] Hinsichtlich der Bepreisung konnten keine konkreten Informationen gewonnen werden. Hierzu muss im Einzelfall direkt Kontakt mit dem Hersteller aufgenommen werden.

Der Editor der Entwicklungsumgebung deckt alle untersuchten Vergleichskriterien ab. Er bietet eine Autovervollständigung ("Code assist") an, die alle nativen Sprachelemente abdeckt. Diese ist in Abbildung 4 dargestellt. Gastsprachen werden hierbei nicht unterstützt. Auf der linken Seite ist das Kontextmenü zu sehen, in welchem COBOL-Befehle angeboten werden.
Ebenfalls verfügt der Editor, dank einer Hintergrundkompilierung durch den integrierten Compiler, über eine automatische Syntaxüberprüfung und Fehlererkennung. Gastsprachen werden nach Konfiguration eines entsprechenden Pre-Prozessors vom Editor als solche erkannt. Auf der rechten Seite von Abbildung 4 ist beispielsweise zu sehen, wie ein fehlerhafter Befehl rot unterstrichen ist.

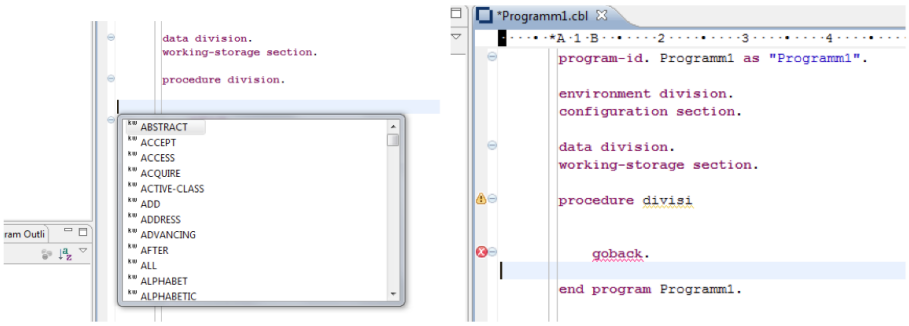


Abbildung 4: Visual COBOL – Editor

Im Hinblick auf Debuggingfunktionalitäten ist das Debugging von lokalen und extern betriebenen Anwendungen, z.B. auf einem System z, möglich. Hierbei kann sich Visual COBOL z.B. an einen bereits auf einem externen System gestarteten Prozess anhängen und diesen beobachten. Auch ist es möglich, dass der Core Dump eines abgestürzten Prozesses eingeholt wird. Wie in Abbildung 5 zu sehen ist, wird die Ausführung dieser Funktionalität, wie innerhalb von Eclipse üblich, über die „Debug Configurations“ Maske angeboten. Dies kann als grafisches Debugging verstanden werden.

Die Einbindung von externen Pre-Prozessoren in den Build-Vorgang ist über eine dedizierte Schnittstelle möglich. Visual COBOL bietet hierbei an, mehrere Pre-Prozessoren miteinander zu verschachteln. Eine Einschränkung ist, dass diese nur unter Nutzung von nativem COBOL (und nicht für Dialekte) möglich ist. Hinsichtlich Fehlererkennung bietet die Entwicklungsumgebung die Möglichkeit, im Editor bei der Kompilierung Fehler anzuzeigen, die aus einer Einbindung von

[36] Produktseite: `http://bit.ly/eESeTb`
[37] Vgl. (2012c)
[38] Vgl. (2012c)

Gastsprachen in den COBOL-Code resultieren. Ergibt sich demzufolge ein Fehler aus fehlerhaftem Code in einer Gastsprache, kann dieser auf die entsprechenden Code-Fragmente zurückgeführt werden.

Für den Zugriff auf IBM DB2 sowie eine generische ODBC/JDBC Schnittstelle ist bereits jeweils ein entsprechender Pre-Prozessor im Lieferumfang enthalten.

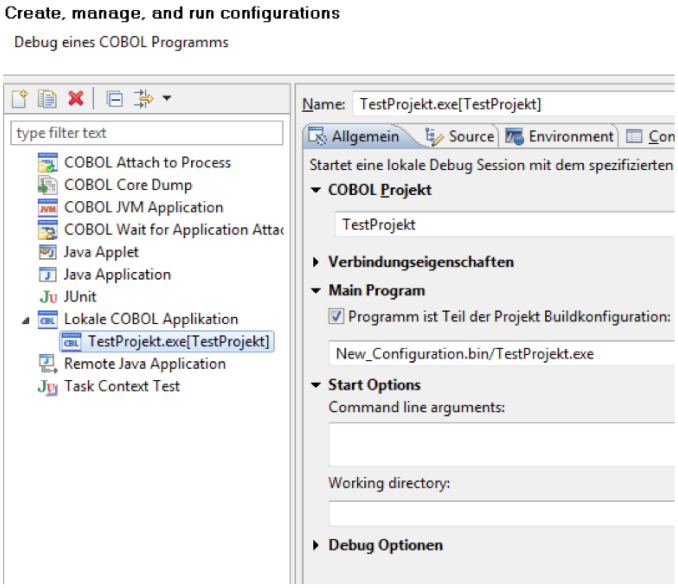


Abbildung 5: Visual COBOL – Debugging

Die Anbindung von externen Systemen ist in Visual COBOL über mehrere Schnittstellen möglich. Hierbei kann eine Kommandozeilenschnittstelle über Secure Shell (SSH) eingebunden werden. Ein grafischer Zugriff auf das Dateisystem eines externen Systems ist über einen Explorer für Linux, Unix und Windows-Umgebungen möglich. Hierzu existiert die dedizierte Eclipse-View "Remote Systems".

Im Bereich von Datenbanken unterstützt Visual COBOL die Anbindung von IBM DB2, Oracle, Sybase, Informix und OpenESQL. Über letztere können beliebige ODBC und JDBC Datenquellen angesprochen werden.

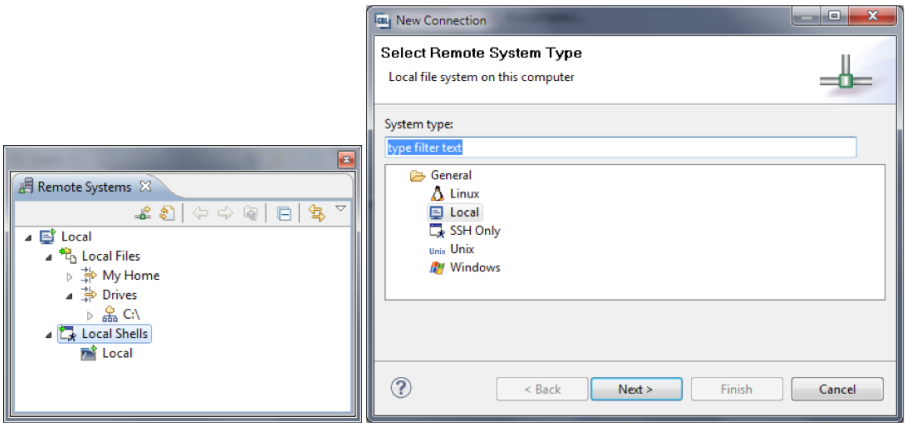


Abbildung 6: Visual COBOL – Anbindung externer Systeme

In Bezug auf Dienstleistungen bietet Micro Focus rund um seine Produkte diverse „Services“ an.[39] Dazu zählen z.B. die Beratung über allgemeine Vorgehensweisen in der Softwareentwicklung oder bei der Umsetzung von Softwarequalitätsmanagement. Neben den proprietären Dienstleistungen existiert ein sogenanntes Partner-Programm, an dem Beratungsunternehmen wie z.B. Capgemini teilnehmen. Diese erbringen u.a. allgemeine „Application Consulting Services“, die z.B. die Unterstützung bei der Einführung oder der Wartung von Micro Focus Produkten umfassen.

Für seine Produkte bietet Micro Focus Support auf verschiedenen Ebenen an. Für Standardprobleme existiert eine 24/7 Telefonhotline. Weiter werden u.a. über eine Fehler- und Problemdatenbank „Self-Help-Services“ zur Verfügung gestellt. Darüber hinaus besteht die Möglichkeit, feste Support-Verträge abzuschließen. Diese sind in einem gestaffelten System organisiert. Pro Support-Level existiert ein fest definierter Leistungsumfang. In umfangreicheren Support-Verträgen wird dem Kunden z.B. ein fester „Account Manager“ zu Verfügung gestellt.

Als vollständige Entwicklungsumgebung existiert eine direkte Abhängigkeit zur vom Hersteller gewählten Eclipse-Version.

[39] Vgl. (2012b)

Allgemeines	
Hersteller	Micro Focus
Softwaretyp	IDE
Quellcode	Geschlossen
Kosten/Lizenzierung	An Hersteller wenden[40]
Systemanforderungen	CPU o.A./RAM o.A./HDD 270 – 420 MB
Funktionale Kriterien	
Editorfunktionen	
Autovervollständigung	Ja
Code-Highlighting	Ja
Fehlererkennung (Syntax-Check)	Ja
Unterstützung von Gastsprachen	Ja (Identifikation und Fehlererkennung)
Debuging	
Grafisches Schritt-für-Schritt Debuging	Ja
Remote Debugging	Ja
Anbindung von externen Systemen	
SSH	Ja
Datenbanken	Ja (IBM DB2, Oracle, Sybase, Informix, OpenESQL (ODBC/JDBC)
Sonstiges	–
Nicht-Funktionale Kriterien	
Beratungsdienstleistungen	Ja (Eigene Dienstleistungen über Partnerunternehmen)
Support	Ja (Hotline/Self-Service/Gestaffelte Supportlevel)
Eclipse-Version	IDE basiert auf Eclipse 3.6
Sonstiges	
• COBOL und Java Integration: Konversion, simultane Nutzung • COBOL Projektunterstützung: spezielle Sichten in der IDE, Erkennung von Dateitypen	
URL	
• Produktseite: `http://bit.ly/M5yCaC` • Testversion des Produkts: `http://bit.ly/P7eLNE`	

Tabelle 2: Visual COBOL – Features

[40] Kontakt zum Hersteller: `http://bit.ly/awy0pa`

3.5.2 isCOBOL

Wie Visual COBOL[41] ist auch das durch den amerikanischen Hersteller Veryant vertriebene Produkt isCOBOL IDE kostenpflichtig.[42] Der Quellcode wird ebenfalls geschlossen gehalten. Rund um die Anwendung werden durch Veryant weitere Entwicklungstools, wie z.B. zur Anbindung von relationalen Datenbanken oder zum Betrieb von COBOL Anwendungen auf einem Java Application Server, angeboten. Grundsätzlich werden die IDE und der Compiler separat vertrieben und lizensiert. Veryant bietet mit seinen Produkten die Entwicklung in einem proprietären COBOL-Dialekt an, der nur über einen eigenen Compiler kompiliert werden kann. Bezüglich Bepreisung und Lizenzgestaltung existieren keine öffentlich einsehbaren Informationen. Diese müssen über den Vertrieb von Veryant individuell angefragt werden.

Der Code-Editor von isCOBOL bietet alle untersuchten Unterstützungsfunktionen für COBOL-Quellcode an. Wie in Abbildung 7 zu sehen ist, zählt dazu u.a. ein individuell konfigurierbares Syntax-Highlighting. Es besteht die Möglichkeit, die Farbdarstellung von Schlüsselwörtern beliebig zu konfigurieren. Über ein Kontextmenü assistiert der Editor mit der Vervollständigung der Syntax bzw. der Nutzung von Code-Templates über Schlüsselwörter. Beispielsweise kann über die Wahl des Schlüsselwortes „move“ in der Autovervollständigung die Generierung der Zeile „move varname to varname“ bewirkt werden. Fehlerhafte Syntax wird vom Editor erkannt und farblich markiert. Ferner werden Syntaxfehler im Eclipse-Problem-View aufgelistet.

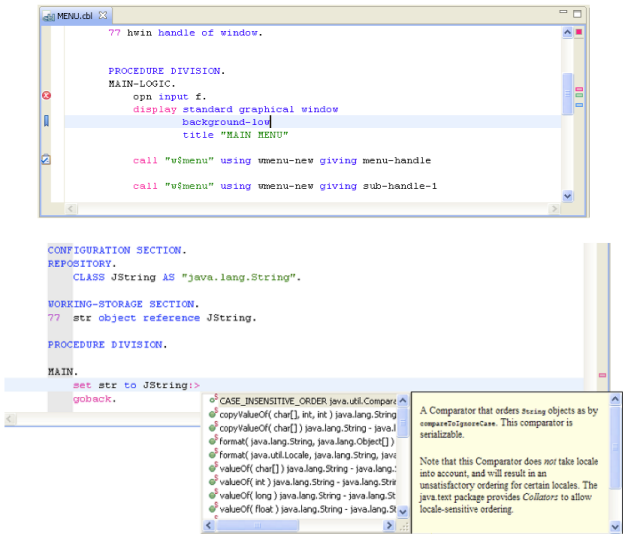


Abbildung 7: isCOBOL – Editor

Wie der nachfolgenden Abbildung 8 zu entnehmen ist, bietet isCOBOL über die Eclipsemaske „Debug Configurations“ an, COBOL Anwendungen zu debuggen. Hierbei unterstützt es grafisches Schritt-für-Schritt Debugging. Der Inhalt von Variablen kann zur Laufzeit über die Variables-View überwacht werden. Ebenfalls können Breakpoints im Quellcode gesetzt werden, um den Ablauf einer Anwendung an ausgewählten Stellen unterbrechen zu können.

[41] Produktseite: `http://bit.ly/N8cJKo`
[42] Vgl. (2012c)

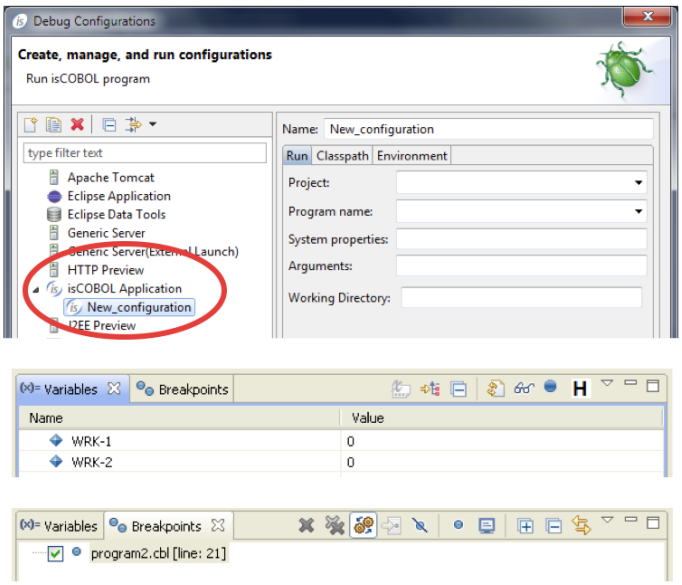


Abbildung 8: isCOBOL – Debugging

Die Anbindung von externen Systemen ist in isCOBOL über mehrere Wege möglich. Neben SSH/Telnet-Verbindungen können Unix, Linux und Windows-Hosts nativ angebunden werden. Ihre Konfiguration ermöglicht neben Shell-Verbindungen einen Direktzugriff auf das Dateisystem. Der Zugriff auf Datenbanken kann über den Zukauf der einleitend erwähnten Erweiterungen realisiert werden.

Veryant bietet neben dem Verkauf seiner Produkte Beratungsdienstleistungen zur Überführung von COBOL-Anwendungen in seinen proprietären Dialekt an.[43] Allgemeine Beratungsdienstleistungen, z. B. für generische Themen der Softwareentwicklung, werden nicht angeboten. Es existiert ebenfalls kein Partnerprogramm, wie es bei anderen Herstellern zu finden ist. In Deutschland werden Beratungsleistungen durch das Unternehmen EasiRun angeboten.

Im Supportbereich gibt es ein zweigeteiltes System.[44] Es existieren zum einen Supportverträge, die in zwei Leistungsumfängen angeboten werden (Standard und Platinum). Zum anderen kann bei Existenz eines Supportvertrages mit dem „Independent Software Vendor (ISV) Program“ ein ergänzendes Leistungsangebot erworben werden. Diese Supportverträge umfassen neben webbasierten Self-Services (z.B. Datenbanken mit Lösungen zu bekannten Fehlern), Remote-Support durch Mitarbeiter von Veryant. Im Standard-Vertrag sind diese zu definierten Geschäftszeiten an Werktagen verfügbar. Im Platinum-Vetrag wird eine 24/7 Erreichbarkeit dieser Mitarbeiter mit fest definierten Reaktionszeiten bereitgestellt.

Als vollständige Entwicklungsumgebung existiert eine direkte Abhängigkeit zur vom Hersteller gewählten Eclipse-Version.

[43] Vgl. (2012a)
[44] Vgl. (2012b)

Allgemeines	
Hersteller	Veryant
Softwaretyp	IDE
Quellcode	Geschlossen
Kosten/Lizenzierung	An Hersteller wenden[45]
Systemanforderungen	CPU o.A./RAM o.A./HDD o.A.
Funktionale Kriterien	
Editorfunktionen	
Autovervollständigung	Ja
Code-Highlighting	Ja
Fehlererkennung (Syntax-Check)	Ja
Unterstützung von Gastsprachen	Nein
Debuging	
Grafisches Schritt-für-Schritt Debuging	Ja
Remote Debugging	Ja
Anbindung von externen Systemen	
SSH	Ja
Datenbanken	Ja (IBM DB2, Oracle, Sybase, Informix, OpenESQL (ODBC/JDBC)
Sonstiges	FTP
Nicht-Funktionale Kriterien	
Beratungsdienstleistungen	Ja, jedoch nur für die Migration von Anwendungen auf den proprietären COBOL-Dialekt.
Support	Ja (Hotline/Self-Service/Gestaffelte Supportlevel)
Eclipse-Version	IDE basiert auf Eclipse 3.7
Sonstiges	
• Kompilierung auf Remote Compiler • Mischung von Java und COBOL	
URL	
• Produktseite: `http://bit.ly/N8cJKo`	

Tabelle 3: isCOBOL – Features

[45] Herstellerseite: `www.veryant.com`

3.5.3 maxenso cobolclipse

Das unter dem Namen maxenso cobolclipse[46] von Innowake vertriebene COBOL-Enwicklungstool ist als reines Plugins sowie in einer eigenständigen, speziell zugeschnittenen Eclipse Version zu erhalten.[47] Aus funktionaler Sicht existiert zwischen diesen Versionen kein Unterschied. Das Bundle ist lediglich eine optisch angepasste Eclipse-Version. Cobolclipse wird ohne eigenen Compiler ausgeliefert. Es bedient sich stets des in der Einsatzumgebung vorhanden Compilers. Innowake vertreibt das Werkzeug als kommerzielles Produkt, dessen Quellcode geschlossen gehalten wird. Die Preis- und Lizenzgestaltung erfolgt in individueller Absprache. Pauschale Informationen konnten nicht gewonnen werden.

Cobolclipse verfügt über einen Code-Editor mit, bezogen auf diese Testreihe, weitreichender Funktionalität. Diese zeichnet sich besonders durch die Analysemöglichkeit von referenziellen Abhängigkeiten zwischen den Dateien einer COBOL-Anwendung aus. Beispielsweise existiert über die „Object Dependencies View“ die Möglichkeit, Abhängigkeitshierarchien in einer Baumstruktur darzustellen. Hierbei ist die Analyse aus zwei Richtungen möglich: Anzeige der von einem Programm verwendeten Module sowie die Anzeige der Programme, die ein spezielles Modul verwenden. Ferner bietet der Editor eine „Hyperlinking“ genannte Funktionalität an. Wie bei der Java-Entwicklung in Eclipse möglich, kann durch eine spezielle Steueranweisung auf eine Variable ihr Definitionsort bzw. ihr Auftreten in Copybüchern angezeigt werden.

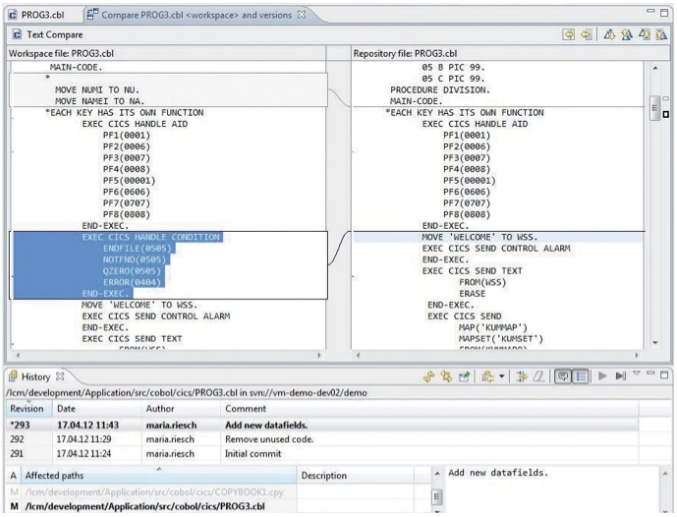


Abbildung 9: cobolclipse – Editor mit Abhängigkeitsanalyse

Über die Kombination von „Strg+Leertaste“ kann die Autovervollständigung des Editors aktiviert werden. Wie Abbildung 10 zu entnehmen ist, assistiert der Editor dem Entwickler ebenfalls durch das Vorschlagen von Befehlen bzw. sonstiger Syntax.

Neben Code-Highlighting verfügt cobolclipse auch über eine Fehlerkennung der Syntax über Hintergrundkompilierung. Die Besonderheit ist hierbei, dass diese Funktion auch ohne Existenz

[46] Produktseite: `http://bit.ly/N8aXcc`
[47] Vgl. (2012) , S. 1

eines lokalen Compilers verfügbar ist. Nach entsprechender Konfiguration sendet cobolclipse den Quellcode im Hintegrund an ein entferntes System, kompiliert diesen und überträgt entdeckte Fehler zurück in den Editor. Der Start dieses Vorgangs kann wahlweise an das Speichern einer Datei gebunden oder durch manuelle Benutzereingabe ausgelöst werden.

Eine Unterstützung von Gastsprachen und zugehörigen Pre-Prozessoren existiert, wenn auch mit limitierter Funktionalität. Cobolclipse unterstützt lediglich Gastsprachen, die mit dem COBOL 88 Standard kompatibel sind. Die Einbindung der Pre-Prozessoren ist mit situationsabhängigem Aufwand manuell möglich.

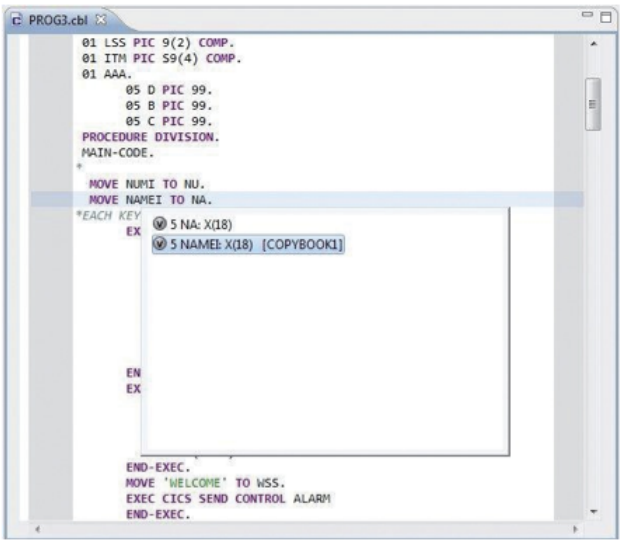


Abbildung 10: cobolclipse – Editor

Das Debugging von COBOL-Anwendungen wird durch die Anbindung entsprechender Schnittstellen der Ausführungsumgebung ermöglicht. Hierbei verwendet cobolclipse die Standardfeatures von Eclipse und stellt keine eigenen zusätzlichen Funktionalitäten zur bereit.

Eine Anbindung von externen Systemen per SSH wird nicht unterstützt, jedoch existiert eine 3270 Terminal Schnittstelle. Hierüber ist die Anzeige des Dateisystems des entfernten Systems in einem Dateiexplorer möglich. Für die Anbindung von Datenbanksystemen existieren keine speziellen Funktionalitäten.

Über den Verkauf des Produktes hinaus bietet innoWake eine Reihe von eigenen Dienstleistungen an. Dazu zählen z.B. Workshops zur Vorstellung von cobolclipse und Anwenderschulungen. Daneben existieren Beratungsdienstleistungen zur Installation und Inbetriebnahme der Anwendung. Im Bereich von Support unterstützt innoWake seine Kunden durch das Angebot von individuell gestaltbaren Wartungsverträgen. Neben einer Telefonhotline, Self-Help-Services (Fehlerdatenbank, Handbücher, etc.) existiert eine Webanwendung, über die der Kunde Hilfeanfragen bei innoWake platzieren kann.

Allgemeines	
Hersteller	innoWake
Softwaretyp	Plugin & IDE
Quellcode	Geschlossen
Kosten/Lizenzierung	
Systemanforderungen	CPU o.A./RAM o.A./HDD o.A.
Funktionale Kriterien	
Editorfunktionen	
Autovervollständigung	Ja
Code-Highlighting	Ja
Fehlererkennung (Syntax-Check)	Ja
Unterstützung von Gastsprachen	Ja, falls COBOL88 kompatibel
Debuging	
Grafisches Schritt-für-Schritt Debuging	Nach manueller Konfiguration
Remote Debugging	Ja
Anbindung von externen Systemen	
SSH	Nein
Datenbanken	Nein
Sonstiges	Dateiexplorer, Terminal-Anbindung 3270
Nicht-Funktionale Kriterien	
Beratungsdienstleistungen	Vorabberatung, Unterstützung bei Installation und Inbetriebnahme, Schulung
Support	Individuelle Wartungsverträge, Online-Ticket-System, Hotline, Wissensdatenbank
Eclipse-Version	IDE basiert auf Eclipse 3.7
Sonstiges	
–	
URL	
• Produktseite: `http://bit.ly/N8aXcc`	

Tabelle 4: cobolclipse – Features

3.5.4 Elastic COBOL

Elastic Cobol vom Hersteller Heirloom Computing[48] kann in mehreren Varianten genutzt werden. Es besteht die Möglichkeit, ein Paket von Plugins zu installieren, welches die Installation und vor allem Integration in einer bestehenden Eclipse-Umgebung ermöglicht. Alternativ kann ein mit allen Plugins ausgerüstetes Eclipse komplett installiert werden. Dies empfiehlt sich bei einer Neueinführung von Eclipse. Die dritte Variante ist es, Elastic Cobol komplett als Clouddienst zu nutzen.[49]

Der cloudbasierte Dienst bietet dem Benutzer eine vollständig eingerichtete Installation der aktuellsten Eclipse-Version inklusive aller benötigten Plugins von Elastic COBOL. Benutzt werden kann die IDE per Remote-Desktop. Dabei stehen dem Benutzer zusätzlich zur eigentlichen IDE ein Tomcat Applikationsserver oder in der Enterprise Edition eine Oracle 11g XE und ein Apache Geronimo Applikationsserver zur Verfügung. Möchte man Elastic COBOL selbst installieren, so können sämtliche Plugins direkt in eine bestehende Eclipse-Umgebung installiert werden oder eine fertig konfigurierte Eclipse-Version heruntergelXXXn werden. Diese steht für Linux und Windows, sowohl in einer 32-Bit als auch einer 64-Bit Version, zur Verfügung. Zur Verwendung der Plugins ist eine properties-Datei notwendig, welche sicherstellt, dass eine geeignete Lizenz erworben wurde.[50]

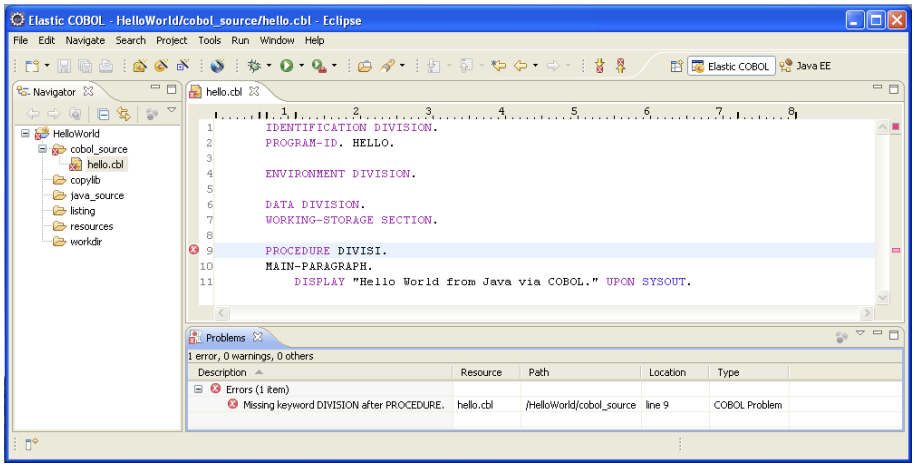


Abbildung 11: Elastic COBOL – Perspektive, Editor und Syntax-Check

In Abbildung 11 ist der von Elastic COBOL bereitgestellte Editor zu sehen. Festzustellen ist, dass eine spezielle Perspektive für Elastic COBOL eingerichtet wurde und angeboten wird. Der Editor selbst unterstützt in Hinblick auf die vordefinierten Untersuchungskriterien Code-Highlighting sowie die Fehlererkennung (Syntax-Check), was ebenfalls Abbildung 11 entnommen werden kann. Die Unterstützung von Gastsprachen ist stark eingeschränkt. So werden laut Language Reference Manual[51], sowie auf Nachfrage durch Heirloom Computings President und Chief Executive Officer Gary Crook[52] bestätigt, lediglich Java, SQL und HTML über den Befehl EXEC unterstützt.

[48] Produktseite: `www.elasticcobol.com`
[49] Vgl. (2012a) , S. 5 f.
[50] Vgl. ebd., S. 4
[51] Vgl. (2012b) , S. 317
[52] Crook, G. (2012)

Hinsichtlich Debugging bietet Elastic COBOL alle aus Eclipse bekannten Funktionen. Es ist dem Benutzer demzufolge möglich, den Programmcode sowohl Schritt-für-Schritt zu prüfen, als auch an unterschiedlichen Stellen Breakpoints zu setzen. Hiermit können beispielsweise sämtliche Variablen an der markierten Stelle eingesehen werden. Auch das Debuggen von Code auf einem entfernten Großrechner ist dank Remote-Debugging möglich.[53] Zusammenfassend werden somit in Bezug auf Debugging und Unterstützung bei der Fehlerbehebung sämtliche Kriterien erfüllt. In Abbildung 12 ist die Perspektive zur Fehlerbehandlung aufgezeigt.

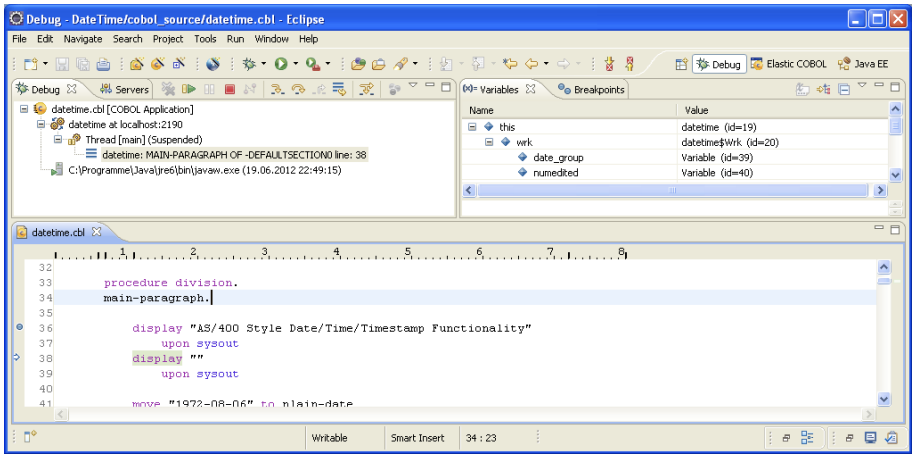


Abbildung 12: Elastic COBOL – Debugging

Defizite weist dieses Produkt besonders vor dem Hintergrund der Verwendung eigener Pre-Prozessoren auf, da deren Unterstützung oder Integration nicht vorgesehen ist.[54] Auch der direkte Zugriff auf das entfernte System über SSH ist nicht möglich. Hierfür müssten weitere Plugins installiert und eingerichtet werden. Customer Information Control System (CICS) wird teilweise unterstützt.

Support bietet Heirloom Computing in Form verschiedener Dokumente und Foren zur Selbsthilfe. Nichtsdestoweniger ist es möglich, ein Support-Ticket zu erstellen, um Fragen an den Hersteller zu richten. Ein telefonischer Support existiert nicht.[55] Die Kosten des Produkts können sich je nach Variante und Turnus unterschiedlich gestalten, wobei sich die nachfolgenden Preise jeweils auf eine Lizenz eines einzelnen Entwicklers beziehen. Für die Nutzung der Cloudlösung fallen Gebühren in Höhe von XX am Tag, XX im Monat oder XX im Jahr an. Für eine eigene Installation auf einem Desktoparbeitsplatz werden XX im Monat oder XX im Jahr erhoben. Darüber hinaus gibt es weitere serverseitige Angebote, welche der Herstellerwebseite entnommen werden können.[56]

[53] Vgl. (2012b) , S. 16 ff.
[54] Crook, G. (2012)
[55] Supportübersicht auf der Herstellerseite: `http://bit.ly/PbcDWk`
[56] Weitere Informationen sind unter `http://bit.ly/Lmbryc` zu finden.

Allgemeines	
Hersteller	Heirloom Computing
Softwaretyp	Plugin, IDE & Clouddienst
Quellcode	Geschlossen
Kosten/Lizenzierung	
Systemanforderungen	CPU o.A./RAM o.A./HDD o.A.
Funktionale Kriterien	
Editorfunktionen	
Autovervollständigung	Nein (auf Roadmap)
Code-Highlighting	Ja
Fehlererkennung (Syntax-Check)	Ja
Unterstützung von Gastsprachen	Eingeschränkt (SQL, Java & HTML)
Debuging	
Grafisches Schritt-für-Schritt Debuging	Ja
Remote Debugging	Ja
Anbindung von externen Systemen	
SSH	Nein
Datenbanken	Nein
Sonstiges	–
Nicht-Funktionale Kriterien	
Beratungsdienstleistungen	Ja (E-Mail innerhalb 24 Stunden)
Support	Ja (Dokumente, Foren & Support-Ticket)[57]
Eclipse-Version	Eclipse 3.7
Sonstiges	
• Ausschließlich Englisch • Einziges Produkt im Vergleich, welches als Clouddienst genutzt werden kann	
URL	
• Produktseite: `https://www.elasticcobol.com`	

Tabelle 5: Elastic COBOL – Features

[57] Preisübersicht auf der Herstellerseite: `http://bit.ly/Lmbryc`
[57] Supportübersicht auf der Herstellerseite: `http://bit.ly/PbcDWk`

3.5.5 Cobos

Das Produkt Cobos vom französischen Hersteller metrixware ist eine kommerzielle OpenSource-Lösung, welche unter dem Metrixware Community License Agreement (MCLA)[58] vertrieben wird. Trotz des offenen Quellcodes fallen für die Verwendung von Cobos außerhalb der testweisen, persönlichen oder schulischen Nutzung Lizenzkosten an. Dabei wird aktuell zwischen den Lizenzmodellen Standalone Edition für XXX pro Jahr und Benutzer, Professional Edition für XXX pro Jahr und Benutzer und Enterprise Edition für XXX pro Jahr und Benutzer unterschieden. Genauere Informationen können diesbezüglich auf der Webseite des Herstellers eingesehen werden.[59]

Cobos setzt bei der Wahl des Editors (Abbildung 13) auf bereits bestehende Open Source Projekte wie IDE Cobol, die dann weitere Funktionen wie Kopieren-Ersetzen, einstellbaren Tabstopp und Großschreibweise bereitstellen.[60] Durch diese Verwendung von bereits vorhandenen Open Source Tools können Abhängigkeiten zwischen den einzelnen Komponenten nicht ausgeschlossen werden. So ist das eben genannte Projekt IDE Cobol bspw. bereits eingestellt, jedoch verspricht metrixware die aktive Wartung.[61] Diese Punkte gilt es insbesondere zu beachten. Von den geforderten Kriterien erfüllt der Editor die Aspekte der Autovervollständigung von Befehlen und das Highlighting von Schlüsselwörtern in der Syntax. Zudem unterstützt er den Benutzer bei der frühzeitigen Behebung von Fehlern in der Syntax. In Bezug auf Gastsprachen wird CICS, DB2, DL1, JCL und REXX unterstützt[62]. Über die Integration eigener Pre-Prozessoren kann keine Aussage getroffen werden.

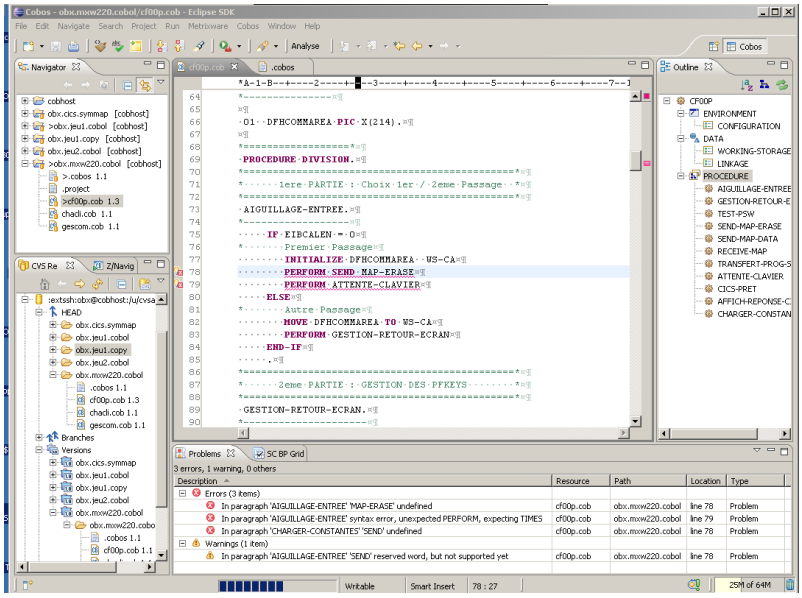


Abbildung 13: Cobos – Editor mit Syntax-Check

Auch Cobos unterstützt ein grafisches Schritt-für-Schritt-Debugging, wie man es von der Java-Programmierung in Eclipse gewohnt ist. Auch eine Remote-Debugging-Funktionalität wird

[58] Lizenzvereinbarung auf der Herstellerseite: `http://bit.ly/MTJ9GB`
[59] Preisübersicht auf der Herstellerseite: `http://bit.ly/LRSrRR`
[60] Produktseite: `http://bit.ly/Ra39I5`
[61] Übersicht von Service und Support auf der Herstellerseite: `http://bit.ly/LRSrRR`
[62] Vgl. (2012)

bereitgestellt.

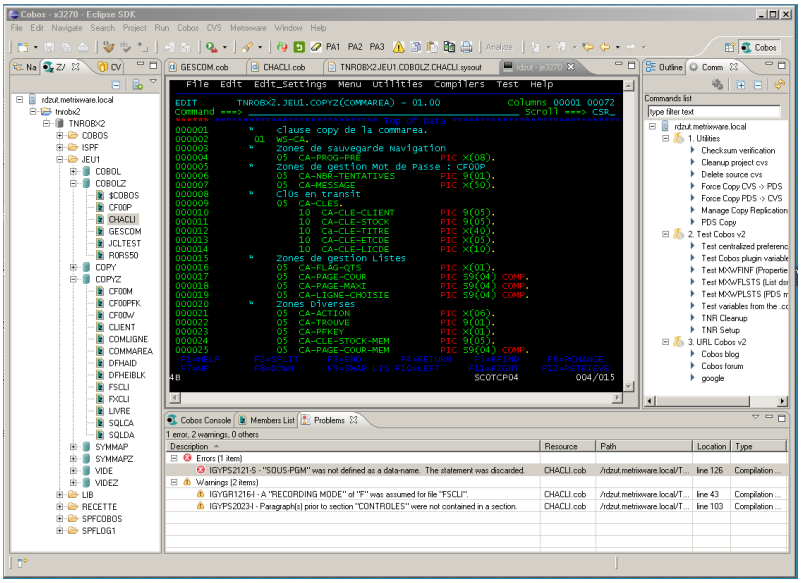


Abbildung 14: Cobos – 3270 Emulator mit entferntem Dateisystem

Eine große Stärke von Cobos liegt in der Integration vieler, für die Entwicklung von COBOL notwendiger Tools. So ist es mit Cobos (ab der Professional Edition) möglich, im Dateisystem des Großrechners, welcher mit z/OS betrieben wird, zu navigieren und Dateien anzulegen und zu bearbeiten. Hierfür kommt ein Tool namens z/Navigator zum Einsatz, welches wiederum vom jeweilig eingesetzten Host-Integration-Modul abhängt. Zudem können (ebenfalls ab der Professional Edition) sehr einfach Befehle an den Host über ein zusätzliches Plugin gesendet werden. Konkret bedeutet dies, dass der Entwickler in die Lage versetzt wird, REXX und Shell Skripte auf dem Großrechner direkt aus Eclipse heraus zu starten. Für weitere administrative Zwecke auf dem Mainframe bietet Cobos letztlich auch einen integrierten 3270 Terminal Emulator (Abbildung 14), mit dem diverse Operationen direkt aus Eclipse auf dem Mainframe durchgeführt werden können. Dieser Emulator basiert auf dem Plugin je3270. Auch hinsichtlich Versionsverwaltung von Quellcode ist eine volle Integration mit bekannten Tools wie CVS, SVN oder Git möglich.

In Bezug auf Support sind in den diversen Preismodellen auch unterschiedliche Serviceleistungen inkludiert. Während bei der Standalone Edition lediglich ein onlinebasiertes Ticketsystem mit festgelegten Antwortzeiten zur Verfügung steht, erhalten Kunden der Professional Edition zusätzlich Telefonsupport. Lediglich Kunden der Enterprise Edition haben auch die Möglichkeit, eine Remotediagnose durch den Hersteller sowie periodische Treffen und Reports zu erhalten.

Allgemeines	
Hersteller	metrixware
Softwaretyp	Plugin
Quellcode	Open-Source (MCLA)
Kosten/Lizenzierung	
Systemanforderungen	CPU o.A./RAM o.A./HDD o.A.
Funktionale Kriterien	
Editorfunktionen	
Autovervollständigung	Ja
Code-Highlighting	Ja
Fehlererkennung (Syntax-Check)	Ja (Nach lokaler Kompilierung)
Unterstützung von Gastsprachen	Ja (CICS, DB2, DL1, JCL, REXX)
Debuging	
Grafisches Schritt-für-Schritt Debuging	Ja
Remote Debugging	Ja
Anbindung von externen Systemen	
SSH	Ja
Datenbanken	Keine Angaben
Sonstiges	Dateiexplorer, Terminal-Anbindung 3270
Nicht-Funktionale Kriterien	
Beratungsdienstleistungen	Keine Angaben
Support	Ja, je nach Preismodell Online-Ticket, Telefonsupport oder persönliche Hilfe
Eclipse-Version	keine Angaben
Sonstiges	
–	
URL	
• Produktseite: `http://bit.ly/Ra39I5`	
• Quellcode bei SourceForge: `http://bit.ly/dsAEui`	

Tabelle 6: Cobos – Features

[62] Übersicht der Kosten und Lizenzen auf der Herstellerseite: `http://bit.ly/LRSrRR`

3.5.6 Rational Developer for System z (RDz)

Die IDE Rational Developer for System z (RDz) ist die kommerzielle Entwicklungsumgebung zur COBOL-Entwicklung der Firma IBM. Wie der Name bereits vermuten lässt, ist dieses Werkzeug für die Entwicklung auf dem, ebenfalls von IBM vertriebenen, System z gedacht. RDz setzt auf der Eclipse-Plattform auf und erweitert diese um wesentliche Funktionen zur COBOL-Entwicklung aber auch um den interaktiven Zugang zum jeweiligen Großrechner mit zOS. Für eine ganzheitliche Entwicklung bietet RDz unterschiedliche Schnittstellen, mit denen es nicht nur möglich ist, COBOL für System z zu entwickeln, sondern bspw. auch Java für Unix-Systeme.

Der Entwicklungsablauf gestaltet sich unter Verwendung des RDz in der Regel wie folgt[63]:

- Zunächst werden die benötigten Dateien in der IDE geöffnet. Dies kann sowohl lokal, als auch entfernt auf dem Großrechner geschehen.
- Während der Bearbeitung des Codes, welche sowohl direkt auf dem Großrechner, als auch auf einem lokalen Laufwerk stattfinden kann, können dem Entwickler Fehler unterlaufen. Um frühzeitig auf Syntaxfehler aufmerksam zu machen, existieren drei Stufen des Syntax-Checks:
  1. Echtzeit-Syntax-Check: Parsing und Fehlererkennung sofort in der IDE
  2. Lokaler Syntax-Check: Fehlererkennung nach vorheriger Kompilierung auf lokalem Rechner
  3. Remote Syntax-Check: Fehlererkennung nach Kompilierung auf dem Großrechner unter Verwendung des produktiv eingesetzten Compilers.
- Zusätzlich findet die Übersetzung von Gastsprachen wie z.B. Structured Query Language (SQL) oder CICS durch hostseitige Pre-Prozessoren
- Abschließend ist die Möglichkeit gegeben, die entfernte Anwendung unter Zuhilfenahme weit verbreiteter Eclipse-Werkzeuge zu debuggen.

Der zur Verfügung gestellte Editor bietet neben dem gewohnten Code-Highlighting auch Möglichkeiten zur Auto-Vervollständigung. Hinsichtlich Syntax-Check wurden bereits die Möglichkeiten des Echtzeit-Syntax-Checks, des Syntax-Checks mit vorhergehender lokaler und entfernter Kompilierung angesprochen. Diese wesentlichen Merkmale sowie die generelle Darstellung des Editors können den Abbildungen 15 und 16 entnommen werden.

An Eclipse-Funktionalitäten angelehnt kann sowohl ein grafisches Schritt-für-Schritt Debugging, auch unter Verwendung von Breakpoints, als auch das entfernte Debugging erfolgen.

[63] XXX, D. (2012)

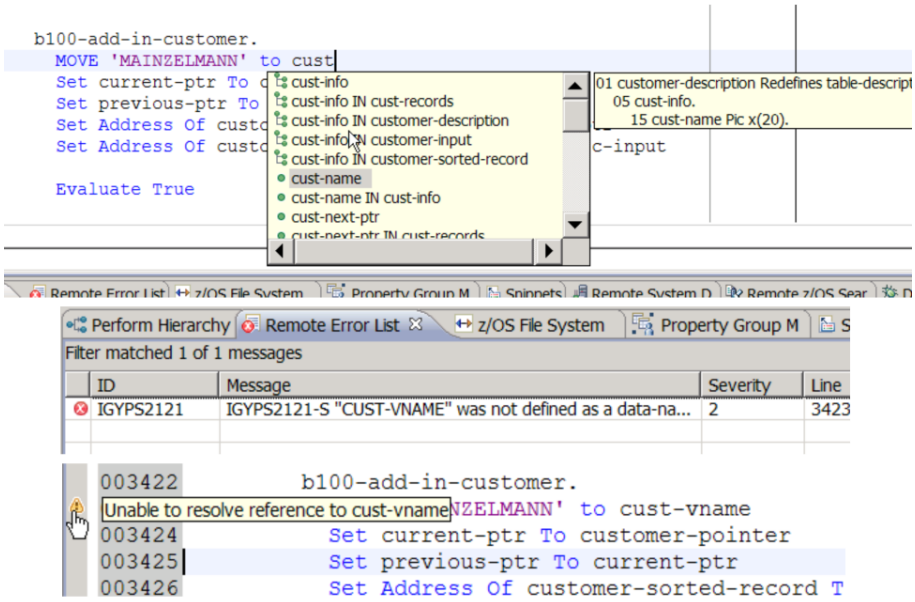


Abbildung 15: RDz – Editor mit Autovervollständigung und Syntax-Check

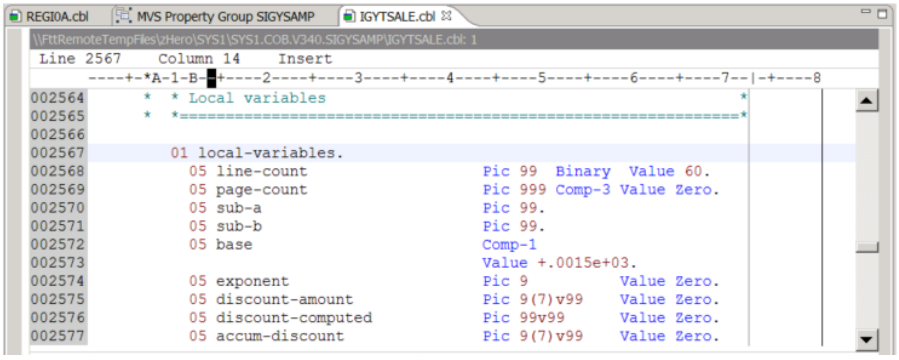


Abbildung 16: RDz – Editor

Zusätzlich können Befehle an den Mainframe gesendet, das Dateisystem durchsucht und die Dateien direkt auf dem Mainframe bearbeitet werden. Zudem steht es dem Benutzer offen, ob er einen an das 3270-Terminal oder einen an die Eclipse-Java-Umgebung angelehnten Editor wählt. Außerdem ist es möglich, mit zwei Editor-Fenstern zur selben Zeit zu arbeiten und Vergleiche zwischen unterschiedlichen Quellen durchzuführen. Auch die Integration von eigenen Pre-Prozessoren ist problemlos möglich, womit Gastsprachen außerhalb des vom Compiler beherrschten Sprachumfangs unterstützt werden können.

RDz kann in mehreren Preismodellen mit unterschiedlichem Leistungsumfang erworben werden. Der empfohlene Verkaufspreis für eine Einzelplatzlizenz wird jährlich mit 6784,19€ beziffert.

Allgemeines	
Hersteller	IBM
Softwaretyp	IDE
Quellcode	Geschlossen
Kosten/Lizenzierung	
Systemanforderungen	CPU 1 GHz/RAM 2 GB/HDD 3 GB[64]
Funktionale Kriterien	
Editorfunktionen	
Autovervollständigung	Ja
Code-Highlighting	Ja
Fehlererkennung (Syntax-Check)	Ja (Echtzeit sowie nach lokalem oder entferntem Kompilieren)
Unterstützung von Gastsprachen	Ja
Debuging	
Grafisches Schritt-für-Schritt Debuging	Ja
Remote Debugging	Ja
Anbindung von externen Systemen	
SSH	Ja
Datenbanken	Ja
Sonstiges	Dateiexplorer, Mainframe-Befehle
Nicht-Funktionale Kriterien	
Beratungsdienstleistungen	Ja (Durch IBM oder Business Partner)
Support	Ja (Meist 12 Monate inbegriffen und frei skalierbar)
Eclipse-Version	Eclipse 3.6
Sonstiges	
URL	
Produktseite: `http://bit.ly/RabaNl`	

Tabelle 7: Rational Developer for System z – Features

[64] Preisübersicht auf der Herstellerseite: `http://bit.ly/Ng5X3R`
[64] Übersicht der Systemanforderungen auf der Herstellerseite: `http://ibm.co/Nr1Srr`

3.6 Priorisierung der Anforderungen

Die in Kapitel 3.4 vorgestellten Anforderungen sollen in einem dritten Schritt validiert und priorisiert werden. Hierzu wird eine simpler Ansatz gewählt.

Die erarbeitete Anforderungsliste wird in einer weiteren Interview-Runde den IT-Führungskräften der Versicherung vorgelegt. Die identifizierten Anforderungen werden mit den Führungskräften diskutiert und in diesem Zuge validiert. Dann werden die Unternehmensvertreter gebeten, jeder Anforderung eine Priorität von 4 (sehr wichtig) bis 1 (weniger wichtig) zuzuordnen.

Die unten stehende Tabelle 8 repräsentiert die durch die Versicherung vorgenommene Priorisierung der Requirements.

Funktionale Kriterien	**Priorität**
Editorfunktionen	
Autovervollständigung	4
Code-Highlighting	4
Fehlererkennung (Syntax-Check)	4
Unterstützung von Gastsprachen	4
Debuging	
Grafisches Schritt-für-Schritt Debuging	2
Remote Debugging	2
Anbindung von externen Systemen	
SSH	1
Datenbanken	4
Nicht-Funktionale Kriterien	
Beratungsdienstleistungen	4
Support	4

Tabelle 8: Gewichtung der Vergleichskriterien

Eine differenzierte Priorisierung der Anforderungen wäre beispielsweise mit Hilfe der Methode des Analytical Hierarchy Processes (AHP)[65] möglich gewesen. Im Zuge dessen werden einzelne Anforderungen in mehreren Iterationen mit Hilfe numerischer Bewertung relativ zueinander gewichtet. Diese relativen Verhältnisse werden auf ihre Plausibilität hin untersucht und es wird schließlich eine Priorisierung der Anforderungen abgeleitet. Diese Methode ist jedoch verhältnismäßig zeitaufwändig.[66,67] Auf Grund der Begrenztheit der Ressourcen Zeit und Mitarbeiterverfügbarkeit wurde im Rahmen der Fallstudie auf das Anwenden einer solcher Methode verzichtet. Der gewählte Ansatz kann jedoch als hinreichend betrachtet werden, da die Versicherung bereits im Voraus klare Vorstellungen über den avisierten Soll-Zustand entwickelt hatte.

[65] Vgl. (2001)
[66] Vgl. (1998)
[67] Vgl. (2010)

3.7 Gegenüberstellung

Auf Basis dieser Gewichtung und der Untersuchungsergebnisse der verschiedenen Entwicklungs-Tools ergibt sich die in Tabelle 8 dargestellte Punktevergabe.

Diese Übersicht zeigt, dass sich die Produkte Rational Developer for System z (RDz) (38 Punkte) und Visual COBOL (37 Punkte) abheben. Alle anderen Werkzeuge erhalten zwischen 20 und 28 Punkten. Diese Zweiteilung kann auf Basis der Punkteverteilung an drei Gründen festgemacht werden. Es zeigt sich, dass Micro Focus und IBM ein vielschichtiges Angebot an Service-Dienstleistungen neben ihren Produkten anbieten. Hierbei profitieren beide Hersteller aus Sicht der Punktevergabe u.a. davon, dass eine Reihe von Drittfirmen ebenfalls IT-Beratung für ihre Produkte anbieten. Dies ist aus Geschäftskundensicht ein wichtiges Kaufargument, da im Krisenfall eine hochverfügbare und leistungsfähige Servicestruktur gewährleistet sein muss. Der deutsche Hersteller innoWake verfügt, aus Sicht der Dienstleistungserbringung, über eine ähnliche Stärke. Als mittelständisches Unternehmen kann innoWake flexibel und zeitnah auf die Anforderungen seiner Kunden im deutschen Raum eingehen. Ausländische Hersteller ohne deutsche Partner können dies sehr wahrscheinlich nicht in ähnlicher Weise.

Neben diesen nicht-funktionalen Aspekten heben sich Visual COBOL sowie der RDz auch im Bereich Funktionalität ab. Im Hinblick auf die Editorfunktionen, welche die Versicherung als sehr wichtig priorisiert hat, wird deutlich, dass beide Produkte die komfortabelsten und umfassendsten Editor-Funktionen bereitstellen. So ist der RDz beispielsweise das einzige Tool, das Autovervollständigung für Gastsprachen anbietet. Dieser Aspekt schlägt sich in den vergebenen Sonderpunkten nieder. Eine Ausnahme bildet hierbei Elastic COBOL, da es als einziges Produkt keine Autovervollständigung von Befehlen besitzt.

Weiterhin verfügen Visual COBOL und RDz über einer Vielzahl von Anbindungsmöglichkeiten externer Systeme. Nur diesen beiden Produkte sind ohne zusätzliche Komponenten in der Lage, Datenbanksysteme anzusprechen. Dies stellt ebenfalls ein Kriterium dar, welches von der Versicherung als wichtig priorisiert wurde. Der RDz profitiert darüber hinaus von der koexistierenden Produktplattform Rational. Über diese kann eine native Integration mit einer Vielzahl weiterer Anwendungen, wie z.B. Source-Code-Management (SCM) oder Quality Assurance (QA)-Systeme, realisiert werden.

An dieser Stelle soll gesondert auf das unter einer OSS-Lizenz vertriebene Produkt Cobos hingewiesen werden. Dessen Funktionsumfang ist weitreichend und bietet dem Entwickler an einigen Stellen mehr als andere Produkte. Dennoch spiegelt die Gesamtbewertung diese funktionale Leistungsfähigkeit nicht in vollem Umfang wider. Grund hierfür ist die Tatsache, dass es im Rahmen dieser Arbeit nicht möglich war, Informationen bzgl. Beratungsdienstleistungen einzuholen. Bezüglich des Unternehmens metrixware ist herauszustellen, dass es sich um einen französischen Hersteller handelt, wodurch ggf. zusätzliche sprachliche Barrieren entstehen könnten.

Abschließend lassen sich drei Punkte festzuhalten, die bei der Auswahl eines der vorgestellten Werkzeuge berücksichtigt werden sollten. Es zeigt sich, dass zu den vermeintlich kostenintensiven Produkten eine *größere Auswahl an Dienstleistungsanbietern* existiert. Hintergrund dessen ist

primär, dass diese Produkte von Großunternehmen angeboten werden. Für Produkte von Unternehmen dieser Größe existiert ein umfangreiches Beratungsangebot durch Drittunternehmen. Hersteller wie Veryant, Heirloom Computing oder Metrixware sind dagegen als Mittelstand bzw. Kleinunternehmen einzustufen, die tendenziell autark auf dem Markt agieren. Bei der Implementierung eines Produktes dieser Hersteller gilt somit abzuwägen, ob die vermeintlich niedrigere Serviceverfügbarkeit als ausreichend einzustufen ist. Relevant ist dieser Aspekt vor allem dann, wenn der Hersteller nicht aus Deutschland bzw. aus dem englischsprachigen Ausland stammt.

Einen weiteren kritischen Faktor stellen die Unterschiede zwischen den *Editor-Funktionen der Werkzeuge* dar. Wie zuvor beschrieben wurde, bietet nur der RDz eine Autovervollständigung für Gastsprachen. Aus Sicht der Versicherung, die nicht ausschließlich Gastsprachen verwendet, die Teil des nativen Sprachumfangs sind, kann das Fehlen dieser Unterstützung u.U. als Grund gegen den Kauf eingestuft werden.

Die Unterschiede in Bezug auf die *Anbindung externer Systeme* können als untergeordnet priorisiert werden, da für Eclipse diesbezüglich eine große Anzahl von spezialisierten Plugins existiert. Hierbei würde der Zugriff auf ein Datenbanksystem z.B. über ein Plugin eines Drittherstellers und nicht über das COBOL-Entwicklungs-Plugin geschehen. Auf Grund der Reduktion von Abhängigkeiten und Heterogenität der Werkzeuglandschaft wäre eine Vermeidung dieses Szenarios natürlich sinnvoll.

4 Diskussion

4.1 Kritische Reflexion

An dieser Stelle soll beleuchtet werden, welche Faktoren bei der Ausarbeitung dieser Fallstudie limitierend wirkten und welche Implikationen sich hieraus ergeben.

COBOL-Kenntnisse: Zunächst ist festzuhalten, dass keiner der Autoren COBOL-Entwickler ist. Das Wissen und insbesondere die praktische Erfahrung hinsichtlich der Spezifika der Programmiersprache müssen daher als begrenzt eingestuft werden. Dies impliziert, dass in zahlreichen Belangen auf die Angaben von Gesprächspartnern und die korrekte Darstellung in der Literatur vertraut wurde.

Test-Infrastruktur: Es ist festzustellen, dass trotz einer Installation und Konfiguration einzelner Plugins in einer Testinstanz keine adäquate Testumgebung für jedes evaluierte Produkt zur Verfügung stand. Dies bedeutet, dass die Evaluation der Plugins vorrangig auf theoretischer Basis stattfand. Eine repräsentative Simulation des Build-Prozesses der Versicherung war im Rahmen der Fallstudie nicht möglich. Dies impliziert das Potenzial der Nichtbeachtung infrastruktureller Spezifika, die erst in der praktischen Anwendung der Lösungen zu Tage treten. Insbesondere wurde die Java-Entwicklung der Versicherung nicht gesondert betrachtet bzw. in die vorliegende Analyse miteinbezogen. Es ist aus diesem Grund nicht auszuschließen, dass bestimmte Entwicklungsmethoden oder IDE-Konfigurationen Implikationen auf die zu integrierenden COBOL-Plugins haben könnten.

Ressourcenknappheit: Als letzter limitierender Aspekt ist die Begrenztheit der Ressourcen zu nennen, die zur Bearbeitung dieser Fallstudie zur Verfügung standen. Am schwersten wiegt dabei die begrenzte Zeit zur Erstellung der Arbeit. Daneben ist jedoch auch die Verfügbarkeit von Experten zu nennen. Die Mitarbeiter der Versicherung standen neben ihren Aufgaben im Tagesgeschäft für Expertengespräche zur Verfügung. Die Anforderungen des Tagesgeschäfts genossen dabei naturgemäß eine übergeordnete Priorität. Auch aus dieser Ressourcenknappheit ergab sich die Notwendigkeit, die Auswahl zu evaluierender Plugins stark einzugrenzen. Die vorgestellte Analyse kann daher keinesfalls als vollständige Marktübersicht aufgefasst werden. Dennoch liefert sie eine praxisrelevante Betrachtung prominenter Lösungen und möglicher Kriterien, diese zu bewerten.

Auf Grund der genannten Limitationen sollte diese Fallstudie als Vorstudie aufgefasst werden. Ihr Mehrwert liegt zum einen in der Identifikation und Priorisierung konkreter Anforderungen an zu untersuchende Plugins bzw. IDEs. Des Weiteren bietet sie eine praxisorientierte Vorauswahl für tiefergehende Evaluationen der empfohlenen Lösungen. Es bietet sich vor diesem Hintergrund an, ressourcenintensivere Anschlussstudien durchzuführen. Im Rahmen dieser sollten konkrete Testfälle der Versicherung mindestens für das am besten bewertete Plugin (idealerweise für alle aufgezeigten Plugins) durchgeführt werden. So kann eine Validität der ausgesprochenen Konfigurationsempfehlung auch vor dem Hintergrund infrastruktureller Spezifika gewährleistet werden. Zudem erscheint es sinnvoll, eine Installation des Plugins auf einer Testinstanz der Versicherung durchzuführen, um die Interoperabilität mit bereits bestehenden Eclipse-Installationen sicherzustellen.

4.2 Zusammenfassung und Ausblick

In den vorangegangenen Kapiteln wurde im Rahmen einer Fallstudie evaluiert, welche COBOL-Plugins für Eclipse existieren und welches dieser Plugins am ehesten für den Auftraggeber der Studie, die Versicherung, geeignet ist.

Dabei erfolgte eine kurze Erläuterung der theoretischen Grundlagen sowie eine Feststellung der Ist-Situation. Im Rahmen diverser Interviews mit der Versicherung wurde der Soll-Zustand konkretisiert. Aus diesem war es möglich, Anforderungen an die zu evaluierenden Plugins abzuleiten. Vor einer Analyse erfolgte zudem eine Priorisierung der einzelnen Anforderungen, sodass final eine Gegenüberstellung der diversen Applikationen und das Aussprechen einer Empfehlung, zugeschnitten auf die Versicherung, möglich ist. Mit Hilfe eines festgelegten Bewertungsschemas wurden Plugins evaluiert. Festzuhalten ist hierbei, dass deren Auswahl auf Basis von Verfügbarkeit und einer Vorsichtung des Marktes erfolgte. Aufgrund der geringen Verfügbarkeit von Open Source Applikationen musste von der ursprünglichen Zielsetzung, primär nicht-kommerzielle Plugins zu analysieren, abgewichen werden. Stattdessen befindet sich unter allen bewerteten Lösungen ausschließlich eine, welche unter einer OSS-Lizenz steht. Im Rahmen des Forschungsauftrags ist diese Anpassung an die Praxis jedoch als vollständig akzeptabel einzustufen.

Die Evaluation der Plugins ergibt, dass eine zentrale Herausforderung vor allem die Gastsprachenunterstützung darstellt. Drei von sechs analysierten Lösungen bieten diese Funktionalität nicht. Eine ausschließliche Unterstützung im nativen COBOL-Sprachbereich wird von zwei Plugins geboten. Hinsichtlich Editor-Funktionalitäten ist durchweg eine breite Unterstützung gewährleistet, sodass hier keine besondere Differenzierung zwischen den Applikationen erfolgen kann. Eine Erwähnung verdient jedoch das Plugin „Cobos“, welches als einzige Lösung eine Terminal-Emulation direkt in Eclipse bietet. Problematisch stellt sich hier jedoch die Unternehmensgröße sowie der Firmensitz im Ausland dar. Vor dem Hintergrund, dass das Dienstleistungsangebot als signifikanter Faktor eingestuft wurde, sollte vor allem bei kleinen Anbietern mit wenigen Mitarbeitern darauf geachtet werden, einen deutschen Dienstleister zu beauftragen, um Sprachbarrieren zu vermeiden. Zudem kann (nicht zwingenderweise muss) möglicherweise ein kleinerer Leistungsumfang erwartet werden.

Auf die Anforderungen der Versicherung ist das Plugin „Visual COBOL“ am ehesten zugeschnitten. Unter Berücksichtigung der Priorisierung erhält diese Lösung im Evaluationsschema die höchste Bewertung (exklusive RDz). Grundlage hierfür sind zum einen die Service-Dienstleistungen, die neben dem Produkt angeboten werden. Zum anderen bestehen funktionale Vorteile, welche die Applikation von anderen abheben. So bietet „Visual COBOL“ neben dem „IBM RDz“ die umfassendsten und komfortabelsten Editor-Funktionen. Zudem kann eine Vielzahl externer Systeme, wie z.B. Datenbanken, angebunden werden.

Wichtig war es im Rahmen dieser Arbeit weiterhin, die ausgesprochene Empfehlung der Referenz RDz gegenüberzustellen. Hintergrund dessen ist die bereits vor einigen Jahren durchgeführte Machbarkeitsstudie zwischen der Versicherung und IBM. Aus dieser resultierte, dass der RDz primär aufgrund finanzieller Gründe nicht eingeführt werden konnte – fachlich jedoch konnte die Lösung alle Anforderungen erfüllen.

Die Empfehlung, das Plugin „Visual Cobol“ in der Versicherung einzusetzen, kann ausschließlich auf Basis der durchgeführten Analyse, nicht jedoch mit vollständiger Sicherheit ausgesprochen werden. Im vorangegangenen Kapitel wurden Restriktionen aufgezeigt, welche dazu führen, dass diese Arbeit nur als Vorstudie angesehen werden sollte. Dies führt dazu, dass Anschlussstudien zu empfehlen sind, welche die beschriebenen Faktoren aufgreifen. In dieser nachfolgenden Arbeit sollten sowohl eine konkrete Kosten-Nutzen-Analyse durchgeführt werden, als auch im Netzwerk der Versicherung Testinstanzen aufgebaut werden, die eine konkrete Evaluierung der Anforderungen ermöglichen. In enger Zusammenarbeit mit den Mitarbeitern der Versicherung sowie ggf. der Berater des Unternehmens, welches das Plugin vertreibt, können die erforderlichen Funktionalitäten weiter konkretisiert und geprüft werden. Zudem ist es denkbar, den Anforderungskatalog und damit den Betrachtungshorizont zu erweitern. Vor dem Hintergrund der Nutzung der modular aufgebauten Eclipse-Plattform und des dazugehörigen Ökosystems besteht das Potenzial, Synergieeffekte zu erzielen. So erscheint es lohnenswert, die COBOL-Entwicklung auch auf eine Kompatibilität mit Eclipse-Features wie z.B. Plugins zur Automatisierung von Entwicklertests zu analysieren. Weiterhin sollte die Interoperabilität mit der internen Java-Entwicklung validiert werden. Letztlich kann hierdurch eine größtmögliche Nutzung der Eclipse-Plattform realisiert werden, wodurch die Produktivität der Entwickler drastisch gesteigert wird.

Evaluation of Open Source Tools for Test Management and Test Automation

Open Source Seminar Paper

Submitted on 05.07.2012

School of Business

International Business Information Management

Course WWI2009I

by

Deborah Fleischer, Jennifer Schwerdtfeger, Patricia Capaul, Verena Thiel

Table of Contents

List of abbreviations

ALM	Application Lifecycle Management
CPM	Critical Path Method
CSS	Closed Source Software
GPL	General Public License
HP	Hewlett-Packard GmbH
IBM	International Business Machines Deutschland GmbH
IDE	Integrated Development Environment
JDK	Java Development Kit
KPI	Key Performance Indicator
LDAP	Lightweight Directory Access Protocol
MPL	Mozilla Public License
OSS	Open Source Software
RTH	Requirement and Testing Hub
SUT	System Under Test
VB	Visual Basic
VNC	Virtual Network Connection

List of Figures

List of Tables

1. Introduction

1.1 Problem Definition

The development of software has been one of the most complex artifacts[1] since systems have been continuously increasing due to new technologies and complex graphical user interfaces. Before releasing new products they need to be tested and their proper function needs to be confirmed.[2] A survey conducted by Electric cloud in partnership with Osterman Research in 2010 showed that the majority of software bugs are attributed to poor testing procedures. Additionally, the software test process is generally regarded as unpleasant by software development professionals. Moreover, the survey found that completely automated testing environments are still rare, with just 12% of software development organizations using fully automated systems, while about 10% reported that all testing was done manually. 46% of software developers complained that they do not have as much time to test as they should, which also becomes apparent in another finding, where more than a third of the developers do not believe their companies perform enough pre-release testing.[3] These tremendous results illustrate the importance of testing in the software division to enable project success, and the challenges faced by companies in their software testing process.[4] Additionally, the great dependency on technology even increases the foundational need of testing to avoid long-term costs due to error correction, which will consume even more resources.

While reducing costs remains a primary priority on the business agenda of many companies, these have started to consider the adoption of Open Source Software (OSS) instead of so-called closed source software (CSS) provided by various software vendors. Businesses are also increasingly seeing it as a way to gain a competitive advantage, according to the results of a Gartner survey released on February 8, 2011. "Gaining a competitive advantage has emerged as a significant reason for adopting an OSS solution, suggesting that users are beginning to look at OSS differently — if they can customize the code to make it unique to their company, they have created a competitive advantage," said Laurie Wurster, research director at Gartner.[5] Nearly 46% of the 547 surveyed companies have already deployed open source applications, and 22% even use open source software consistently across all departments. 25% of the re-

[1] Cf. Ludewig, L. J. (2010), p.37
[2] Cf. Sommerville, I. (2007), p.30
[3] Cf. Electric Cloud (2010)
[4] Cf. Wieczorek, M./Meyerhoff, D. (2001), p.6
[5] Gartner (2011)

spondents stated that open source adoption was part of a strategic business decision, while 31% mentioned it as part of the company's IT strategy. Other motivations included the maturity of applications and total cost of ownership, according to the report.[6]

As a result of the above-mentioned findings regarding test management software as well as the role of open source software in today's organizations, this paper focuses on OSS for test management and automated testing. The intention is to take a closer look on leading OSS available on the market and to evaluate these based on a pre-defined set of criteria to establish a final ranking. Furthermore, a comparison with CSS will be made with solutions provided by the leading software vendors IBM and HP to get an impression of the different features commercial solutions provide and where discrepancies may be found between OSS and CSS.

All in all, this guideline is developed to support *a* company in their assessment whether OSS will meet the desired requirements or if only traditional CSS can sat- isfy the requirements. Therefore, the developed criteria catalogue evaluates critical success factors and dependencies, which are essential to choosing the suitable testing solution.

1.2 Structure

The present paper is structured as follows. In Chapter 2, the authors intend to lay necessary foundations for the study. This includes the definition and delimitation of test management and test automation. The identified differences justify a separate examination of the two within the third chapter, which on the one hand determines specific characteristics regarding test management and test automation and on the other hand describes the tool selection process in both areas. Furthermore, chapter 3 presents the criteria, which make up the evaluation matrix and therefore serve as basis for the tool assessment. In chapter 4, the results of the evaluation process of the selected test management and test automation tools will be introduced. The outcome is utilized to compare and evaluate the tools with the intention to create a final ranking. Moreover, the company will receive an individualized and customized recommendation for implementing test management and test automation software in chapter 5. Finally, the conclusion will provide a summary on the presented findings.

[6] Cf. Gartner Inc. (2011)

2. Theoretical Principles

2.1 Open Source Software

The traditionally applied software within a company is known as proprietary software[7], whose license was purchased from a software vendor. Due to the inaccessible source code of this commercialization, it is called Closed Source Software.

In contrast, the source code of OSS is public and can be modified to fit the company's need. As the improvement progress of this software type is openly accessible and publicly documented, it promotes cost efficiency, since upgrades remain free.[8] Thus, low expenses mainly influence the popularity of OSS. The development of OSS and its progress is driven by communities, which includes private people as well as enterprises. The motivation for improvement varies between simply gaining experience and sharing information, staying up-to-date and having inside knowledge of modern technology, or promoting OSS financially and logistically to increase interest and progress. Due to the development team's dependence on the financial, logistical and technical requirements of sponsors as well as the process of OSS implementation into a company's business, licensing is a controversial issue.[9] Therefore, the improvement of increasing professionalism and quality is supported from two different perspectives. The community in the background provides OSS as an alternative to proprietary enterprise solutions. Consequently, the integration needs to be carefully analyzed by the strength of supporters as the progress critically depends on the cooperation between private households and public companies. Commonly known examples of Open Source Software are Android, supported by Google[10], and Firefox, sponsored by Mozilla, as opposed to the Internet Explorer, funded by Microsoft.

2.2 Definition of testing

Before stepping into the process of examining test management and test automation, it is necessary to define the term testing. While several sources focus on "establishing confidence that a program of system does what it is supposed to"[11], this paper refers to the increase in value of

[7] Lanier, J. (2007)
[8] Soto, M./ Ciolkowski, M. (2009)
[9] BITKOM (w.y.)
[10] Android.com (w.y.)
[11] Cf. Hetzel, B. (1993), p.110

the program. This means that the reliability can be increased if defects are found and removed. Therefore, software should not be tested to show that the functionalities are working without defects, but act on the assumptions that defects are included in the code and need to be found. According to ISO/IEC Standards from 1991, a test is "a technical operation that consists of the determination of one of more characteristics of a given product, process, or service according to a specified procedure"[12].

Additionally to the ISO/IEC Standards, the IEE Standard 829 from 1998 defines testing as "the process of analyzing a software item to detect the differences between existing and required conditions and to evaluate the features of the software item".[13] IEEE Standard 610.12 from 1990 describes testing as "the process of operating a system or component under specified conditions, observing or recording the results, and making an evaluation of some aspect of the system or component".[14]

Klaus Franz summarizes the testing process from a software development view. He refers to all test activities that have the aim to fully and correctly realize the requirements for the software product and prove the accomplishment of the quality requirements. [15] Accordingly, testing and reviews are static processes designed to detect existing differences and therefore ensuring proper functionality. So testing consists of static verification as well as dynamic validation. Furthermore, testing includes activities such as planning, controlling, preparation and measurement. This summarizes all attributes of testing in a comprehensive definition and will be used henceforth.

2.3 Test Management

All tests that are executed within a company need to be carefully planned, organized, and scheduled in order to prevent that the tested applications interfere with running systems. The action of creating an efficient plan is called test management. Thus, all resources that are involved in the testing area need to be analyzed. Therefore, a system under test (SUT) requires

[12] Cf. Prins, A. (2009)
[13] Cf. Moore, W. J. (2002), p.67
[14] Cf. Copeland, L. (2004), p.53
[15] Cf. Franz, K. (2007), p.24

specifications, which are defined as requirement test matrix. The major tasks of test management can be associated with different phases within the product lifecycle management:[16]

- planning
- authoring
- executing
- reporting

The implementation of these four phases is further specified at IBM: “Test management is where the goals, the metrics used to measure such goals, and how the data for them will be collected are defined.”[17]

Additionally, test management includes previous tests by linking its results to future trials. This aspect refers to the sequential relationships, especially within the same testing family. If an environment includes two subcategories A and B, where B inherits specific characteristics of A and A fails, it is useless to test B independent from A as parallels can provide an easier investigation of errors and positively impact parameters. The process of managing tests includes methodologies and strategies of project management like the critical path method (CPM). The results of all tests are stored within one common system that delivers reports and metrics to evaluate the quality of SUT. Critical success factors will be analyzed and improved before the system’s release.

The main advantages of test management are limited to the following aspects[18]:

- Test documentation provides an overview and avoids redundancy.
- Test management increases the efficiency of testing, which saves time and costs.
- Test management enhances effectiveness, as less-improved characteristics will be easily pointed out.

In contrast, test management also causes several problems:

- It is very time-consuming, as each testing tool requires an individualized treatment.
- Test management is very complex and uses lots of human resources[19].
- Sometimes the validation of requirements is disregarded.

Instead of coordinating tests manually, several steps can be automated to further increase efficiency and effectiveness as described in the following section.

[16] Cf. IBM (2011), p. 1
[17] IBM (2011), p. 1
[18] IBM Deutschland GmbH (w.y.)
[19] Testmanagement.com (w.y.)

2.4 Test Automation

Test management can be applied on manual as well as automated testing tools. This part will outline the advantages and disadvantages of automatizing test management.

Test automation technologies have not reached every industry yet, due to missing standards and best practices. As test cases become more and more complex and the time pressure increases, companies start focusing on these technologies. The intention of test automation is to establish high test coverage, decrease costs and reducing errors by using a testing tool:[20] Therefore, an automated test is able to provide the following advantages when applied properly:

- Reduction of testing effort
- Cost reduction
- Discharge of testing personnel through automated regression tests[21]
- High test coverage and optimized software quality[22]
- Enhanced performance
- Increased efficiency
- Shortened test duration
- Reusable test sets
- Continuous testing throughout the whole development process, not just in the end

However, test automation involves several risks:

- If the test activities have not been planned, costs may increase
- Initial expenditure is high
- Maintenance costs of the testing scripts can increase
- Unrealistic expectations regarding the tool
- Technical problems within the testing software[23]

The decision, if software should be tested automatically, depends on several economic aspects. Before the actual project starts it needs to be evaluated if automated testing significantly increases the efficiency. The following table examples decision based criteria:

[20] Cf. Bommer, C./Spindler, M./Barr, V. (2008), p. 258
[21] Cf. Dustin, E./Rashka, J./Paul, J. (2001), p. 57
[22] Cf. Strahringer, S. (2011), p. 88
[23] Cf. Menzel, M. (2006), p.66

	Automatic testing	Manual testing
Initial investment	- Depending on the software tool it is higher than for manual testing, especially when regarding familiarization	+ No investment for further software tools
Effort of test preparation	- Approximately twice as high, as the tests need to be configured and are often more complex[24]	+ only the determination of test cases is required
Effort during execution	+ Less effort as the test runs automatically, even during non-working hours	- The execution and documentation is very time- consuming
Analysis of test results	+ The analysis and reporting is less time consuming as most tools create reports themselves	- Very time consuming
Maintenance	- The maintenance costs for the software and test scripts are higher[25]	+ Changes can be adapted by the tester
Regression tests	The investment in an automated tool is only efficient, if regression tests need to be made and very complex test cases need to be run[26]	An advantage when testing small applications or running tests that only need to be repeated once

Table 1: Automatic and manual testing

The decision has to be made by every company individually regarding internal as well as external influences. Generally, testing tools can increase efficiency and effectiveness of a company.[27]

[24] Cf. Grechenig, T. u.a. (2010), p. 332
[25] Cf. Grechenig, T. u.a. (2010), p. 332
[26] Cf. Bath, G./ McKay, J. (2010), p. 343
[27] Cf. Strahringer, S. (2011), p. 92

3. Methodology

3.1 Tool selection

Open source test management and test automation tools are in great demand due to the exposable source code as well as publicly accessible documentations. However, test management or automation tools or frameworks cannot cover all projects because of the individual nature of each project regarding test management or automation. Requirements concerning software functionality therefore have to be defined unambiguously in order to realize the defined aim of the project.

An extensive literature and internet research identified 122 open source test automation tools[28] and 26 open source test management tools.[29] A comprehensive evaluation of all existing test management and test automation tools is not possible within the context of this elaboration due to the limited quantitative as well as temporal extent. Therefore, the emphasis is put on the four leading Open Source Software tools regarding test management and test automation for the purpose of an extensive software analysis.

Although Open Source Software tools will be the primary topic of this paper, leading commercial tools will also be evaluated in order to compare them to open source solutions.

Market leadership is usually defined by largest market share per provider or highest profitability margin.[30] According to literature, market share represents merely “the proportion of total market or industry sales made by one of the competing companies”[31] and may be measured either on value basis by multiplying sales price per volume or on a unit basis, considering amount of units shipped.[32] Nevertheless, the aforementioned definition is not applicable when considering the expense less character of Open Source Software.

Market leadership in context of Open Source Software should be therefore defined by means of obtaining user acceptance as well as popularity, whereby user acceptance refers to already tried and trusted functionality of Open Source Software, while popularity implies its usage in

[28] Cf. Aberdour, M. (2012a)
[29] Cf. Aberdour, M. (2012b)
[30] Cf, Simon, H. (2007), p.49
[31] Xu, J. (2005), p.69
[32] Cf. Rouse, M. (2007)

various large projects and companies. A large community providing expertise to users and continue developing the Open Source Software steadily also indicates popularity and reliability.

The four most popular test management and test automation systems regarding Open Source Software are identified below based on an exhaustive literature research.

Test Management tools

QA Test Lab carried out a study regarding commercial and open source test management **tools** by surveying 120 software testing and quality assurance professionals. TestLink was identified as most popular open source test management software developed with focus on usability, maintainability and effectiveness[33] by 53% of the interviewees.

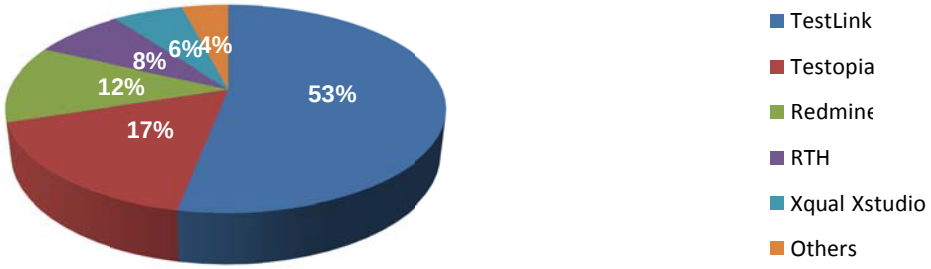


Figure 1: Popularity of Open Source Test Management Software[34]

Testopia represents the next popular open source test management tool with 17% of popularity, followed closely by Redmine with 12%. Testopia is a test case management extension for the Bugzilla bug tracking and testing tool. Testopia Bugzilla provides a complete suite of features of test and defect management. Bugzilla is used by more than 1268 international companies, organizations, and projects which reflect its popularity as well as maturity.[35] The Requirements and Testing Hub (RTH) tool should also be taken under consideration with a popularity of 8%.

Test Automation tools

Appropriate test automation tools are identified by Tools Journal, including the open source tools Selenium and WatiN along with a number of commercial test automation products.[36] Selenium as well as WatiN represent test automation tools for browser-based testing of web applica-

[33] Cf. Jhureley, A.(2010)
[34] Contained in: QA TestLab (2009), S.6
[35] Cf. Bugzilla (2012a)
[36] Cf. TestJournal (2011)

tions. Due to various surveys as well as evaluations, the Open Source Software T-Plan Robot and Canoo Web Tests complement the four most widely used test automation systems.[37]

The open source tools regarding test management and test automation selected in order to consider and evaluate in detail are as followed:

Test Management Tools:

1. TestLink
2. Testopia Bugzilla
3. Redmine
4. Requirement and Testing Hub

Test Automation Tools:

1. Selenium
2. WatiN
3. TPlan Robot
4. Canoo WebTest

Closed Software Test Management and Test Automation Tools

QA TestLab also considered the popularity of commercial systems for test management, test automation and bug tracking. According to that, Atlassian´s JIRA software dominates with a popularity of 33% of popularity closely followed by HP Quality Center with 31%. IBM Rational is the succeeding software for test management and automation representing 12% of popularity.[38]

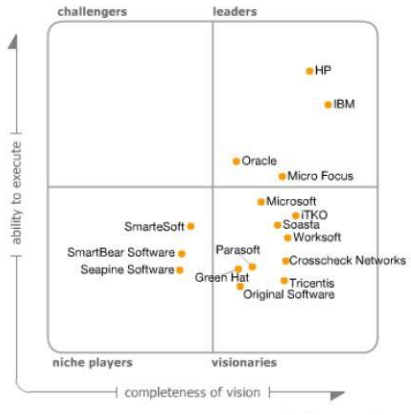


Figure 2: Magic Quadrant for Integrated Software Quality Suites

A study regarding Integrated Software Quality Suites carried out by Gartner Inc. confirms the leading position of the multinational technology and consulting corporations Hewlett-Packard and IBM within the context of commercial test management and test automation systems.[39] HP

[37] Cf. Opensourcetesting.org (2012); Whichtestingtool.com (2012)
[38] Contained in:QA TestLab (2009), S. 8
[39] Cf. Gartner Inc. (2011), p.8

and IBM as well as Oracle and Micro Focus are identified at market leader according to Gartner Inc. Vendors like Microsoft and Soasta are declared as visionaries, whereas Seapine Software and SmarteSoft are classified as niche players.

The closed source tools incorporating test management and test automation components evaluated within this paper as follows:

1. HP Quality Center
2. IBM Rational Quality Manager

3.1.1 Test Management

The following paragraph describes four test management tools, which were carved out in chapter 3.1 based on literary research.

Bugzilla Testopia

Bugzilla is a bug tracker end-user solution documenting bug reports. This web application is supported by Mozilla. [40] Additionally, such OSS is compatible[41] with Linux Kernel, Gnome, KDE, Apache Projects, Open Office and Eclipse. It can also be used with Linux Distributions. All in all, Bugzilla offers a wide range of compatibility, which is also recognized and actively used by 1268 companies, organizations and projects. The most famous companies and organizations are NASA, Facebook, Wikipedia, Nokia and The New York Times.

Bugzilla has turned into a group undertaking. It was first developed for internal use only, but spread out very soon. Therefore, it has changed from being a mozilla.org implementation into a general bug tracking system with to a strong community. The continuous improvement is strongly driven by anonymous sponsoring and donations of Mozilla. Furthermore, it is known for its light-weighted implementation leading to increased speed and efficiency. Thus, several bug system projects can be tracked simultaneously to define progress used for project's comparison. The status can be viewed by sending pre-defined queries to a data base as depicted in Figure 3. Selected criteria will be modified for receiving a customized view.

[40] Bugzilla (2012a)
[41] Bugzilla (2012a)

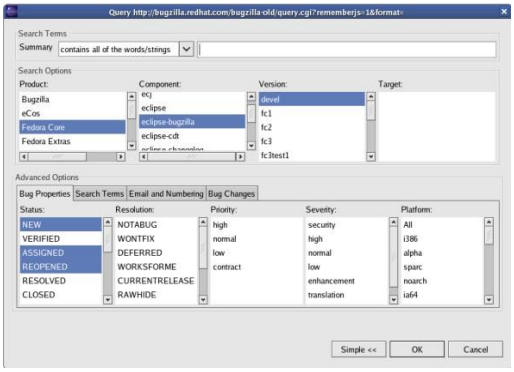


Figure 3: Query interface allows user to track results[42]

In addition, Bugzilla can be extended by Testopia, which is designed to organize and track test cases. As a result, the implementation supports Bugzilla from a project management perspective as it offers detailed bug reporting and documentation, which can be used for further test cases.

In summary, Bugzilla Testopia offers great functionalities to ensure quality and maintain effectiveness and efficiency of software. Further tool specifications can be easily implemented by focusing on a lightweight system. On the one hand, redundancy within the database is avoided. On the other hand, several bug tracking entries are used for comparison of debugging and further development. Therefore, the interaction of involved systems and applications is regarded closely.

Redmine

Redmine is a project management tool that focuses on planning testing projects. It provides features such as access control and issue tracking systems, which are required for further analysis. Moreover, Gantt charts and calendars[43] provide a great overview of scheduled events; notifications can be implemented into wikis, forums as well as into feeds and an email system. For example, it automatically creates issue tracking via emails. The current status of tickets can be tracked by an integrated progress feature. Thus, Redmine offers a great graphical interface to illustrate critical topics, such as open project tasks split into individual assigned workloads while representing coordination and responsible job roles for each specific topic. In addition, forums, wikis and documentation interfaces promote communication to ensure a project's suc-

[42] Bugzilla (2012a)
[43] Redmine (2012)

cess. Figure 4 represents different possibilities to state the project's success depending on time spent on the project as well as approximated time left. Grouping and selecting criteria for a more detailed analysis is demonstrated below.

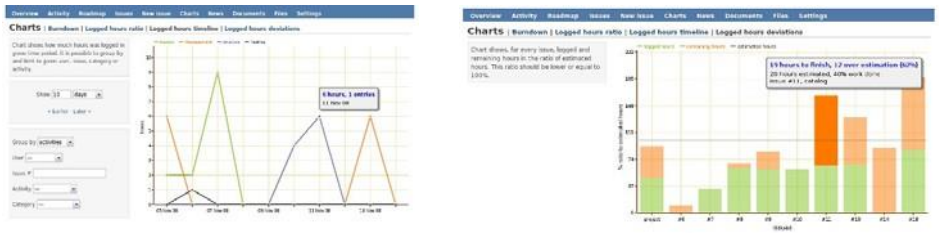


Figure 4: Redmine – Graphical User Interface for project tracking[44]

Moreover, SCM tools such as SVN, CVS, Git, Mercurial, Bazaar and Darcs can be integrated into Redmine. The community has established the compatibility of additional third party tools within the past years. However, its support differs significantly from Redmine.

All in all, Redmine is a multi-client project management tool used for planning and organizing issue-tracking systems. Discussion forums ensure a well-coordinative administration system to define accesses and roles. Furthermore, customer language can be chosen to secure a better communication and include news and messaging systems.

Requirement and Testing Hub

Requirements and Testing Hub (RTH) is an open source test-management tool with integrated requirement management and bug tracking capabilities. RTH can be integrated into projects supporting multiple operating systems (Windows, Linux), PHP programming language, SQL databases as well as Apache HTTP Server according to standards and providing an API connection to integrate other software.

RTH provides a CKEditor component, an open source HTML and text editor from CKSource, in order to facilitate the creation of test cases. Already created test cases can be easily imported using a Microsoft xls file. The ability to create, manipulate and delete test requirements is given by RTH. Moreover, a traceability matrix covers all requirements related through test case id to a coverage statistic.

[44] Redmine (2012)

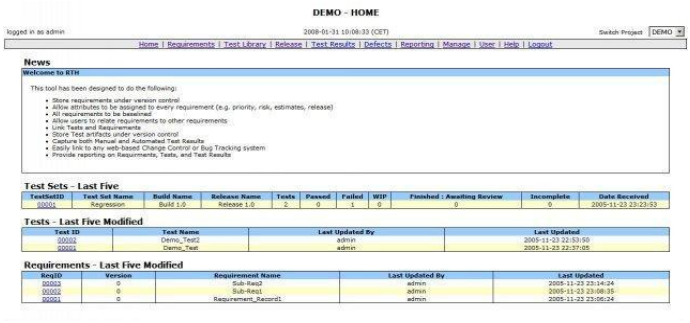


Figure 5: RTH main page

RTH presents a light weight and easy to use test case management tool, which supports different user groups with defined user rights addressing the user right administration. However, proper documentation and support activities cannot provided sufficiently due to a small team of developers.

TestLink

TestLink is the most popular open source test management and execution system including test specification, planning, reporting and requirements tracking. The web-based, multilingual user interface of TestLink runs on browsers supporting JavaScript, XHTML and CSS, facilitating maintenance and upgrades of TestLink because client installations are no longer required.

TestLink is platform independent test management software supporting multiple programming languages (JavaScript, PHP) and database systems (MySQL, PostgreSQL, MS-SQL). Moreover, it is developed with emphasis on usability, maintainability as well as effectiveness and can be integrated with the most common bug tracking systems such as Bugzilla and Mantis. Internal and external authentication support is provided by Lightweight Directory Access Protocol (LDAP) including flexible role based access control with multiple permission levels. Due to a centralized repository for all test cases and results, it facilitates furthermore the maintenance of multiple projects.

Moreover, it provides excellent reporting features including the ability to generate reports in various formats like HTML, CSV, MS Word and MS Excel. Reports can be generated based on the results from test case executions e.g. build wise reports and can also be visualized by usage of graphs. TestLink is owned by Teamtest, a strong open community of international testers and an active development team, providing User and Developer Guides as well as Release notes.

Figure 6: TestLink test specification and test report interface

To summarize, TestLink presents a robust and flexible framework for managing testing processes and ensures compliance with standard testing processes as specified by IEEE 829 or BCS SIGIST.

3.1.2 Test Automation

In the following, the four test automation tools, which were pre-defined are explained. Special characteristics are outlined and used for further evaluation.

Canoo WebTest

Canoo WebTest is free Java open source tool for automated testing of web applications and helps its users to reduce the defect rate of their web application. The downside of the tool being free of charge is that there is no guaranteed support available. However, Canoo can be contacted for special support incidents, a support contract or on-site help for introducing automated testing into the user's project. Canoo WebTest has been used successfully in thousands of organizations ranging from small internet startup companies up to global players, for intranet and internet sites, for portals and B2B applications.[45]

Canoo WebTest offers many advantages to its users. For instance, it has an easy syntax, which can be understood even when the user is not familiar with WebTest. The recorder allows for a quick generation of a first draft of the tests, which can then be adapted and refactored to build robust test suites. Canoo WebTest is fast in executing its test cases, as it does not download CSS and does not need to compute page rendering. Furthermore, Canoo WebTest provides excellent reporting with all information required to understand the failure cause. Another advantage is that it is plain Java and therefore runs everywhere as long as a Java Develoopment Kit (JDK) is installed. Very appealing for the user is also that the tool can be easily extended

[45] Cf. Canoo Engineering AG (2012a)

and adapted to custom needs. Canoo WebTest also offers a straightforward integration. WebTest scripts are ANT scripts, which allow for an easy integration. Moreover, it is possible to test web applications written in various programming languages such as PHP, ASP, .NET, etc. Canoo WebTest executes client side JavaScript almost like a regular browser does. For this purpose, it uses HtmlUnit, which can be seen as a faceless browser that is able to mime the behavior of e.g. Internet Explorer or Firefox. This also represents a disadvantage of WebTest as it only simulates Firefox' or Internet Explorer's way to execute JavaScript, which implies that there may be discrepancies between the simulated behavior and the actual behavior in a real browser.[46] Other features of WebTest are the possibility to be integrated into Maven as well as the exemplary reporting it provides.

Selenium

Selenium is the most renowned and versatile Open Source web automation tool suite that supports agile development of test automation for web-based applications across many platforms. It is a package of various test components, which consists of the following three major tools:

1. Selenium IDE – a Firefox add-on that will do simple record-and-playback of interactions with the browser. This is used to record and playback actions taken in the browser.
2. Selenium RC – Used to run tests on different browsers and systems. It furthermore supports replay of Selenium scripts through browsers
3. Selenium Grid – Automated replay of Selenium scripts on many browsers simultaneously, also known as mirrored replay.
4. Qualitia and Tellurium – A wrapper for the Selenium engine.

Each one has a specific role in aiding the development of test automation for a Web application.[47] It can also be easily integrated into various projects, supporting multiple programming languages such as .NET, Perl, Python, Ruby, and the Java and is running on all platforms (Windows, Mac, Linux, Solaris and others) and Browsers (Firefox 2/3, IE 7/8, Safari 2/3, Opera 8/9). Additionally Selenium can be integrated with the most common Testing Frameworks (JUnit, Nunit, TestNG, Bromine,Test::Unit, etc.). Selenium supports capture and replay very well, so test cases can be created easy and fast. Features that are missing in Selenium are a sufficient

[46] Cf. Canoo Engineering AG (2012b)
[47] SeleniumHQ (2012)

reporting function and a native keyword and data driven approach, but for these missing features, further tools have been developed, e.g. Fitnesse or SeleniumCamp. The use of the product grew within Thoughtworks, and the company decided to release it as open source software to the wider community. In 2008 the Selenium project was merged with WebDriver to allow for server-side replay of the Selenium scripts, and since then Selenium GRID has been released to support large scale parallel testing. Google adopted it and many of the Thoughtworks engineers moved to work there to continue development of the tool.

Moreover there are many third party vendors that provide commercial support, an increasing knowledge base (Selenium-HQ) and an active Google group.

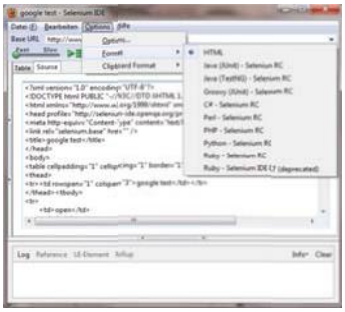


Figure 7: Graphical User Interface for Selenium IDE

T-Plan Robot

T-Plan Robot is a flexible and universal black box automated testing tool. It offers a new approach for black box testing due to its almost platform independent connection via Virtual Network Connection (VNC) and the image-based automation. Developed on generic image based testing principles, T-Plan robot provides a human-like approach to software testing and performs well in situations others may fail. As VNC servers also support most significant systems including some embedded ones (such as Windows Mobile devices), the tool can be effectively used to automate any desktop GUI application on any common system.[48]

T-Plan Robot is available in two forms:

1. T-Plan Robot Enterprise - a commercial and actively developed product with support and customization services

[48] Cf. T-Plan (2012a)

2. T-Plan Robot Open Source – a legacy open source version with limited functionality based on the initial 2.0 release with no stable support guaranteed and little development activities[49]

T-Plan Robot Open Source is available under GNU General Public License (GPL). According to the T-Plan Robot Website, the project is not backed up by any professional support and only maintained in terms of major bug fixing with no features being developed. The website also provides an overview on the differences between the commercial T-Plan Robot Enterprise and the open source version for enhanced decision-making when considering T-Plan Robot as Test Automation software.[50] Major advantages of T-Plan Robot include:

- Platform independence (Java)
- Wide range of testable applications
- Support of Java test scripts as well as proprietary scripting language
- Record & Replay capability
- Support of testing over the RFB protocol (better known as VNC)
- Ability to perform black box GUI testing of Windows Mobile applications directly on mobile phones[51]

As T-Plan Robot is based on image based testing, it inherits certain advantages that come with that approach. It follows a true end user approach, as the tool navigates through the GUI the same way as the user does and verifies the test results visually seeing the same image as the user. Furthermore, it has complete technology independence. Due to automation inputs and outputs being produced on open source (OS) level, the application technology becomes irrelevant and a single testing tool can automate every GUI application or a set of application based in various technologies. Another advantage is the short learning curve even for engineers with little programming experience and/or no knowledge of the underlying application platform. Beside the advantages, this testing concept also comes with certain disadvantages, such as limited verification means, as image comparison does not always suffice to verify that the application works well. Moreover, a very stable environment is required as just a mere change of desktop background color may result in an image search failure.[52]
The following illustration shows the graphical user interface of T-Plan Robot:

[49] Cf. T-Plan (2012b)
[50] Cf. T-Plan (2012c)
[51] Cf. T-Plan (2012a)
[52] Cf. T-Plan (2012d)

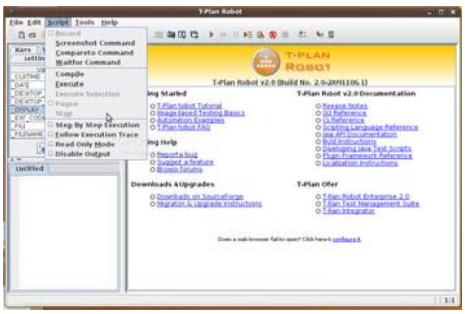


Figure 8: GUI of T-Plan Robot[53]

WatiN

WatiN provides an easy way to automate Web Application Testing in .Net. Inspired by Watir, WatiN has grown into stable framework offering many features as well as automation of Internet Explorer and Firefox.[54]

The tool puts emphasis on simplicity, flexibility and maintenance when writing test cases. It is able to do basic capture replay functions and also checks the expected results on a page. It supports web applications that are developed in any language and a wide range of browsers, but it is very limited in terms of platform independency, as it is very dependent on Microsoft Windows and .Net.

The community has grown significantly and offers an FAQ site and a wiki, but no commercial support is yet offered, as the developer himself is the counterpart regarding questions.

```
[Test]
public void SearchForWatiNOnGoogle()
{
  using (var browser = new IE("http://www.google.com"))
  {
    browser.TextField(Find.ByName("q")).TypeText("WatiN");
    browser.Button(Find.ByName("btnG")).Click();

    Assert.IsTrue(browser.ContainsText("WatiN"));
  }
}
```

Figure 9: Hello world example of web Test Automation in WatiN: searching Google

[53] Contained in: T-Plan (2012b)
[54] Cf. WatiN (2011)

3.2 Selection of Tools for Closed Source Software

Closed source software provides advantages in terms of all-around packages. For a particular fee a customized support throughout the entire application lifecycle is included as such service suppliers provide specialized expertise.

In the following paragraph, two leading service providers represent CSS for test management.

3.2.1 HP Application Lifecycle Management / HP Quality Center

HP Quality Center comes in two editions: HP Application Lifecycle Management (ALM) and HP Quality Center (QC). HP ALM, as shown in figure 4, is the larger edition. It contains all the HP Quality Center functionalities plus integration into the development life cycle "HP Application Lifecycle Intelligence" and cross-project data sharing and reporting capabilities. HP QC and HP ALM are web-based applications that aim to manage the lifecycle of key applications from its requirements to its implementation. It offers a central platform that manages all the activities regarding requirement coverage and management by the possibility to derive test cases from requirements and business processes, test management and control, test performance and logging. Furthermore, the platform supports the company by developing a framework and environment for workflows of the application lifecycle at a central location and dashboards that can reflect the current state of the test. This enables the company to identify bottlenecks at an early stage.[55]

HP QC & ALM provide real time visibility of requirement coverage and thus the visibility of all defects that are connected with it. It also supports collaboration during the whole lifecycle and has standardized test and quality processes that can increase the productivity with workflows and automatic notifications.

[55] Cf. Hewlett-Packard (2011), p.1

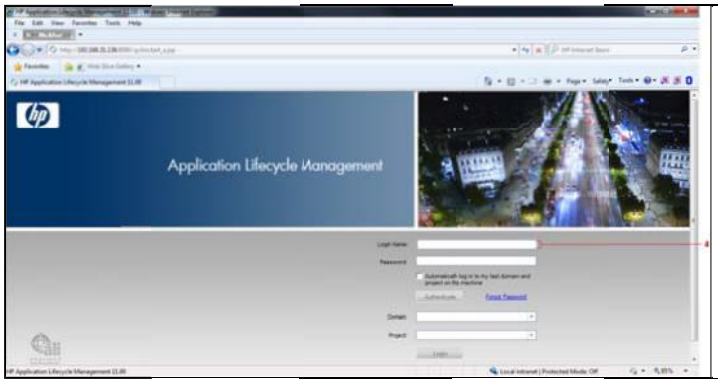


Figure 10: Application Lifecycle Management

The HP Quality Center software can be expanded with additional tools and interfaces to address the needs of large, global enterprises with initiatives that span up to hundreds of applications and geographically distributed teams. Additional tools that can be integrated are for instance HP QuickTest Professional, HP Service Test or HP Sprinter.[56] HP Sprinter is included in HP Quality Center & ALM and supports an automatic and more detailed test run documentation, smart defects that include the test run history and storyboard, parallel execution of manual tests and semi-automatic manual tests, which can be an advantage when combining test automation and test management in one tool.

During the utilization of the application a logical test process is followed. This process includes five independent phases that are shown in the figure below. Project management supports the whole process with reporting and analysis.

Figure 11: Logical test process of HP Quality Center[57]

One of the core functionalities of HP Quality Center is the test planning and execution. Test cases can be stored in a structured way to allow a uniformly classification and description.

The central test case catalogue in HP QC & ALM includes all testing types from functional to performance and security tests and can be executed manually or automated.

[56] Cf. Hewlett-Packard (2011), p.3
[57] Hewlett-Packard, p.3

The benefits of HP Quality Center are therefore various:[58]

- Track and measure project milestones and key performance indicators (KPIs)
- Enable centralized management and enforcement of consistent workflows and processes
- Reduce duplication of effort through asset sharing and reuse
- Gain visibility into entire application portfolio
- Increase communication and collaboration between key stakeholders

3.2.2 IBM Application Lifecycle Management/ Rational Quality Manager

As IBM focuses on an all-around solution, it integrates its test management tools into an application lifecycle. One of the most important tools that ensure a project's quality is IBM's Rational Quality Manager (RQM)[59] with approach of collaborative software. It deals with scheduling and evaluating software's success in terms of the error ratio. Therefore, it is necessary to integrate customer requirements and regard machines' interactions. These conditions will be implemented into test cases.

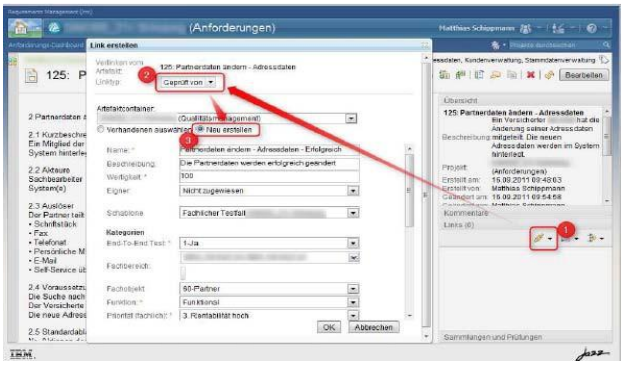


Figure 12: IBM Rational Quality Manager – Test cases based on requirements

RQM offers the possibility to create test cases, which directly depend on requirements as pictured in Figure 12. This test case will be linked to the specific requirements. Accordingly, changes within the requirements are synchronized automatically in the test case or can be integrated manually.

The ability of creating test cases is based on standardization. Each test case is defined by a large amount of independent parameters, which can be filled by a drop-down menu listed in

[58] Cf. Hewlett-Packard, p.3
[59] IBM (2011)

categories and attributes like pictured in Figure 13. The user chooses the most applicable characters for a particular test case. A detailed test description can be added within a test editor.

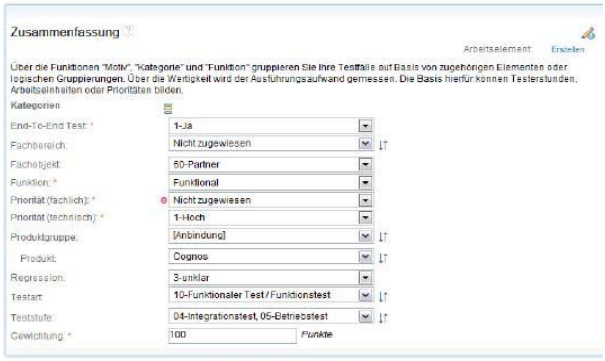


Figure 13: IBM Rational Quality Manager – Categories and attributes

Developed test cases can be multiplied and used for further specialized testing. Modifications of these master templates are documented. Furthermore, test cases and its scripts can be imported into the RQM via xml or Microsoft Excel. Within the RQM it is possible to sort individual tests within a project to determine the test cases' execution order. Moreover, execution date and time can be scheduled manually. Within the execution phase, results are documented for each step and marked as "passed", "failed", "without result" or "blocked". If an error occurred within the test execution phase, a new test case will be generated automatically and referred to existing bugs, to detect the specific bug.

In order to track the entire life cycle of test management, several IBM tools need to work together. Therefore, RQM is known as test management hub, since it is implementable throughout any platform and type of test. Moreover, it has the ability to integrate several testing solutions such as:

- IBM Rational Performance Tester, which ensures application scalability and spots systems' bottlenecks.
- IBM Rational Functional Tester, which automates functional regression testing.
- IBM Rational Service Tester, which provides performance and functional testing of web-based service applications.
- IBM Rational AppScan®, which scans and tests web applications to define vulnerabilities.
- IBM Rational Policy Tester, which compares and implements special requirements.

- IBM Rational Software Analyzer, which drives static analysis.[60]

In addition to the presented extraction of software testing tools, several other IBM applications can be integrated to build on ALM. The wide range of compatibility drives a customized solution for quality and test management with the focus on planning, executing, measuring, reporting and detecting defects. The main vision of IBM is to drive a quality-oriented approach as it includes and ensures code quality, unit testing as well as test data extraction and masking.

All in all, RQM provides a great application for test and quality management[61]. Test cases are created and executed. Its results are compared to existing tests and similarities are pointed out to evaluate its success, which will be included in future plans and schedules to define test cycles. The accomplishments plus administrative details such as the date and duration are documented. Based on these records, the further defect management is scheduled. Simultaneously, resources are considered and used to improve efficiency like within the testing environment. The planning process includes the interaction of several test cases with a focus on the user's requirements, which can be imported from xml data. Test cases can be individually created and customized to match the specific user requirements. Additional features can be implemented as mentioned above.

3.2.3 Comparison of Open Source and Closed Source Software

The biggest difference between open and closed source software considers the fee paid for licenses. While commercial service providers charge the customer for implementation, the Open Source Software can be embedded by well-trained internal staff. Accordingly, no additional charges will occur. However, in-house support requires very detailed knowledge of specific tools and therefore does not imply that the total cost of ownership of Open Source Software is less than the costs of closed source software. Instead of providing a cost-intensive training, the implementation as well as support could be outsourced. This interferes with the idea of closed source software. Thereby, the product, usability, security, innovation, service and support will be bundled as an all-around package and sold by the service provider. Thus, the service provider can provide the best knowledge for this specific area.

[60] Gartner Inc.(2011)
[61] IBM (2011)

All in all, the application of closed source software benefits from an integrated solution, which is modified by a service provider that knows its products in detail. However, such service leads into a vendor lock-in. In contrast to applying external knowledge, internal training is cost-intensive but provides independency and personalized innovation. Finally, the major difference is based on the company's strategy, which encounters the advantages of a commercial or free software solution. By considering the closed software examples in this paper, HP Quality Center and IBM Rational are both tools for test management that can be enhanced with test automation features as well. Furthermore, the support, implementation, further development and training are guaranteed.

3.3 Concept of Evaluation matrix

To evaluate the pre-selected tools an evaluation matrix was designed, consisting of different criteria in three core areas: generally open source criteria, common software criteria and functional criteria regarding test management and test automation tools. The selected tools are reviewed for each of these areas. Thus, at the end of the evaluation of each tool three measures are provided, which allow for better comparison among the different tools. The individual areas are composed of a set of predefined criteria, which were subjected to a weighting. For each feature the tools will be assigned a score between 0 and 5, with 5 being the best score attainable and 0 the worst, i.e. a tool with a rating of 5 meets the requirements satisfactorily, while a tool with a rating of 0 fails to meet them at all. For some criteria it may be the case that the required information were not available. This is reflected in the evaluation with a weighting of zero, while the weighting of the other criteria remains the same. Thus, the total score, which a tool can receive, decreases. The formula used to calculate the overall amount of points a tool may obtain during evaluation is: Amount received = score/5 * % weight. If based on the information available all criteria could be evaluated the maximum amount a tool can get equals to 100.

In the following, the different criteria for the three sections will be introduced along with a brief description to provide a better understanding of the valuation basis for each tool.

3.3.1 Open source criteria

The first section ‘Open source criteria’ takes a look at all those criteria that make up a good Open Source Software. The selected open source criteria to be evaluated have been identified through a detailed literature research and can be named as follows: Licensing & Pricing, Improvement & Release Activity, Community, Maturity and Documentation. Please note at this point, that there may be many more criteria, which could be taken into account. For the purpose of this paper, however, a selection in favor of the most important criteria that fit into the scope was made. The assessment of a criterion as important was based their numerous occurrence in various literature sources.[62]

As the price plays a very important role and is in fact the main reason for a company to choose Open Source Software, **Licensing & Pricing** is the first criterion to be measured. Licenses that accompany the Open Source Software are especially for programmers of significance. Unlike commercial software, open source licenses provide users with access to the code, so that they are able to modify and customize the existing code according to their needs.[63] The most well-known example of an open source license is the GNU GPL, which is also one of the most used licenses.[64] **Improvement & Release Activity** is also of major importance when evaluating Open Source Software. Release activities are a good indicator for the effort a vendor is making to improve its product. Furthermore, it indicates that the application is actually used. The **community** is a vital open source resource. Community activities, such as Wikis, a community site, mailing lists, are a good indicator for the acceptance level of Open Source Software. Of major significance regarding the community of an OSS are its size and the overall frequency of activities. A large community leads to the assumption that the software is widely accepted. An active user group provides feedback on improving the quality of a product, e.g. regarding bugs and user experience, and supports other users, who use the product.[65] This is crucial for Open Source Software as it often lacks the support that usually accompanies the purchase of commercial software. **Maturity** measures the reign of a product. Software, which has existed for a longer time, may be more likely to be maintained and supported in the future, as it has already proven itself in practice, while new software may face the risk of failing to get the desired acceptance level, which may lead to the termination of further development activities. Thus, the criterion is closely related to release activities and the community. The last criterion to be evalu-

[62] Cf. Abiodun, O. M. (2011), p. 6; Van den Berg, K. (2005), p. 11
[63] Cf. Abiodun, O. M. (2011), p. 7
[64] Cf. Weber, S. (2004), p. 48
[65] Cf. Van den Berg, K. (2005), p. 12

ated in this section is **documentation.** Here, a differentiation is made between developers' documentation and users' documentation[66], where the first provides information developers may need for further improvement and adjustment of the code and the latter shall function as a manual that instructs the user on how to use the application. Different information resources such as wikis, forums and newsletters fall into the category user documentation.

The weighting of the criteria is as follows: Licensing & Pricing – 20%, Improvement & Release Activity – 20%, Community – 30%, Maturity – 10%, Documentation – 20%. Community was regarded more important in comparison to the remaining, because a large and stable community, which actively uses the tool and supports further development activities is a crucial factor when deciding in favor of Open Source Software. Maturity, however, was weighted as less important, since an older product is not necessarily guaranteed to be the best product on the market. When evaluating Open Source Software with a longer longevity a closer look should be taken at improvement and release activities, as these are good indicators for the market positioning of the provider and the acceptance level of the software. Only if software is actually used, it is likely to be maintained and improved on a regular basis.

3.3.2 Common Software Criteria

This section includes a selection of criteria, which general test tool criteria. These criteria are applicable to any test tool, and may be used to not only evaluate Open Source Software, but also commercial or proprietary software. Thus, this set of criteria will be applied in the evaluation of the previously selected open source tools as well as the commercial tools by HP and IBM. The criteria are: System requirement, Usability, Interoperability, Support, and Tool customization.[67]

The first criterion to be assessed is **system requirement**. This includes the minimum hardware and software requirement for the system on which the test will run as well as the support environment of the tool, e.g. the supported operating systems, supported browsers and used programming language used. The Integrated Development Environment used is also considered in this criterion, as this is relevant for further development and customization, if the customer decides to engage in any related activity - the more common the IDE (e.g. Eclipse or Netbeans) the better the assessment. In a next step, the **usability** of the test tools is evaluated. Here, the

[66] Cf. Abiodun, O. M. (2011), p. 8
[67] Cf. Abiodun, O. M. (2011), p. 8; Illes, T. et al. (2005), p. 2 et sqq.

ease of installation and the ease of use are assessed. The third criterion is **interoperability**, which describes the level of integration with other tools. This criterion is of major importance and thus, a closer look should be taken at user requirements regarding compatibility and interfaces to be supported. Furthermore, the level of **support** receivable by the user from the tool provider or community is evaluated. Open source support is mainly provided by the community of users and developers. However, when choosing OSS, receiving personal support may lead to additional expenses, which need to be considered from the very beginning. The last criterion to be assessed is **tool customization**, which measures the degree to which a tool can be customized. This may include the availability of additional functions to be added to existing settings or the possibility to adapt the code to specific customer requirements, provided the know-how is available within the organization.[68]

The following weighting was applied to the previously mentioned criteria: System requirements – 30%, Usability – 15%, Interoperability – 20%, Support – 15%, Tool customization – 20%. System requirements were regarded as most important as these are crucial for the tool to even run on the desired platform. If the tool is not compatible with the system used, it cannot be considered at all, which justifies the weighting. Usability and support were not assessed as important as interoperability and tool customization, because the latter two are important conditions that the tool needs to meet in order to test the desired applications in their environment. If these criteria are not met, the selected tools may not be suitable options. Usability and support, however, can be compensated. A precise user manual may make up for bad usability and support may (when it is not available for free) be obtainable at an additional expense.

3.3.3 Functional criteria Test Management

The following section focuses on an additional set of criteria, which are used to evaluate the functionality of the previously selected open source test management tools. Unlike the two sets introduced above, these criteria are only applicable to test management tools, while another set of additional functional criteria will be used to evaluate open source test automation tools. The following criteria have been identified through an extensive review of test management tool re-

[68] Cf. Abiodun, O. M. (2011), p. 8 et sqq.

lated requirements: Test planning, Test requirement plan, design test cases, execution of test cases, detect defects, user right administration concept, reports and web access. [69]

The first criterion to be assessed is **Test planning**, which describes the ability to create and manipulate a test plan. This criterion also includes the evaluation of the ability to link the test plan to test requirements. The next criterion to be evaluated refers to the **Test requirement plan**, or in other word the ability to create, modify and delete test requirements. [70] The third criterion in this section describes the ability to **design test cases** for the required test level. Key activities of designing test cases are among others the choice of test techniques and the identification of test conditions and may also include the possibility to add parameters. Another feature is the **execution of test cases**, which is assessed based on the ability and flexibility to execute test cases as well as the degree of managing test cases. [71] These activities can either be done manually or automated by running a test script. The ability to **detect defects** and track bugs also represents an important functionality for a test management tool. Defects and issues are the outcome of testing and as the intention of test management tools is to manage the whole lifecycle of a testing process, it is important to keep track of these issues and ensure their resolution. Furthermore, the feature to implement a **user right administration** concept is desirable in a test management tool, as an organization may wish to make a distinction between the different users regarding the permission to see and write within the tool.[72] Another criterion, which should be paid attention to during the evaluation process, is the option to generate and customize **reports**, which reflect the entire testing process in a comprehensible manner. A favorable export file format would be a .pdf or any other read-only file. Finally, it may be beneficial for test management tools to provide features that enable **web access**. This eliminates the restriction to a specific workstation, which may improve the overall effectiveness of the tool.[73]

The weighting of functional test management criteria was assigned as follows: Test planning – 15%, Test requirement management – 15%, Design test cases – 15%, Execute test cases – 10%, Detect defects – 15%, User right administration – 10%, Test reports – 15%, Web access – 5%. Test planning was regarded as one of the most important factors as it helps to manage releases and cycles (iterations) more effectively. User can easily plan and track application re-

[69] Cf. Abiodun, O. M. (2011), p. 10, Illes, T. et al. (2005), p. 4 et sqq.
[70] Cf. Abiodun, O. M. (2011), p. 10 et sqq.
[71] Cf. Illes, T. et al. (2005), p.4
[72] Cf. Company (2012)
[73] Cf. Abiodun, O. M. (2011), p. 11

lease progress using this feature. Another crucial factor is the test requirement management. This module is important, because it enables the user to connect the requirements to the test cases and ensures traceability of requirements, which is important to keep track of the project state. A high importance lies also on the design of test cases. If test cases are specified wrong test results can pass, but do not test the required functionality. Therefore the design of test cases can be critical and needs to be supported by the tool. The test execution is important but not as central as the criteria mentioned before as it is usually only listing the entire required test steps and the user is supposed to update status of each step with passed, failed or not complete. Because of this it is an important feature to have, but usually does not lead to incorrect assumptions. Defect detection although is substantial for test management and thus rated with 15%. Defects should be logged and mapped to the corresponding test cases which failed and hence to the requirements in order to make sure that the defect is treated. Reports are important for the project state reporting to the rest of the team and management.

3.3.4 Functional criteria Test Automation

This section deals with the functional criteria, which the open source test automation tools are evaluated against, after concluding the assessment of the first two sections. After reviewing various resources and evaluation reports on open source test automation tools, the following criteria to be assessed were determined: Test automation approach, Script language, Predefined functions & methods, Integrated development environment, Modularity of test cases, Supported test methods (white/black box testing), and Reporting.[74]

First of all, the **test automation approach** will be evaluated. This includes the ability of the tool to execute Capture & Replay or Keyword & data driven replay. Furthermore, the **script language** is assessed in a next step. Here, it is of significance, which programming languages are supported, if it is object oriented, if it has modularization features and other relevant aspects. The third criterion evaluates if the tool comes with **predefined functions & methods** e.g. regarding browser handling, xml handling and csv handling. Another relevant criterion to be assessed is the **Integrated development environment**. In this context the IDE-integration, version management, test management and the ability to debug is evaluated.[75] As fifth criterion the **modularity of test cases** was determined. Here, special attention is paid to the possibility to

[74] Cf. Whichtestingtool.com (2012)
[75] Cf. ibidem

add parameters, to import and export excel files and the overall flexibility in the execution.[76] Moreover, it will be assessed if white box testing and/or black box testing are **supported test methods**.[77] Also of great importance is the last criterion to be evaluated: **Reporting**.[78] This criterion considers output formats, customization of reports, anonymity of users and the possibility to export the report as a read-only file such as pdf. A minimum requirement for this criterion is the possibility to export the data into an xls file, as from here manual reports may be created using Pivot etc. An additional field is included in the evaluation matrix to collect further features of the tool, which have not been assessed in a previous step.

The weighting of the criteria in the third section was determined as follows: Test automation approach – 20%, Script Language – 10%. Predefined functions/methods – 10%, Integrated development environment – 15%, Modularity of test cases – 10%, Supported test methods (white box testing) – 5%, Supported test methods (black box testing) – 10%, Reporting – 20%. The most crucial factors in this evaluation are the test automation approach and the reporting functionality. The test management approach includes the capture & replay functionality, typical for testing GUI applications, which meets the need of saving time and effort during the project. If this criterion is not included, the tool fails to meet this requirement. Reporting is inevitable as the test automation runs without human intervention and without reporting of the result it can be very time consuming to find the step where the error has occurred. Furthermore reporting is crucial for keeping track of the project state. Not as highly evaluated are the supported test methods black and white box testing. This methodology differs by the approach of testing. Within the black box testing the internal structure/ design/ implementation of the item being tested is not known to the tester whereas in white box testing it is known. As these factors collude together they are regarded similarly to the other criteria, but are not a decisive factor once one of the two methods is not supported by the tool.

[76] Cf. Company (2012)
[77] Cf. ibidem
[78] Cf. Whichtestingtool.com (2012)

4. Evaluation

4.1 Test Management

Tool	Open Source Criteria	Common Software Criteria	Test Management criteria	Total	Ranking
BugzillaTestopia	92	68	76	236	2
Testlink	96	91	88	275	1
Redmine	66	66	63	195	3
RTH	50	54	49	153	4

Table 2: Overview of Evaluation Results for Test Management Tools

The test management tools evaluated above are open source licensed under GPL, Bugzilla Testopia developed by Mozilla Foundation is furthermore licensed under MPL (Mozilla Public Licence). TestLink as well as Bugzilla Testopia take leading positions according to the community support, which becomes apparent in FAQ´s and forums. TestLink is supported by a large group of international testers and developing experts whereas Bugzilla Testopia is maintained and greatly supported by Mozilla Foundation; additionally commercial support is provided by Mozilla, while Redmine offers a public forum permitting open discussions, FAQ´s and active chat rooms. The Requirement and Testing Hub support is restricted to a public discussion forum and no community support is provided due to an one-man-developer-team. Consequently, Redmine and RTH received a lower ranking than TestLink and Bugzilla Testopia for the criterion community. TestLink, Bugzilla Testopia and Redmine published regular releases, while RTH provided less improvement activity to their user groups as a result of its limited human resources. Its last release was made available in 2009. TestLink and Bugzilla Testopia also achieved the highest rating regarding the documentation as user and developer manuals, release notes and stepwise installation guides are provided to potential users. RTH is ranked lower, since its documentation is mainly composed of immature FAQ´s and release notes.

TestLink is best rated regarding the open source criteria mentioned above with 96 out of 100 possible points, closely followed by Bugzilla Testopia receiving 92 out of 100 points. Redmine came off badly especially when considering the documentation whereas RTH has underperformed additionally in terms of community support and release activities.

System requirements, which is the first criterion evaluated in the common software criteria section, is higher rated for TestLink and RTH due to their standardized requirements. Both tools are written in JavaScript and PHP while Bugzilla Testopia and Redmine depend on Perl 5 and Ruby on Rails, which are both uncommon programming languages. RTH runs under Linux and Win-

dows, whereas the remaining three products represent platform independent systems. All of them support multiple databases like MySQL, which represents a commonly used standard database. All of the four open source test management tools provide a satisfying usability due to an intuitive web interface. However, regarding the ease of installation, TestLink represents the leading product as it offers an automatic scripted installation. TestLink as well as Redmine offer stepwise installation guides in order to facilitate the installation process; for RTH there is no installation guide available. The lowest ranked tool in this category is Bugzilla Testopia due to a command line driven installation, which is very uncommon nowadays. Integration with other tools is provided by TestLink, Redmine and RTH over an API interface, moreover TestLink enables the integration with LDAP server used for user authentication processes. Mozilla provides various add-ins for Bugzilla, but it is lower ranked in comparison because of the missing API. TestLink and Bugzilla Testopia offer various possibilities for customization. Bugzilla Extensions facilitates the modification of both, code and user interface, and the distribution to other users. Moreover, Bugzilla offers customization of css and images leading to an individual product skin. By using smarty templates with user-defined fields, TestLink is also highly customizable without requiring editing of the source code. RTH and Redmine are ranked moderately; because the framework can be extended and customized by add-ons, but they cannot keep compete with the degree of customization provided by Bugzilla and TestLink.

TestLink tops the list of all considered test management products when taking all evaluated software criteria into account. The Open Source Software achieved 91 from 100 points, followed by Bugzilla Testopia and Redmine. RTH received the lowest score of 54 points due to missing installation instructions and the lack of support.

Within the section of functional criteria for test management, TestLink exceeds the other test management tools by reaching 88 points, closely followed by Bugzilla Testopia scoring 76 points out of 100. Redmine as well as RTH lag behind with scores of 63 and 49 points respectively.

Regarding test planning, TestLink enables in addition to the creation and management of test cases also the organization of test cases into test plans. When using TestLink, test cases can be prioritized, assigned to testers and milestones can be defined. The test management extension for Bugzilla, Testopia, supports the export as well as the import of test plans (XML), whereas Redmine contains only features for scheduling test plans. RTH is also restricted to the creation and management of test cases. In summary, the features regarding test planning are far

more comprehensive for TestLink than for the remaining three and therefore it scored the highest within this section. The ability to create, manipulate and delete test requirements is best integrated in TestLink and Bugzilla Testopia by enabling imports as CSV files or by creating and deleting them manually via the interface. Using a CKEditor, which is an intuitive open source editor from CKSource that can be used in web pages, test cases can be manually created within RTH, and also be imported by using an .xls template, while Redmine enables test requirement management only by additional tool implementations and is accordingly scored the lowest. TestLink and Bugzilla Testopia support the import and export of Test Cases to and from XML file formats contrary to Redmine and RTH. Moreover, TestLink contains also configuration parameter allowing editing and adapting of the executed test cases. For this reason, it is ranked best in terms of designing test cases. Regarding the tracking of software problems, Bugzilla and TestLink are the leading ones, due to the fact that Bugzilla represents a bug-tracking-system itself containing all features required for defect or bug tracking. Within TestLink, defect management can be extended by integrating leading bug tracking systems, such as Mantis or even Bugzilla, which is for the main part a bug tracking system. RTH and Redmine only record and document occurred problems. User right administration is realized via authentication support (internal or external LDAP) with user self-registration support, which allows flexible role based access control when using TestLink. Various built-in permission levels assign the access privileges to different user groups. Bugzilla Testopia defines default users to whom individual privileges can be assigned. RTH only supports user groups with defined user rights, which cannot be modified. Consequently, it received a lower score.

All of the evaluated tools contain features for generating reports, whereas TestLink supports a variety of different reports, which can be exported to different formats (HTML, MS Word, MS Excel). The reports can also be sent directly via mail from the tool, which results in a higher score regarding the generation of reports.

Considering all three core areas - general open source criteria, common software criteria and functional criteria of test management - TestLink could be identified as the best test management tool of the ones examined, receiving 275 out of 300 points. The features that are not sufficiently provided are concerned with missing customization of reports as well as exporting reports to the pdf file format. Bugzilla Testopia is ranked as second best test management tool due to the fact that Bugzilla Testopia represents a bug tracking system (Bugzilla) with an integrated test management extension (Testopia). Although, Testopia provides test management features, like test planning and test requirement management, it cannot compete with TestLink.

Moreover, an uncommon scripting language (Perl 5) and a command line driven installation process result in point deduction leading to 236 points. Redmine also uses an uncommon scripting language (Ruby on Rails), provides only slight possibilities of customization and does not offer proper documentation. The main drawbacks of Redmine in the section of the functional test management criteria were identified for the criteria requirement management, the creation of test cases as well as their execution. When considering all core areas, Redmine reached an overall score of 195 points. The Requirement and Testing Hub cannot compete with the other test management tools when considering the open source criteria due to missing documentation and a low degree of maturity. RTH is rated poorly regarding the area of common software criteria as a result of missing support activities and a low degree of tool customization possibilities. Furthermore, test planning and requirement management as well as bug tracking features are not sufficiently provided. With a score of 153 pints out of 300, RTH received the lowest ranking of the evaluated tools.

The following adaption of Gartner´s Magic Quadrant to the evaluated open source test management tools summarizes the obtained assessment results within an evaluation matrix.

Figure 14: Adaptation of Gartner's Magic Quadrant to OSS Test Management tools

Gartner´s Magic Quadrant research method divides competing Open Source Software into four distinct sections, Leaders, Visionaries, Niche Players and challengers, by means of the *completeness of vision* and the *ability to execute* it. In order to adapt the Magic Quadrant to the characteristics of open source test management tools, the functional ability of the system including test tool criteria and functional criteria for test management tools was equated with Gartner´s *ability to execute*. The *completeness of vision* was adapted to the consideration of open source

test management tools by equalizing it with the performance in Open Source Software related criteria in order to consider aspects such as continuous improvement and proper documentation activities.

The preceding evaluation of the four test management tools TestLink, Bugzilla Testopia, Redmine and RTH has identified TestLink as distinct leader of Open Source Software for test management due to its excellent performance in functional and Open Source Software related criteria. Bugzilla Testopia´s classification within the leader quadrant is reasoned by good performance within the considered criteria due to the well-marked bug tracking functions and the extension of test management features. Therefore, Bugzilla Testopia may be particularly suitable if distinctive bug tracking features are required. Redmine and RTH are placed within the quadrant of niche players due to their moderate performance regarding the overall functionality as well as the fulfillment of Open Source Software criteria. When comparing Redmine and RTH, Redmine performs slightly better in both areas, the functional and the open source one.

The classification of the four considered test management tools may serve as a basis of decision-making when selecting an appropriate open source test management tool. Nevertheless, individual requirements should be determined as well as prioritized for the purpose of defining the expected functionality before selecting one particular tool.

4.2 Test Automation

Tool	Open Source Criteria	Common Software Criteria	Test Automation criteria	Total	Ranking
WatiN	64	69	49	182	4
Selenium	98	93	82	273	1
Canoo WebTest	86	86	75	247	2
T-Plan Robot	70	69	57	196	3

Table 3: Overview of Evaluation Results

All four products are open source licensed under Apache 2.0 or GPL V2.0. Community support is ranked highest for Selenium, as it has an active community, a mailing list, a wiki and a forum. Support is also offered commercially from third party vendors. T-Plan Robot only offers an FAQ site and a wiki, but limited commercial support as there is also a closed software license available. WatiN has FAQ's, but the developer himself is responsible for the support. Therefore, WatiN and T-Plan robot are ranked lower than Selenium. Releases and improvements are frequently made available to the user groups of Selenium and Canoo Web Test, whereas WatiN

and T-Plan Robot show less development activity. This is due to their popularity and user acceptance. The documentation is ranked highest for Selenium and T-Paln Robot. Selenium's documentation starts with the installation of the IDE and describes every step to a working test case with many screenshots. Furthermore, a functional reference is available inside the Selenium IDE. T-Plan Robot offers a good documentation as well as a step by step tutorial including images. Code samples are also made available. Canoo Web Test has a good online documentation with examples of how to create web tests. WatiN has no documentation, but a functional reference and is therefore only ranked moderately. Regarding the criteria mentioned above, Selenium is best rated for the open source criteria with 98 from 100 points while WatiN performs poorly especially in terms of documentation and community support.

Considering the common software criteria system requirements all four products are ranked high, as the test execution is either done inside the IDE or independent of the operating system. The overall usability and handling of Selenium is quite good. With the Selenium IDE testcases are created easily and fast and can directly run inside the IDE. Selenium RC is a little bit more complex, because the installation and configuration is not very easy, but well documented. The testcases can be developed in Java, C#, Python and Ruby and Selenium supports parallel execution. Within Canoo Web Test the installation is easy, as Java is provided during the installation and the download only needs to be unpacked. The installation of WatiN is also very easy and well documented. WatiN test recorder is able to capture test cases and export them to C# for further editing, which makes it easy to use. The execution can be done via the command line or inside the IDE, but WatiN is a .Net library, so the only way to use it is by developing an own test project using a .Net compatible language. T-Plan robot is a pure java application so there is no installation required, but the development of test cases is time consuming as there is no capture engine available. Furthermore, the tool is not very intuitive. The execution is done inside the IDE or via the command line. In terms of interoperability T-Plan robot is able to automate every platform providing a VNC server, so compatibility is ranked high. WatiN is compatible with the most common used browsers like Internet Explorer and Mozilla Firefox, but limited to Windows Microsoft and .Net and hence does not perform as good as T-Plan robot. Selenium runs on every platform supporting Java 1.5+. With its wide support for programming languages it fits into most existing environments. Canoo Web Test is also compatible as it uses Ant scripts, which allows direct and easy integration with continuous integration tools. Canoo Web Test, WatiN and T-Plan robot have no guaranteed support. Contrary to the other tools, Selenium has community support, an own IRC channel and a bug tracker. Moreover, numerous companies are

listed to provide commercial support, which guarantee support in urgent situations. WatiN can only be customized when writing own code and T-Plan robot offers customization and development of features on request which implies additional costs. Selenium and Canoo Web Test have a plugin that makes it easier to customize the tools for special requirements. When considering the common software criteria as whole, Selenium is the tool with the best ranking due to its score of 93 points, followed by Canoo Web Test with 86 points. WatiN and T-Plan robot received the lowest ranking due to a lack of support and customization.

As with the last two criteria, Selenium also outperforms the other three testing tools in the functional test automation section. By scoring 82 points in this section it is the undisputed leader, as Canoo Web Test only scores 73, T-Plan Robot 55 and WatiN 49 points. This already begins to show in the very first criterion - the test automation approach. Selenium interacts with the web application by injecting java script into the page. The captured tests are initially created in Selense, the programming of Selenium, but can be exported to Java, C#, Python or Ruby for further processing. Furthermore, Selenium is able to synchronize automatically with the web application. No native keyword & data driven approach is possible, but can be featured when using the tool Fitnesse in addition. WatiN interacts via COM with the browser. It has built-in synchronization features and verification of the web application is done by the WatiN API. It is not able to capture interactions with the application under test, but WatiN TestRecorder can be installed to enhance the tool with this feature.

T-Plan Robot works with image recognition and therefore only image capturing, but not capturing of whole test cases is possible. Record & Replay capability is not included and due to the restricted proprietary scripting language a development will not be possible. As T-Plan Robot also records the interaction via mouse clicks and keyboard strokes, it reflects reality best in comparison to the other tools. Verification features are limited to image and text verification. Canoo Web Test has no capture & replay feature, but for the test case development, it is possible to install the WebTest recorder, which can record the actions of the user and converts it into the desired scripting language. Selenium supports many programming languages (C#, Java, Perl, PHP, Python, & Ruby); WatiN only supports .Net-able languages; Canoo Web Test supports XML and Groove and T-Plan Robot has its own very limited scripting language. In regards of predefined functions and methods, Selenium offers no native support for file handling or user interrupts, but supports browser handling, xml handing, csv handling and file systems. WatiN is able to deal with various file formats using the .Net functionality, but does not support xml, csv and file systems. T-Plan Robot is not able to deal with files in any way, but supports any brows-

er. Canoo Web Test simulates Firefox' or Internet Explorer's way to execute JavaScript. This behavior is simulated, which means that it does not work as good as in a "real" browser. Regarding the Integrated Development Environment Selenium comes with the Selenium IDE, which supports auto completion and correction of syntax. The IDE can be integrated with most common Testing Frameworks (JUnit, Nunit, TestNG, Bromine, Test::Unit, etc.). While running the test cases it is possible to debug them step-by-step. With WatiN the utilization of every .Net compatible IDE is possible. Debugging is also possible, but it does not feature any grouping of test cases nor other test management features. The IDE of T-Plan Robot does not help the user in many ways. Version control is not possible, test management features not available and difficult to implement, but some debugging features are implemented. Canoo Web Test has support for IDE integrations. Reporting is the only criteria in which Selenium is not as strong as its competitors. It automatically creates XML reports with the result of each automation step. Customer reports are not possible, but can be introduced with some programming effort or by additionally installing the tool "SeleniumCamp", which is developed to provide this missing feature. WatiN also does not support this feature, but .Net can be used to develop a report engine. However, T-Plan Robot is able to create HTML reports with included screenshots of the application under test and is able to send mails to report test automation states. Canoo Web Test uses XSLT (extensible style sheet language transformations) to provide the presentation of the test results.

When considering the assessment area as whole Selenium is the best-ranked tool throughout all categories. The only features that are missing is sufficient reporting and native keyword and data driven approach, but for these missing features already tools have been developed, e.g. Fitnesse or SeleniumCamp, which make it easy to overcome the drawbacks of this tool. Canoo Web Test is ranked second best, especially due to an uncommon scripting language, a missing IDE and no guaranteed support, but it is a great option for users that only require doing automated black box tests with Web Browser. WatiN is very dependent on the Microsoft Windows platform and .Net, which is a disadvantage if platform independency is required. It furthermore does not include reports, offers any help for the customization of the tool, documentation is insufficient and the developer himself does the support, which on the one hand can be an advantage as he knows the full functionality, but on the other hand can be a risk when urgent problems occur. T-Plan Robot is the best option if only platform independency is required, but test case creation can be very time consuming and the technique still has a lot of limitations. Additionally there is a closed source license available, which is more developed, has great support and offers a lot more functionalities and leads to a neglect of this tool.

In accordance with Gartner's Magic Quadrant research methodology, a similar evaluation matrix is derived at this point for the assessed open source test automation tools to visualize and summarize the evaluation results. It also elucidates in which areas drawbacks were observed.

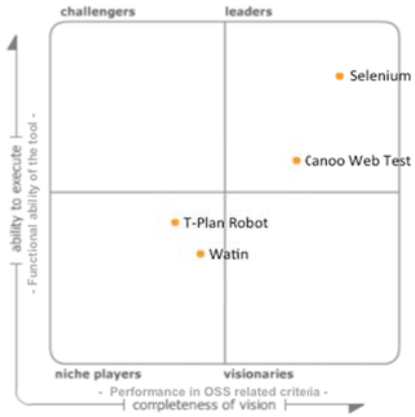


Figure 15: Adaptation of Gartner's Magic Quadrant to OSS Test Automation tools

The *ability to execute* was equalized with the overall functional ability of the tool, where mainly the common test tool criteria and functional criteria for test automation were considered, and hence the overall functionality of the test automation tool is assessed. The *completeness of vision* was adapted to the overall performance in open source related criteria regardless of the actual application field; since this is a good representation of how committed a community or provider is in extending and improving its Open Source Software and actually making the offered software a serious alternative to commercial software. The graphical competitive positioning identifies four types of technology providers: Leaders, Visionaries, Niche Players and challengers. The evaluation of the 4 tools has produced a clear winner, Selenium, with an outstanding performance in functional as well as open source-related criteria, making it a clear leader in the field of Open Source Software for test automation. Canoo WebTest's good performance in these areas also justifies its positing in the leader quadrant, though it has to be emphasized once more that Canoo WebTest is a great tool for automated testing of web applications, whereas Selenium has a wider application range, which goes beyond the testing of web applications. However, Canoo WebTest may be the better choice, if only that application area needs to be covered. Due to their evaluation T-Plan Robot and Watin were allocated in the quadrant Niche players, implying that these tools may be a good solution for some user groups, but in terms of overall performance in meeting the average requirements of all users regarding a good Open Source Software for test automation Canoo WebTest and Selenium represent the better options. When comparing the two to one another, the matrix illustrates that T-Plan Robot per-

formed slightly better than Watin in terms of functionality, while Watin meets more OSS requirements that identify a good Open Source Software than T-Plan Robot. When assessing the average expectations in terms of OSS and functionality of our four tools, the presented ranking may serve as a guide for selecting the right tool. Though, before making a decision in favor of a particular tool, the individual requirements need to be determined and mapped on to the functionality of the particular tool. If an Open Source Software is the preferred choice, an outstanding performance in the open source related criteria is of major importance as the continuous improvement and release activity and active support are crucial for the utilization of the software. Regardless of the overall quality and functionality of the tool, if the development activities are abandoned the potential benefits of the tool cannot be leveraged and a new tool would have to be found. Furthermore, the switching costs in this case may exceed the costs for a commercial product, where continuous improvement and support is ensured by the vendor.

5. Recommendation

Current Situation

The company has currently no testing tool in use. The testing is done manually by the developers themselves and the relevant division. Utilized programming languages are: Java, C++, and VB.

Requirements

The customer defined a set of requirements a suitable solution should meet. The requirements include the ability to test applications with graphical user interfaces, such as browser applications, as well as java and COBOL programs. The tool should be compatible with Windows 7 and Linux operating systems. Black and white box testing should be supported by the test automation tools. Furthermore, it is of high importance that the selected test management tool can be integrated with the test automation tool. Another requirement the selected tools should meet is stability as multiple users should be able to work with the tool simultaneously. These also need to be provided with an easily comprehensible user documentation to ensure proper usability. Functional criteria that should be incorporated are: user right administration, test case excel import/export, plan/manage test resources, requirement management, modularity of test cases (parameter insertion, flexible execution) and reports.

Requirement Mapping

In the following an overview of the selected tools is given and the extent to which they meet the above listed requirements are presented.

	OPEN SOURCE SOFTWARE								CLOSED SOURCE SOFTWARE	
	Test Management				Test Automation					
	Bugzilla Testopia	Redmine	RTH	TestLink	Cannoo Web Test	Selenium	T-Plan Robot	WantiN	HP ALM	IBM Rational
Testing of graphical user interfaces										
Browser/Internet Applications					√	√	√	√	√	√
Lotus Notes Applications					X	X	X	X	X	√
COBOL programs	X	X	X	X	X	X	X	X		√
Java programs	√	√	√	√	√	√	√	√	√	√
Programming languages										
Java					X	√	√	X	√	√
C++					X	X	X	√	√	√
VB					X	X	X	√	√	√
XML					√	√	X	√	√	
System requirements										
Windows 7 Server/ Linux	√	√	√	√	√	√	√	√	√	√
DB SQL	√	√	√	√	√	√	√	√	√	√
Test methods										
White Box					X	√	X	X		√
Black Box					√	√	√	√	√	√
Test Tool/ Open Source Criteria										
Interoperability of test management/ test automation software					√				√	√
Stability	√	√	√	√	√	√	√	√	√	√
User documentation/ FAQ	√	√	√	√	√	√	√	√	√	√
Functional criteria										
User right administration	√	√	√	√					√	√
Test Case Excel import/export	√	√	√	√					√	√
Plan/Manage test resources	X	√	X	√					√	√
Requirement management	X	√	√	√		√		√	√	√
Modularity of test cases (paramter insertion, flexible					√	√	X	√	√	√
Reports	√	√	√	√	√	√	√	X	√	√

Figure 16: Requirement Mapping

As seen above, while the ability of testing graphical user interfaces within the browser or the internet is given by all examined tools, IBM Rational Quality Manager is the only tool that can test Lotus Notes applications as well. Java applications can be easily tested with any of these tools; however COBOL applications represent a challenge for all of them, except for IBM Rational Quality Manager, which can be extended by a plug-in to address this issue. The programming languages used at the company are Java, C++ and Visual Basic (VB). As XML is also a widely spread markup language it is considered in the evaluation as well. Only the commercial tools support all languages. However, it is not a necessity that all tools support all languages; it is more important that at least one of the above mentioned is sup- ported. For further information on which tool supports what language, please see Figure 16. When examining the tools for the predefined system requirements (operating system, database)

all tools performed satisfactorily. Also of significance for the company is if black box and/or white box testing is included. The commercial tools (IBM Rational and HP Quality Center) and the best rated test automation tool Selenium are capable of both, while the remaining open source test automation tools only support black box testing.

The interoperability between test management tools and test automation tools is guaranteed for all of them, as they all have an API interface, which enables the communication between software components. The commercial tools have test automation and test management capabilities and can also be combined with all other open source tools as they provide an API interface. This could be relevant when an open source tool is already utilized within the company and should be extended with additional functionalities. Concerning stability and user documentation the evaluated tools fulfill the requirements. User right administration is a functionality which applies to the test management tools and is included in RedMine, TestLink, Bugzilla Testopia, RTH, HP Quality Center and IBM Rational, whereas the planning and management of test resources is only available within Redmine, TestLink and the commercial tools. A further functional requirement is the ability of importing and exporting test cases from Excel-sheets. This requirement is fulfilled by all test management tools, but it could be possible that an additional Plug-In needs to be downloaded to provide this functionality. Another feature that is also very important is requirement management in order to ensure that all needs are fulfilled. This feature is provided by all test management tools except of Bugzilla Testopia. The customer also defined the modularity of test cases as a desired functionality, which is satisfied by the commercial tools and almost all test automation tools; T-Plan Robot is the only tool that lacks this kind of functionality. The last requirement was the reporting functionality. Regarding this functionality report customization and the ability to export the reports was important, where pdf is the preferred output format in order to maintain the inalterability when reporting the results to the management and to the auditors. Furthermore, it should not show information that allow an association between the users and the executed work, as this is not supported by the works council. During the collection of requirements, usability was also mentioned as an important soft factor. In the appendix, a comprehensive evaluation of different criteria is included, which also provides an assessment of the usability of the different tools. The best ranking was obtained by TestLink and Redmine for test management tools and by Canoo WebTest and Selenium for test automation tools.

Recommendation

Looking at the big picture, Selenium and TestLink have emerged once again as the best rated open source tools and are therefore the most recommendable tools that can find their application at the company. All demanded requirements are met by TestLink, which makes it a suitable option for the customer. Selenium only lacks in the ability of testing Lotus Notes applications and COBOL programs, which is not provided by the other considered open source tools as well, but covers all remaining requirements. However, if the testing of Lo- tus Notes and COBOL applications is of major importance to the customer, commercial tools should be considered as an alternative to also address these requirements.

6. Conclusion

The presented work is based on the fact that many companies and organizations see the need to test their software before release to ensure proper functioning, but do not have the necessary resources to perform this kind of testing due to either a lack in human or financial resources. The paper addresses these issues by examining open source test management and test automation tools, which can compensate these two factors. The fact that software is open source keeps financial expenses down, while test automation intends to disburden the personnel usually responsible for manual testing.

For this purpose, a number of tools of both areas were selected for further examination. The selection process involved a review of multiple surveys and rankings, which resulted in eight tools, four for test management, and four for test automation. Furthermore, two commercial tools were selected in order to present the differences in scope and functionality between open source and Closed Source Software. Following this, an evaluation matrix was developed to serve as basis for the tool assessment. Through an extensive literature research, various criteria were defined, which were grouped into three distinct categories: Open source criteria, Common software criteria and functional criteria. After assessing the tools in the evaluation matrix, the results were compared and a test winner for each area identified. The winners were: TestLink for Test management and Selenium for Test automation. The insights that were gained in that process were then utilized to formulate a recommendation, by matching the individual customer requirements of the company to the capabilities of the

open source and commercial tools. Once again, Selenium and TestLink were concluded as the most suitable open source option for the customer as the majority of requirements are met.

The findings of this paper elucidate that Open Source Software has in fact become a valuable business alternative to commercial tools, as the pre-defined customer requirements could be satisfactorily met. Of major importance in this process is to ensure that open source criteria, such as a reliable community and useful user documentation are provided. For the company open source tools are a good option. This may not be generally the case. To ensure the suitability of an open source tool, all requirements should be defined and evaluated in detail to find the best solution in practice.

.

List of references

Books

Bath, G./McKay, J. (2010): Praxiswissen Softwaretest - Test Analyst und Technical Test Analyst: Aus- und Weiterbildung zum Certified Tester - Advanced Level nach ISTQB-Standard, Heidelberg: dpunkt Verlag

Bommer, C./Spindler, M./Barr, V. (2008): Software-Wartung: Grundlagen, Management und Wartungstechniken, Heidelberg: dpunkt Verlag

Copeland, L. (2004): A Practitionor's Guide to Software Testing, 2nd Edn., London/Norwood: Artech House Publishers

Dustin, E./Rashka, J./Paul, J. (2001): Software automatisch testen : Verfahren, Handhabung und Leistung; mit 100 Tabellen, Berlin: Springer Verlag

Grechenig, T. u.a. (2010): Softwaretechnik: Mit Fallbeispielen aus realen Entwicklungsprojekten, München: Pearson

Hetzel, B. (1993): The Complete Guide to Software Testing, San Francisco: John Wiley & Sons Inc.

Franz, K. (2007): Handbuch zum Testen von Web-Applikationen: Testverfahren, Werkzeuge, Praxistipps, Berlin/Heidelberg: Springer Verlag

Ludewig, L. J. (2010): Software Enginieering. Grundlagen, Menschen, Prozesse, Techniken, 2nd Edn., Heidelberg: dpunkt Verlag

Menzel, M. (2006): Software-Testautomatisierung: Leitfaden für die effiziente Einführung, Saarbrücken: VDM Verlag

Moore, W. J. (2002): Encyclopedia of Software Engineering, Indianapolis: John Wiley & Sons, Inc.

Myers, J. G. (2001): Methodisches Testen von Programmen, 7th Edn., München: Oldenbourg Wissenschaftsverlag

Pol, M./Koomen, T./Spillner, A. Management und Optimierung des Testprozesses: ein

(2002): praktischer Leitfaden für erfolgreiches Testen von Software mit TPI und TMap, 2nd Edn., Berlin/Heidelberg: Springer Verlag

Sommerville, I. (2007): Software Engineering 8, Edinburgh: Pearson

Strahringer, S. (2011): Application Management, Berlin: dpunkt Verlag

Weber, S. (2004): The Success of Open Source, Cambridge: Harvard University Press

Wieczorek, M./Meyerhoff, D. (2001): Software Quality, Köln: Springer Verlag

Articles

Gartner (2011): Magic Quadrant for Integrated Softwrae Quality Suites.pdf, p. 8, 10, 12, 13

Hewlett Packard (2010): HP Application Lifecycle Management Software

IBM (2011a:) Quality management across the product and lappication life cycle.pdf, p. 5ff.

IBM (2011b): RQM – Eine kurzübersicht aus Practitioner Sicht – 16-09-2011.pdf

Soto, M./ Ciolkowski, M. (2009): The QualOSS Open Source Assessment Model- Measuring the Performance of Open Source Communities, in: Third International Symposiumm on Empirical Software Engineering and Measurement, Frauenhofer Institute, p. 1

Online Articles

BITKOM (w.y.): Open Source Software, http://www.bitkom.org/files/documents/BITKOM_Publikation_OSS_Version_1.0.pdf, retrieval: 20.05.2012, p. 20

Paper

Abiodun, O. M. (2011): Open Testing – Open Source Software Testing tools Evaluation model: Kemi-Tornio University of Applied Sciences

Illes, T. et al. (2005): Criteria for Software Testing Tool Evaluation – A Task Oriented View, Heidelberg: University of Heidelberg

Van den Berg, K. (2005): Finding open options – An Open Source Software evaluation modelwith a case study on Course Management Systems, Tilburg: Tilburg University

Internet sources

Aberdour, M. (2012b): Test management tools, http://www.opensourcetesting.org /testmgt.php, retrieval: 15.07.2012

Android.com (w.y.) : Philosophy and Goals, in: Android Open Source Project, http://source.android.com/index.html, retrieval: 20.05.2012

Bugzilla (2012a) Bugzilla, www.bugzilla.org, retrieval: 07.06.2012

Bugzilla (2012b): Who uses Bugzilla?, http://www.bugzilla.org/installation-list/, retrieval: 30.06.2012

Bugzilla (2012c): The bugzilla guide – Installation, http://www.bugzilla.org/ docs/2.18/html/installation.html, retrieval: 29.06.2012

Bugzilla (2012d): Bugzilla Screenshots, http://sourceware.org/eclipse/ bugzilla/images/query.png, retrieval: 03.07.2012

Canoo Engineering AG (2012a): Canoo Web Test White Paper, http://webtest.canoo.com /webtest/manual/whitepaper.html, retrieval: 21.05.2012

Canoo Engineering AG (2012b): Web Test Key Characteristics, http://webtest.canoo.com/webtest/manual/keyCharacteristics.html, retrieval: 15.06.2012

Electric Cloud (2010): Survey founds 58% of Software Bugs Result from Test Infrastructure and Process, Not Design Defects, http://www.electric-cloud.com/news/2010-0602.php, retrieval: 03.07.2012

Gartner Inc. (2011): Gartner Survey Reveals More than Half of Respondents Have Adopted Open source Software Solutions as Part of IT strategy, http://www.gartner.com/it/page.jsp?id=1541414, retrieval: 03.07.2012

IBM (2011): Developer Library, http ://www.ibm.com/developerworks /rational/library/06/1107_davis/, retrieval : 05.06.2012

Jhureley, A. (2010): Recommended open source tools for test management and defect tracking, http://www.ezdia.com/epad/recommended-open source-tool-test-management-defect-tracking/681/,

retrieval: 17.05.2012

Lanier, J. (2007): Discover Magazine, Long-Live Closed Source Software, http://discovermagazine.com/2007/dec/long-live-closed-source-software/article_view?b_start:int=1&-C=, retrieval: 29.06.2012

Opensourcetesting.org (2012): Functional testing, http://www.opensourcetesting.org/functional.php, retrieval: 20.09.2012

Prins, A. (2009): Software Testing and more, http://www.testingthefuture.net/2009/08/275/, retrieval: 12.03.2012

QA TestLab (2009): Best Open Source Test Management and Bug Tracking Tools, http://www.scribd.com/doc/47882053/Best-Open source-Test-Management-and-Bug-Tracking-Tools, retrieval: 25.06.2012

Redmine (2011): Wiki-Charts Plug-In, http://www.redmine.org/projects/redmine/wiki/PluginCharts, retrieval: 03.07.2012

Redmine (2012) Redmine, www.redmine.org, retrieval: 07.06.2012

Rouse, M. (2007): TechTarget, Market Leadership, http://searchcrm.techtarget.com/definition/market-le%20provides, retrieval: 25.06.2012

Selenium (2011): What is Selenium?, http://seleniumhq.org/, retrieval: 30.06.2012

Selenium (2012): What is Selenium?, http://seleniumhq.org, retrieval: 07.06.2012

Simon, H. (2007): Hidden Champions of the Twenty-First Century: Success Strategies of unknown, http://books.google.de/books?id=zoj3R9DuXwwC&pg=PA49&dq=market+leader+definition&hl=de&sa=X&ei=91TrT7ftNI7Hsgbw4uTVBQ&ved=0CDcQ6AEwAA#v=onepage&q=market%20leader%20definition&f=false, retrieval: 15.06.2012

TestJournal (2011): 10 Best Tools for Test Automation, http://www.toolsjournal.com/articles/item/195-10-best-tools-for-test-automation , retrieval: 20.05.2012

Testmanagement (w.y.a): IBM Rational Anforderungs- und Testmanagement, http://www-01.ibm.com/software/de/rational/offerings/manage.html, retrieval: 29.06.2012

Testmanagement (w.y.b): Take control over your test process with the right test management tools, http://www.testmanagement.com/, retrieval: 29.06.2012

T-Plan (2012a): T-Plan Robot Enterprise, http://www.t-plan.com/robot/index.html, retrieval: 15.06.2012

T-Plan (2012b): T-Plan Robot Projects, http://www.t-plan.com/robot/docs/versions.html, retrieval: 15.06.2012

T-Plan (2012c): T-Plan Robot FAQ, http://www.t-plan.com/robot/docs/faq.html, retrieval: 15.06.2012

T-Plan (2012d): Image based versus object oriented testing, http://www.t-plan.com/robot/docs/articles/img_based_testing.html, 15.06.2012

WatiN (2011): WatiN Overview, http://watin.org/, retrieval: 30.06.2012

Whichtestingtool.com (2012): Open Source Test Automation Tool/ Framework Evaluation, http://www.whichtestingtool.com/tool-evaluation.html, retrieval: 07.06.2012

Xu, J. (2005): Market Research Handbook- Measurement, Approach and Practice, http://books.google.de/books?id=73-eSezp7sAC&pg=PA69&dq=market+share+definition&hl=de&sa=X&ei=bVrrT-nGM4TSsgbCgtnmBQ&ved=0CDcQ6AEwAA#v=onepage&q=market%20share%20definition&f=false

Vergleich von Open Source Web-Applikation-Honeypots zum Aufbau einer Penetrations-Test-Umgebung

Seminararbeit 6. Semester

vorgelegt am 05.07.2012

Fakultät Wirtschaft

Studiengang Wirtschaftsinformatik- International Business Information Management

Kurs WWI2009I

Modul Open Source

Prof. Dr. Thomas Kessel

von

Robert Bruchhardt, Maximilian Heinemeyer, Gerd Radecke, Jennifer Zohar

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einleitung

1.1 Einführung und Problemstellung

„Hacker knacken Server der SPD“ - so der Titel eines Spiegel Online Artikels vom 20. Mai 2012 zum Thema Sicherheitslücken auf Webseiten. Eine Gruppe anonymer Personen hat sich zum Ziel gesetzt unsichere Webseiten aufzudecken. Im Zuge dessen hat diese Gruppe unter anderem Sicherheitslücken auf der Website der SPD genutzt, um sich so Zugriff zu Login-Daten und Passwörtern zu verschaffen. Dieser Artikel zeigt sowohl die Aktualität als auch die Relevanz und Bedeutung des Themas Sicherheitslücken auf Webseiten.[1]

Nahezu alle Unternehmen haben in der heutigen Zeit eine signifikante Webpräsenz mit essentieller Funktionalität. Die Webseite eines Unternehmens ist zugleich das Portal zu Datenbanken, welche im Hintergrund teils geschäftsrelevante Informationen speichern, wie beispielsweise die zuvor erwähnten Login-Daten. In diesem Zusammenhang nutzen Angreifer Sicherheitslücken auf Unternehmenswebseiten, um relevante Daten zu extrahieren und zu missbrauchen. Ein erfolgreicher Angriff hat einen negativen Einfluss auf die Unternehmensreputation und sorgt daraus resultierend für den Verlust von Kunden. Zusätzlich folgen nicht selten direkte oder indirekte finanzielle Schäden für das Unternehmen.

Zur Vermeidung bzw. Aufdeckung von bereits bestehenden Sicherheitslücken wenden Unternehmen verschiedene Methoden an. Eine Möglichkeit stellen sogenannte Penetration-Test-Labore zur Schulung der internen IT-Abteilung dar. Hierfür werden Web-Applikations-Honeypots genutzt. Honeypots sind Systeme mit integrierten Sicherheitslücken, welche auf einem Server installiert werden und in einer Testumgebung IT-Sicherheitspersonal eines Unternehmens die Möglichkeit bieten das System zu attackieren. Auf diesem Weg können verschiedene Angriffsmethoden und -Quellen analysiert, getestet und erkannt werden.[2]
Vor diesem Hintergrund ergibt sich die herausfordernde Wahl des passenden Honeypots für ein Unternehmen. Verschiedene Kriterien müssen bei der Wahl des Test-Systems untersucht werden. Aus dieser Herausforderung ergibt sich die Motivation der vorliegenden Arbeit.
Jedes Unternehmen hat das Bestreben der Kostenersparnis, weshalb insbesondere frei zugängliche und kostenfreie Open Source Honeypots zum Erstellen von Penetration-Test-Laboren in Frage kommen. Hinzu kommt, dass kaum kommerzielle Honeypot-Alternativen existieren.

[1] Vgl. SpiegelOnline (2012) http://www.spiegel.de/netzwelt/web/partei-gehackt-unbekannte-knacken-server-der-spd-a-833905.html
[2] Vgl. Harwood, M. (2011), S.371

1.2 Zielsetzung

Ziel dieser Arbeit ist es, verschiedene Open-Source-Honeypots, welche zu Trainingszwecken eingesetzt werden, zu installieren, auf Grund von verschiedenen Kriterien zu evaluieren und für Penetration-Test-Labore zu empfehlen. Diese Kriterien umfassen Aspekte wie beispielsweise das Vorhandensein von relevanten Sicherheitslücken bei Webapplikationen oder aber die Einfachheit der Installation und die Usability des Honeypots. Das Ergebnis dieser Arbeit enthält im Wesentlichen

- eine **modulare Bewertungsmatrix** zur weiteren Verwendung und optional individuellen Modifikation durch den Leser und
- konkrete **Empfehlungen von Honeypots**, welche sich zur Schulung von Mitarbeitern eignen, um typische Sicherheitslücken aufzuzeigen.

Die Bewertungsmatrix enthält mehrere Kategorien, welche wiederum von verschiedenen Kriterien bestimmt werden. Dieser Kategorien können je nach Schwerpunktsetzung des Unternehmens unterschiedlich gewichtet werden. Vorteilhaft ist dabei die Modularität der Matrix.
Weiterhin sollen diese ausgewählten Honeypots Unternehmen dabei unterstützen, Sicherheitslücken auf deren eigenen Webapplikationen zu vermeiden. Der Einsatz von Honeypots kann in Penetration-Test-Laboren, Workshops zur IT-Sicherheit oder anderen Trainings stattfinden. Neben dem Lerneffekt für Webentwickler ist IT-Sicherheit in einem Unternehmen unabdingbar.

1.3 Vorgehensweise

Die Arbeit gibt zunächst Aufschluss über verschiedene Arten von Honeypots, die Definition dieser und eine Erläuterung von Penetration-Test-Laboren, welches ein notwendiger Schritt ist, um zu verstehen, welche Honeypots zu welchen Zwecken in dieser Ausarbeitung näher betrachtet werden. Darauf aufbauend folgt die Beschreibung der angewandten Methodik zur Bewertung der Open Source Honeypots. Hierzu werden alle in der Bewertungsmatrix enthaltenen Kriterien näher ausgeführt. Daraufhin erfolgt eine Produktvorstellung von möglichen Honeypots, aus welcher eine definierte Anzahl an Produkten ausgewählt und näher untersucht wird. Nach der Auswahl erfolgen die Evaluation der Honeypots und die Bewertung. Schließlich werden die Ergebnisse der Auswertung beschrieben. Handlungsempfehlungen, eine Zusammenfassung der Arbeit und ein Ausblick folgen im Schlusskapitel.

2. Grundlagen

Im Folgenden Kapitel werden die notwendigen Termini erläutert, um so die Basis für die nachfolgenden Kapitel zu schaffen. Definiert werden die Begriffe Honeypot, Penetrationtest und Web-Honeypots werden von Server-Honeypots abgegrenzt.

2.1 Honeypots

Der Begriff Honeypot stammt aus dem englischen und kann mit “Honigtopf” übersetzt werden. Im Bereich der Informationstechnologie (IT) wird unter diesem Terminus eine „Zielmaschine“[3] verstanden, auf welcher absichtlich Sicherheitslücken im System integriert werden, um so einen Angreifer bei dessen Vorgehen zu beobachten.
Eine einheitliche und allgemein akzeptierte Definition von Honeypot ist nicht existent, da unterschiedliche Honeypots für verschiedene Zwecke genutzt werden.[4]
Vor diesem Hintergrund ist insbesondere im Rahmen dieser Arbeit eine Unterscheidung zwischen Honeypots, die als Zielmaschine für externe Angreifer zu Kontrollzwecken genutzt werden und Honeypots als Testsysteme, um mögliche Angriffe zu demonstrieren und zu evaluieren, zu machen. Weiterhin erfolgt die Unterscheidung zwischen Honeypots, welche Sicherheitslücken auf Webseiten darstellen und solche, die im Serverumfeld zu Übungszwecken eingesetzt werden. Im Folgenden werden die genannten Honeypot-Arten voneinander abgegrenzt.

2.2 Unterscheidung Web-Honeypot und Server-Honeypot

Unter dem Begriff **Web-Honeypot** werden im Rahmen dieser Arbeit Honeypots verstanden, welche Sicherheitslücken auf Webapplikationen nachbilden, wie z.B. SQL-Injections oder eine kryptografisch unsichere Speicherung. Im Vergleich dazu existieren auch Honeypots, welche Sicherheitslücken auf **Serverebene** simulieren, wie z.B. Lücken in der Konfiguration eines Servers. Auf einem Server-Honeypot befinden sich angreifbare Applikationen, die darauf warten, dass Angreifer Lücken entdecken und ausnutzen.[5]
Neben den genannten Arten von Honeypots muss zudem die Absicht, welche hinter dem Einsatz von Honeypots steckt, abgegrenzt werden. Dabei gibt es zum einen die Möglichkeit Honeypots zum Studieren von Angreifern einzusetzen (=Intrusion Detection System). Hierbei werden bewusst Sicherheitslücken auf einem aktiven Server oder einer Webapplikation integriert, um auf diese

[3] Peikari, C./ Chuvakin, A. (2004), S.488
[4] Vgl. Spitzner, L. (2003), http://www.tracking-hackers.com/papers/honeypots.html ; vgl. dazu auch: Krooß, M. (2003), S.8
[5] Vgl. Gerrit Göbel, J. / Dewald, A. (2011), S.11

Weise eine Angriffsfläche für Hacker zu schaffen und deren Vorgehen bei einem Angriff zu analysieren.

Daneben besteht die Möglichkeit Honeypots zum internen Training zu nutzen. Diese Trainings-Honeypots können beispielsweise Applikationen auf einem Server sein, in welchem Sicherheitslücken integriert sind, um internen Entwicklern und IT-Sicherheitsverantwortlichen die Chance zu geben, als Angreifer in einer Testumgebung zu agieren und Sicherheitslücken auszunutzen.

Im Rahmen dieser Arbeit werden Web-Honeypot-Produkte evaluiert, welche Webapplikationen mit Sicherheitslücken simulieren und zu Trainingszwecken in einer internen Testumgebung dienen, wie Abbildung 1 zeigt.

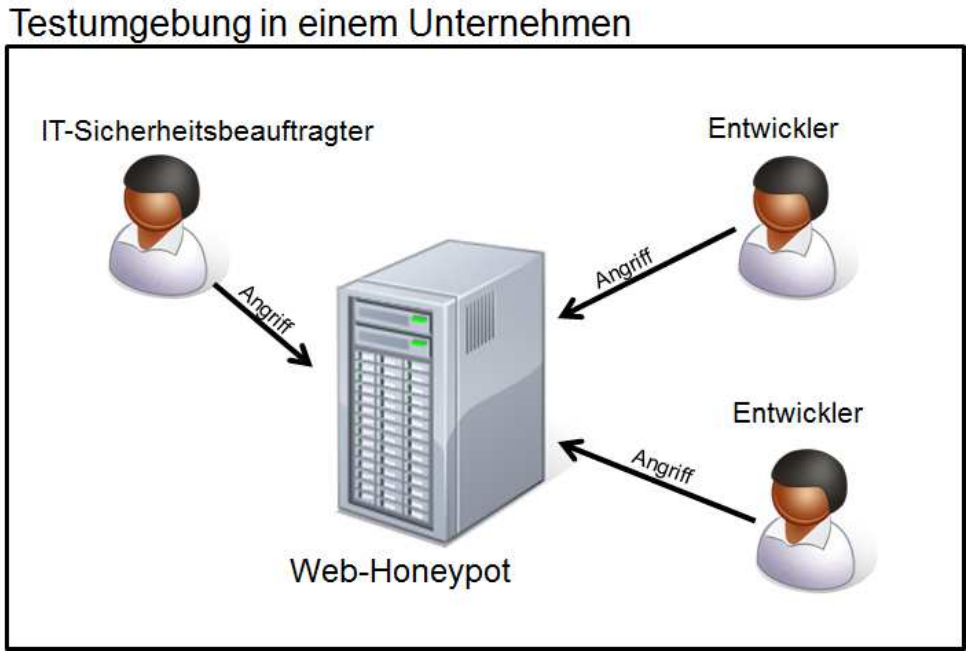


Abb. 1: Modell eines Web-Honeypots in einem Testlabor

Als Testumgebung werden häufig Penetrationtest-Labore in Unternehmen verwendet. Penetrationtesting im Allgemeinen wird im nachfolgenden Kapitel näher ausgeführt.

2.3 Penetrationstest

In der Literatur wird ein Penetrationtest wie folgt definiert: “Penetration Test ist die Bezeichnung für eine Sicherheitsüberprüfung eines Netzwerk- oder Softwaresystems aus Sicht eines Hackers”.[6] Der Schluss der Definition “Aus Sicht eines Hackers”, impliziert die Frage nach der Herangehensweise, d. h. wie ein Angreifer sich Zugang zu einem System verschafft und dieses steuert. Ziel ist

[6] Wilhelm, T. (2010), S.166

es mit Hilfe von Penetrationtests Schwachstellen im System zu identifizieren und die Sicherheit der Systeme zu erhöhen.

Weiterhin erfolgt die Unterscheidung zwischen zwei Typen von Penetrationtests: White-Box Test und Black-Box Test. Bei einem **White-Box Test** verfügt der Tester über das Wissen und die Informationen über das System, wie z. B. die IP Adresse, Open Source Software oder Quellcode, welches getestet werden soll. Basierend auf der Definition kann bei einem White-Box Test davon ausgegangen werden, dass der Tester einen internen Angreifer simuliert.

Bei einem **Black-Box Test** wird hingegen ein externer Angreifer simuliert, welcher keinerlei Informationen über das Zielsystem bzw. -netzwerk hat. Der Penetration-Tester, welcher zugleich den externen Angreifer darstellt, muss sich alle notwendigen Informationen besorgen, um den Test durchführen zu können.[7]

Penetrationtesting wird in vier Phasen durchgeführt wie die nachfolgende Abbildung 2 zeigt.

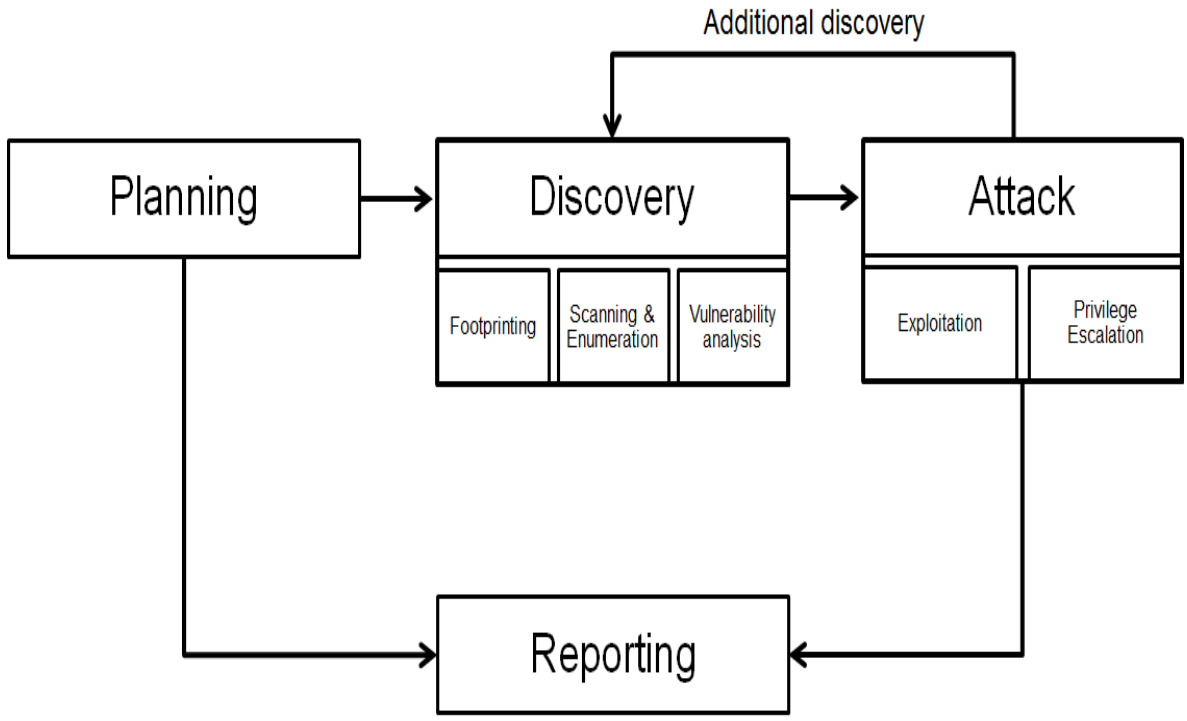


Abb. 2: Ablauf eines Penetration-Tests[8]

1. Planning

In der ersten Phase erfolgt die Zielsetzung und Planung, in welcher vor allem der Umfang des Tests bestimmt wird. Zusätzlich müssen verschiedene Vereinbarungen, wie beispielsweise ein

[7] Vgl. Thomas, S. (2007); http://www.trivadis.com/uploads/tx_cabagdownloadarea/Penetration_Final_070802.pdf

[8] Mit Änderungen entnommen aus: Saindane, M. (o.J.), S.3

Non Disclosure Agreement[9], geschlossen und unterzeichnet werden. Weiterhin wird die Strategie- und Vorgehensdefinition des Testing Teams bestimmt.

2. Discovery

Das Testen beginnt in dieser Phase. Alle nötigen Informationen werden gesammelt und zusammengefasst. Der erste Schritt innerhalb der Phase ist das Footprinting. Hier wird Recherche betrieben, um jegliche Informationen über die Organisation, in welcher das Penetrationtesting stattfindet, und deren Systeme herauszufinden. Nachfolgend kommt die Scanning & Enumeration Phase in der alle möglichen Systeme, offene Ports oder laufende Services der Zielorganisation auf Lücken hin untersucht werden. In dieser Phase wird viel ausprobiert und getestet. Nachdem das Zielsystem identifiziert und alle nötigen Informationen aus vorherigen Phasen konsolidiert wurden, erfolgt die Vulnerability Analysis Phase. In dieser hat das Penetrationtest-Team das Ziel alle möglichen Schwachstellen in den verschiedenen Zielsystemen aufzudecken.

3. Attack

Die “Attack”-Phase stellt die Hauptphase des Penetrationtests dar. Diese Phase beginnt mit der “Exploitation”, dem Ausnutzen der Schwachstellen. Hier wendet das Penetrationtest-Team verschieden Methoden und Techniken an, um Schwachstellen auszunutzen und sich so Zugriff zu dem Zielsystem zu verschaffen. Das Ausnutzen der Schwachstellen führt nicht immer direkt zum Ziel. Oft geschieht es, dass der Zugriff auf Administrationsrechte nicht vollständig erfolgt. An dieser Stelle befindet sich der Prozess in der “Privilege Escalation” Phase. Tritt dieser Fall ein, muss eine tiefergehende Recherche betrieben werden, wodurch erneut die “Discovery”- Phase durchlaufen wird, wie der Pfeil in Abbildung 2 zeigt.

4. Reporting

Das Reporting stellt die vierte Phase des Penetrationtest dar, kann aber auch parallel zu den anderen Phasen auftreten. In dieser Phase werden sowohl alle rechtlichen Bestimmungen und Informationen aus der “Planning”-Phase gesammelt, als auch Ergebnisse und sonstige Erkenntnisse nach Beendigung der “Attack”-Phase im Report mit aufgenommen werden.

[9] Ein Non disclosure agreement hilft, nicht-öffentliche Informationen mit einem potentiellen Investor zu teilen, da es die entsprechend notwendige Verschwiegenheit zusichert. Vgl. Gründerszene (o.J.), http://www.gruenderszene.de/lexikon/begriffe/non-disclosure-agreement-nda

3. Bewertungskriterien

Zur Bewertung der im Rahmen dieser Arbeit ausgewählten Honeypots, werden im Folgenden ausgewählte Kriterien zur Evaluation dieser festgelegt und im Detail erläutert. Beginnend mit den OWASP (Open Web Application Security Project) Top 10 Sicherheitslücken, über zu weiteren Kriterien zur Bewertung eines Honeypots zu Trainingszwecken.

3.1 Sicherheitslücken

Die bewusste Existenz von Sicherheitslücken ist eine grundlegende Eigenschaft von Honeypots. Um zu bewerten ob ein Honeypot eine ausreichende Menge und die richtigen Arten von Sicherheitslücken bietet wird in dieser Arbeit jeder diskutierte Honeypot auf die OWASP Top 10[10] geprüft. Das Open Web Application Security Project[11] ist eine offene, internationale, herstellerunabhängige Community mit dem Ziel Organisationen dabei zu unterstützen Web Applikationen so zu konzipieren, entwickeln, erwerben, betreiben und warten, dass sie sicher und vertrauenswürdig sind.[12] Bei der Auswahl der Sicherheitslücken wird nicht nur nach Häufigkeit sondern nach Risiko d. h. einer Summe aus den folgenden Faktoren zusammengestellt:

- Angriffsvektor - Wer hat die Möglichkeit für den Angriff? Welche Informationen stehen den Angreifern zur Verfügung?
- Verbreitung - Wie häufig tritt die Sicherheitslücke auf?
- Auffindbarkeit - Wie leicht ist es für einen Angreifer die Lücke zu finden?
- Technischer Auswirkung - Welche Möglichkeiten hat der Angreifer, falls die Lücke ausgenutzt wird?

Die aktuellen OWASP Top 10 sind weit verbreitet und akzeptiert und decken sich größtenteils mit der anerkannten Liste der 25 größten Softwarefehler die jährlich von MITRE[13] und SANS[14] herausgegeben wird.[15] Darüber hinaus werden die OWASP Top 10 von der US Federal Trade Commission und der Defense Information Systems Agency des Verteidigungsministeriums als Richtlinie vorgegeben.

[10] Vgl. OWASP (2012a), http://owasp.de/top10
[11] Vgl. OWASP (2012b), https://www.owasp.org
[12] Vgl. OWASP (2012c), https://www.owasp.org/index.php/About_OWASP
[13] Eine Non-Profit-Organisation die eine Vielzahl an US-Behörden (v. A. Verteidigung) unterstützt und aus dem Massachusetts Institute of Technology (MIT) hervorging. Vgl. MIT (2012), http://en.wikipedia.org/wiki/Massachusetts_Institute_of_Technology
[14] **S**ysAdmin, **A**udit, **N**etworking, and **S**ecurity - ein auf Internet-Sicherheit spezialisiertes Unternehmen
[15] Vgl. MITRE (2011), http://cwe.mitre.org/top25/index.html#AppendixD

Für die Bewertung der Honeypots ist dabei ausschlaggebend wie viele der OWASP Top 10 Sicherheitsprobleme in einem Honeypot vorhanden sind. Im Folgenden wird ein kurzer Überblick über diese gegeben, dabei wird jeweils die offizielle prägnante Erklärung verwendet.[16]

3.1.1 Injection

“Injection-Schwachstellen, wie beispielsweise SQL-, OS- oder LDAP-Injection treten auf, wenn nicht vertrauenswürdige Daten als Teil eines Kommandos oder einer Abfrage von einem Interpreter verarbeitet werden. Ein Angreifer kann Eingabedaten dann so manipulieren, dass er nicht vorgesehene Kommandos ausführen oder unautorisiert auf Daten zugreifen kann.”

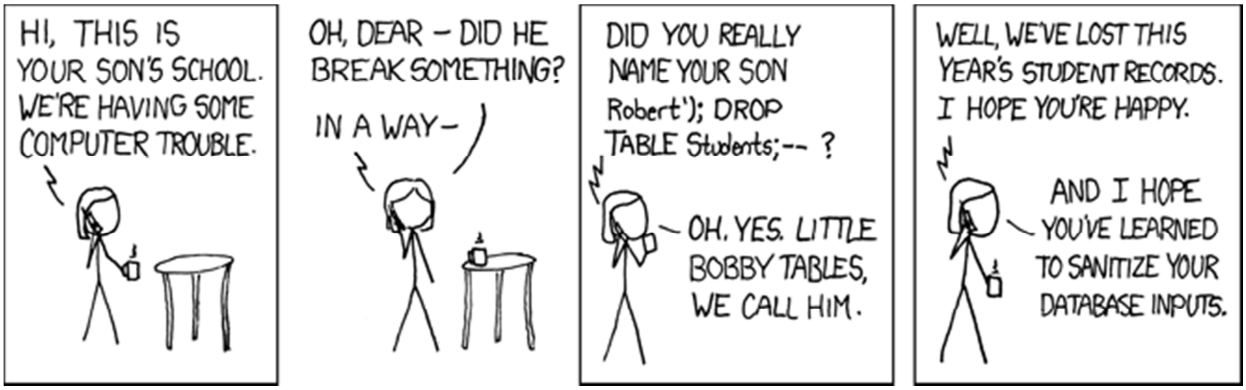


Abb. 3: Beispiel SQL-Injection[17]

Das Ausnutzen einer SQL-Injection Lücke wird in Abbildung 3 an einem Vornamen, dessen einfügen in das anfällige Schülerverwaltungssystem zur Löschung der Schülerdatenbank führt, illustriert.

3.1.2 Cross-Site Scripting (XSS)

“XSS-Schwachstellen treten auf, wenn eine Anwendung nicht vertrauenswürdige Daten entgegennimmt und ohne entsprechende Validierung und Kodierung an einen Webbrowser sendet. XSS erlaubt es einem Angreifer Scriptcode im Browser eines Opfers auszuführen und somit Benutzersitzungen zu übernehmen, Seiteninhalte zu verändern oder den Benutzer auf bösartige Seiten umzuleiten.“

[16] OWASP (2012a), http://owasp.de/top10
[17] Vgl. Munroe, R. (2007), http://xkcd.com/327/

3.1.3 Fehler in Authentifizierung und Session Management

“Anwendungsfunktionen, die die Authentifizierung und das Session Management umsetzen, werden oft nicht korrekt implementiert. Dies erlaubt es Angreifern, Passwörter oder Sessiontoken zu kompromittieren oder die Schwachstellen so auszunutzen, dass sie die Identität anderer Benutzer annehmen können.“

3.1.4 Unsichere direkte Objektreferenzen

“Unsichere direkte Objektreferenzen treten auf, wenn Entwickler Referenzen zu internen Implementierungsobjekten, wie Dateien, Ordner oder Datenbankschlüssel von außen zugänglich machen. Ohne Zugriffskontrolle oder anderen Schutz können Angreifer diese Referenzen manipulieren, um unautorisiert Zugriff auf Daten zu erlangen.“

3.1.5 Cross-Site Request Forgery (CSRF)

“Ein CSRF-Angriff bringt den Browser eines angemeldeten Benutzers dazu, einen manipulierten HTTP- Request an die verwundbare Anwendung zu senden. Session Cookies und andere Authentifizierungsinformationen werden dabei automatisch vom Browser mitgesendet. Dies erlaubt es dem Angreifer, Aktionen innerhalb der betroffenen Anwendungen im Namen und Kontext des angegriffen Benutzers auszuführen.“

3.1.6 Sicherheitsrelevante Fehlkonfiguration

“Sicherheit erfordert die Festlegung und Umsetzung einer sicheren Konfiguration für Anwendungen, Frameworks, Applikations-, Web- und Datenbankserver sowie deren Plattformen. Alle entsprechenden Einstellungen müssen definiert, umgesetzt und gewartet werden, da sie meist nicht mit sicheren Grundeinstellungen ausgeliefert werden. Dies umfasst auch die regelmäßige Aktualisierung aller Software, inklusive verwendeten Bibliotheken und Komponenten.“

3.1.7 Kryptografisch unsichere Speicherung

“Viele Anwendungen schützen sensible Daten, wie Kreditkartendaten oder Zugangsinformationen nicht ausreichend durch Verschlüsselung oder Hashing. Angreifer können solche nicht angemessen geschützten Daten auslesen oder modifizieren und mit ihnen weitere Straftaten, wie beispielsweise Kreditkartenbetrug, begehen.“

3.1.8 Mangelhafter URL-Zugriffsschutz

“Viele Anwendungen realisieren Zugriffsberechtigungen nur durch das Anzeigen oder Ausblenden von Links oder Buttons. Allerdings muss auch beim direkten Zugriff auf eine geschützte URL eine Prüfung der Zugriffsberechtigung stattfinden, ansonsten können Angreifer durch gezieltes Manipulieren von URLs trotzdem auf diese zugreifen.“

3.1.9 Unzureichende Absicherung der Transportschicht

“Viele Anwendungen stellen die Vertraulichkeit und Integrität von sensiblem Netzwerkverkehr nicht durch entsprechende Verschlüsselung und Authentisierung sicher. Wird Verschlüsselung eingesetzt, werden oft schwache Algorithmen, abgelaufene oder ungültige Zertifikate verwendet oder falsch eingesetzt.“

3.1.10 Ungeprüfte Um- und Weiterleitung

“Viele Anwendungen leiten Benutzer auf andere Seiten oder Anwendungen um oder weiter. Dabei werden für die Bestimmung des Ziels oft nicht vertrauenswürdige Daten verwendet. Ohne eine entsprechende Prüfung können Angreifer ihre Opfer auf Phishing-Seiten oder Seiten mit Schadcode um- oder weiterleiten.“

3.2 Community, Dokumentation und Support

Für Open Source Produkte gibt es in den meisten Fällen kein Unternehmen, das für Qualitäts- oder Sicherheitsmängel zur Verantwortung gezogen werden kann. Im Falle von Honeypots sind Sicherheitslücken selbstverständlich kein Problem, sondern ein gewünschtes Feature. Allerdings sollte bedacht werden, dass es für Nutzer trotzdem zu Problemen und Fragen kommen kann, wenn z.B. die Installation nicht fehlerfrei funktioniert, das Wiederherstellen eines kompromittierten Systems fehlschlägt oder das Ausnutzen der vorhandenen Sicherheitslücken nicht gelingt. An die Stelle eines Unternehmens rückt bei Open Source Software die Entwickler-Community. Speziell für den Aufbau eines Labors zur Fortbildung der eigenen Entwickler ist es natürlich relevant in wie weit Lernhilfen wie z.B. Tutorials und eine allgemeine Betreuung zur Nutzung offeriert wird. Bei der Bewertung der Honeypots wird folglich überprüft ob und in welchem Umfang die folgenden Funktionen, von der Community bereitgestellt werden:

1. Dokumentation
    a. Gibt es eine Installationsanleitung?
    b. Gibt es eine Anleitung oder ein Tutorial, das die ersten Schritte aufzeigt?

   c. Erwähnt die Dokumentation alle vorhandenen Sicherheitslücken ausreichend?
   d. Gibt es eine Anleitung die in die Weboberfläche integriert ist?
   e. Wie gut ist der Quellcode dokumentiert?

2. Fehlerbehebung
   a. Gibt es aktive Entwickler, die Fehler beheben können? (Wie aktiv ist die Community diesbezüglich?
   b. Gibt es einen öffentlichen Bugtracker? Wird dieser aktiv betreut?

3. Support bei Anwendungsproblemen
   a. Gibt es ein Forum?
   b. Gibt es einen Chat für schnelle Hilfe?

4. Anpassung des Honeypots an neue Sicherheitslücken
   a. Gibt es aktive Entwickler, die neue Sicherheitslücken einbauen können? Wie aktiv ist die Community diesbezüglich? Gibt es unterstützende Unternehmen?
   b. Alternativ, wie einfach ist es Sicherheitslücken selbst einzubauen oder bestehende zu ändern?

Aufgrund der Tatsache, dass im Bereich Web-Applikation Sicherheit der Großteil der vorhandenen Informationen in Englisch vorliegt, wird davon ausgegangen, dass der Umgang mit der englischen Sprache keine Probleme darstellen sollte. Bei der Bewertung wird daher nicht unterschieden, ob Dokumentation und Support von der Community in Deutsch oder Englisch angeboten werden.

3.3 Installation

Die Installation steht vom lateinischen Ursprung her allgemein für die Einrichtung eines bestimmten Objektes. Im Rahmen dieser Arbeit ist dieser Begriff als die initiale Konfiguration des Open Source Honeypots zum Einsatz z. B. in einem Penetrationtest Labor definiert. Da insbesondere in Open Source Projekten, wenn auch oft fälschlicherweise, eine große Vorkenntnis bei Anwendern vorausgesetzt wird, ist es wichtig, die Einfachheit und Nachvollziehbarkeit des Einrichtungsprozesses des Honeypots zu untersuchen und in die Gesamt-Bewertung als Kriterium mitaufzunehmen. Verschiedene Aspekte, die im Folgenden aufgeführt werden, kommen dabei insbesondere zum Tragen.

1. Existiert eine ausreichende Anleitung für die Installation und Konfiguration des Honeypots?[18] Diese kann von einem einfachen Text bis zu einer Schritt-für-Schritt Bildschirmfoto-Anleitung reichen. Auch ist bei der Anleitungsevaluierung die Aktualität der Ressourcen zu berücksichtigen. Zu beachten ist außerdem, ob es zusätzliche externe Quellen wie Foren, Wiki oder Blogs gibt, die den Installationsprozess beschreiben oder Hilfestellung geben.

2. Ist der Honeypot ohne weiteres in bestehende Strukturen zu integrieren oder setzt er bestimmte Betriebssysteme oder Programme voraus, auf denen er aufsetzt? Da diese Systemvoraussetzungen kommerzieller Art sein können (z.B. Windows für eine .NET Umgebung), sind diese Installationsvoraussetzungen im Vorhinein zu evaluieren. Außerdem sind die Systemanforderungen des Honeypots etwa in Bezug auf Hard- und Software zu evaluieren.

3. Existieren “frequently occurring errors”, also häufig auftretende Fehler, während des Installationsprozesses, welche von Benutzern gemeldet wurden? Auch können unbekannte Fehler bei einer Probeinstallation und -konfiguration in die Bewertung miteinfließen. Beispiel hierfür könnte die Abhängigkeit von bestimmten Bibliotheken sein, welche nicht ausreichend dokumentiert ist und umständlich manuell nachgetragen werden muss.[19] Schlimmstenfalls kann dies sogar die Installation des Honeypots komplett verhindern und somit ein weiteres Testen unmöglich machen.

4. Wie lange dauert die Installation voraussichtlich und wie viele manuelle Schritte sind vonnöten? Wie ist der Ressourcenaufwand in Bezug auf Zeit, Mensch und somit Geld? Gerade in Bezug auf den Labor-Einsatz wichtig, da es oft zu Neuinstallationen kommen kann.

5. Gibt es ein Installationsinterface, also eine GUI, oder ein Skript zur Vereinfachung des Installationsprozesses? Ist damit vielleicht sogar eine Dokumentation obsolet? Oder müssen viele Schritte auf Konsolenebene getätigt werden? Wenn ja, welche Vorkenntnisse sind nötig?

6. Kann es zu Konfigurationsschwierigkeiten kommen, beispielsweise zu komplizierte Netzwerkeinstellungen? Wie klar ist die Konfiguration? Ist sie überhaupt nötig oder fehlen grundlegende Konfigurationsmöglichkeiten?

7. Existieren sonstige Außergewöhnlichkeiten oder Einschränkungen insbesondere im Laufe der Konfiguration (Mitunter subjektive Einschätzung)?

[18] Vgl. Hecktor, R. u.a (2012), S.39
[19] Vgl. Mikhalenko, P. (2006), http://www.oreillynet.com/pub/a/sysadmin/2006/09/28/honeypots.html

3.4 Usability

Da der zu evaluierende Honeypot Entwicklern vornehmlich schnelle, motivierende und einfache Einblicke in Web-Sicherheit gewähren soll, ist eine einfache Gebrauchstauglichkeit ein wichtiger Punkt zur Bewertung des Open Source Honeypot-Programms. Inwiefern ein Programm intuitiv zu benutzen ist, ist natürlich immer eine subjektive Einschätzung des Betrachters, und kann nur bedingt durch objektive Kriterien begründet werden.[20] Jedoch bietet die Definition der ISO-Norm 9241-10 “Grundsätze der Dialoggestaltung”[21] Einblicke, wie Honeypots in Bezug auf Benutzbarkeit bewerten werden können. Die Hauptpunkte aus der Norm werden im Folgenden kurz zusammengefasst, für eine detaillierte Darstellung mit Beispielen empfiehlt sich weiterführend die direkte Lektüre der Norm.

3.4.1 Aufgabenangemessenheit

Aufgabenangemessenheit beschreibt die unterstützende Aufgabe von Dialogen und Schnittstellen, dem Benutzer die benötigten Funktionen einfach, ohne Probleme und schnell zur Verfügung zu stellen. Ein Honeypot gilt in dieser Arbeit als aufgabenangemessen, wenn vorhandene Dialoge, Menüs und sonstige Kontroll-Objekte schnell und einfach zu einem vom Benutzer gewünschten Ziel führen und nicht überladen sind. Bietet der Honeypot beispielsweise ein Menü, welches nicht überladen ist und schnell zu elementaren Funktionen und Sicherheitslücken führt, ist er mit gut zu bewerten.

3.4.2 Selbstbeschreibungsfähigkeit

Selbstbeschreibungsfähigkeit umfasst die Eigenschaft, dass das zu verwendende Programm selbsterklärend ist und Unverständlichkeiten wie Doppeldeutigkeiten vermeidet. Damit sind insbesondere Beschriftungen und Formulierungen gemeint. Sind die Kontrolleinheiten wie Knöpfe und Titel selbsterklärend und führen genau zum beschriebenen, ist er positiv zu bewerten, etwa wenn eine Schaltfläche mit “Sicherheitslücke XY” beschriftet ist und auch dorthin navigiert.

3.4.3 Steuerbarkeit

Steuerbarkeit erlangt ein Programm, wenn es vom Anwender ohne weiteres bedient werden kann, um zum gewünschten Ziel zu gelangen. Bei einem Honeypot wird dieses Kriterium positiv bewertet, wenn alle Funktionalitäten einfach über die Benutzeroberfläche aufgerufen werden können. Ist beispielsweise eine bestimmte Seite, auf der eine Sicherheitslücke erklärt wird, nur durch eine ma-

[20] Vgl. Hecktor, R/. u.a (2012), S.36
[21] EUROPÄISCHE NORM (1995), www.interactive-quality.de/site/DE/int/pdf/**ISO_9241**-10.pdf

nuelle Browser-Eingabe zu erreichen und nicht wie alle anderen über ein Menü, ist die Steuerbarkeit negativ zu bewerten.

3.4.4 Erwartungskonformität

Erwartungskonformität steht für die Einhaltung von allgemeinen Standards, so dass der Benutzer schon aus Erfahrungen mit anderen Programmen Kenntnisse anwenden kann. Bei Honeypots ist damit insbesondere gemeint, inwiefern die Simulation von Webseiten mit realistischen übereinstimmen und inwiefern sich die Steuerung von anderen Webseiten unterscheidet. Befindet sich beispielsweise das Menü auf der rechten Seite, ist das für den Nutzer ungewohnt und demnach in Hinblick auf Realismus negativ zu bewerten. Alternativ kann sich ein Honeypot auch von realistischen Webseiten unterscheiden, muss dann aber gemäß den Erwartungen des Benutzers an eine Lern-Applikation / E-Learning gemessen werden.

3.4.5 Fehlertoleranz

Fehlertoleranz beschreibt die Eigenschaft, fehlerhafte Eingaben des Benutzers zu erkennen oder zu akzeptieren, aber trotzdem zum gewünschten Ergebnis zu führen. Bei einem Honeypot bringt dieses Kriterium keinen Mehrwert, da gezielt Fehler provoziert und ausgenutzt werden sollen.

3.4.6 Individualisierbarkeit

Ein Programm ist individualisierbar, wenn es den Vorlieben des Benutzers entsprechend anpassbar ist. Im Rahmen dieser Arbeit ist dieses Kriterium positiv zu bewerten, wenn individuelle Konfigurationen vorgenommen werden können, z. B. Ein- und Ausschalten von Hinweisen oder Oberflächenanpassung.

3.4.7 Lernförderlichkeit

Lernförderlichkeit wird dadurch erreicht, dass das Programm den Benutzer etwa durch ein Tutorials o. Ä. das Lernen der Software erleichtert. Dieses Kriterium lässt sich gut auf Honeypots anwenden und zielt darauf ab, das Vorhandensein und den Inhalt von Hinweisen und Tutorials innerhalb des Programms zu analysieren und zu bewerten. Ein Honeypot beispielsweise, der jede Sicherheitslücke umfangreich erläutert und Hilfestellung gibt, fördert den Lernprozess ungemein und weist gezielt auf mögliche Sicherheitslücken hin.

Weitere Beispiele und genauere Umschreibungen können außerdem aus KommDesign[22] entnommen werden.

Diese Hauptkriterien lassen sich bedingt auch auf Honeypots anwenden, schließlich werden diese in Schulungen oder Workshops regelmäßig von Anwendern benutzt. Als Simulation von Web-Anwendungen müssen diese sowohl gut bedienbar im Front-End, als auch unter der Prämisse der Sicherheitslücken-Erprobung im Back-End gut bedienbar sein. Einzig das 5. Kriterium der Fehlertoleranz kann nicht ausschlaggebend für die Usability des zu evaluierenden Honeypots sein, denn ein Aspekt der Honeypots in dieser Arbeit ist, bewusst Fehler zu produzieren und zu anderen Ergebnissen als es der Web-Anwendungsdesigner gewünscht hat zu gelangen. Die sechs anderen Punkte jedoch sind gerade im Hinblick auf Anwendung im E-Learning, Blended Learning[23] oder Workshop-Bereich sinnvoll und sollten genauestens analysiert werden.

Darüber hinaus ist es für eine glaubwürdige Webseiten-Simulation wichtig, ein ansprechendes Design vorweisen zu können. Wenn die Weboberfläche des Honeypots nicht realistisch modelliert wurde, kann es Benutzern mitunter schwer fallen, eine Parallele des Honeypots zu realen Web-Applikationen, wie etwa der Web-Präsenz des eigenen Unternehmens, zu ziehen. Dies kann dann zu einer mangelnden Ernsthaftigkeit oder Berücksichtigung der möglichen Sicherheitslücken seitens der Workshopteilnehmer führen. Auch wirkt sich ein gutes Design der Honeypot-Anwendung motivierend auf den Benutzer aus. Daher wird als zusätzliches Bewertungskriterium in dieser Arbeit die mitunter subjektive Einschätzung des Designs der Honeypot Web-Anwendung neben den 6 Punkten der ISO-Norm berücksichtigt.

Eine weitere Evaluationsmöglichkeit bietet das “DATech-Prüfverfahren für die Konformitätsprüfung interaktiver Systeme auf Grundlage von DIN EN ISO 9241, Teile 11 und 110”,[24] welches weitere Normen und Werkzeuge bietet, die Benutzerfreundlichkeit zu bewerten. Im Rahmen der Arbeit wird jedoch ausschließlich auf die ISO-Norm Bezug genommen, da die dort gegebenen Kriterien ausreichend für die Bewertung der Benutzerfreundlichkeit sind und eine Integration des DATech-Prüfverfahren den Rahmen in diesem Bewertungsbereich übersteigt.

[22] Vgl. Wirth, T. (2009), http://kommdesign.de/texte/din.htm
[23] Blended Learning ist integriertes Lernen, also “die Mischung von Präsenz- und Online-Anteilen in der Lehre” O.V. (2011), http://www.e-learning.fu-berlin.de/lehren_mit_neuen_medien/einsatzszenarien/blended_learning/index.html
[24] DATech (2009), http://www.datech.de/share/files/Leitfaden-Usability.pdf

3.5 Zusätzliche Features

In dieser Kategorie werden Features erfasst, die zu Beginn der Evaluation nicht erwartet wurden aber einen Mehrwert für die Nutzer bieten können.
Da es keine Norm gibt, die vorschreibt welche Funktionen ein Honeypot bieten muss, ist zu erwarten, dass jeder Honeypot andere Features enthält. Die zusätzlichen Features können auch den Ausschlag geben, wenn bereits ein sehr konkreter Use-Case vorhanden ist, der besondere Funktionen erfordert.
Denkbar sind etwa zusätzliche Features wie, besondere Einfachheit der Konfiguration, außerordentliche Anleitungen und Hilfestellung als auch unerwartete Funktionalitäten wie beispielsweise den Honeypot per Knopfdruck in den Installationszustand zu versetzen.

4. Honeypot-Analysen

Im folgenden Kapitel werden die Ergebnisse der Untersuchungen der für das Projekt relevanten Honeypots vorgestellt. Die hier analysierten Honeypots stellen eine Auswahl aus einer größeren Anzahl an Honeypots dar.

Selektion relevanter Honeypots

Aus erschöpfender Recherchearbeit ergab sich, dass ca. 15 frei verfügbare Open Source Honeypots existieren. Die meisten dieser Honeypots konnten direkt aussortiert werden, da sie nicht mehr unterstützt werden oder sich in einem unfertigen Zustand befinden. Ein Beispiel für eine Beta Version ist der Badstore.[25]

Um die Studie in der benötigten Tiefe umsetzen zu können wurde eine Teilmenge der Gesamtzahl an verfügbaren Honeypots ausgewählt. Dadurch wird gewährleistet, dass die Untersuchungen in der entsprechenden technischen und detaillierten Tiefe stattfinden können.

Die in Tabelle 1 gelisteten Honeypots fanden in der Vorauswahl Berücksichtigung. Aus der Tabelle ist erkennbar, welche fünf Honeypots für eine detailreiche Analyse ausgewählt wurden. Die Gründe für einen Ausschluss aus der zweiten, tieferen Untersuchung sind ebenfalls in der Tabelle aufgeführt.

Nr.	Bezeichnung	In Hauptuntersuchung?	Ausschlussgrund
1	Acunetic Online	Nein	Kommerziell und keine Dokumentation verfügbar
2	CrackMeBank	Nein	Kommerziell und keine Dokumentation verfügbar
3	DamnVulnerableWebApplication (DVWA)	Ja	
4	Gruyere	Ja	
5	Ghost	Nein	Eingeschränkte Funktionalität, kaum Dokumentation vorhanden
6	HacMe Series	Ja	
7	Insecure Web App	Nein	Blackbox-Honeypot mit sehr geringer Dokumentation, aktuellerer Honeypot (WebGoat) vom selben Hersteller verfügbar

[25] Vgl. o.V. (o.J.), http://www.badstore.net/

8	Mutillidae	Ja	
9	WebGoat	Ja	
10	Web Maven	Nein	Wird nicht mehr unterstützt vom Hersteller, er verweist selbst auf andere Honeypot-Projekte
11	ZAP-Wave	Nein	Stark auf ein bestimmtes Proxy-Tool zugeschnitten, daher eingeschränkte Features

Tab. 1: Aufzählung möglicher Honeypots für die Bewertung

Somit sind die fünf Honeypots, die weitergehend untersucht werden, folgende:

1. **DamnVulnerableWebApplication**
2. **Gruyere**
3. **HacMe Series**
4. **Mutillidae**
5. **WebGoat**

Neben den erwähnten Open Source Honeypots existieren vergleichbare, proprietäre Honeypots. Da diese nur unter Zuhilfenahme von finanziellen Mitteln als Applikation für einen Workshop o. Ä. möglich wäre und einen langen Prozess mit sich führt, können die proprietären Honeypots nur erwähnt und nicht in die Hauptanalyse mit einbezogen werden. Die entdeckten proprietären Angebote setzen eine enge Kooperation zwischen Honeypot-Anbieter und dem Kunden voraus und gehen häufig mit Schulungen oder verpflichtenden Trainings einher.
Ein Beispiel für eine proprietäre Lösung ist das “Offensive Security Intelligent Penetration Testing Lab” von Offensive Security. In diesem Modell können über Web-Applikationen hinausgehend auch andere Labore, zum Beispiel Infrastrukturen wie Netzwerke, virtuell bezogen werden. Das Unternehmen konzipiert das zu beziehende Netzwerk bzw. die Web-Applikation basierend auf den individuellen Kundenwünschen, wie beispielsweise Schwierigkeit des Labors, Umfang oder Support.

5. Produktvergleich

Um die in Kapitel 4 ausgewählten Honeypots systematisch zu bewerten wurde eine Bewertungsmatrix gemäß der Kriterien aus Kapitel 3 entworfen. Jedes der fünf Kriterien wird darin mit seinen Unterkriterien und einer Gewichtung gelistet. Die Gewichtung erlaubt es dabei sowohl den Autoren als auch Lesern dieser Arbeit Schwerpunkte zu setzen um Kriterien zu priorisieren. Die Bewertungsmatrizen werden daher auch in digitaler Form bereitgestellt, um individuelle Anpassungen vorzunehmen.
Die von den Autoren vorgenommene Gewichtung stellt sich dabei folgendermaßen zusammen:

- Sicherheitslücken 25%
- Community, Dokumentation & Support 30%
- Usability 20%
- Installation 15%
- Zusätzlich Features 10%

Die Gewichtungen wurden dabei so gewählt, dass das Gesamtergebnis einen allgemeinen Eindruck, wie er für den Aufbau eines neuen Labors nötig ist, erlaubt. Eine stärkere Priorisierung der Sicherheitslücken und Dokumentation ist durchaus denkbar, sofern ein gewisses Vorwissen besteht. Die ausgefüllten Bewertungsmatrizen können in Anhang 1 bis Anhang 5 dieser Arbeit eingesehen werden. In den folgenden Kapiteln werden die untersuchten Honeypots jeweils vorgestellt und die wichtigsten Erkenntnisse aus der Bewertung erläutert.

5.1 DVWA

“DVWA is a PHP/MySQL web application that is damn vulnerable.” Die Abkürzung DVWA steht für Damn Vulnerable Web Application und ist ein Open Source Projekt, welches von der kommerziellen Firma Random Storm unterstützt wird. Ziel dieses Projektes ist es, IT-Sicherheitsspezialisten dabei zu unterstützen, ihre Fähigkeiten und Methoden in einer legalen Umgebung unter Beweis zu stellen. Web-Entwicklern wird gezeigt, wie wichtig IT-Sicherheit ist und wie einfach vorhandene Sicherheitslücken ausgenutzt werden können.[26] Im Folgenden gibt Abbildung 4 einen Eindruck wie der Honeypot DVWA aufgebaut ist.

[26] Vgl. DVWA (2012a), http://www.dvwa.co.uk/

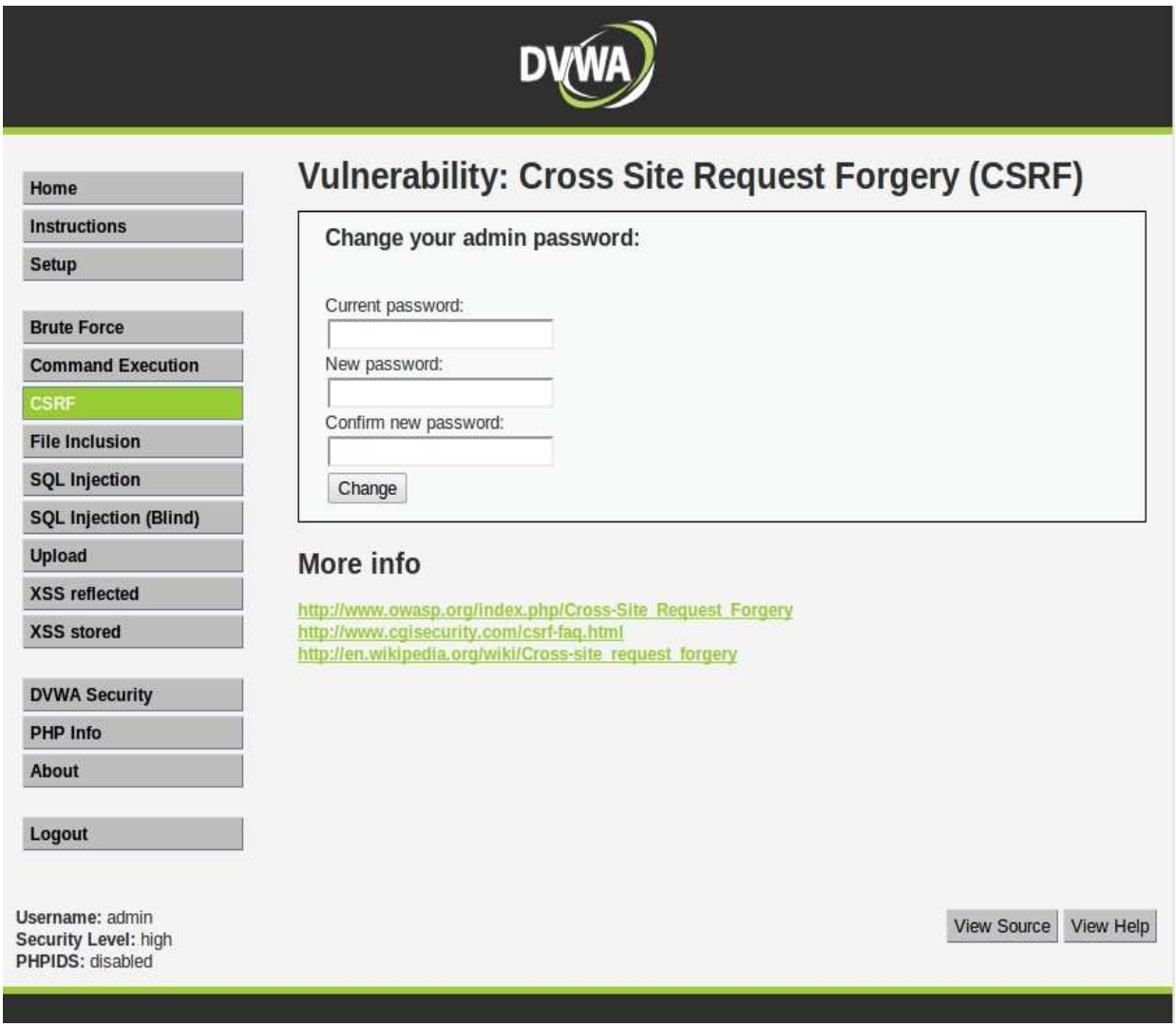


Abb. 4: DVWA Startseite

Im Folgenden werden die Hauptaussagen, welche aus der Auswertung des DVWA-Projekts hervorgehen, zusammengefasst. Zusätzlich findet sich in Anhang 1 eine ausführliche Bewertungsmatrix des Projektes.

Bei der Bewertung des Projekts konnten lediglich vier der Top 10 von OWASP bestimmten Sicherheitslücken gefunden werden. Diese waren SQL-Injection, XSS, CSRF und sicherheitsrelevante Fehlkonfiguration. Im Bereich der Dokumentation und Community ist insbesondere ein Youtube-Video positiv anzumerken, welches eine Anleitung zu den ersten Schritten im Umgang mit DVWA zeigt. Durch das Vorhandensein dieses Anleitungsvideos hat DVWA in der Bewertung die maximale Punktezahl (3 von 3) erreicht. Weiterhin wird lediglich eine von vier Sicherheitslücke exemplarisch erklärt. Jedoch sind Verweise auf Erklärungen der jeweiligen Sicherheitslücken und eine Quellcode-Ansicht in der Weboberfläche integriert. Aus den Erklärungen wird ersichtlich, dass schon länger nicht mehr an dem Quellcode gearbeitet wurde (letzte allgemeine Änderung 14. Oktober 2011),[27] lediglich vereinzelte Änderungen die sich auf Grund von Fehlern ergeben haben

[27] Vgl. DVWA (2012b), http://code.google.com/feeds/p/dvwa/svnchanges/basic

wurden im März diesen Jahres umgesetzt. Infolgedessen konnten in diesem Fall nur zwei von drei Punkten bei der Bewertung vergeben werden.

In Bezug auf die Community zeigt die Recherche, dass zwei dedizierte Personen als aktive Entwickler eingetragen sind.

Positiv hervorzuheben ist das Bugtracking, welches gut kommentiert und übersichtlich dargestellt wird. Hinsichtlich der Community ist weder ein Forum noch ein Chat auffindbar, weshalb in diesem Fall keine Punkte vergeben werden konnten. Weiterhin können lediglich zwei aktive Entwickler genannten werden, welche auf Anfragen reagieren und vermutlich in der Lage sind neue Sicherheitslücken zu implementieren. Allerdings fand in den letzten vier Monaten keine Reaktion auf diese statt.

Im Bereich der Installation hat sich gezeigt, dass DVWA ohne Aufwand in die bestehende Umgebung zu integrieren ist und nur wenige Konfigurationen notwendig sind, um den Honeypot zu installieren.

Hinsichtlich der Usability hat das Open Source Projekt in der Bewertung weitestgehend gut abgeschnitten. Was die Aufgabenangemessenheit betrifft ist ein hohes Lernpotenzial zu verzeichnen.

In Bezug auf die Steuerbarkeit wird der Benutzer klar durch die Anwendung geführt. Die Individualisierbarkeit wird in Abbildung 5 veranschaulicht.

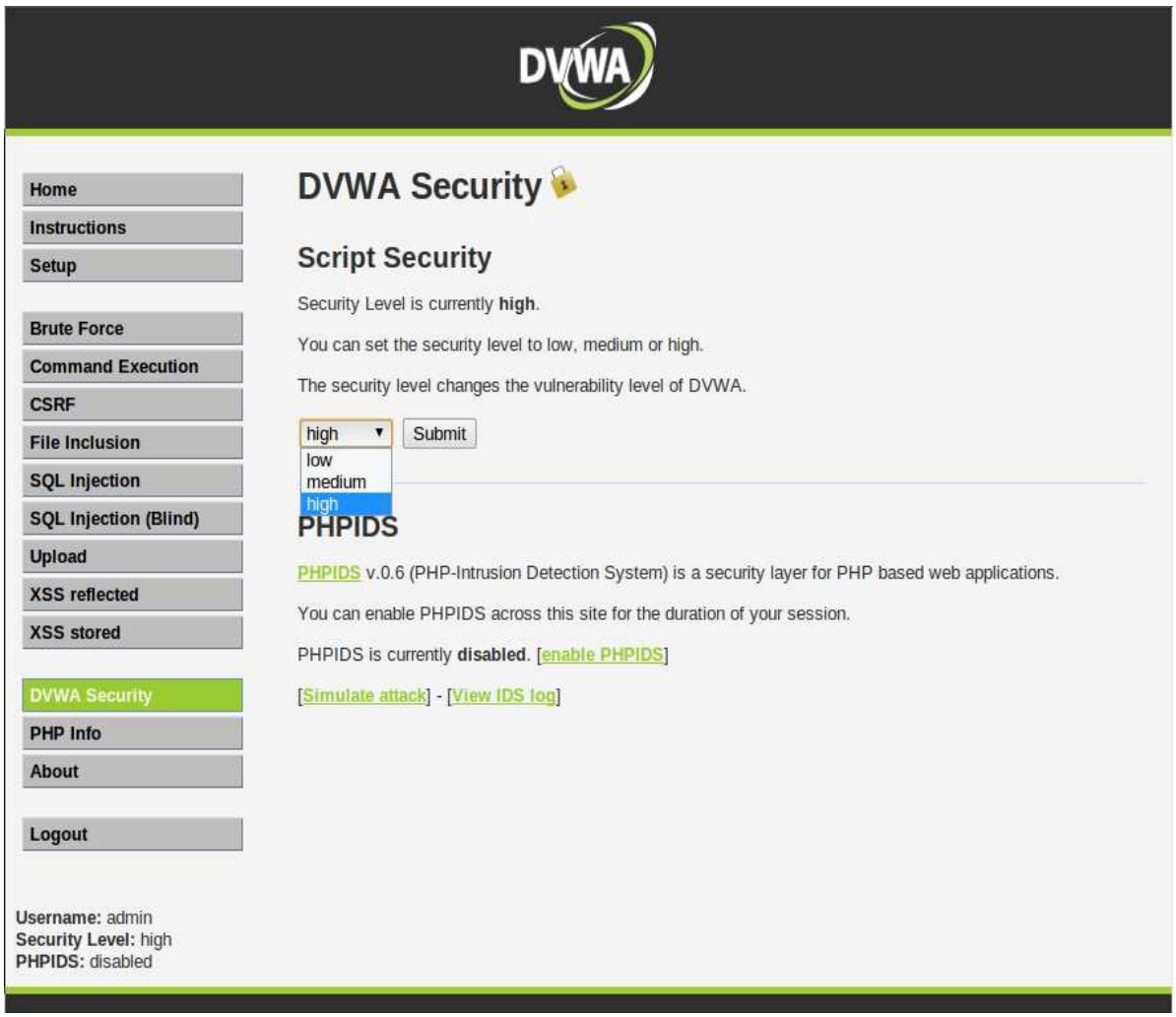


Abb. 5: DVWA Security Auswahlmöglichkeiten

Der Benutzer kann hier zwischen verschiedenen Schwierigkeitsstufen auswählen, welche bei “low” beginnen, über zu “medium” gehen und in “high” gipfeln. Die Abstufungen werden mit der vollen Punktzahl bewertet. Auch die Lernförderlichkeit ist sehr hoch, da dem User die Lösungen nicht vorgegeben werden, sondern dieser sich über gegebene Referenzen Wissen zum Teil selbst erarbeiten muss.

Resümierend ist das DVWA-Honeypot als mittelmäßig zu bewerten. Von insgesamt 100% konnten 59% erreicht werden, insbesondere die Beständigkeit des Projektes ist fragwürdig, da nur wenige Entwickler vorhanden und kaum aktiv waren in den letzten zwei Jahren. Trotz der Tatsache, dass das Projekt von Random Storm betreut wird, ist die Aktivität nur mäßig. Aller Voraussicht nach erzielt das kommerzielle Unternehmen keinen finanziellen Mehrwert durch die Förderung des Projektes, weshalb dieses nicht aktiv vorangetrieben wird.

5.2 Gruyere

Gruyere ist eine von Google Labs und der Google Code University entwickelter Honeypot, der vor der Veröffentlichung unter dem Projektnamen "Web Application Exploits and Defenses" für Google-interne Schulungen benutzt wurde. Zugriff auf die Twitter-ähnliche Anwendung kann entweder online auf den Google-Servern stattfinden[28] oder auf einer lokalen Instanz. Eine Installation ist dabei unnötig - Gruyere ist eine eigenständige Anwendung, die selbst einen Server startet. Die Wahl der Programmiersprache ist mit Python dabei eher ungewöhnlich, macht Gruyere aber zu einem interessanten Kandidaten. Bei der Erstellung einer lokalen Instanz nimmt daher der Download der Gruyere-Dateien die meiste Zeit in Anspruch. Das Starten beschränkt sich auf den Aufruf einer Datei - damit ist Gruyere bereit gehackt zu werden. Aus Sicherheitsgründen reagiert der Gruyere-Server standardmäßig nur auf lokale Anfragen, um auch "externen Angreifern" Zugriff zu gestatten muss die Datei "gruyere.py" per Hand bearbeitet werden. In diesem Fall ist selbstverständlich sicherzustellen, dass der Server nur für organisationsinterne "Angreifer" erreichbar ist.

Abb. 6: Gruyere Startseite

Obwohl Google die OWASP Top 10 in der offiziellen Gruyere Dokumentation nicht erwähnt, enthält Gruyere 8 der Lücken. Diese 8 und weitere nicht in den Top 10 enthaltenen Lücken werden jeweils ausführlich erklärt. Dabei wird unterschieden in Lücken, die mit oder ohne Code-Kenntnisse (White- vs. BlackBox) ausgenutzt werden können, sowie für jede Lücke erst Tipps und dann eine Anlei-

[28] Vgl. Gruyere (2010), http://google-gruyere.appspot.com/

tung zum Ausnutzen geliefert. Zusätzlich wird angegeben, wie ein möglicher Fix der Sicherheitslücke aussehen könnte. Tipps zum Ausnutzen können dabei separat ausgeklappt werden um versehentliches Lesen der Lösung zu verhindern. Die Schritte zum Ausnutzen einer Lücke werden dabei sehr detailliert angegeben, falls die Onlineversion Gruyeres genutzt wird, werden URLs sogar direkt an die jeweilige Instanz angepasst.

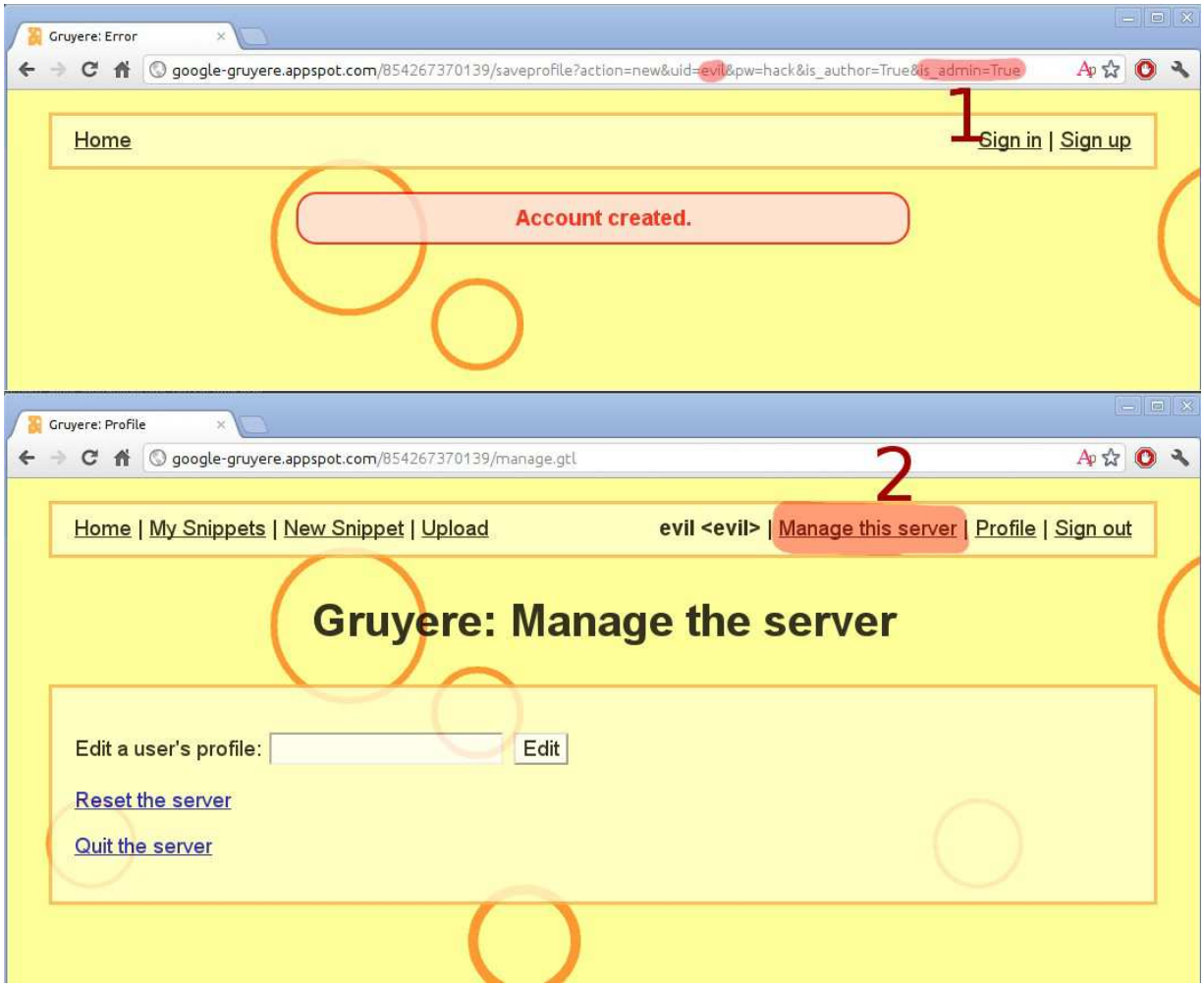


Abb. 7: Illustration einer Sicherheitslücke vom Typ „Mangelhafter URL-Zugriffsschutz“ in Gruyere[29] [30]

Die Betreuung des Projektes ist nicht offiziell beendet worden, es ist allerdings nicht erkennbar, dass das Projekt noch weiterentwickelt wird; so gibt es weder neue Versionen noch ein Blog das über mögliche Änderungen berichtet. Seit der Veröffentlichung im Juni 2010 hat sich allerdings auch kein dringender Änderungsbedarf ergeben. Für Probleme ist ein Diskussionsforum zwar vorhanden aber so gut wie nie genutzt worden, was sich allerdings durch die ausführliche Dokumentation erklären lässt. Das Einbauen oder Schließen von Sicherheitslücken ist für versierte Nutzer

[29] Schritt 1: Erstellung eines neuen Nutzers mit manipulierter URL. Schritt 2: Der neu erstellte Nutzer hat nun die Administratorrechte und kann z.B. den Server stoppen.
[30] Die Hinleitung zum Ausnutzen dieser Lücke lässt sich unter http://google-gruyere.appspot.com/part3#3__elevation_of_privilege begutachten.

sicherlich möglich und wird durch ausführliche Kommentare im Quellcode sowie die Erklärungen in der Onlinedokumentation erleichtert, es gibt jedoch kein Tutorial, das eine konkrete Änderung aufzeigt.

Die Oberfläche Gruyeres ist minimalistisch und ungewöhnlich bunt, und wirkt daher eher wie ein Spiel- als ein Übungsplatz für Web-Entwickler. Tipps zum Ausnutzen der Sicherheitslücken finden sich auf der Oberfläche genauso wenig wie Einstellungsmöglichkeiten z. B. bezüglich der Schwierigkeit.
Alle genannten Aspekte finden sich nochmal zusammengefasst in einer Bewertungsmatrix in Anhang 2.

Zusammenfassend lässt sich sagen, dass Gruyere zwar viele Sicherheitslücken abdeckt und gut erklärt, aber bezüglich der Nutzungsfreundlichkeit Defizite hat. Als Erweiterung eines bereits bestehenden Labors oder für die Fortbildung von Pythonentwicklern hat es selbstverständlich trotzdem eine Daseinsberechtigung. Die Onlineversion bietet darüber hinaus eine Möglichkeit Gruyere ohne eigenen Aufwand zu testen oder damit zu arbeiten. Dies ist eine sehr attraktive Möglichkeit um sich einen schnellen Überblick über Gruyere zu verschaffen ohne selbst das Risiko eingehen zu müssen eine stark unsichere Applikation auf einem internen Server ausführen zu müssen.

5.3 HacMe Series

Die HacMe Series von Foundstone umfasst eine Reihe von absichtlich unsicher gestalteten Applikationen. Foundstone gehört mittlerweile zum Unternehmen McAfee. Da die Applikationen zum Teil annähernd 8 Jahre alt sind, konzentriert sich diese Abhandlung auf den aktuellsten Teil der Serie, das HacMe Casino.[31] Es existieren zwar neuere Anwendungen, allerdings setzen diese einen anderen Fokus wie z. B. HacMe Bank (Android), welche auf mobile Sicherheit abzielt. Foundstone hat seine Serie mit verschiedenen Frameworks umgesetzt, das HacMe Casino ist dabei in Ruby on Rails geschrieben. Windows XP wird als Plattform zwingend benötigt, woran sich das Alter der Anwendung erahnen lässt. HacMe Casino ist eine BlackBox Applikation, die den Nutzer die Sicherheitslücken ohne integrierte Anleitung entdecken und ausnutzen lässt und den Quellcode nicht beim primären Download mitliefert.
Die Untersuchung des HacMe Casinos hat ergeben, dass es sich um eine mittelmäßige Anwendung handelt. Dies ist hauptsächlich auf die fortgeschrittene Schwierigkeit der Anwendung (aufgrund von wenig integrierter Dokumentation und Anleitung) als auch auf die Kompatibilitätsprobleme mit neueren Betriebssystemen zurückzuführen. Sofern mit Windows XP gearbeitet wird, ist die

[31] Vgl. Hacme Casino (2012), http://www.mcafee.com/us/downloads/free-tools/hacme-casino.aspx

Installation der Anwendung sehr einfach, da sie in direkt ausführbarer Form geliefert wird (.exe Datei). Die Anwendung muss nur ausgeführt werden und wird wie eine herkömmliche Windows-Anwendung in wenigen Schritten bis zur Fertigstellung “durchgeklickt”, was i. d. R. in unter 5 Minuten erfolgt. Die Dokumentation des Hacme Casinos, welche Installation und intentionale Sicherheitslücken beinhaltet, ist in einem PDF Dokument auf der McAfee Website auffindbar. Neben der offiziellen Dokumentation existieren diverse Youtube-Videos, in denen verschiedene Sicherheitslücken erklärt und ausgenutzt werden. Wird allerdings eine andere Plattform als Windows XP genutzt, kann es zu erheblichen Komplikationen bei der Ausführung kommen. Die offiziell dokumentierten Sicherheitslücken umfassen nicht alle vorhandenen Sicherheitslücken, außerdem entsprechen sie nicht in vollem Umfang den OWASP Top 10 sondern decken lediglich 7 von 10 ab.
Interessant an dem HacMe Casino sind durchführbare Cross-Site-Scripting-Attacken. Diese sind hier gut durchführbar, da mehrere Nutzer im Casino angemeldet sein können und die Auswirkungen sehr plastisch erlebbar sind. So kann ein von einem Angreifer geschickter Link, der von einem eingeloggten Nutzer angeklickt wird, dazu führen, dass der ahnungslose Nutzer all sein “Online-Geld” zum Angreifer überweist.

Abb. 8: Startbildschirm des HacMe Casinos

Das HacMe Casino stellt die Imitation einer echten Web-Anwendung dar und verzichtet daher auf integrierte Hinweise und Informationen zu den einzelnen Sicherheitslücken. Dies erschwert das Erkunden der Anwendung für weniger geübte Nutzer. Nutzer mit Sicherheitskenntnissen können es als Herausforderung betrachten, diese reale Anwendung wie in einem lebensnahen Szenario

auf Schwachstellen zu untersuchen und diese auszunutzen. So bietet die Anwendung wie in Abbildung 9 sichtbar viele Optionen, die in einem Online-Casino ebenfalls auffindbar sind.

Abb. 9: HacMe Casino Optionsmenü (Eingeloggt)

Obwohl McAfee die HacMe Series frei zur Verfügung stellt, scheint kein aktiver Support für die Produkte vorhanden zu sein. Die Recherche nach aktiven Foren und anderen Anlaufstellen für Supportanfragen ergab, dass es öffentlich keine direkte Anlaufstelle für Fragen, Hinweise und Probleme mit der HacMe Series gibt.
Im Detail lassen sich alle genannten Bewertungskriterien zu HacMe in Anhang 3 finden.

Ist jedoch etwas Vorkenntnis beim Nutzer vorhanden, so bietet das HacMe Casino eine gelungene Immersion in einem “Real-Life” Szenario. Hierzu tragen das Design und das realistische Verhalten der Anwendung bei.
Zusammenfassend handelt es sich bei dem HacMe Casino um einen etwas veralteten Honeypot, der für erfahrene Nutzer nach Startbemühungen (Windows XP Maschine real oder virtuell aufsetzen) durchaus nützlich und lehrreich sein kann.

5.4 Mutillidae

Mutillidae ist ein von Adrian Crenshaw und Jeremy Druin entwickelter Honeypot, welcher auf PHP basiert. Für die Installation wird ein Webserver mit PHP-Unterstützung und eine MySQL Datenbank benötigt. Mutillidae beinhaltet ebenfalls die OWASP Top 10[32] Sicherheitslücken und hilft mit optional einschaltbaren Hinweisen dabei, diese auszunutzen.
Im Folgenden werden alle Bewertungskriterien und der Erfüllungsgrad dieser im Projekt Mutillidae näher erläutert. Eine ausführliche Übersicht findet sich zusätzlich in Anhang 4.

Die Dokumentation des Honeypots im Allgemeinen ist gelungen, so existieren ausführliche Anleitungen und ein eigener YouTube-Kanal dabei, Sicherheitslücken zu erkennen und auszunutzen. Im Honeypot selbst sind sogenannte “Hints”, also Tipps und Hinweise, ein- und ausschaltbar, um umfangreiche Tutorials für die Sicherheitslücken und deren Ausnutzung aufzuzeigen. Außerdem ist ein Schwierigkeitsgrad einstellbar, der die jeweiligen Hacking-Versuche erschwert und jederzeit frei einstellbar ist.

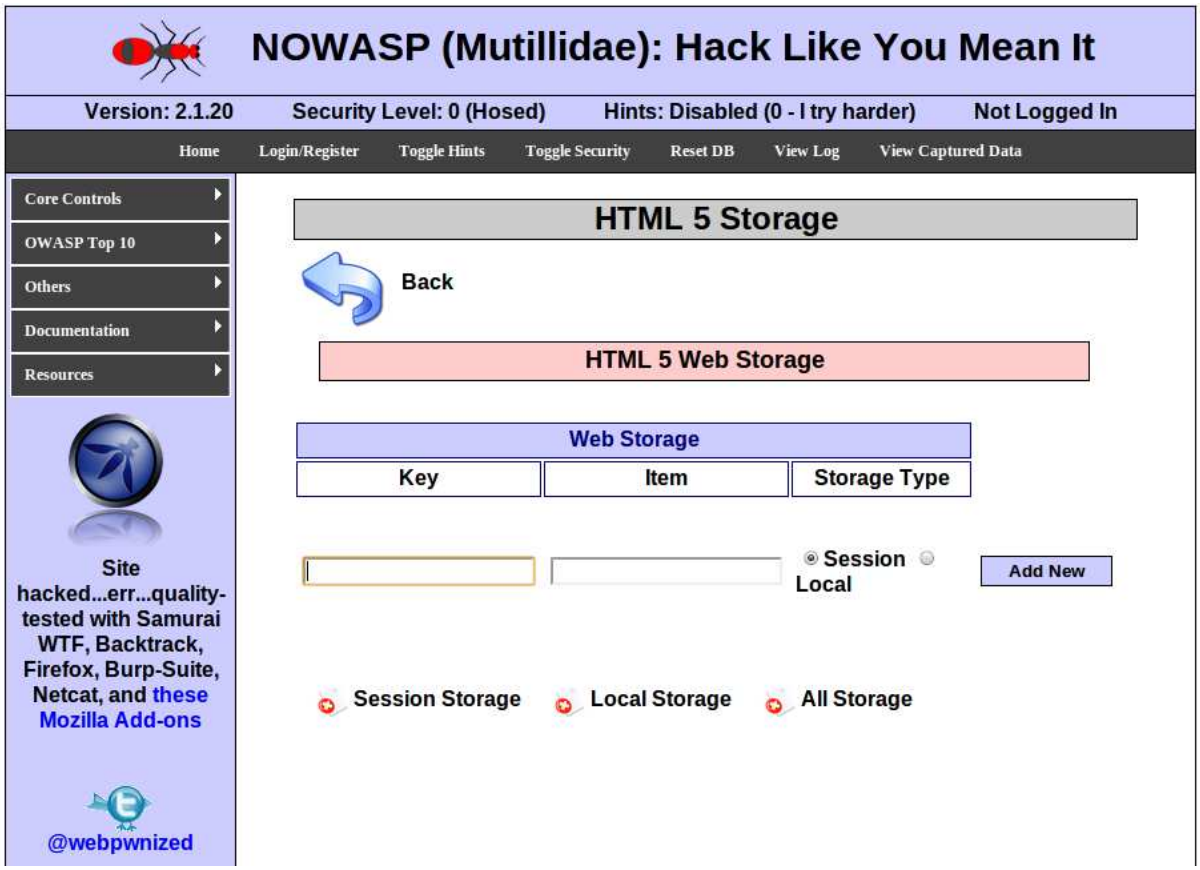


Abb. 10: Mutillidae-Oberfläche Benutzer-Menü links & Konfigurationsmöglichkeiten im oberen Menü

[32] Vgl. OWASP (2012a), http://owasp.de/top10

Das Projekt ist auf SourceForge angesiedelt und wird aktuell von nur einem aktiven Entwickler betreut und weiterentwickelt. Die Kommunikation mit dem Projekt kann über direkte Ansprache an den Entwickler erfolgen, das SourceForge-Forum ist jedoch eher als inaktiv anzusehen und externe Foren versprechen nicht unbedingt schnelle Hilfe. Eine große, aktive und leicht zu erreichende Community jedoch fehlt. Eigene Sicherheitslücken können nur mit entsprechendem Code-Verständnis hinzugefügt werden, Anleitungen hierfür waren im Rahmen der Evaluation dieser Arbeit nicht auszumachen. Weiterhin wird die Security-Live-CD “Samurai-WTF”, die als eines ihrer Tools den Honeypot Mutillidae beinhaltet, oft in externen Veranstaltungen oder Schulungen genutzt, wie etwa auch auf der diesjährigen OWASP AppSec Research 2012 im Juli.[33]

Die Installation des Honeypots ist sehr einfach ausgefallen und kann in unter 5 Minuten erfolgen, vorausgesetzt ein laufender PHP-Server wie beispielsweise der Apache Web Server ist vorhanden. Nach einem einfachen Kopieren des Mutillidae-Ordners in den htdocs-Ordner des Servers kann der Honeypot im Browser unter *Serveradresse/Mutillidae* aufgerufen werden.
Die Usability des Honeypots ist insgesamt von hoher Qualität, auch wenn das Design etwas altmodisch wirkt. Jedoch sind die wichtigen Funktionalitäten schnell zu erreichen und die Anleitungen und Beschriftungen sehr humorvoll gehalten (Siehe Abbildung 10 “Site hacked … err... qualitiy test with ...”), was den Benutzer zusätzlich motiviert. Durch die ausführlichen Tipps und Anleitungen eignet sich Mutillidae außerdem auch für den Blended- oder E-Learning Einsatz. Die Hinweise und Tutorials erwiesen sich im Rahmen des Bewertungsprozesses dieser Arbeit als so hochwertig, dass erste Sicherheitslücken in unter 30 Minuten nachvollzogen und sogar ausgenutzt werden konnten.

[33] Vgl. AppSecEU (2012), http://www.appsecresearch.org/training-3/

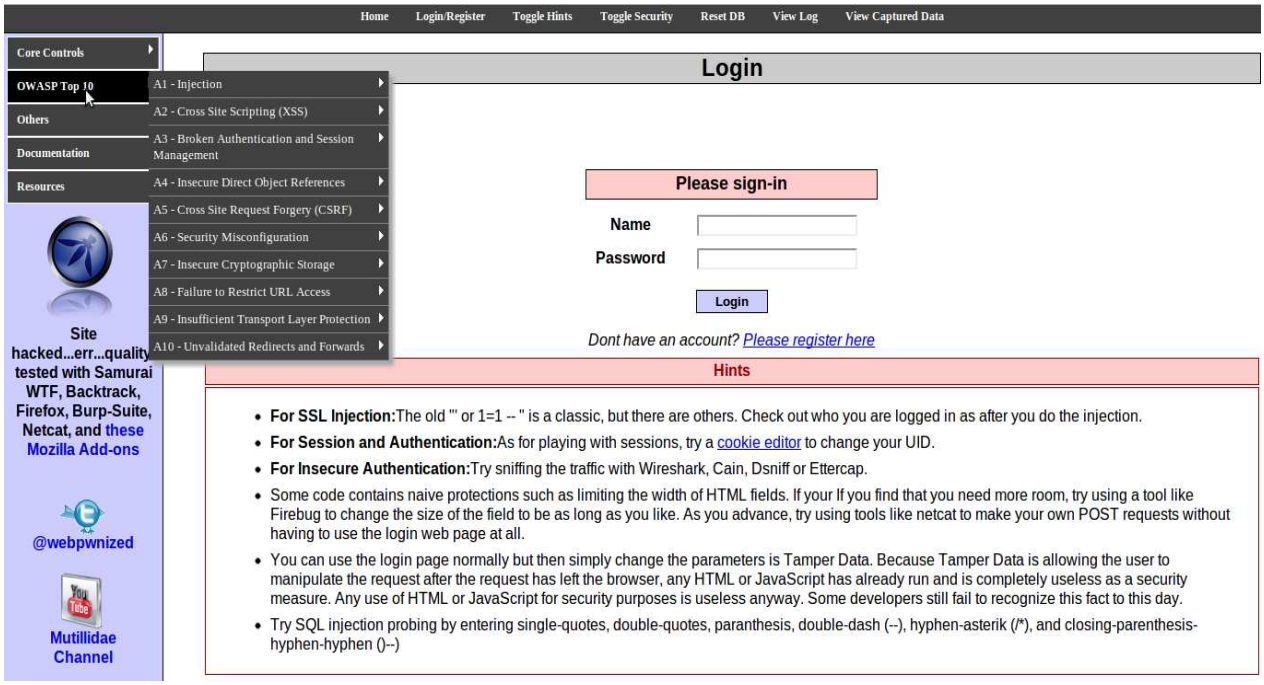


Abb. 11: Szenen-Screenshot aus Mutillidae mit ausgeklapptem Menü und eingeschalteten Hinweisen

Wie Abbildung 11 deutlich wird, kann sowohl die Dokumentation, als auch Tutorial-Videos und eine Beschreibung der OWASP-Top-10 Lücken schnell über das Linke Menü angesteuert werden. Die obere Menüleiste erlaubt Anpassungen zur Laufzeit und umfasst u. A. optionale Hinweis-Einblendungen, einen einstellbaren Schwierigkeitsgrad und eine Wiederherstellungsfunktion der Datenbank, wenn diese z. B. durch SQL-Injections bereits attackiert wurde und wieder in den Originalzustand versetzt werden soll, etwa um andere Angriffe aufzuzeigen. Im gezeigten Screenshot wurden die Hinweise eingeblendet und zeigen für den Log-In Bildschirm u. A. die Möglichkeit einer Passwort-Umgehung durch Eintragen von SQL-Code im Passwort-Feld auf. Die Hinweise fallen dabei nicht so detailliert wie etwa bei Webgoat aus, sind aber in Kombination mit angebotenen Tutorials und Videos ausreichend und motivierend, was insbesondere für den Einstieg wichtig ist.

Insgesamt liegt mit Mutillidae ein qualitativ hochwertiger PHP-basierter Honeypot vor, der mit umfangreichen Tipps und Funktionalitäten glänzen kann, jedoch Abstriche in Hinblick auf eine sehr kleine Programmierer-Community und altmodischem Design vorzuweisen hat.

5.5 Webgoat

Webgoat ist ein vom OWASP getriebenes Honeypot-Projekt. Es läuft auf einem Apache Tomcat-Server und ist daher sowohl mit Linux als auch mit Windows kompatibel. Das auf Java-Server-

Pages (JSP) basierende Projekt wird seit 2002 entwickelt[34] und orientiert sich an den aktuellen OWASP Top 10[35] Sicherheitslücken, die es in sogenannten “Lessons” aufzeigt und erklärt.

Nachfolgend werden die Hauptaussagen der Bewertung erläutert. Im Detail sind diese in Anhang 5 dieser Arbeit zu finden.

Eine Evaluierung dieses Honeypots zeigt ein äußerst ausgereiftes Projekt: Neben den 10 Top Sicherheitslücken existieren noch über 20 weitere Lessons über andere mögliche Web-Anwendungsfehler, welche außerdem durch die aktive Community erweitert werden können. Alle diese Lessons sind zudem dokumentiert.[36] Bei diesem erweiterbaren “Lesson”-System handelt es sich außerdem um ein herausragendes Merkmal, ermöglicht dieses Feature nicht nur einfache Erstellung von E-Learnings und Blended Learnings innerhalb von Unternehmen, sondern bietet außerdem eine einfache und gut dokumentierte Möglichkeit für Unternehmen, den Honeypot um eigene relevante Unterrichtseinheiten zu erweitern.

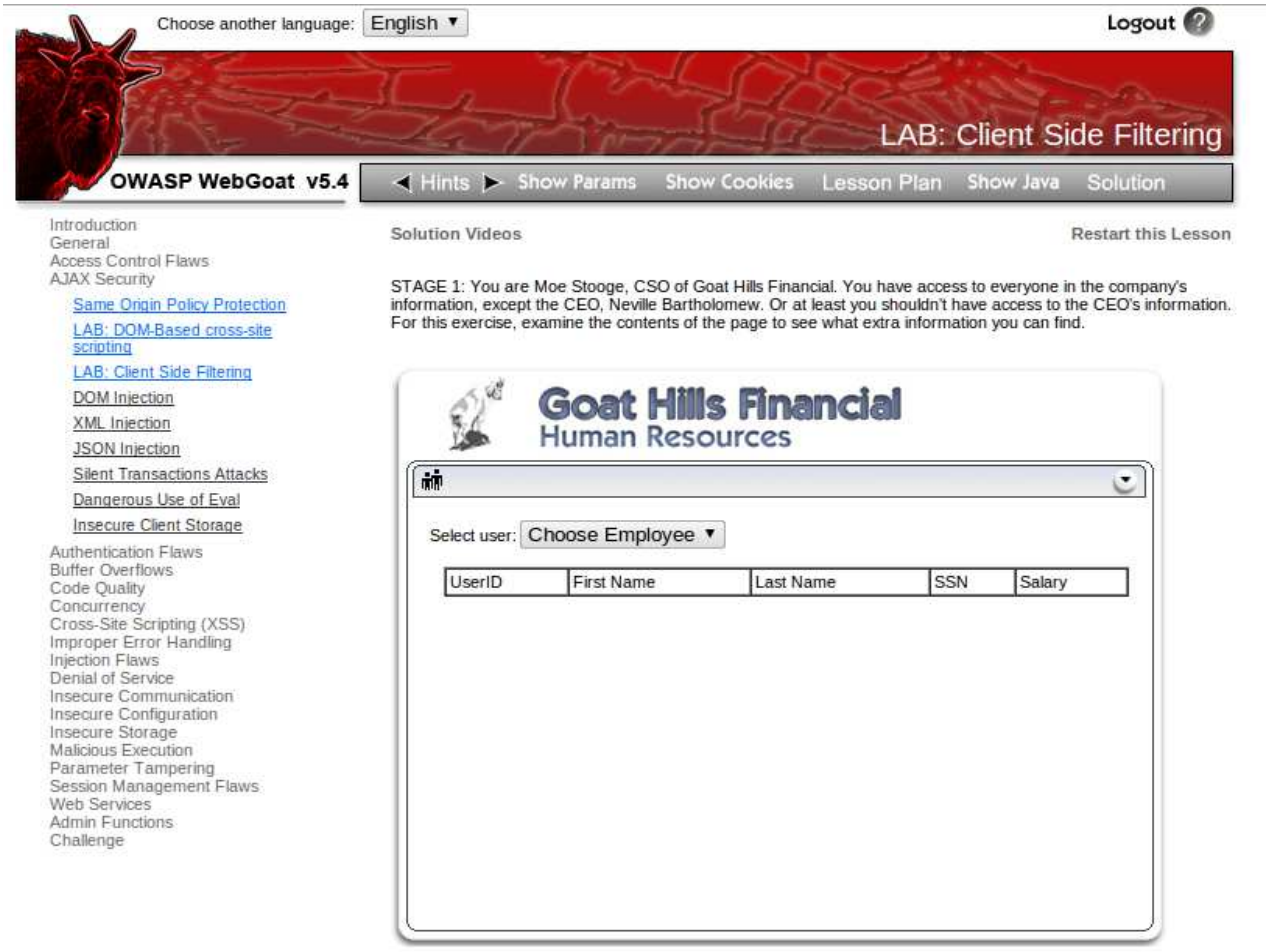


Abb. 12: Webgoat-Lesson mit Interface, Erklärung und dem Menü mit allen verfügbaren Lessons, kategorisiert nach Sicherheitslücken

[34] Vgl. OWASP (o.J.), https://lists.owasp.org/pipermail/owasp-webgoat/
[35] Vgl. OWASP (2010), https://www.owasp.org/images/b/b8/OWASPTop10_DE_Version_1_0.pdf
[36] Vgl. OWASP (2009), https://www.owasp.org/index.php/Lesson_Plans

Auch in der Dokumentation macht das Projekt einen professionellen Eindruck, denn das Webgoat-Wiki hilft sowohl bei der Installation, als auch bei ersten Schritten und den schon vorhanden Lessons. Auch ist eine FAQ vorhanden. Im Rahmen der Evaluation dieser Arbeit konnten erste Sicherheitslücken bereits in 30 Minuten problemlos ausgenutzt und nachvollzogen werden.
Das Projekt ist aktuell auf der Plattform “GoogleCode” angesiedelt und zählt dort insgesamt 23 Projekt-Mitglieder, wovon mindestens 7 aktiv programmieren. Auch weisen aktuelle Einträge und Antworten auf eine aktive Community hin, weshalb der Honeypot auch in Zukunft aktuelle Sicherheitslücken beinhalten dürfte.
Die Installation des Honeypots ist einfach und in weniger als 10 Minuten fertiggestellt, vorausgesetzt, ein Apache Tomcat Server wurde bereits vorab installiert. Ein Kopieren der .war-Datei in den “Catalina-Home”-Webapps-Ordner und das Hinzufügen bestimmter Rollen auf dem Tomcat-Server, welches in der Webgoat Dokumentation umfangreich beschrieben ist, sorgt für einen schnellen Start.

Die Weboberfläche ist benutzerfreundlich und übersichtlich, die Beschreibung der Sicherheitslücken ist meist gelungen und ausführlich. Dabei können zusätzliche Hinweise, Parameter und sogar Lösungen jederzeit aktiviert werden, was zu einem gewissen Maß an Individualisierung und Konfigurierbarkeit führt. Das Design ist aufgeräumt und wirkt nicht überladen. Wie in Abbildung 13 zu sehen ist, werden dabei auf der rechten Menüseite alle verfügbaren Lessons nach Kategorien sortiert und können so schnell vom Benutzer angesteuert werden.

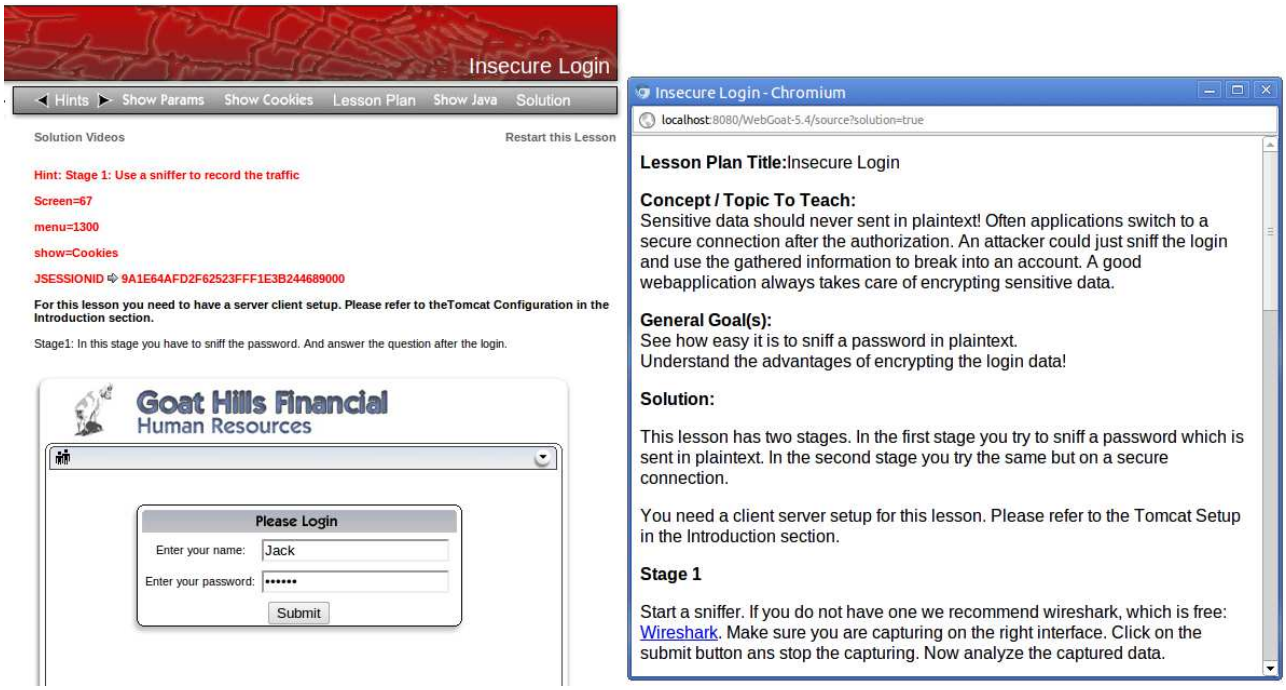


Abb. 13: Webgoat-Lesson mit eingeblendeten Hinweisen und der zugehörigen Lösung

In Abbildung 13 wurden zahlreiche Individualisierungen über das obere Menü vorgenommen. So werden Hinweise (rote Schrift links oben im Bild) und die Lösung (Fenster rechts im Bild) für die Lesson angezeigt. In diesem Beispiel wäre es nötig, einen frei verfügbaren Network-Sniffer wie Wireshark[37] zu benutzen, um die Sicherheitslücke auszunutzen.

Die wenigen Nachteile des Honeypots sind ein fehlendes Forum des Projekts und mangelnde optische Individualisierbarkeitsmöglichkeiten. Beide können jedoch vernachlässigt werden, wenn die große Anzahl an Vorteilen betrachtet wird: Eine Vielzahl an Sicherheitslücken, gute Erklärungen, E-Learning-Qualität der Lessons und ein ausgereiftes Design machen dieses Projekt zu einem sehr guten Honeypot. Das hohe Roadmap-Ziel des Projektes das "...defacto standard web application security training environment"[38] zu sein, wird nach den Maßstäben dieser Arbeit erfüllt.

Webgoat wird außerdem von der Security-Firma AspectSecurity[39] gesponsert, welche ebenfalls Schulungen im Bereich Websecurity mit diesem Honeypot anbieten. Zahlreiche weitere externe Tutorials und Ressourcen sind außerdem verfügbar wie etwa die "OWASP WebGoat v5.3 RC Web Hacking Simulation WalkThrough Series" der YGN Ethical Hacking Group.[40]

[37] Vgl. Wireshark (o.J.), http://www.wireshark.org/
[38] Vgl. OWASP (2011), https://www.owasp.org/index.php/OWASP_WebGoat_Project_Roadmap
[39] Vgl. ASPECT (2012), https://www.aspectsecurity.com/services/training/
[40] OWASP Webgoat (o.J.), http://yehg.net/lab/pr0js/training/webgoat.php

6. Schlussbetrachtung

Im folgenden Kapitel wird die Arbeit zunächst zusammengefasst, eine abschließende Empfehlung ausgesprochen und ein Ausblick auf weitere mögliche Forschungsfelder in diesem Zusammenhang gegeben. Die Arbeit endet schließlich mit einem Fazit.

6.1 Zusammenfassung

Um eine Empfehlung von Honeypots für Penetration-Test-Labore aussprechen zu können, wurde im Rahmen dieser Arbeit zunächst der Begriff des Honeypots als Testumgebung zum Aufzeigen von Sicherheitslücken definiert und zu Honeypots als Intrusion-Detection-System abgegrenzt.
Um im Folgenden eine detaillierte Bewertung von verschiedenen Lösungen vornehmen zu können, wurde daraufhin die Methodik der Arbeit erläutert. Honeypots wurden im Rahmen dieser Arbeit nach den OWASP-Top-10 Sicherheitslücken, der generellen Dokumentation des Projekts, der Einfachheit des Installationsprozesses und der Usability evaluiert. Die daraus resultierende Bewertungsmatrix ist Teil dieser Arbeit und kann genutzt werden, um eine individuelle Gewichtung der einzelnen Kriterien vorzunehmen.
Ausgehend von diesen Kriterien wurde daraufhin definiert, welche Honeypot-Projekte untersucht werden. Dazu wurden verschiedene mögliche Lösungen kurz vorgestellt und anschließend selektiert. Diese Vorauswahl ergab die Projekte Mutillidae, Webgoat, HacMe Series, DVWA und Gruyere, welche dann mit Hilfe der zuvor erstellten Bewertungsmatrix untersucht wurden. Hierzu wurde sowohl online recherchiert, als auch durch eine Installation des Honeypots auf einer virtuellen Maschine praktische Erfahrung gesammelt. Der direkte Vergleich der verschiedenen Open Source Honeypots mit Hilfe der im Rahmen dieser Arbeit erstellten Bewertungsmatrix, die ebenfalls zur eigenen Evaluation weiter verwendet werden kann, stellt den Hauptmehrwert dieser Ausarbeitung dar. Mit Hilfe dieser Evaluation kann nun im Folgenden eine Empfehlung ausgesprochen werden.

6.2 Handlungsempfehlungen

Bevor konkrete Projektnamen genannt werden können, muss zunächst eines klargestellt werden: Eine Empfehlung von Honeypots zur Benutzung in Workshops oder Penetration-Test-Laboren kann nicht uneingeschränkt ausgesprochen werden. Es hängt zu einem gewissen Teil vom “Use Case” des Benutzers ab, welcher Honeypot sich am besten für welche Zwecke eignet. So ist ein Honeypot immer in Bezug auf die Programmiersprache zu sehen, auf der er basiert. Diese sollte wenn möglich mit der Unternehmenswebpräsenz übereinstimmen. Zum Beispiel kann es wichtig sein, einen Java-basierten Honeypot zur Schulung der eigenen Mitarbeiter zu benutzen, wenn die Unternehmens-Website ebenfalls auf Java aufbaut. Auch ist die Auswahl des richtigen Honeypots

abhängig von der zur Verfügung stehenden Infrastruktur. So schränken beispielsweise die benutzten Betriebssysteme innerhalb des Unternehmens die Wahl des Honeypots zusätzlich ein, d. h. es ergibt wenig Sinn, einen auf Windows ausgerichteten Honeypot als Schulungsprogramm zu verwenden, wenn intern alle Applikationen auf Linux basieren.

Nichtsdestotrotz empfiehlt diese Arbeit insbesondere zwei Honeypots zum Einsatz in Workshops und Penetration-Test-Laboren:

1. Für einen PHP basierten Honeypot, welcher auf dem bekannten Web-Server-Paket Apache XAMMP und damit dem Apache Webserver aufbaut, empfiehlt sich der Einsatz von **Mutillidae.** Der Honeypot deckt alle relevanten Sicherheitslücken ab, ist sehr gut zu bedienen und liefert umfangreiche Anleitungen. Letzteres in Verbindung mit einer zur Verfügung stehenden Video-Plattform (YouTube-Channel) ermöglichen es, einfach E-Learnings oder hybride Lerntechniken (Klassenraum und Selbstlernprozesse) im Rahmen von Workshops anzuwenden, was sowohl Kosten sparen kann als auch zusätzliche Motivation und ein tieferes Verständnis ermöglicht.

2. Für einen auf Java Server Pages aufbauenden Honeypot, welcher auf dem populären Webserver Apache Tomcat aufsetzt, empfiehlt sich das Honeypot-Projekt **Webgoat**. Der Honeypot glänzt mit einer großen Vielzahl an Sicherheitslücken, die in sogenannten “Lessons” detailliert erläutert werden. Wie auch bei Mutillidae ist der Umfang an Tutorials, Anleitungen, Hilfestellungen und Videos so groß, dass es für Unternehmen attraktiv ist, E-Learnings und Klassenraumschulungen parallel oder kombiniert im Rahmen eines Penetration-Test-Labors oder -Workshops zu integrieren. Insbesondere das erweiterbare “Lesson”-System und eine große aktive Open Source Community machen diesen Honeypot im Rahmen dieser Arbeit zur besten Anwendung zur Schulung von Personen in Web-Anwendungs-Sicherheit.

6.3 Ausblick

Zahlreiche Themen bieten Potenzial zur Bearbeitung in weiteren Arbeiten und Forschungen. So bietet sich als weiteres betriebswirtschaftliches Forschungsthema insbesondere eine **Kostengegenüberstellung und -kalkulation** in Bezug auf Penetration-Testing und Schulung der eigenen Mitarbeiter an. Welche Kosten entstehen durch die Ein- und Durchführung von auf Open Source Honeypot basierten Workshops? Bietet es Vorteile oder Ersparnisse, auf vorhandene externe An-

bieter zurückzugreifen?[41][42] Sind Kalkulationen in Richtung der Unterlassung von solchen Schulungen, welche Verluste durch resultierende erfolgreiche Angriffe drohen, möglich?

Auch wäre eine weitere Arbeit in Hinblick auf die **Umsetzung des Penetration-Test-Labors** denkbar. Empfehlen sich E-Learnings als Selbstlernprozesse vielleicht sogar im Homeoffice der Mitarbeiter oder sind Klassenraumschulungen mit Tutor besser geeignet? Oder empfehlen sich Blended-Learning Techniken, also eine Kombination aus Arbeit in der Gruppe im Klassenraum und Hausaufgaben alleine oder von Zuhause aus? Sind andere Techniken ebenfalls denkbar und gut, wie etwa die Erstellung einer virtuellen Klassenraumschulung oder der einfachen Lektüre von Literatur zum Thema Sicherheitslücken?

Auch die andere Seite im Hacking-Prozess kann in weiteren Arbeiten beleuchtet werden. So fokussiert diese Arbeit ausschließlich die Seite des “Opfers”, also der Anwendung, die eine lückenhafte Sicherheit aufweist und infolge dessen attackiert wird. Interessant ist jedoch auch die **Seite des Angreifers**: Wie können Lücken erkannt werden? Welche Software kann für Angriffe genutzt werden und unter welchen Umständen ist sie zu empfehlen? Welche Erkenntnisse resultieren daraus für das Opfer? Wie kann sich ein Unternehmen das Wissen um Angriffstechniken zu Nutze machen und dagegen agieren? Im Rahmen von Forschungen in diesem Themenbereich können sowohl Open Source (z.B. Backtrack) als auch kommerzielle Lösungen beleuchtet werden (z.B. HP WebInspect oder IBM Rational AppScan).

6.4 Fazit

IT-Sicherheit ist mit der steigenden Verschiebung von Diensten und Informationen mehr denn je ein sehr wichtiges Thema. Vor allem Unternehmen müssen dem Anspruch einer sicheren Applikation insbesondere im Web-Umfeld gerecht werden. Um dies zu ermöglichen können Schulungen in Testumgebungen stattfinden, die mögliche Sicherheitslücken aufzeigen und verantwortlichen Mitarbeitern näherbringen. So ist es möglich, auch auf der wirklichen Unternehmens-Webpräsenz ein sicherheitstechnisch fehlerfreies Programm zu unterstützen.
Diese Arbeit hat hierbei als Mehrwert die Empfehlung von besonders geeigneten Honeypots ausgesprochen, wobei der Bewertungsprozess bewusst transparent gehalten wurde, um dem Anwender der Bewertungsmatrix eigene Gewichtung, Erweiterung und Anwendung der Kriterien auf andere Software zu ermöglichen. Mit Hilfe der empfohlenen Honeypots oder anderen, in der Bewertungsmatrix gut abschneidenden Lösungen ist es möglich, einen Workshop zur Sicherheitsschu-

[41] Vgl. Offensive Security (2012), http://www.offensive-security.com/offensive-security-solutions/virtual-penetration-testing-labs/; vgl. dazu auch: Compass Security AG (o.J.), http://www.csnc.ch/misc/files/2011/hl-remote-for-staff-trainings.pdf
[42] Vgl. Pentestlab (2012), http://pentestlab.org/about-us/

lung aufzubauen, als E-Learning, Klassenraumschulung oder hybrid umzusetzen und so die Sicherheit von Unternehmens-Web-Applikationen deutlich zu erhöhen.

Anhang

Anhangverzeichnis

Anhang 1 DVWA- Bewertungsmatrix

Kriterium	Punkte	Kommentar	Gewichtung
Sicherheitslücken		*1 Punkt pro vorhandener Lücke*	0,25
A1 – Injection	1	Gemäß der DVWA Dokumentation und https://code.google.com/p/dvwa/issues/detail?id=4	
A2 – Cross-Site Scripting (XSS)	1		
A3 – Fehler in Authentifizierung und Session Management	0		
A4 – Unsichere direkte Objektreferenzen	0		
A5 – Cross-Site Request Forgery (CSRF)	1		
A6 – Sicherheitsrelevante Fehlkonfiguration	1		
A7 – Kryptografisch unsichere Speicherung	0		
A8 – Mangelhafter URL-Zugriffsschutz	0		
A9 – Unzureichende Absicherung der Transportschicht	0		
A10 – Ungeprüfte Um- und Weiterleitungen	0		
Bewertungssumme:	**4**		
Maximum:	**10**		
Erfüllungsgrad:	**40,00%**		
Community & Dokumentation & Support		*Von 0 bis 3 Punkte (Anzuwenden auf alle Kriterien bis auf Sicherheitslücken und Features)*	0,3
Dokumentation		*0=nichts,1=vorhanden aber nicht gut, 2=ok gut, 3=sehr gut*	
Gibt es eine Anleitung, die die ersten Schritte aufzeigt?	3	Ein Youtube Video ist auf der Homepage vorhanden (http://www.youtube.com/watch?feature=player_embedded&v=Q3p_joL7X8E)	
Erwähnt die Dokumentation alle vorhandenen Sicherheitslücken ausreichend?	1	Der jeweilige Ort für jede Lücke wird erwähnt aber nur eine Sicherheitslücke wird exemplarisch erklärt.	
Gibt es eine Anleitung die in die Weboberfläche integriert ist?	2	Verweise auf Erklärungen der jeweiligen Sicherheitslücken und Quellcodeansicht sind in die Oberfläche eingebaut.	

Wie gut ist der Quellcode dokumentiert?	2	Die letzten allgemeinen Änderungen die gemacht und dokumentiert wurden sind vom 14.Oktober 2011, längere Zeit wurde nichts an dem Code geändert (http://code.google.com/feeds/p/dvwa/svnchanges/basic), vereinzelte Änderungen die sich auf Grund von Fehlern ergeben haben sind aktueller (letzte Änderung 19.März 2012) http://code.google.com/feeds/p/dvwa/issueupdates/basic;	
Fehlerbehebung			
Gibt es aktive Entwickler, die Fehler beheben können? (Wie aktiv ist die Community diesbezüglich?	2	Zwei Personen werden genannten die beide als Owner eingetragen sind (http://code.google.com/p/dvwa/people/list), darüber hinaus ist DVWA ein Projekt das von dem Unternehmen Random Storm betreut wird, die Aktivität ist allerdings beschränkt.	
Gibt es einen öffentlichen Bugtracker?	3	Ja, Bugreporting mit Priorität, Statusangabe, Fehlertyp, Besitzer und Kommentar, gut kommentiert und übersichtlich (http://code.google.com/p/dvwa/issues/list)	
Support bei Anwendungsproblemen			
Gibt es ein Forum?	0	Nicht auffindbar, ein Changelock ist vorhanden in dem Community-Mitglieder verschiedene Änderungen bewerten und kommentieren (kann als eine Art statisches Forum angesehen werden) http://code.google.com/p/dvwa/wiki/CHANGELOG	
Gibt es einen Chat für schnelle Hilfe?	0	Nicht auffindbar	
Anpassung des Honeypots an neue Sicherheitslücken			
Gibt es aktive Entwickler, die neue Sicherheitslücken einbauen können? (Wie aktiv ist die Community diesbezüglich?	2	Zwei Entwickler sind auf dem Bugtracker aktiv und reagieren auf Anfragen. Allerdings ist auf die Anfragen in den letzten 4 Monaten (Stand Juni 2012) keine Reaktion vorhanden.	
Alternativ, wie einfach ist es Sicherheitslücken selbst einzubauen	1	Keine Erklärung für die Erweiterung, Quellcode ist allerdings sehr modular und simpel aufgebaut. Erweiterungen sind also prinzipiell möglich.	
Bewertungssumme:	**16**		
Maximum:	**30**		
Erfüllungsgrad:	**53,33%**		
Installation			0,15
Installationsanleitung	3	Sehr einfache Installation: auspacken, im Browser aufrufen. Es gibt außerdem ein Anleitungsvideo. (http://code.google.com/p/dvwa/wiki/README, http://www.youtube.com/watch?v=GzIj07jt8rM)	

In bestehende Umgebung integrierbar?	3	Benötigt nur eine Standard XAMPP Umgebung und eine Datenbank mit beliebigem Namen.	
Frequently occurring errors dokumentiert?	1	Link im offline Readme ist tot, der Link im Onlinereadme zeigt auf den Bugtracker, nicht besonders nutzerfreundlich.	
Geschwindigkeit der Installation und Anzahl der manuellen Schritte	2	Auspacken aufrufen, danach gibt es einen Medienbruch, da die Config-Datei per Hand konfiguriert werden muss.	
Installationsinterface / GUI oder CLI?	1	DB Tabellen werden über Klick in der GUI angelegt, die DB Konfiguration muss aber ohne GUI bzw. mit externer GUI gemacht werden.	
Einfachheit der Installation	2	Nicht besonders kompliziert, die Fehlermeldungen bei DB-Verbindungsschwierigkeiten sind aber extrem generisch.	
Klarheit der Konfiguration	3	Bis auf die Datenbank, keine Konfiguration notwendig. Sehr übersichtliche und dokumentierte Konfigurationsdatei.	
Bewertungssumme:	**15**		
Maximum:	**21**		
Erfüllungsgrad:	**90,48%**		
Usability			0,2
ISO 9241-10			
Aufgabenangemessenheit	3	Sehr hohes Lehrpotential. Durch Schwierigkeitsgrade und umfangreicher Dokumentation sehr gute Aufgabenangemessenheit.	
Selbstbeschreibungsfähigkeit	2	Gut, dank integrierten Verweisen auf Literatur und Dokumentation der implementierten Sicherheitslücken.	
Steuerbarkeit	3	Der Nutzer wird sehr klar durch die Anwendung geführt, eine Steuerung macht keine Probleme.	
Erwartungskonformität	2	Die Anwendung verhält sich größtenteils, wie es von ihr erwartet wird.	
Individualisierbarkeit	3	Schwierigkeitsgrad kann zwischen "einfach", "mittel" und "schwer" gewählt werden. Die Datenbank kann nach Bedarf per Knopfdruck auf die Ursprungseinstellungen gesetzt werden.	
Lernförderlichkeit	3	Sehr hoch, da der Nutzer die Lösungen nicht einfach präsentiert bekommt, sondern sich über die Referenzen das Wissen zum Teil selbst erarbeiten muss.	
Design			
Ist das Design ansprechend?	3	Sehr ansprechend, einer modernen Webapplikation angepasst.	
Bewertungssumme:	**19**		
Maximum:	**21**		
Erfüllungsgrad:	**90,48%**		
Features			0,1

Unerwartete und positive Zusätze (1 pro Feature, Maximal 5)	1	Es existiert eine Reset-Funktion um bereits veränderte Daten in der Datenbank wieder auf den Installationsstand zurückzusetzen.	
Maximum:	**5**		
Erfüllungsgrad:	**20,00%**		
Gesamt	**59,67%**		

Anhang 2 Gruyere-Bewertungsmatrix

Kriterium	Punkte	Kommentar	Gewichtung
Sicherheitslücken		*1 Punkt pro vorhandener Lücke*	0,25
A1 – Injection	1	http://google-gruyere.appspot.com/part4#4__code_execution Code injection, SQL nicht verfügbar da kein SQL verwendet wird	
A2 – Cross-Site Scripting (XSS)	1	http://google-gruyere.appspot.com/part2#2__cross_site_scripting	
A3 – Fehler in Authentifizierung und Session Management	1	http://google-gruyere.appspot.com/part3#3__cookie_manipulation	
A4 – Unsichere direkte Objektreferenzen	1	http://google-gruyere.appspot.com/part4#4__path_traversal	
A5 – Cross-Site Request Forgery (CSRF)	1	http://google-gruye-re.appspot.com/part3#3__cross_site_request_forgery	
A6 – Sicherheitsrelevante Fehlkonfiguration	1	http://google-gruye-re.appspot.com/part5#5__configuration_vulnerabilities	
A7 – Kryptografisch unsichere Speicherung	1	http://google-gruye-re.appspot.com/part5#5__information_disclosure_config_2	
A8 – Mangelhafter URL-Zugriffsschutz	1	http://google-gruyere.appspot.com/part3#3__elevation_of_privilege	
A9 – Unzureichende Absicherung der Transportschicht	0		
A10 – Ungeprüfte Um- und Weiterleitungen	0		
Bewertungssumme:	**8**		
Maximum:	**10**		
Erfüllungsgrad:	**80,00%**		
Community & Dokumentation & Support		*Von 0 bis 3 Punkte (Anzuwenden auf alle Kriterien bis auf Sicherheitslücken und Features)*	0,3

Dokumentation		*0=nichts,1=vorhanden aber nicht gut, 2=ok gut, 3=sehr gut*	
Gibt es eine Anleitung, die die ersten Schritte aufzeigt?	2	http://google-gruyere.appspot.com/part1#1__using_gruyere - nicht sehr detailliert, aber Gruyere ist auch recht simpel aufgebaut	
Erwähnt die Dokumentation alle vorhandenen Sicherheitslücken ausreichend?	3	http://google-gruyere.appspot.com/part2 - Das Menü listet alle vorhandenen auf und erklärt auch was nicht vorhanden ist. Es gibt jeweils 2 Tipps und eine Anleitung für das Ausnutzen jeder Lücke	
Gibt es eine Anleitung die in die Weboberfläche integriert ist?	0	Nicht vorhanden.	
Wie gut ist der Quellcode dokumentiert?	3	Ausführliche Kommentare für jede Funktion, plus Erklärungen zum Fixen jeder Sicherheitslücke auf der Homepage	
Fehlerbehebung			
Gibt es aktive Entwickler, die Fehler beheben können? (Wie aktiv ist die Community diesbezüglich?	1	Google betreut das Projekt noch, es ist unklar ob bei Bedarf Änderungen vorgenommen werden.	
Gibt es einen öffentlichen Bugtracker?	0	Nicht vorhanden.	
Support bei Anwendungsproblemen			
Gibt es ein Forum?	1	http://code.google.com/edu/forum.html?place=forum/web-security-gcu-forum	
Gibt es einen Chat für schnelle Hilfe?	0	Nicht vorhanden.	
Anpassung des Honeypots an neue Sicherheitslücken			
Gibt es aktive Entwickler, die neue Sicherheitslücken einbauen können? (Wie aktiv ist die Community diesbezüglich?	1	Google betreut das Projekt noch, es ist unklar ob bei Bedarf Änderungen vorgenommen werden. Projekt ist vom Juni 2010, also sind noch nicht unbedingt Updates nötig gewesen.	
Alternativ, wie einfach ist es Sicherheitslücken selbst einzubauen	2	Gut kommentierter Code sollte Erweiterungen erleichtern, es lässt sich jedoch nichts darüber finden, ob es schon mal gemacht wurde.	
Bewertungssumme:	**13**		
Maximum:	**30**		
Erfüllungsgrad:	**43,33%**		
Installation			0,15
Installationsanleitung	3	Code auspacken, Datei mit Python starten, mehr Anleitung ist nicht nötig.	
In bestehende Umgebung integrierbar?	3	Solang Python installierbar ist und Port 8008 frei ist sollte kein Problem auftreten.	
Frequently occurring errors dokumentiert?	2	Nicht dokumentiert, es wurden allerdings sowohl beim eigenen Test als bei der Recherche keine möglichen Fehler gefunden.	
Geschwindigkeit der Installation und Anzahl der	3	Auspacken, Ausführen, keine Konfiguration oder weitere Schritte nötig.	

manuellen Schritte			
Installationsinterface / GUI oder CLI?	3	Keine GUI vorhanden, allerdings auch keine Installation nötig, daher positive Bewertung.	
Einfachheit der Installation	3	Siehe vorherige Punkte.	
Klarheit der Konfiguration	2	Konfiguration muss per Hand in den Dateien gemacht werden, ist aber an den jeweiligen Stellen gut erklärt und dokumentiert.	
Bewertungssumme:	**19**		
Maximum:	**21**		
Erfüllungsgrad:	**90,48%**		
Usability			0,2
ISO 9241-10			
Aufgabenangemessenheit	3	Alle Funktionen stehen direkt zur Verfügung, unnötige Komplexität wurde nicht gefunden.	
Selbstbeschreibungsfähigkeit	2	Homepage-Link ist inkonsistent, das Löschen von Snippets geschieht über ein [x] das nicht weiter erklärt wird. Lücken sind nur in der Dokumentation erklärt.	
Steuerbarkeit	3	Alle Funktionen stehen direkt zur Verfügung, unnötige Komplexität wurde nicht gefunden.	
Erwartungskonformität	2	Hochgeladene Dateien können nicht angesehen werden ohne Kenntnisse der URL.	
Individualisierbarkeit	0	Die Farbe des eigenen Namens lässt sich anpassen, der Rest ist hardcoded.	
Lernförderlichkeit	1	Oberfläche bietet keine ersten Schritte, nur durch das Handbuch.	
Design			
Ist das Design ansprechend?	1	Einheitliches Design, nicht besonders ansprechend, wirkt eher spielerisch.	
Bewertungssumme:	**12**		
Maximum:	**21**		
Erfüllungsgrad:	**57,14%**		
Features			0,1
Unerwartete und positive Zusätze (1 pro Feature, Maximal 5)	4	Reset Funktion, Denial of Service Lücke erklärt, spezielle AJAX-Lücken erklärt, eigene Sandbox für jeden User d.h. prinzipiell können mehrere Nutzer eine Installation nutzen.	
Maximum:	**5**		
Erfüllungsgrad:	**80,00%**		
Gesamt	**66,00%**		

Anhang 3 HacMe Series-Bewertungsmatrix

Kriterium	Punkte	Kommentar	Gewich-tung
Sicherheitslücken		*1 Punkt pro vorhandener Lücke*	0,25
A1 – Injection	1	Offizielle Sicherheitslücken aus der McAfee Dokumentation entnommen. Zusätzlich inoffizielle Sicherheitslücken aus Internetquellen entnommen.	
A2 – Cross-Site Scripting (XSS)	1		
A3 – Fehler in Authentifizierung und Session Management	1		
A4 – Unsichere direkte Objektreferenzen	1		
A5 – Cross-Site Request Forgery (CSRF)	1		
A6 – Sicherheitsrelevante Fehlkonfiguration	1		
A7 – Kryptografisch unsichere Speicherung	0		
A8 – Mangelhafter URL-Zugriffsschutz	1		
A9 – Unzureichende Absicherung der Transportschicht	0		
A10 – Ungeprüfte Um- und Weiterleitungen	0		
Bewertungssumme:	**7**		
Maximum:	**10**		
Erfüllungsgrad:	**70,00%**		
Community & Dokumentation & Support		*Von 0 bis 3 Punkte (Anzuwenden auf alle Kriterien bis auf Sicherheitslücken und Features)*	0,3
Dokumentation		*0=nichts,1=vorhanden aber nicht gut, 2=ok gut, 3=sehr gut*	
Gibt es eine Anleitung, die die ersten Schritte aufzeigt?	3	Ausführliche PDF Dokumentation vorhanden	
Erwähnt die Dokumentation alle vorhandenen Sicherheitslücken ausreichend?	2	Alle offiziellen vorhanden, im Internet lassen sich allerdings noch mehr dokumentiert finden	
Gibt es eine Anleitung die in die Weboberfläche integriert ist?	0	Nein	
Wie gut ist der Quellcode dokumentiert?	0	Nicht einsehbar ohne viel Aufwand, der Quellcode ist nicht auf die Produkte verlinkt	
Fehlerbehebung			
Gibt es aktive Entwickler, die Fehler beheben können? (Wie aktiv ist die Community diesbezüglich?	0	Keine Entwickler, keine aktive Community	
Gibt es einen öffentlichen Bugtracker?	0	Nein	
Support bei Anwen-			

dungsproblemen			
Gibt es ein Forum?	1	Vorhanden über McAfee allgemein, aber keine Posts zu HacMe	
Gibt es einen Chat für schnelle Hilfe?	0	Nein	
Anpassung des Honeypots an neue Sicherheitslücken			
Gibt es aktive Entwickler, die neue Sicherheitslücken einbauen können? (Wie aktiv ist die Community diesbezüglich?	1	Es gibt noch Entwickler, allerdings wird aktuell nicht entwickelt	
Alternativ, wie einfach ist es Sicherheitslücken selbst einzubauen	0	Kaum möglich ohne Expertenwissen, da Quellen schwer auffindbar sind	
Bewertungssumme:	**7**		
Maximum:	**30**		
Erfüllungsgrad:	**23,33%**		
Installation			0,15
Installationsanleitung	3	Sehr gut, in PDF ausführlich erklärt	
In bestehende Umgebung integrierbar?	2	Sofern diese Umgebung vorhanden ist, ja, sehr einfach als .exe Datei ausführbar	
Frequently occurring errors dokumentiert?	0	Nicht vorhanden	
Geschwindigkeit der Installation und Anzahl der manuellen Schritte	3	Sehr schnell, wenige Windows-Typische Installationsschritte	
Installationsinterface / GUI oder CLI?	3	GUI vorhanden und einfach zu bedienen	
Einfachheit der Installation	3	Sehr einfach	
Klarheit der Konfiguration	3	Sehr klar, keine Möglichkeiten zur Misskonfiguration	
Bewertungssumme:	**17**		
Maximum:	**21**		
Erfüllungsgrad:	**80,95%**		
Usability			0,2
ISO 9241-10			
Aufgabenangemessenheit	2	Zwar nicht mehr topaktuell (von 2006) aber dennoch zielführend	
Selbstbeschreibungsfähigkeit	1	Black-Box-Typische Anwendung, beschreibt kaum etwas selbst	
Steuerbarkeit	0	Nicht vorhanden, man muss sich die Anwendung selbst erschließen und wird nicht geführt	
Erwartungskonformität	2	Erfüllt den Zweck, Sicherheitslücken werden realitätsnah dargestellt	
Individualisierbarkeit	0	Nicht vorhanden	
Lernförderlichkeit	2	Ist gegeben, sofern etwas Eigeninitiative vorhanden ist	
Design			
Ist das Design ansprechend?	2	Wirkt sehr realitätsnah, allerdings ist das Design per se nicht sehr "schön"	
Bewertungssumme:	**9**		

Maximum:	**21**		
Erfüllungsgrad:	**42,86%**		
Features			0,1
Unerwartete und positive Zusätze (1 pro Feature, Maximal 5)			
Maximum:	**5**		
Erfüllungsgrad:	**0,00%**		
Gesamt	**45,21%**		

Anhang 4 Mutillidea-Bewertungsmatrix

Kriterium	Punkte	Kommentar	Gewichtung
Sicherheitslücken		*1 Punkt pro vorhandener Lücke*	0,25
A1 – Injection	1	Anspruch der Seite: "Mutillidae contains all of the vulnerabilties from the OWASP Top 10."	
A2 – Cross-Site Scripting (XSS)	1		
A3 – Fehler in Authentifizierung und Session Management	1		
A4 – Unsichere direkte Objektreferenzen	1		
A5 – Cross-Site Request Forgery (CSRF)	1		
A6 – Sicherheitsrelevante Fehlkonfiguration	1		
A7 – Kryptografisch unsichere Speicherung	1		
A8 – Mangelhafter URL-Zugriffsschutz	1		
A9 – Unzureichende Absicherung der Transportschicht	1		
A10 – Ungeprüfte Um- und Weiterleitungen	1		
Bewertungssumme:	**10**		
Maximum:	**10**		
Erfüllungsgrad:	**100,00 %**		
Community & Dokumentation & Support		*Von 0 bis 3 Punkte (Anzuwenden auf alle Kriterien bis auf Sicherheitslücken und Features)*	0,3
Dokumentation		*0=nichts,1=vorhanden aber nicht gut, 2=ok gut, 3=sehr gut*	
Gibt es eine Anleitung, die die ersten Schritte aufzeigt?	3	Sowohl ausführliche Anleitungen als auch ein eigener YouTube-Channel vereinfachen den Einstieg ungemein	

Erwähnt die Dokumentation alle vorhandenen Sicherheitslücken ausreichend?	3	Sowohl die Dokumentation als auch im Programm selbst über Tipps werden alle vorhanden Lücken genauestens aufgezeigt. Bestens für Workshops geeignet	
Gibt es eine Anleitung die in die Weboberfläche integriert ist?	3	Über sogenannte "hints", die ein- und ausblendbar sind, ist für jede OWASP Top 10 Lücke ein umfangreiches Tutorial bereitgestellt	
Wie gut ist der Quellcode dokumentiert?	3	Umfangreiche Dokumentation vorhanden, mit Installationsanleitung und genauen Beschreibungen der Sicherheitslücken	
Fehlerbehebung			
Gibt es aktive Entwickler, die Fehler beheben können? (Wie aktiv ist die Community diesbezüglich?	2	Ein sehr aktiver Entwickler vorhanden (Jeremy Druin)	
Gibt es einen öffentlichen Bugtracker?	3	Ja, auf SourceForge vorhanden	
Support bei Anwendungsproblemen			
Gibt es ein Forum?	1	Es existieren externe Foren und ein SourceForge Forum, letzteres ist aber eher als inaktiv anzusehen.	
Gibt es einen Chat für schnelle Hilfe?	0	Nicht vorhanden, höchstens extern dann aber eher keine schnelle Hilfe zu erwarten.	
Anpassung des Honeypots an neue Sicherheitslücken			
Gibt es aktive Entwickler, die neue Sicherheitslücken einbauen können? (Wie aktiv ist die Community diesbezüglich?	2	Ein sehr aktiver Entwickler, aber keine große Community.	
Alternativ, wie einfach ist es Sicherheitslücken selbst einzubauen	1	Es können Requests gestellt werden, mit Code-Verständnis ist auch ein eigenes Einbauen möglich, beim derzeitigen Umfang aber nicht nötig.	
Bewertungssumme:	**21**		
Maximum:	**30**		
Erfüllungsgrad:	**70,00%**		
Installation			0,15
Installationsanleitung	3	Sehr einfach ohnehin und trotzdem eine bis ins Detail beschriebene Anleitung (http://www.irongeek.com/i.php?page=mutillidae/installation)	
In bestehende Umgebung integrierbar?	3	Einfach auf bestehendem Apache Web Server installierbar durch einfachen Copy Paste, ansonsten auch auf der Linux Live CD Samurai enthalten	
Frequently occurring errors dokumentiert?	3		
Geschwindigkeit der Installation und Anzahl der manuellen Schritte	3	Sehr schnell und wenige Schritte, wenn Apache Web Server vorhanden	
Installationsinterface / GUI oder CLI?	3	Nach Copy/Paste durch Aufruf der Seite Interface gegeben	

Einfachheit der Installation	3	Konfiguration sehr einfach über übersichtliches Interface	
Klarheit der Konfiguration	3	Sicherheitslücken, Hilfestellungen und Anforderungen sind sehr gut und schnell einzustellen. Konfiguration ist jederzeit erreichbar über ein oberes Menü.	
Bewertungssumme:	**21**		
Maximum:	**21**		
Erfüllungsgrad:	**100,00 %**		
Usability			0,2
ISO 9241-10			
Aufgabenangemessenheit	3	Dialoge und Interaktionsmöglichkeiten sind stets angemessen.	
Selbstbeschreibungsfähigkeit	3	Durch gute Beschriftungen und Titel sowie ausreichende und gut erreichbare Hilfestellung kann jeder Teil schnell vom Benutzer verstanden werden.	
Steuerbarkeit	3	Befehle führen ohne Probleme zum gewünschten Ziel, umständliche Bedienung konnte nicht gefunden werden	
Erwartungskonformität	2	Da man allgemein keine Erwartungen oder Referenzen für eine Honeypot-Oberfläche erwartet, sind einige Hinweise oder Optionen zunächst ungewohnt, wie z.B. die Hints nach unten hin angeordnet. Trotzdem ist eine schnelle Eingewöhnung möglich.	
Individualisierbarkeit	1	Tipps und Schwierigkeit sind einstellbar, die Oberfläche kann aber nicht ohne weiteres angepasst werden. Dies ist aber auch nicht weiter schlimm oder wichtig im Benutzerkontext.	
Lernförderlichkeit	3	Höchstnote durch zuschaltbare umfangreiche Tipps und schnell erreichbare Dokumentation.	
Design			
Ist das Design ansprechend?	1	Das Design erscheint etwas altmodisch, ist aber vollkommen ausreichend. Teilweise erscheinen einem insbesondere die Tipps etwas überladen.	
Bewertungssumme:	**16**		
Maximum:	**21**		
Erfüllungsgrad:	**76,19%**		
Features			0,1

Unerwartete und positive Zusätze (1 pro Feature, Maximal 5)	5	Humorvolle Kommentare und Beschreibungen vereinfachen und motivieren die Bedienung für den Endbenutzer. Zusätzlich kann der HoneyPot schon fast als Standalone betrachtet werden, was ihn E-Learning und Blended-Learning tauglich macht. Zusätzlich kann der HoneyPot einfach in den Originalzustand versetzt werden innerhalb des Programms und ein Aktivitäts-Log ist vorhanden. Sicherheitslücken die in den OWASP Top 10 2007 enthalten und 2010 entfallen sind, sind Teil des Projekts.	
Maximum:	**5**		
Erfüllungsgrad:	**100,00 %**		
Gesamt	**86,24%**		

Anhang 5 Webgoat-Bewertungsmatrix

Kriterium	Punkte	Kommentar	Gewichtung
Sicherheitslücken		*1 Punkt pro vorhandener Lücke*	0,25
A1 – Injection	1	Da das Projekt OWASP getrieben ist, finden sich alle OWASP-Sicherheitslücken wieder	
A2 – Cross-Site Scripting (XSS)	1		
A3 – Fehler in Authentifizierung und Session Management	1		
A4 – Unsichere direkte Objektreferenzen	1		
A5 – Cross-Site Request Forgery (CSRF)	1		
A6 – Sicherheitsrelevante Fehlkonfiguration	1		
A7 – Kryptografisch unsichere Speicherung	1		
A8 – Mangelhafter URL-Zugriffsschutz	1		
A9 – Unzureichende Absicherung der Transportschicht	1		
A10 – Ungeprüfte Um- und Weiterleitungen	1		
Bewertungssumme:	**10**		
Maximum:	**10**		
Erfüllungsgrad:	**100,00 %**		

Community & Dokumentation & Support		*Von 0 bis 3 Punkte (Anzuwenden auf alle Kriterien bis auf Sicherheitslücken und Features)*	0,3
Dokumentation		*0=nichts,1=vorhanden aber nicht gut, 2=ok gut, 3=sehr gut*	
Gibt es eine Anleitung, die die ersten Schritte aufzeigt?	3	Es existiert ein ausführliches Wiki mit ersten Schritten	
Erwähnt die Dokumentation alle vorhandenen Sicherheitslücken ausreichend?	3	Das Wiki der Seite zeigt alle vorhandenen Sicherheitslücken klar auf, https://www.owasp.org/index.php/Lesson_Plans	
Gibt es eine Anleitung die in die Weboberfläche integriert ist?	3	Über sogenannte "Lessons" wird jede Lücke ausführlich erklärt, während gleich auf derselben Oberfläche praktische Beispiele gegeben werden.	
Wie gut ist der Quellcode dokumentiert?	2	Quellcode weist zum größten Teil selbsterklärende Bezeichnungen, aber wenig Kommentare auf	
Fehlerbehebung			
Gibt es aktive Entwickler, die Fehler beheben können? (Wie aktiv ist die Community diesbezüglich?	3	23 aktive Projekt-Mitglieder, davon 1 Besitzer und laut GoogleCode 7 Commiters. Auf Issues wird auch aktuell geantwortet (Stand Juni 2012)	
Gibt es einen öffentlichen Bugtracker?	3	Ja, auf GoogleCode vorhanden	
Support bei Anwendungsproblemen			
Gibt es ein Forum?	1	Die Kommunikation findet hauptsächlich über eine Mailingliste, Feature-Requests und Wiki statt. Ein öffentliches Forum mit offenen Fragen konnte nicht gefunden werden, es existiert jedoch ein FAQ.	
Gibt es einen Chat für schnelle Hilfe?	1	Allgemeiner Chat nicht vorhanden, aber evtl. über Google Dienst Kontakt herstellbar	
Anpassung des Honeypots an neue Sicherheitslücken		Da das Projekt OWASP getrieben ist, ist der Honeypot stets auf dem neuesten Stand in Bezug auf Sicherheitslücken. Zusätzliche Lücken können jedoch sehr einfach hinzugefügt werden, indem eigene, sogenannte "Lessons" geschrieben werden können. Hierzu existiert eine umfangreiche Anleitung.	
Gibt es aktive Entwickler, die neue Sicherheitslücken einbauen können? (Wie aktiv ist die Community diesbezüglich?	3	23 Mitglieder, davon 7 Commiter und aktuelle Problem-Diskussionen weisen auf eine sehr aktive Community hin.	
Alternativ, wie einfach ist es Sicherheitslücken selbst einzubauen	3	Es ist sehr einfach, über sogenannte eigenen "Lessons" eigenen Lücken mit Anleitung zu integrieren. Hierzu existiert eine umfangreiche Dokumentation.	
Bewertungssumme:	**25**		
Maximum:	**30**		
Erfüllungsgrad:	**83,33%**		

Installation			0,15
Installationsanleitung	3	Eine sehr gut beschriebene Anleitung für verschiedene Systeme. Einfach das Programm auf einen existierenden Tomcat deployen.	
In bestehende Umgebung integrierbar?	3	Ein Tomcat wird vorausgesetzt, dann ist weiteres an sich kein Problem.	
Frequently occurring errors dokumentiert?	3	FAQ für häufig auftauchende Probleme vorhanden	
Geschwindigkeit der Installation und Anzahl der manuellen Schritte	3	Sehr schnell durch einfaches deployen, evtl. noch ein paar Konfigurationsschritte für Benutzer nötig	
Installationsinterface / GUI oder CLI?	2	Nach Copy/Paste des war.Files, danach Bearbeitung einer Konfigurationsdatei	
Einfachheit der Installation	2	Bis auf die Benutzerrechte im Tomcat keine Probleme	
Klarheit der Konfiguration	3	Die Weboberfläche ist sehr Benutzerfreundlich und sämtliche Konfigurationen, die nötig sind, sind einfach vorzunehmen	
Bewertungssumme:	**19**		
Maximum:	**21**		
Erfüllungsgrad:	**90,48%**		
Usability			0,2
ISO 9241-10			
Aufgabenangemessenheit	3	Dialoge und Interaktionsmöglichkeiten sind allesamt aussagekräftig und eher zu ausführlich	
Selbstbeschreibungsfähigkeit	3	Alle Funktionen und "Lessons" sind gut betitelt und beschreiben die dahinterliegende Funktion ausreichend	
Steuerbarkeit	3	Sehr gute Menüsteuerung und einfache "Lessons", die schnell anzusteuern sind	
Erwartungskonformität	3	Die Weboberfläche des Honeypots ist intuitiv bedienbar und die Menüs sind so, wie man sie von andere Websites kennt und erwartet	
Individualisierbarkeit	2	Eigenen "Lessons" können geschrieben werden, Hinweise und Tipps können einfach angeschaltet werden. Eine visuelle Individualisierung ist jedoch nur mit Webdesign-Kenntnissen möglich und nicht durch das Projekt unterstützt.	
Lernförderlichkeit	3	Durch umfangreiche Tipps, Dokumentation und guter Beschriftung lernt der Benutzer schnell den Umgang mit "Webgoat"	
Design			
Ist das Design ansprechend?	3	Das Design erscheint absolut angemessen und aufgeräumt	
Bewertungssumme:	**20**		
Maximum:	**21**		
Erfüllungsgrad:	**95,24%**		
Features			0,1

Unerwartete und positive Zusätze (1 pro Feature, Maximal 5)	3	Das Programm kann schon ohne Anleitung benutzt werden, ist also E-Learning und Blended Learning geeignet. Die Möglichkeit, eigene "Lessons" zu schreiben, ist sehr gut ausgeprägt und gibt einen Extrapunkt. Der Javacode, Cookies und Lösungen können außerdem einfach von den Benutzern eingesehen werden.	
Maximum:	**5**		
Erfüllungsgrad:	**60,00%**		
Gesamt	**88,62%**		

Anhang 6 Benutzerhandbuch DHBWA

Inhaltsverzeichnis

Achtung! Diese virtuelle Maschine enthält vier Honeypots die absichtlich Sicherheitslücken enthalten.
Die Benutzung geschieht auf eigene Gefahr!

Benutzer und Passwörter

Der Nutzer mit dem Sie beim Start dieser virtuellen Maschine standardmäßig eingeloggt werden ist `osstest`. Das Passwort lautet `honey`. Dieser Nutzer ist als Admin für diese virtuelle Maschine eingetragen.

Der MySQL-Server lässt sich mit dem Administrator-Account `root` verwalten. Das Passwort lautet ebenfalls `honey`.

Der Tomcat-Server lässt sich mit dem Administrator-Account `tom` verwalten. Das Passwort lautet ebenfalls `honey`.

Um den WebGoat Honeypot zu nutzen können Sie auf die folgenden verschiedene Nutzer zurückgreifen:

Rolle	User	Passwort
webgoat_admin	`wgadmin`	`wgadmin`
webgoat_user,webgoat_basic	`wgbasic`	`wgbasic`
webgoat_user	`wgguest`	`wgguest`

Um den DVWA Honeypot zu nutzen, existiert nur ein standardmäßig eingerichteter Account: admin mit dem Passwort password.

Installierte Honeypots

Es sind 4 Honeypots auf dieser virtuellen Maschine installiert, die auch allesamt sofort nachdem Systemstart verfügbar sind. Es handelt sich dabei um DVWA, Gruyere, Mutillidae und WebGoat.

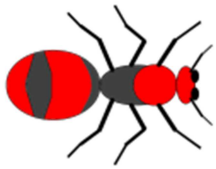
DVWA

Website: http://dvwa.co.uk/
Lokale Installation: http://localhost/dvwa/login.php (User: admin Password: password)
Lokaler Ordner: /var/www/dvwa

Gruyere

Website: https://google-gruyere.appspot.com/
Online Installation: https://google-gruyere.appspot.com/start
Lokale Installation: http://localhost:8008
Lokaler Ordner: /home/osstest/gruyere-code/

Mutillidae



Website: http://www.irongeek.com/i.php?page=mutillidae/mutillidae-deliberately-vulnerable-php-owasp-top-10

Lokale Installation: http://localhost/mutillidae/index.php

Lokaler Ordner: /var/www/mutillidae/

WebGoat

Website: https://www.owasp.org/index.php/Category:OWASP_WebGoat_Project

Lokale Installation: http://localhost:8080/WebGoat-5.4/attack (Benutzer siehe Seite 1)

Lokaler Ordner: /var/lib/tomcat6/webapps/WebGoat-5.4

Quellenverzeichnis

Literaturverzeichnis

Gerrit Göbel, J. / Dewald, A. (2011)	Client-Honeypots- Exploring Malicious Websites, München: Oldenbourg Wissenschaftsverlag
Harwood, M. (2011)	Security Strategies in Web Applications and Social Networking, Sudbury: Jones &Bartlett Learning
Hecktor, R./ Hohmann, D./ Radecke, G./ Klink, M./ Petrovic, J./ Pfister, T. (2012)	Leitfaden zur Einführung von Open Source Software in Organisationen, Duale Hochschule Baden-Württemberg Stuttgart
Peikari, C./ Chuvakin, A. (2004)	Kenne deinen Feind: Fortgeschrittene Sicherheitstechniken, Köln: O'Reilly Verlag
Wilhelm, T. (2010)	Professional Penetration Testing: Creating and Operating a Formal Hacking Lab, Oxford: Elsevier

Verzeichnis der Internet und Intranet-Quellen

AppSecEU (2012) — OWASP Appsec Research 2012 conference, http://www.appsecresearch.org/training-3/, Abruf: 16.06.2012

ASPECT (2012) — Instructor Led Training Services, https://www.aspectsecurity.com/services/training/, Abruf: 18.06.2012

Compass Security AG (o.J.) — Professional Security Lab, http://www.csnc.ch/misc/files/2011/hl-remote-for-staff-trainings.pdf, Abruf: 18.06.2012

DATech (2009) — Leitfaden Usability, http://www.datech.de/share/files/Leitfaden-Usability.pdf, Abruf: 12.06.2012

DVWA (2012a) — DVWA Webpage, http://www.dvwa.co.uk/, Abruf: 12.06.2012

DVWA (2012b) — Subversion commits to project dvwa on Google Code, http://code.google.com/feeds/p/dvwa/svnchanges/basic%29, Abruf: 15.06.2012

EUROPÄISCHE NORM (1995) — Europäische Norm, www.interactive-quality.de/site/DE/int/pdf/ISO_9241-10.pdf, Abruf: 09.06.2012

Gründerszene (o.J.) — Gründerszene- das Magazin für Gründer, http://www.gruenderszene.de/lexikon/begriffe/non-disclosure-agreement-nda, Abruf: 05.06.2012

Gruyere (2010) Web Application Exploits and Defenses, http://google-gruyere.appspot.com/, Abruf: 15.06.2012

Gruyere (2010) Client-State Manipulation, http://google-gruy-ere.appspot.com/part3#3__elevation_of_privilege, Abruf: 15.06.2012

Hacme Casino (2003) Free Tools McAfee, http://www.mcafee.com/us/downloads/free-tools/hacme-casino.aspx, Abruf: 16.06.2012

Krooß, M. (2003) Honeypots Einsatzmöglichkeiten beim Schutz von IT-Systemen, http://www.informatik.uni-ham-burg.de/RZ/lehre/18.415/seminararbeit/11_Honeypots.pdf, Abruf: 25.05.2012

Mikhalenko, P. (2006) Managing a Honeypot, http://www.oreillynet.com/pub/a/sysadmin/2006/09/28/honeypots.html, Abruf: 09.06.2012

MIT (2012) Massachusetts Institute of Technology, http://en.wikipedia.org/wiki/Massachusetts_Institute_of_Technology, Abruf: 07.06.2012

MITRE (2011) Comparison to OWASP Top Ten 2010, http://cwe.mitre.org/top25/index.html#AppendixD, Abruf: 07.06.2012

o.V. (0.J.) http://www.badstore.net/, Abruf: 12.06.2012

o.V. (2011) Blended Learning, http://www.e-learning.fu-ber-lin.de/lehren_mit_neuen_medien/einsatzszenarien/blended_learning/index.html, Abruf: 12.06.2012

Munroe, R. (2007) Exploits of a mum, http://xkcd.com/327/, Abruf: 09.06.2012

Offensive Security (2012) Virtual Penetration Testing labs, http://www.offensive-security.com/offensive-security-solutions/virtual-penetration-testing-labs/, Abruf:18.06.2012

OWASP (2009) Lesson Plans, https://www.owasp.org/index.php/Lesson_Plans, Abruf: 18.06.2012

OWASP (2011) OWASP WebGoat Project Roadmap, https://www.owasp.org/index.php/OWASP_WebGoat_Project_Roadmap, Abruf: 18.06.2012

OWASP (2012a) OWASP Top 10 Sicherheitslücken, http://owasp.de/top10, Abruf: 06.06.2012

OWASP (2012b) OWASP-Startseite, https://www.owasp.org, Abruf: 06.06.2012

OWASP (2012c) OWASP- About page, https://www.owasp.org/index.php/About_OWASP, Abruf: 06.06.2012

OWASP (o.J.) The Owasp-webgoat Archives , https://lists.owasp.org/pipermail/owasp-

webgoat/, Abruf: 18.06.2012

OWASP Webgoat (o.J.) Security/ Hacking Training Movies, http://yehg.net/lab/pr0js/training/webgoat.php, Abruf: 18.06.2012

Pentestlab (2012) PenTestLaboratory, http://pentestlab.org/about-us/, Abruf: 25.06.2012

Saindane, M. (2006) Penetration testing- a systematic approach, http://www.infosecwriters.com/text_resources/pdf/PenTest_MSaindane.pdf, Abruf: 31.05.2012

SpiegelOnline (2012) Hacker knacken Server der SPD, http://www.spiegel.de/netzwelt/web/partei-gehackt-unbekannte-knacken-server-der-spd-a-833905.html, Abruf: 15.05.2012

Spitzner, L. (2003) Honeypots
Definitions and Value of Honeypots, http://www.tracking-hackers.com/papers/honeypots.html, Abruf: 25.05.2012

Thomas, S. (2007) Penetration Test- Sicherheitsüberprüfung von Netzwerk- oder Softwaresystemen aus Hacker Sicht, http://www.trivadis.com/uploads/tx_cabagdownloadar-ea/Penetration_Final_070802.pdf ,Abruf: 30.05.2012

Wireshark (o.J.) Wireshark-Homepage, http://www.wireshark.org/, Abruf.

18.06.2012

Wirth, T. (2009) Usability, http://kommdesign.de/texte/din.htm, Abruf: 09.06.2012

Vergleichsstudie zu Open Source Produkten für Last- / Performancetesttools

Seminararbeit

vorgelegt am 5.7.2012

Fakultät Wirtschaft

Studiengang WI-International Business Information Management

Kurs WWI2009I

von

Lisa Aust, Thomas Dorsch, Timo Pfister, Annkathrin Schwarz

DHBW Stuttgart:

Prof. Dr. Thomas Kessel

Inhaltsverzeichnis

Abkürzungsverzeichnis

AJAX	= Asynchronous JavaScript and XML
BSD	= Berkeley Software Distribution
CFG	= Config
CGI	= Common Gateway Interface
CLI	= Command-line interface
CPU	= Central Processing Unit
CGI	= Common Gateway Interface
FTP	= File Transfer Protocol
GNU	= GNU's Not Unix
GPL	= General Public License
GUI	= Graphical User Interface
HTML	= Hypertext Markup Language
HTTP	= Hypertext Transfer Protocol
HTTPS	= Hypertext Transfer Protocol Secure
IMAP	= Internet Access Message Protocol
IDE	= Integrated Development Environment
JDBC	= Java Database Connectivity
JDK	= Java Development Kit
JMS	=Java Message Service
JRE	= Java Runtime Environment
JSSE	= Java Secure Socket Extension
JVM	= Java Virtual Machine
LDAP	= Lightweight Directory Access Protocol
MS	=Microsoft
POP3	= Post Office Protocol 3
PTTMonitor	= PushtoTest TestMaker Monitor
RPC	= Remote Procedure Call
SMTP	= Simple Mail Transfer Protocol
SOAP	= Simple Object Access Protocol
SSL	= Secure Sockets Layer
TCP	= Transmission Control Protocol
TLS	= Transport Layer Security
URL	= Uniform Resource Locator
XLS	= eXcel Sheet
XML	= Extensible Markup Language
xPath	= XML Path Language

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Innerhalb eines jeden Software-Entwicklungszykluses sollte eine erfolgreich konzipierte und entwickelte Anwendung getestet werden. Entwicklern stehen hierfür eine Vielzahl an unterschiedlichen Werkzeugen zum Testen eines Programms zur Verfügung. Viele dieser Werkzeuge sind kommerzielle Produkte von Unternehmen, die mit der Bereitstellung von Testingtools und entsprechender Beratung Gewinn erzielen möchten. Ihre Nutzung ist folglich mit Kosten für die Organisation verbunden, die eine neu entwickelte oder bereits bestehende Anwendung testen möchte. Eine Alternative zu den kommerziellen Testingtools stellen deshalb Open Source Werkzeuge dar, die durch Entwickler und Organisationen frei verwendet werden können.

Ziel dieser Seminararbeit ist die Erstellung einer Vergleichsstudie von Open Source Produkten für Performance- und Lasttests von Anwendungen. Dazu wird ein Vergleichssystem entwickelt, welches anhand einheitlicher Vergleichskriterien Bewertungen der verschiedenen Produkteigenschaften vornimmt. Die Vergleichskriterien werden dabei in vier Kategorien Bedienbarkeit, Technologie, Funktionalität und Sonstiges gegliedert. Das Vergleichssystem soll so entwickelt werden, dass die unterschiedlichen Vergleichskriterien von späteren Nutzern individuell gewichtet werden können, um den spezifischen Anforderungen gerecht zu werden.

Im ersten Teil der Vergleichsstudie wird zunächst erläutert, was in der Literatur unter Performance- und Lasttests verstanden wird. Es werden die Kriterien eingeführt, die durchgängig für die Beschreibung und Bewertung der Tools genutzt werden. Anschließend werden vier ausgewählte Open Source Produkte für Performance- und Lasttests unter Berücksichtigung dieser Kriterien beschrieben. Die Auswahl der Tools wird dabei durch ihren Verbreitungsgrad sowie die Arten von Anwendungen, die getestet werden können, begründet.

Nach der Vorstellung der einzelnen Open Source Produkte wird die Methodik zur Bewertung der Tools eingeführt. Mithilfe einer Bewertungsmatrix werden die Test-Programme in den unterschiedlichen Kriterien bewertet, abhängig vom Umfang, in welchem sie das jeweilige Kriterium erfüllen. Die Bewertungsmatrix liefert nach Evaluierung aller vier Produkte ein ganzheitliches Bild über die Leistungsfähigkeit der Tools im direkten Vergleich. Die Ergebnisse werden anschließend interpretiert, um Schlussfolgerungen aus der Bewertung zu ziehen. Diese zeigen auf, welches Testtool für welchen Anwendungsbereich besonders geeignet ist.

In einem weiteren Teilkapitel wird erörtert, worin die wesentlichen Unterschiede zwischen Open Source und kommerziellen Testingtools liegen. Dabei soll ein Vergleich gezogen werden, der im Allgemeinen die Vor- und Nachteile der jeweiligen Lizenzierungsart (Open Source und kommerzieller Software) beleuchtet. Aufgrund der Vielzahl unterschiedlicher Produkte, die sowohl kommerziell als auch als Open Source auf dem Markt bereitstehen und der damit einhergehenden deutlich steigenden Komplexität wurde im Rahmen dieser Arbeit auf eine direkte Gegenüberstellung spezieller kommerzieller und Open Source Testtools verzichtet.

Diese Arbeit beschäftigt sich ausschließlich mit Performance- und Lasttest. Andere Tests, wie beispielsweise Funktionstests, sind nicht Bestandteil dieser Vergleichsstudie. Aufgrund des vorgegebenen Umfangs der Seminararbeit wird lediglich eine Auswahl von vier Open Source Produkten als sinnvoll erachtet. Folglich wird kein vollständiges Bild über alle verfügbaren Open Source Produkte für Performance- und Lasttest erarbeitet.

2 Definition von Performance- / Lasttests

Einer der wichtigsten Aspekte für den Erfolg eines Produktes und einer Web Applikation ist dessen Performance,[1] weil Web Benutzer nicht Antwortzeiten warten wollen.[2] Aus diesem Grund ist es entscheidend zu wissen, wie sich die jeweilige Homepage oder IT Komponenten während des Betriebs und dem Zugriff von Usern verhalten.[3]
Last- bzw. Performancetesttools berechnen bei Ihrer Ausführung Kennzahlen über verbrauchte Kapazitäten und Ressourcen von Anwendungen. Laut der Inventage AG lässt sich durch dieses Vorgehen sicherstellen, dass vorhandene Systemressourcen den Bedürfnissen der jeweiligen Anwender gerecht werden. Mit Lasttests werden Skalierbarkeit, Durchsatz und Stabilität der folgenden Komponenten und Produkte analysiert:

- Anwendung
- Betriebssystem
- Applikationsserver
- Datenbank
- Netzwerk[4]

[1] Vgl. Inventage AG (2012)
[2] Vgl. Lucca, G./ Fasolino, A. (2006), S.65
[3] Vgl. Menascé, d. (2002), S.2
[4] Vgl. Inventage AG (2012)

Die Inventage AG versteht unter einem Lasttest einen (nicht funktionalen) Softwaretest, mit dem eine gewisse und zu erwartende Last auf dem laufenden System erzeugt und das Verhalten desselbigen beobachtet und untersucht wird. Meist werden hierzu Simulationen verwendet, deren Ziel es ist:

- Fehler aufzudecken, die im funktional orientierten Systemtest/Integrationstest nicht gefunden wurden.
- Nicht-funktionale Anforderungen (z. B. geforderte Antwortzeiten der Mengenverarbeitung) nachzuweisen.
- Die Dimensionierung der Hardwareausstattung zu überprüfen.[5]

Wird ein Lasttest außerhalb der normalen betrieblichen Belastung durchgeführt, wird von einem Stresstest gesprochen.[6] Mit einem Stresstest werden sowohl die Stabilität, als auch die Grenzen der Performance des Systems getestet[7]. Es können zudem Komponenten erkannt werden, die möglicherweise erweitert werden müssen, um die Performance zu erhöhen.[8] Wir ein Lasttest über einen längeren Zeitraum, beispielsweise 48 bis 72 Stunden, durchgeführt, wird von einem Dauerlasttest gesprochen.[9]

Nachdem unterschiedliche Kennzahlen ermittelt wurden, werden mithilfe dieser Kennzahlen verschiedene Lasttestskripts erstellt. Verschiedene Skripte, Definitionen und unterschiedliche Runtime Einstellungen werden in ihrem Zusammenschluss als Szenario bezeichnet.[10] Für einen erfolgreichen und aussagekräftigen Last- bzw. Performancetest wird eine gewisse Anzahl an Benutzern simuliert. Die verschiedenen Tools unterscheiden sich hierbei bei der möglichen Anzahl an zu simulierenden Benutzern. Laut der Inventage AG lässt sich eine erhöhte Systemlast erzeugen, indem die Wartezeit zwischen Interaktionen bewusst niedrig gehalten wird.

Ein Last- bzw. Performancetest ist einem funktionalen Test nachgelagert. Das bedeutet, dass das zu testende System, bzw. die Systemkomponente in einem funktional stabilen Zustand sein muss, um auf Lastbewältigung getestet werden zu können.[11]

[5] Vgl. Wikipedia (2012a)
[6] Vgl. Wikipedia (2012b)
[7] Vgl. Klinger,J. /Schipfer, G. (2004)
[8] Vgl. Inventage AG (2012)
[9] Vgl. Ebenda
[10] Vgl. Ebenda
[11] Vgl. Wikipedia (2012a)

3 Toolauswahl und Kriterienerläuterung

Last- und Performancetests sind für jede Art von Entwicklungsprojekten unabdingbar. Vor allem für Webanwendungen sind sie ein wichtiges Instrument, um die Stabilität und Funktionsfähigkeit eines Systems sicherzustellen. Aus diesem Grund ist es essenziell, bei der Auswahl eines Last- bzw. Performancetesttools die Anforderungen an das zu entwickelnden Systems zu bestimmen, um mögliche Fehlerquellen frühzeitig zu erkennen, zu eliminieren und nicht kalkulierten Kosten entgegenzuwirken. Im Open Source Bereich gibt es bereits eine Vielzahl an Tools, die verschiedenste Systeme oder Anwendungen von Web- bis Java Applikationen auf unterschiedlichste Arten testen können.[12] Jedes Tool bietet diverse Funktionen, die in Abhängigkeit des Anwendungsfalls mehr oder weniger hilfreich bzw. relevant sind. Jedoch sind die meisten Open Source Tools laut eines Artikels der Computerwoche nur bedingt für beliebige Systemlandschaften einsetzbar, sondern hauptsächlich für bestimmte Anwendungsbereiche, wie z. B. Web- und Java-Applikationen geeignet.[13] Dies resultiert aus der Tatsache, dass der Code von proprietären Protokollen für Open Source Tools in der Regel nicht zugänglich ist und somit Systeme, die auf solchen Protokollen basieren, nur schwer mit Open Source Tools getestet werden können. Aus diesem Grund werden in dieser Vergleichsstudie lediglich Tools vorgestellt, die in der Lage sind Webapplikationen zu testen.

Einige der Open Source Testtools umfassen neben den Performance- und Lasttests auch die Möglichkeit für funktionales Testing, Test Management, Fehlerverfolgungstools sowie Sicherheitstesttools.[14] Wie bereits in der Einleitung erwähnt, wird sich diese Ausarbeitung auf die Analyse der Testtools hinsichtlich ihrer Funktionalitäten für Performance- und Lasttests beschränken.

Eine umfangreiche Auflistung von sowohl Open Source funktionalen Testing Tools und Test Management Tools, als auch Fehlerverfolgungstools findet sich unter der folgenden Homepage: http://www.opensourcetesting.org

[12] Vgl. Bradtke, R (2008), S. 147
[13] Vgl. Billens, A. (2007)
[14] Vgl. Bradtke, R (2008), S. 147

3.1 Toolauswahl

Auf dem Markt existieren unterschiedlichste Werkzeuge zur Lastgenerierung auf Servern. Viele dieser Tools sind darauf ausgelegt, mit einer Vielzahl simpler Anfragen das System zu penetrieren.[15] Heutzutage jedoch sollten Systeme und vor allem Webanwendungen darauf ausgerichtet sein, eine hohe Anzahl an unterschiedlichsten Nutzeranfragen gleichzeitig zu managen. Daraus ergibt sich die Notwendigkeit an Lasttesttools, die Interaktion des Nutzers mit dem Testsystem möglichst realistisch zu simulieren.[16] Diese Testsituationen lassen sich unter Verwendung von Skripten und Testfällen abbilden. Da die Erzeugung von Testskripten und die parallele Nutzerverwaltung (Multithreading) ein sehr wichtiger Punkt bei der Auswahl eines Lasttesttools sind, war es der Anspruch dieser Vergleichsstudie, Testtools zu evaluieren, die diese Anforderungen erfüllen. Darüber hinaus sollten die zu evaluierenden Werkzeuge das Testen verteilter Anwendungen unterstützen, um eine Anwendung auf mehreren Systemknoten zu testen und dadurch möglichst realistische Testfälle zu simulieren.

Allen Open Source Programmen ist gemein, dass ihr Quellcode frei zugänglich ist und unter den unterschiedlichen Open Source Lizenzen weiterverbreitet werden dürfen. Weber beschreibt Eigentum im Open Source Umfeld wie folgt: „Property in Open Source is configured fundamentally around the right to distribute, not the right to exclude."[17]

Bei Open Source Lizenzen wird grundsätzlich zwischen der GNU General Public License (GPL) und der Berkeley Software Distribution (BSD) Copyright License unterschieden.
Software unter einer GNU GPL Lizenz darf lizenzgebührenfrei erworben, vervielfältigt, veröffentlicht und verändert werden, jedoch ist es nicht erlaubt, geänderte Programme proprietär weiterzuvertreiben.[18] Software, die der BSD Lizenz unterliegt, darf frei verwendet, kopiert, verändert und veröffentlicht werden und ist somit der GNU GPL Lizenz in den Grundzügen ähnlich. Ein Produkt unter der BSD Lizenz muss nicht quelloffen veröffentlicht werden.[19]

Es gibt unterschiedlichste Tools, die unter den beiden Open Source Lizenzen veröffentlicht sind. Einige der Tools werden von der Community nicht mehr weiterentwickelt, wie zum Beispiel OpenSTA, welches vor ein paar Jahren in diversen Rankings sehr gute Bewertungen

[15] Vgl. Schimrazki, O. (2009), S. 20
[16] Vgl. ebenda, S. 20
[17] Weber, S. (2004), S. 1
[18] Vgl. Weber, S. (2004), S. 6
[19] Vgl. ebenda, S. 6

erhielt. Jedoch wurden keine neuen Updates seit 2007 veröffentlicht. Es ist daher anzunehmen, dass dieses Tool folglich nicht unterstützt wird. Dies ist ein zu beachtendes Problem bei Wahl und Anwendung von Open Source Software, da die verantwortlichen Communities jederzeit Releases und Support einstellen können und so keine fortwährende Aktualität und Unterstützung gewährleistet werden können.

In Anhang 1 befindet sich eine Auflistung von verschieden Open Source Performance Testtools.

Um eine hohe Aktualität dieser Arbeit zu gewährleisten, wurden für die Vergleichsstudie nur Programme ausgewählt, von denen im letzten Jahr bzw. bereits dieses Jahr Updates veröffentlicht wurden. Es wurde eine Auswahl an vier Tools getroffen, die anhand der in 3.2 erläuterten Krieterien bewertet wurden: JMeter, The Grinder, PushtoTest TestMaker und FunkLoad. Aus Tabelle 1 geht hervor, wann die neuste Version der jeweiligen Tools veröffentlicht wurde.

	JMeter	The Grinder	TestMaker	Funk Load
Aktuellste Version	2.7 (Mai 2012)	3.9.2(Mai 2012)	Version 6.5 (Juni 2012)	1.16.1 (Juli 2011)

Tabelle 1: Übersicht über die Versionen der evaluierten Tools

Die Auswahl der Tools wurde basierend auf zwei unterschiedlichen Kriterien getroffen. Zum einen wurde nach der Funktionalität, die die Tools abdecken, ausgewählt und zum anderen, welche Tools bereits von Sekundärliteratur in Vergleichen behandelt wurden. Im Rahmen des ersten Kriteriums wurde entschieden, dass zwei Tools, TestMaker und FunkLoad, ausgewählt werden sollten, die speziell zum Testen von Webanwendungen geeignet sind und zwei Tools, die weit aus mehr Funktionalitäten wie Datenbank- und Middlewaretesting unterstützen und zudem auch Java Anwendungen testen können.
Unter dem zweiten Kriterium wurde während der Literaturrecherche eine Liste mit Tools aufgestellt, die bereits analysiert wurden (siehe Tabelle 2). Hierzu wurden sowohl diverse Diplom- und Bachelorarbeiten als auch Fachartikel zur Rate gezogen.

Sun Faban
The Grinder
PushToTest TestMaker
Apache JMeter
FunkLoad
OpenSTA
WebLoad Basic Edition
Dieseltest

Tabelle 2: Häufig behandelte Tools in der Literatur

Basierend auf den Erkenntnissen der Literaturrecherche und den eigens festgelegten Kriterien wurden der Apache JMeter, das wohl bekannteste Lasttesttool im Open Source Umfeld und The Grinder als Tools, welche eine hohe Bandbreite an Funktionalitäten mitliefern, ausgewählt. Außerdem werden diese beiden im Unterschied zum TestMaker von der Open Source Community effektiv weiterentwickelt, weshalb auch die aktuellsten Updates aus diesem Jahr stammen. Der TestMaker unterscheidet sich von allen anderen drei Tools dadurch, dass er durch das Unternehmen PushtoTest unterstützt und weiterentwickelt wird.

3.2 Kriterienerläuterung

Im Folgenden wird erläutert, welche Kriterien ausgewählt wurden, um die vorher genannte Auswahl an Last-und Performancetesttools zu bewerten. Hierbei wurden diverse, bereits existierende Vergleichsstudien herangezogen, um eine Liste relevanter Vergleichskriterien, zu erstellen.

Die Anforderungen an ein Bewertungssystem für Performance- und Lasttesttools sollten sich auf verschiedene Bereiche beziehen und alle relevanten Merkmale, die für Anwender ausschlaggebend sind, miteinbeziehen. Hierbei ist vor allem die Generierung und Durchführung der Tests ein essenziell zu bewertendes Merkmal.[20] Auch die unterschiedlichen Protokolle und Anwendungen, die sich mithilfe der Tools testen lassen, können ausschlaggebend für die Bewertung sein.

Wie bereits in der Einleitung erwähnt wurden vier Hauptkriterien Bedienbarkeit, Funktionalität, Technologie und Sonstiges definiert. Alle nicht zuordnungsfähigen Kriterien wurden unter dem Punkt *„Sonstiges“* zusammengefasst. Tabelle 14 in

[20] Vgl. Schimrazki, O. (2009), S. 33

Anhang 2 ordnet die Vergleichsmerkmale aus unterschiedlichen Quellen zu den festgelegten Hauptkategorien zu und veranschaulicht Überscheidungen zwischen den jeweiligen Kriterienlisten, die in bereits existierenden Vergleichsstudien verwendet wurden.

3.2.1 Bedienbarkeit

Aus Tabelle 14 in

Anhang 2 wird ersichtlich, dass das Kriterium *„Bedienbarkeit“* von keiner der Vergleichsstudien berücksichtigt wurde. Möglicherweise aus dem Grund, weil es sich hierbei um eine subjektive Einschätzung handelt. Diese Vergleichsstudie wird versuchen diesen Bereich basierend auf Sekundärliteratur für die ausgewählten Tools zu bewerten und einzuschätzen.
Unter Bedienbarkeit wurden Aspekte der einfachen, intuitiven und schnellen Benutzung des Tools für unerfahrene Tester betrachtet. Dabei wurde analysiert, ob das zu betrachtende Tool über eine graphische Oberfläche verfügt, oder über die Kommandozeile gesteuert wird. Auch die einfache Installation ist ein Kriterium, welches aussagekräftig für die Einschätzung der Bedienbarkeit eines Testtools ist. Tabelle 3 liefert einen Überblick über die in der Kategorie Bedienbarkeit evaluierten Merkmale.

1. Bedienbarkeit
Einfache Installation
Intuitive Bedienbarkeit
Graphische Oberfläche (GUI)
Schnelle Einarbeitung

Tabelle 3: Überblick über die Kategorie „Bedienbarkeit“

3.2.2 Technologie

Die Kategorie *„Technologie“* als Hauptkriterium wird in vielen bereits existierenden Vergleichsstudien abgehandelt. Dabei dient sie zunächst der Angabe, in welcher Sprache das Tool programmiert wurde und welche Scripting Sprache/n zur Erzeugung von Testfällen genutzt werden. Eine wichtige Voraussetzung für ein Lasttesttool ist, dass es mit den Systemvoraussetzungen des zu testenden Systems kompatibel ist. Aus diesem Grund sollte man sich im Vorfeld informieren, welche Protokolle unterstützt werden müssen, auf welchen Plattformen das System läuft und somit auch testbar sein muss, oder ob sogar ein gänzliche plattformunabhängig erforderlich ist. So sind manche der getesteten Tools plattformunabhängig, andere wiederum nur für Anwendungen unter Windows oder Linux geeignet. Für diese Vergleichsstudie wurde die Plattformabhängigkeit bzw. -unabhängigkeit unter Berück-

sichtigung der drei am weitesten verbreiteten Betriebssysteme Windows, Linux und Unix verglichen.
Ein weiteres Merkmal sind die vom Testtool unterstützten Protokolle, die es möglich machen, nicht nur Webanwendungen, sondern zum Beispiel auch Datenbanken und Java Applikationen zu testen. Hierbei wurden die folgenden gängigen Protokolle in die Vergleichsliste aufgenommen: das HTTP und HTTPs Protokoll, PoP3, Java Database Connectivity (JDBC) Protokoll für Datenbanken, File Transfer Protocol (FTP), Asynchronous JavaScript and XML (AJAX) und Secure Sockets Layer (SSL).

Des Weiteren fällt unter die Kategorie Technolgie die Möglichkeit, bei den Last- und Performancetesttools eine parallele Nutzerverwaltung (Multithreading) und verteilte Anwendungen zu simulieren. Vor allem die Erzeugung einer großen Anzahl an parallelen Nutzern, die unabhängig das System testen, ist eine wichtige Voraussetzung für ein Performance- und Lasttesttool. Ohne das Erreichen des gewünschten parallelen Testvolumens wird ein Lasttesttool nutzlos.[21] Durch die Erzeugung hoher paralleler Nutzerzugriffe wird ein extremer Anstieg von Protokolldaten erzeugt, die verwaltet und ausgewertet werden müssen. Da dies auch von kommerziellen Anbietern erkannt wurde, wird dieses Kriterium meist als Staffelungsmechnanismus für den Preis eines Lasttesttools verwendet. Bei der Auswahl der zu vergleichenden Tools wurde darauf geachtet, dass alle Tools dieses Kriterium vollständig erfüllen. Handelt es sich also um ein Projekt mit hohen parallelen Nutzerzahlen, lassen sich diese meist kaum durch die Leistung eines einzelnen Rechners abbilden. Um die gewünschte Last überhaupt erzeugen zu können, werden in der Regel die Lasten auf mehreren Rechner verteilt, um eine erhöhte Last auf den Servern zu erzeugen. Im Anschluss ist eine zentrale Zusammenführung aller Protokolldaten der generierten Lasten für die spätere Auswertung wichtig.

[21] Vgl. Stolze, J. (2008)

2. Technologie
Zugrundeliegende Technologien
Scriptingsprache
Plattformunabhängigkeit
Betriebsysteme
Windows
Linux
Unix
Testbare Protokolle
HTTP
HTTPS
POP3
JDBC
FTP
Ajax
SSL
Parallele Nutzerverwaltung (Multithreading)
Verteilte Anwendung

Tabelle 4: Überblick über die Kategorie „Technologie“

3.2.3 Funktionalität

Die Kategorie *„Funktionalität“* wird generell durch die zwei Hauptmerkmale Testing und Analyse in Tabelle 5 geprägt. Des Weiteren sind die Systeme und Anwendungen relevant, welche mit dem jeweiligen Tool testbar sind. Dabei beschränkt sich diese Vergleichsstudie auf vier wesentliche Systeme: Webservices, Webanwendung, Datenbanken und Middleware-Komponenten.

Die wichtigste Funktionalität eines Lasttesttools ist die Erzeugung von Testfällen bzw. Testskripte. Die Erstellung eines speziellen Testfalls für ein System kann zeit- und kostenaufwendig sein und Vorkenntnisse über das zu testende System erfordern. Aber nur die Durchführung passender Testfälle validieren Aussagen über die Konstanz und Performance eines Systems. Lasttesttools unterstützen die Generierung von Testskripten, jedoch gibt es Unterschiede bei der Umsetzung. Aus diesem Grund wurden Merkmale festgelegt, die Aussagen darüber zulassen, ob eine manuelle Testfallgereierung erforderlich ist, ob das Aufzeichnen eines Testfalls möglich ist und ob überhaupt ein integrierter Testfallgenerator vorhanden ist. Desweitern wird untersucht, ob sowohl client-seitige also auch server-seitige Tests möglich sind. Ein weiterer Punkt ist die Frage, ob die Abbildung komplexer Szenarien genau so möglich ist, wie die einfache Abfolge von Protokollabfragen.

Neben der Testfallgenerierung wird untersucht, wie die gewonnenen Testdaten im Anschluss an einen Test analysiert und interpretiert werden. Aufgrund der speziellen Anforderungen der getesteten Systeme ist dies nicht von den Lasttesttools selbst durchführbar, sondern sollte individuell gestaltet werden. Die Lasttesttools können den Anwender hier lediglich bei der Darstellung der Informationen unterstützen. Deshalb wird untersucht, in welchem Datenformat die Testergebnisse vorliegen. Dies können einfache Textdateien sein, eine graphische Darstellung der Informationen mittels Graphen und Diagrammen, das Erzeugen von Berichten, bis hin zu der Möglichkeit alle gesammelten Rohdaten in Tabellen oder Datenbanken zu exportieren.[22]

3. Funktionalität
Testbare Funktionalitäten
Webservices(XML)
Webanwendung
Datenbanken
Middleware
Testfallgenerierung & Scenariounterstützung
Manuelle Testfallgenerierung
Recording von Testfällen
Integrierte Testfallgenerierung
Client-seitiges Testing
Server-seitiges Testing
Parametrisierung
Komplexität der Szenarien
Auswertung&Genauigkeit der Testergebnisse
Graphische Ausgabe der Testergebnisse
Speicherung in Log Files
Exportierbarkeit in Spreadsheettools
Format der Auswertung

Tabelle 5: Überblick über die Kategorie „Funktionalität“

[22] Vgl. Stolze, J. (2008)

3.2.4 Sonstiges

Das letzte vergleichbare Merkmal ist gerade bei Open Source Tools die Support-Möglichkeiten durch Communities und eine Produkt-Dokumentation sowie die Frage, ob das Tool von der Community weiterentwickelt wird und ob es Support durch externe Firmen gibt. Zu dieser Kategorie gehört ebenfalls das Kriterium „externer Support“. Dieser bewertet, in welchem Maße Kurse und Schulungen für ein Produkt angeboten werden und, ob externe Firmen bei der Konfiguration unterstützen bzw. generellen Support zur Verfügung stellen.

4. Sonstiges
Support
Dokumentation
Weiterentwicklung
Externer Support
Aktuellste Version
Community

Tabelle 6: Überblick über die Kategorie „Sonstiges“

4 Produktvorstellungen

4.1 Apache JMeter

Das Programm JMeter von Apache basiert vollständig auf Java. Die Desktopanwendung wurde entwickelt, um Last- und Performancetests für Client/Server-Architekturen durchzuführen. Es unterstützt sowohl das Testen von statischen, als auch dynamischen Ressourcen.[23] Laut Apache sind Tests durchführbar mit:[24]

- Statischen Dateien,
- Java Servlets,
- Common Gateway Interface (CGI) Skripten,
- Java Objekte,
- Datenbanken,
- FTP-Server.

Laut der Produktbeschreibung ist JMeter dafür geeignet Server und deren Performance genau zu testen, da eine hohe Last auf dem zu testenden Server erzeugt werden kann. Im JMeter Wiki,[25] welches viele wertvolle Informationen über den JMeter enthält, wird zu dieser Last ein Vergleich aufgestellt: Es wird beschrieben, dass JMeter eine weitaus höhere Last auf einem System generieren kann, als ein normaler Benutzer, da Anfragen schneller durchgeführt werden. Ein durchschnittlicher Benutzer wartet ungefähr 10 Sekunden, bis er auf einen neuen Link klickt. Der JMeter wartet eine Sekunde und kann somit die zehnfache Last generieren. Zudem kann das Tool Anwendungen auf korrekte Rückgabewerte testen, indem Testskripte erstellt werden, die eine Validierung der vorhandenen Rückgabewerte vornehmen.[26]

4.1.1 Voraussetzungen

Ursprünglich wurde der JMeter für Windows entwickelt, mittlerweile finden sich jedoch auch Unix (Linux, Solaris etc.) Versionen. Um die Version von JMeter (1.9.1) selbst zu kompilieren bzw. auszuführen, wird mindestens das Java Development Kit (JDK) 1.4 benötigt. Als Extensible Markup Language (XML) Parser wird standardmäßig der Xerces XML Parser3 verwendet, jedoch ist die Verwendung eines anderen Parsers ebenso möglich. Falls Webseiten getestet werden sollen, die eine Verschlüsselung verwenden, muss eine Java-SSL-

[23] Vgl. Stoll, C./ Pommerening, T. (2004), S. 6
[24] Vgl. Apache Software Foundation (2012)
[25] Vgl. JMeter Wiki (2011)
[26] Vgl. Stoll, C./ Pommerening, T. (2004), S. 6

Implementierung installiert werden.[27] Da die Version 1.9.1 das Java Runtime Environment (JRE) 1.4 benötigt, kann die darin integrierte JSSE (Java Secure Socket Extension) verwendet werden.

4.1.2 Bedienbarkeit

Laut Testern (diese Einschätzung basiert auf eigenen Erfahrungen sowie der anderer Nutzer) des Produktes ist die Installation das Apache JMeter sehr einfach. Das Programm muss zunächst im gewünschten Verzeichnis entpackt werden und anschließend die im bin-Ordner zu findende Datei JMeter.bat (bei Windows) oder das JMeter Skript (bei Unix) ausgeführt werden. Wenn das JDK 1.4 von Sun verwendet wird, werden keine weiteren Patches oder .jar Pakete benötigt. Zur Deinstallation muss lediglich der JMeter Ordner gelöscht werden, da es für den JMeter keine Einträge in der Windows Registry oder Dateien in Systemverzeichnissen gibt. Die Steuerung der Anwendung ist sowohl über ein dynamisches Graphical User Interface (GUI), als auch über ein statisches, jedoch sehr schnelles Call Level Interface (CLI) möglich.[28] Die Bedienung des Tools über die GUI (siehe Abb. 1) geschieht meist intuitiv und benötigt keine lange Einarbeitungszeit. Da die Menüs eingeschränkte Auswahlmöglichkeiten zur Verfügung stellen, ist die grafische Oberfläche sehr übersichtlich und erleichtert zusätzlich die Bedienbarkeit. Laut den Erfahrungen einiger Nutzer werden bei großen und schweren Lasttests viele Ressourcen des Hauptspeichers verbraucht, was die Reaktionszeit des Programmes und damit auch die Bedienbarkeit deutlich erschwert.

[27] Vgl. Stoll, C./ Pommerening, T. (2004), S. 7
[28] Vgl. ebenda, S. 8

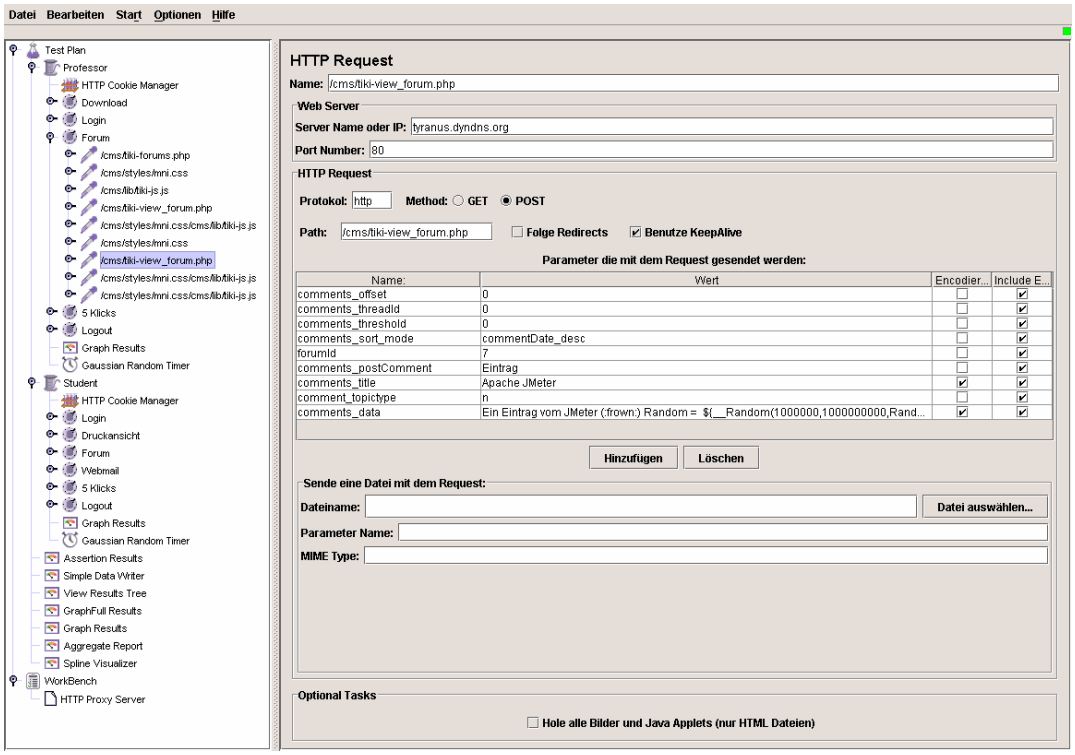


Abb. 1: Apache JMeter GUI[29]

4.1.3 Technologie

Mithilfe des JMeters können sowohl Java-, als auch reine HTML-Seiten getestet werden.[30] Da es sich, wie bereits zu Beginn beschrieben, um eine auf Java basierende Anwendung handelt, ist der JMeter auf allen Systemen einsetzbar, die eine JVM (Java Virtual Machine) implementiert haben. JMeter ist in der Lage Java APIs zu verwenden. Dadurch kann er folglich HTTP, HTTPS, Simple Object Access Protocol (SOAP), FTP, JAVA, LDAP, Java Message Service (JMS), POP3, JUnit und Datenbankzugriffe per JDBC zur Lastgenerierung verwenden. Was JMeter allerdings nicht leisten kann, ist der Umgang mit JavaScript und damit sämtlicher AJAX-Umgebungen. [31]

[29] Enthalten in: Stoll, C./ Pommerening, T. (2004), S. 10
[30] Vgl. JMeter Wiki (2011)
[31] Vgl. Schimratzki, O. (2009), S. 24

4.1.4 Funktionalität

Die Aufzeichnung der Interaktion auf Webseiten erfolgt über einen Proxy Server. Diese Aufzeichnung wird anschließend als Baumstruktur angezeigt. Als Skriptsprache dient XML, wodurch Skripte auch manuell angepasst werden können. Jedoch sind keine XML Kenntnisse nötig, da der Benutzer Skripte auch über die JMeter Einstellungen schnell und einfach bearbeiten kann. Durch XML-Verschachtelungen werden Blöcke gruppiert. Innerhalb des Tools wird dies durch verschiedene Controller-Elemente realisiert. Standard-Controller z. B. dienen nur als Gruppierung, darüber hinaus gibt es auch Aufnahme-Controller, die die Proxy-Daten aufzeichnen oder Zufalls-Controller, die für jeden Durchlauf einen zufälligen Eintrag aus ihrer Liste auswählen. Bereits gespeicherte Skripte können einfach in die Testumgebung geladen werden. Jedes Skript bildet dabei eine sogenannte Thread-Gruppe, die unabhängig oder gleichzeitig mit anderen Skripten ausgeführt werden kann. Für jede dieser Thread-Gruppen kann festgelegt werden, wie viele Benutzer simuliert und wie oft die Skripte wiederholt werden sollen. JMeter erlaubt es zudem, Eingabewerte zu parametrisieren. Dabei können verschiedene Funktionen, wie beispielsweise die random() Funktion, verwendet werden, die eine Zufallszahl aus einem gewünschten Bereich zurückgibt. Möglich ist es zudem bereits aufgezeichnete Skripte manuell oder automatisch zu erweitern. Das Trennen oder Zusammenführen von Skripten kann jedoch nur manuell durchgeführt werden. Die Auswertung von Tests fällt laut Testern „äußerst dürftig“ aus.[32] Zwar können Testergebnisse und sogar Graphen im XML-Format abgespeichert werden, jedoch wurde die Anzeige und das automatische Auswerten als mangelhaft bewertet. In einem Graphen lassen sich unterschiedliche Werte, wie „Daten“; „Durchschnitt“, „Median“, „Abweichung“ und „Durchsatz“ anzeigen, siehe Abb. 2.

[32] Vgl. Stoll, C. / Pommerening, T. (2004), S. 9

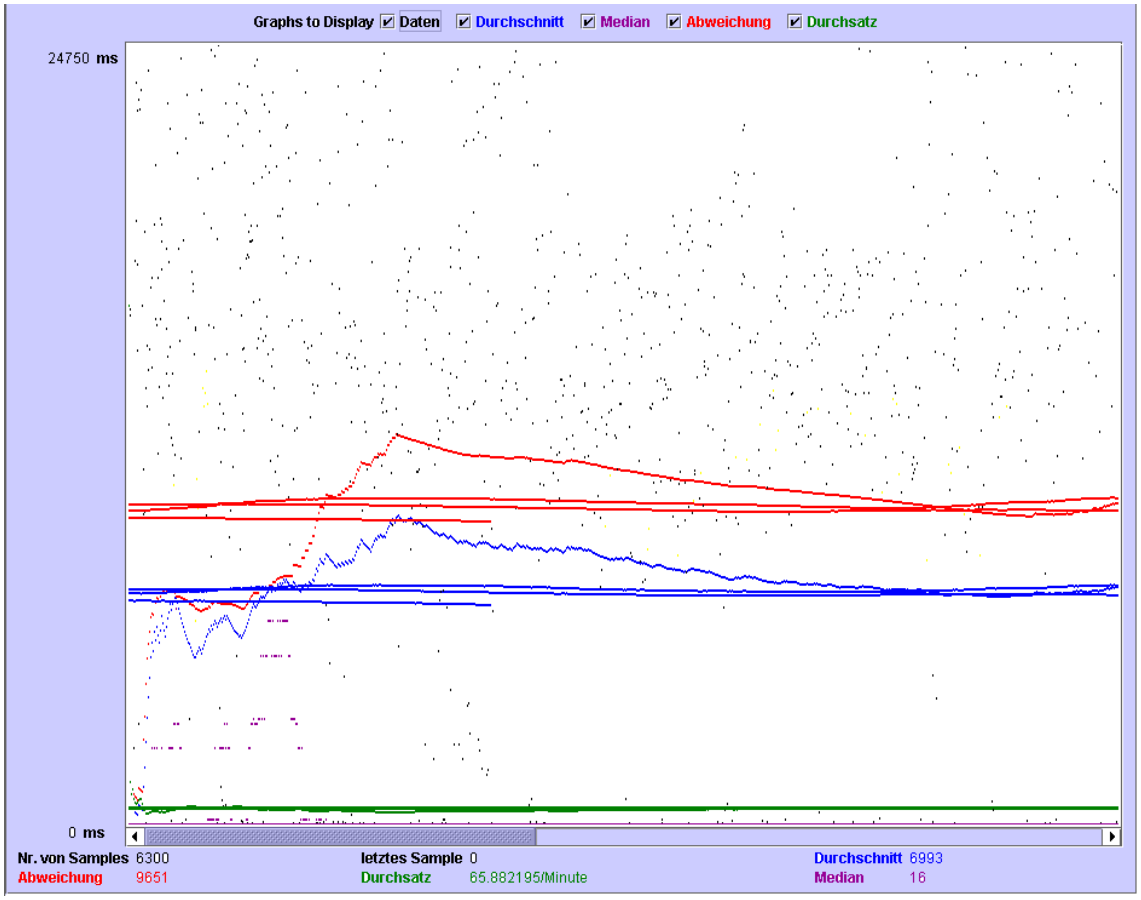


Abb. 2: Graphische Auswertung JMeter[33]

Leider ist der Graph nicht veränderbar und es bleibt unklar, welche Werte auf der x-Achse abgetragen werden. Zudem können keine Vergleiche von unterschiedlichen Testreihen durchgeführt werden. Nichtsdestotrotz konnten die Ergebnisse in ein Excel Sheet geladen werden und dort präzise ausgewertet werden. JMeter bietet jedoch eine Reihe von Plug-ins an, mit deren Hilfe unter anderem die graphische Darstellung verbessert werden kann. Ein Beispiel für ein solches Plug-in ist der Composite Graph. Mit diesem Plug-in können in einem Graph verschiedene Zusammenhänge visualisiert werden.[34] Beispielsweise lassen sich in einer Auswertung sowohl die Auswirkungen der Benutzeranzahl, als auch der Antwortzeit darstellen (siehe Abb. 3). Messdaten über die getestete Maschine oder das Betriebssystem kann JMeter nicht anzeigen.

[33] Enthalten in: Stoll, C./ Pommerening, T. (2004), S. 11
[34] Vgl. JMeter-plugins (2011)

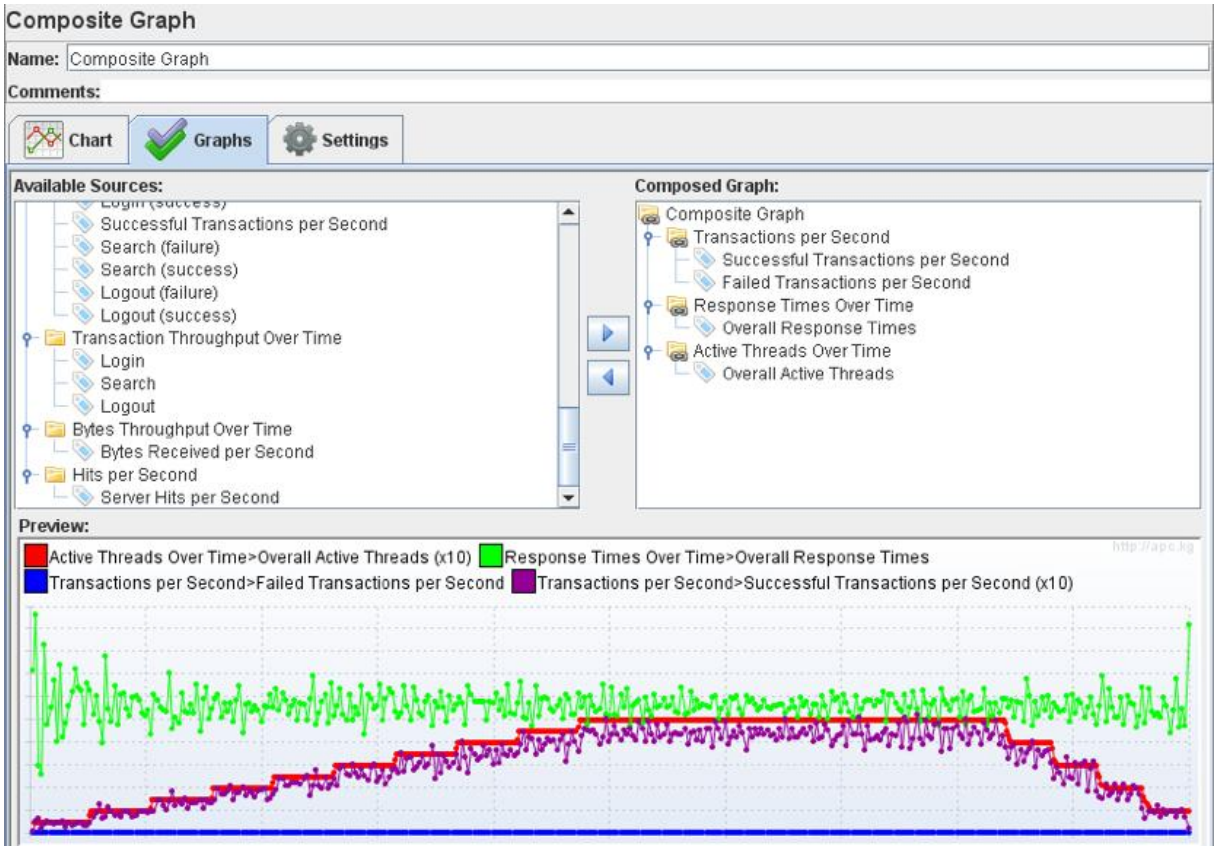


Abb. 3: JMeter Plug-ins: Composite Graph[35]

4.1.5 Sonstiges

Als Support werden lediglich Mailinglisten angeboten. Da jedoch die Entwickler der Anwendung selbst aktiv auf Fragen per Mail reagieren, ist anzunehmen, dass Support schnell und einfach zur Verfügung steht.

Durch eine umfangreiche Dokumentation,[36] welche in Englisch zur Verfügung steht, können schnell kleinere Probleme gelöst und verschiedene Fragen beantwortet werden. Zudem steht ein Tutorium zur Verfügung, welches die Einarbeitungsphase deutlich erleichtert. Darüber hinaus steht eine Onlinehilfe zur Verfügung, die übersichtlich aufgebaut ist und als Referenz hinzugezogen werden kann.

Einige Unternehmen und Hochschulen bieten für den JMeter verschiedene Kurse und Seminare an. Innerhalb dieser können sich Anwender über unterschiedliche Funktionen informieren und lernen den Umgang mit dem Tool. Diese Firmen unterstützen zudem teilweise bei der Konfiguration.

[35] Enthalten in: JMeter-plugins (2011)
[36] Vgl. The Apache Software Foundation (2012)

Vergleichskriterien	JMeter
1. Bedienbarkeit	
Einfache Installation	!
Intuitive Bedienbarkeit	✗
Graphische Oberfläche (GUI)	✗
Schnelle Einarbeitung	!
2. Technologie	
Zugrundeliegende Technologien	Python, pyUnit
Scriptingsprache	Python
Plattformunabhängigkeit	✗
Betriebsysteme	
Windows	!
Linux	✓
Unix	!
Testbare Protokolle	
HTTP	✓
HTTPS	✓
POP3	✗
JDBC	✗
FTP	✗
Ajax	✗
SSL	✓
Parallele Nutzerverwaltung (Multithreading)	✓
Verteilte Anwendung	✓
3. Funktionalität	
Testbare Funktionalitäten	
Webservices(XML)	!
Webanwendung	✓
Datenbanken	✗
Middleware	✗
Testfallgenerierung & Scenariounterstützung	
Manuelle Testfallgenerierung	✓
Recording von Testfällen	✓
Integrierte Testfallgenerierung	✗
Client-seitiges Testing	✓
Server-seitiges Testing	✓
Parametrisierung	✓
Komplexität der Szenarien	✓
Auswertung&Genauigkeit der Testergebnisse	
Graphische Ausgabe der Testergebnisse	✗
Speicherung in Log Files	!
Exportierbarkeit in Spreadsheettools	✓
Format der Auswertung	pdf, xls, html
4. Sonstiges	
Support	✓
Dokumentation	!
Weiterentwicklung	✓
Externer Support	✗
Aktuellste Version	1.16.1 (Juli 2011)
Community	Nuxeo

Tabelle 7: Bewertung des Apache JMeter anhand der Vergleichskriterien

4.2 The Grinder

Der Grinder ist ein Java basiertes Lastest-Framework, der unter der BSD Lizenz frei verfügbar ist. Die erste Version wurde für das Buch "Professional Java 2 Enterprise Edition with BEA WebLogic Server" von Paco Gómez und Peter Zadrozny entwickelt und danach von Philip Aston zur zweiten Version weiterentwickelt. [37]
Das Framework ist ein universelles Testframework und stellt Komponenten für Last- und Performancetests für Webanwendungen zur Verfügung. Darüberhinaus lassen sich auch Webservices, Datenbanken und beliebiger Java Code testen.
Für den Vergleich von Open Source Performance- und Lasttesttools wurde dieses Framework ausgewählt, da es Tests auf mehreren Maschinen ermöglicht. Die Tests können in einem Netzwerk verteilt und koordiniert durchgeführt werden, um die simulierte Last zu erhöhen und möglicherweise mehrere Szenarien parallel zu testen.

4.2.1 Bedienbarkeit

Die Bedienbarkeit von Grinder 2 und 3 ist einfach gehalten. Beide Versionen lassen sich leicht installieren und verfügen über eine grafische Benutzeroberfläche zur Steuerung und zum Monitoren von Tests. Jedoch existiert keine benutzerfreundliche Oberfläche zum Skripten und Coden von Tests. Ein weiterer Nachteil ist, dass die einzelnen Workerprozesse auf jedem Lastclient aus der Kommandozeile gestartet werden müssen. Es existiert ferner keine dedizierter Benutzeroberfläche zur Testauswertung.

4.2.2 Technologie

Die Tests werden in der Scriptingsprache Jython erstellt, welche eine JavaVM-basierte Implementierung der Programmiersprache Python darstellt.
Es lassen sich Tests basierend auf den Protokollen HTTP und HTTPs durchführen sowie JDBC Schnittstellen zum Testen von Datenbanken nutzen. Ferner werden Webservices via SOAP und RPC und MOM basierte Middleware via XML- Remote Procedure Calls (RPC) unterstützt. Andere Internetanwendungen, die POP3, Simple Mail Transfer Protocol (SMTP), FTP und andere Protokolle verwenden, können ebenfalls getestet werden.

Der entscheidende Vorteil des Grinders ist, dass er im Rahmen der Verfügbarkeit von J2SE1.4[38] komplett plattformunabhängig ist[39] und somit auch unter Unix bzw. Linux und Windows ausführbar ist.

[37] Vgl. Dempfle, M. (2007), S. 74
[38] Java Plattform Standard Edition ist eine Sammlung von Java APIs
[39] Vgl. Schimrazki, O. (2009), S. 22

Im Rahmen einer Lastteststudie von Webanwendungen mittels The Grinder an der Uni Stralsund wurden ein Lasttest mit 400 virtuellen Usern verteilt auf vier Agenten und ein Stresstest mit 1000 virtuellen Usern simuliert. Dies soll verdeutlichen, in welchen Dimensionen sich mit The Grinder Userinteraktionen simulieren lassen.

4.2.3 Funktionalität

Die Testfallgenerierung mittels des Grinders erfolgt über das Schreiben oder Aufnehmen von Skripten in der Sprache Jython.
Der Grinder 2 hatte den Nachteil, dass Testfälle nicht aufgenommen und somit auch die Testskripte nicht integriert erstellt werden konnten. Testskripte mussten manuell in einem Editor erstellt und konfiguriert werden.
The Grinder 3 wurde stark erweitert und bietet die Möglichkeit Testskripte auszuführen, die in Java oder in Jython geschrieben sind. Desweiteren können nun Testskripte über einen Transmission Control Protocol (TCP) Proxy aufgenommen und anschließend zum Beispiel durch des Eclipse Plug-in GrinderStone nachbearbeitet werden. Der TCP Proxy ist ein Testfallgenerator zur automatischen Erstellung von Grinder-Testskripten. Somit ist die integrierte Aufnahme von Testfällen in diesem Tool nur bedingt über Erweiterungen gegeben.

Während der Durchführung der Tests werden verschiedene Log Files erstellt, mit dessen Daten die im System vordefinierten Graphen zur Visualisierung von Antwortzeiten und Datendurchsätzen erzeugt werden (siehe Abb. 4). Dabei werden die Anzahl der Anfragen, die Anzahl der Fehler, die Minimum-, Maximum- und Durchschnittsantwortzeiten und die Anzahl der Tests pro Sekunde für jeden Test angezeigt und gespeichert.

Alle aufgetretenen Fehler werden in einer Error-Datei gespeichert und können später separat in einem Spreadsheettool wie Microsoft Excel ausgewertet werden. Desweiteren wird eine Zusammenfassung erstellt, die alle Tests mit der jeweiligen Anzahl der erfolgreichen und fehlerhaften Durchläufe und die dazugehörigen mittleren Testzeiten in Millisekunden und die Teststandardabweichung in Millisekunden auflistet. Im Allgemeinen wird die Ausgabe der Messwerte des Grinders als schwach eingeschätzt, da ein Großteil der Informationen nur in Logdateien vorliegt und durch zusätzliche Tools nachträglich manuell grafisch aufbereitet werden muss. Daher sind für die grafischen Auswertungen externe Tools nötig, wie zum Beispiel das Open Source Zusatzprogramm „Grinder Analyzer“, das die Auswertung von Standard Log-Dateien ermöglicht. Es wird empfohlen solche Programme zu nutzen, da konventionelle Programme, wie z. B. Microsoft Excel die Masse der anfallenden Testdaten nicht

adäquat verarbeiten können und eine Aggregation der Daten zu einer Verfälschung der Ergebnisse führen kann.[40]

Der Grinder erzeugt Last auf einem System und protokolliert die Zeiten, die einzelne Methoden auf der Client-Seite zur Ausführung benötigen. Es werden jedoch keine Systemmessungen auf der Serverseite unterstützt.[41]

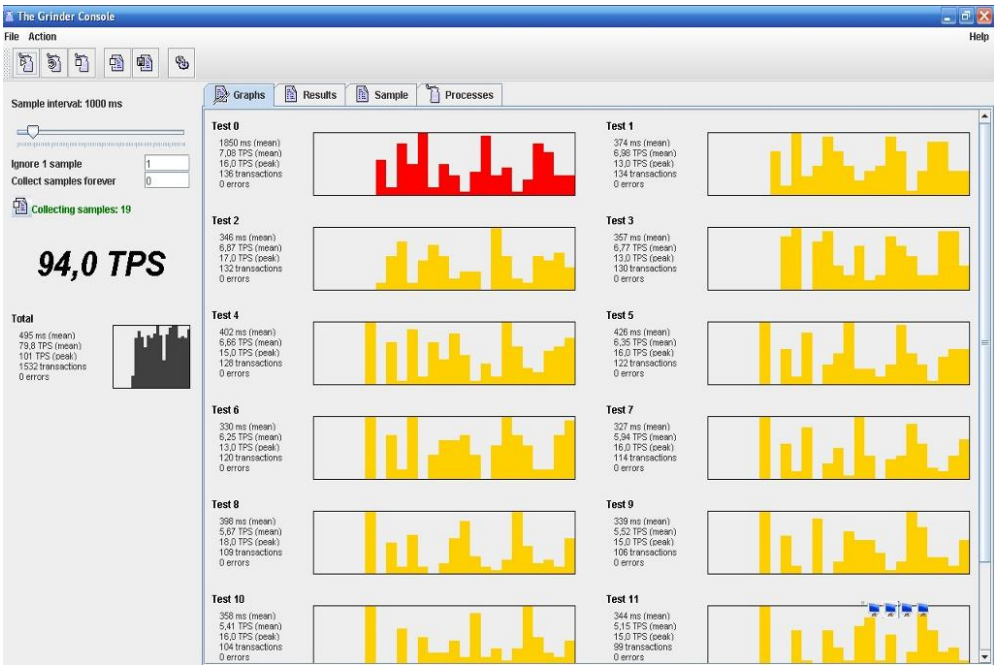


Abb. 4: Visualisierung der Testergebnisse durch „The Grinder“

4.2.4 Sonstiges

Es gibt bereits einige Unternehmen, die Kurse und Seminare zum Thema Lasttests mithilfe von Grinder anbieten. Im Unterschied zum TestMaker gibt es kein dediziertes Unternehmen, das sich mit der Weiterentwicklung des Tools beschäftigt. Dafür widmet sich die Open Source Community „A Developer Group“ dieser Aufgabe. Außerdem existieren Mailing Listen, die genutzt werden können, um Teil der Grinder Community zu werden. Nachfolgend eine Auswahl:

- ***grinder-announce:*** Low-volume notifications of new releases
- ***grinder-use:*** The place to ask for help
- ***grinder-development:*** For those interested in developing The Grinder [42]

[40] Vgl. Falk, R./ Salchow, M./ Diederichs, J./ Kasper, T. (2009)
[41] Vgl. Dempfle, M. (2007), S. 73
[42] Vgl. Aston, P./ Fitzgerald, C. (2012)

Die Dokumentation des Grinder ist sehr umfangreich und umfasst kommerzielle Literatur:

User Guide	http://grinder.sourceforge.net/g3/whats-new.html
FAQs	http://grinder.sourceforge.net/faq.html
Tutorial	http://grinder.sourceforge.net/g3/tutorial-perks.html
Script Gallery	http://grinder.sourceforge.net/g3/script-gallery.html
Articles	http://grinder.sourceforge.net/links.html
Commercial books	*Professional Java 2 Enterprise Edition with BEA WebLogic Server J2EE Performance Testing*[43]

Tabelle 8: Literatur zur Dokumentation von „The Grinder“

4.2.5 The Grinder Architektur

Zum Abschluss wird auf die Architektur des Grinders eingegangen. Im Allgemeinen lässt sich die Architektur in drei verschiedene Prozessgruppen unterteilen:

1. Konsolenprozesse: Dienen der Aggregation und Koordinierung aller externen Prozesse, um Statusinformationen und eine Übersicht der Messergebnisse anzuzeigen.
2. Agentenprozesse: Beschreiben die Verwaltungsprozesse, die separat auf jedem Client-Rechner gestartet werden müssen. Diese Prozesse starten die Workerprozesse/ Tests und sind für deren Überwachung verantwortlich.
3. Workerprozesse: Dies sind Prozesse, die der eigentlichen Durchführung der Tests dienen. Die vorkonfigurierten Module werden von diesen Prozessen geladen, konfiguriert und durchgeführt.[44]

Eine Testinstanz besteht aus einem Agent Prozess und einem oder mehreren Worker Prozessen.

Der Agent dient der Administration der einzelnen Worker und erhält seine Befehle und die zu startenden Testskripte von der Konsole. Hierzu verbindet sich der Agent nach dem Start mit der Konsole und wartet auf deren Befehle. Der Agent startet nach Befehl der Konsole - entsprechend seiner Konfiguration - die Worker und übermittelt diesen die Anzahl der zu startenden Worker-Threads, das Testskript und einige andere Parameter.

Die Workerprozesse beginnen entsprechend ihrer Parameter mit den im Testskript definierten definierten Anfragen an das zu testende System. Die Ergebnisse der Anfragen, wie z. B. Antwortzeiten werden in einer Log Datei gespeichert und an die Konsole übermittelt (siehe

[43] Vgl. Aston, P./ Fitzgerald, C. (2012)
[44] Vgl. Dempfle, M. (2007), S. 74

Abb. 5).

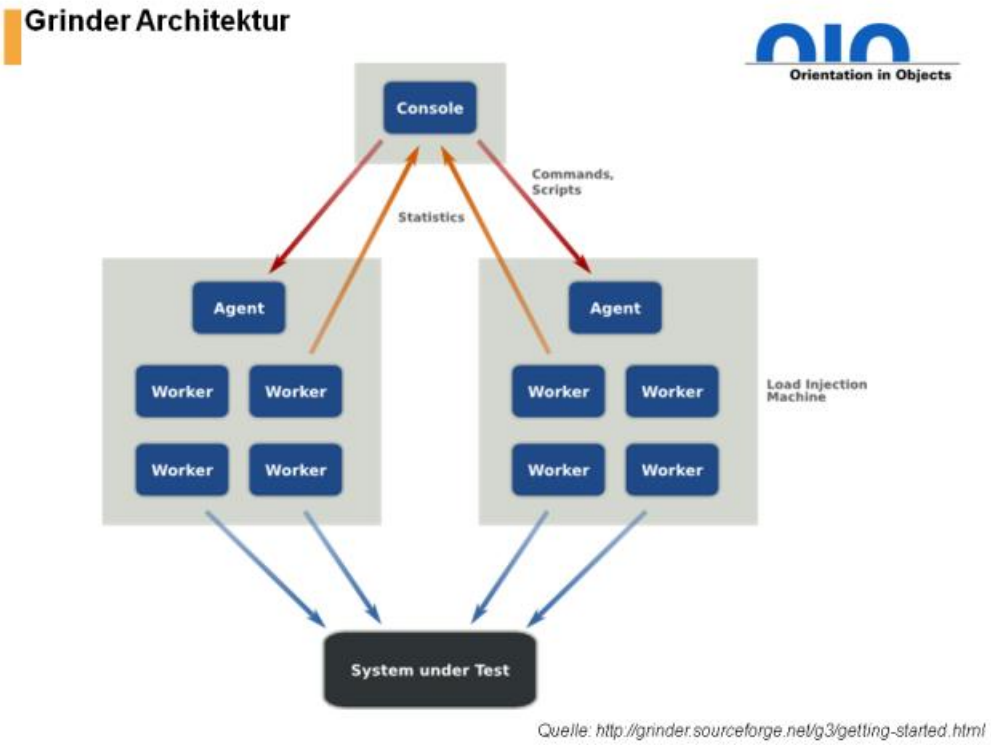


Abb. 5: The Grinder Architektur[45]

[45] Enthalten in: MyCore (2010)

Vergleichskriterien	The Grinder
1. Bedienbarkeit	
Einfache Installation	✓
Intuitive Bedienbarkeit	!
Graphische Oberfläche (GUI)	!
Schnelle Einarbeitung	✓
2. Technologie	
Zugrundeliegende Technologien	Java, Python
Scriptingsprache	Jython
Plattformunabhängigkeit	✓
Betriebsysteme	
Windows	✓
Linux	✓
Unix	✓
Testbare Protokolle	
HTTP	✓
HTTPS	✓
POP3	✓
JDBC	✓
FTP	✓
Ajax	✗
SSL	✗
Parallele Nutzerverwaltung (Multithreading)	✓
Verteilte Anwendung	✓
3. Funktionalität	
Testbare Funktionalitäten	
Webservices(XML)	✓
Webanwendung	✓
Datenbanken	✓
Middleware	✓
Testfallgenerierung & Scenariounterstützung	
Manuelle Testfallgenerierung	✓
Recording von Testfällen	!
Integrierte Testfallgenerierung	!
Client-seitiges Testing	✓
Server-seitiges Testing	✗
Parametrisierung	✓
Komplexität der Szenarien	✓
Auswertung&Genauigkeit der Testergebnisse	
Graphische Ausgabe der Testergebnisse	!
Speicherung in Log Files	✓
Exportierbarkeit in Spreadsheettools	✓
Format der Auswertung	xls
4. Sonstiges	
Support	✓
Dokumentation	✓
Weiterentwicklung	✓
Externer Support	!
Aktuellste Version	3.9.2(Mai 2012)
Community	A Developer Group

Tabelle 9: Bewertung von „The Grinder“ anhand der Vergleichskriterien

4.3 PushtoTest TestMaker

Der TestMaker ist eine Open Source Plattform des Unternehmens PushtoTest, mit der sowohl Funktionstest, als auch Performance- und Lasttests durchgeführt werden können.[46] Auf der Webseite des Unternehmens stehen zwei Versionen des TestMakers zum Download bereit: zum einen die Version „TestMakerCommunity“, die dem individuellen Nutzer ein Tool zum Testen von Webanwendungen und sogenannten Rich Internet Applications zur Verfügung stellt. Damit können Funktions-, Load- und Performancetests sowie Produktionsmonitorings durchgeführt werden. Des Weiteren gibt es die Version „TestMakerEnterprise“ die für den Unternehmenseinsatz und das Testen von großen geschäftskritischen Web- und Rich Internetanwendungen gedacht ist. Beide Versionen des Tools können kostenlos heruntergeladen werden. In der Version „TestMakerCommunity“ können jedoch nur 50 virtuelle Benutzer simuliert werden. Um weitere virtuelle Benutzer simulieren zu können, verlangt das Unternehmen PushtoTest eine Lizenzgebühr pro Nutzer. Die Version „TestMakerEnterprise“ kann innerhalb der ersten 15 Tage kostenlos genutzt werden und beinhaltet eine voll funktionsfähige Version des Programms sowie den vollen Service des Unternehmens. Ab dem 15 Tag muss für die Unternehmensversion eine Lizenzgebühr an PushtoTest entrichtet werden.[47]
Der Quellcode des TestMakers ist unter der General Public License v2 frei verfügbar. Neben den beiden genannten vorkonfigurierten Versionen, die nach dem Herunterladen direkt installiert werden können, können die Nutzer des TestMakers mithilfe des Quellcodes ihr eigenes Programm kompilieren, dass allerdings nicht durch PushtoTest unterstützt wird. Voraussetzung dafür sind laut Homepage des Herstellers Expertenkenntnisse in Java, NetBeans, XML, Web Services und im Testen von Anwendungen.[48]

4.3.1 Voraussetzungen

Der PushtoTest TestMaker ist javabasiert und benötigt die Java Runtime Environment.[49] In der aktuellen Version 6 des TestMakers benötigt man die Javaversion 1.6. Laut der Homepage des Herstellers liefern die Installationsdateien alles mit, was man zur Installation und zum Ausführen des Programms benötigt. Zur Ausführung von verteilten Tests muss jedoch die Zusatzkomponente TestNode heruntergeladen werden, die auf Servern und in virtuellen Maschinen installiert werden kann. Des Weiteren empfiehlt es sich, den PushtoTest TestMaker Monitor (PTTMonitor) herunterzuladen, um die Möglichkeiten des Monitorings der

[46] Vgl. Schimrazki, O. (2009), S. 25
[47] Vgl.PushToTest (2012)
[48] Vgl. ebenda
[49] Vgl. Stoll, C. / Pommering, T. (2004), S. 12

Auslastung von Central Processing Unit (CPU), Hauptspeicher und des Netzwerks nutzen zu können.[50]

4.3.2 Bedienbarkeit

Die Installation des TestMaker ist sehr einfach und ohne Vorwissen über das Produkt möglich. Der Benutzer muss lediglich das Programm auf der Seite des Herstellers in der gewünschten Version herunterladen, die Installationsdateien entpacken und anschließend das Programm ausführen.[51] PushtoTest stellt die Installationspakete für Windows (32 und 64 bit), Linux (32 bit) und Mac OS X zur Verfügung.
Die allgemeine Bedienung des Programms zur Generierung von Testfällen sowie der Durchführung von Tests gestaltet sich relativ einfach und intuitiv. Dem TestMaker fehlt jedoch die Möglichkeit, ein Skript zur Kontrolle auszuführen und anschließend Informationen zum Debugging zu erhalten. Im Allgemeinen nachteilig ist zudem die spärliche Hilfe, die dem Nutzer nur wenige und unkonkrete Informationen anbietet und im Zweifel auf den Hersteller verweist. Hieraus könnte man ableiten, dass das Unternehmen PushtoTest durch Service-Angebote und durch Schulungen Umsätze mit dem TestMaker erzielen möchte.[52]

4.3.3 Technologie

Mit dem TestMaker können Funktions- und Lasttests für komplexe Webanwendungen, die AJAX, Flash und Flex verwenden, durchgeführt werden. Da das Programm zu 100 % java-basiert ist, läuft es auf allen Systemen, die Java unterstützen.[53] Die Ausführung des Programms ist dadurch unter allen gängigen Betriebssystemen, wie Windows, Linux und Mac OS problemlos möglich,
Die Testfälle können im TestMaker in den folgenden Sprachen geschrieben werden: Java, Jython, Python, Ruby, Groovy und JavaScript. Um die erstellten Skripte zu übertragen, können die folgenden Protokolle genutzt werden: HTTP, HTTPS, SOAP, XML-RPC, SMTP, POP3 und Internet Access Message Protocol (IMAP).[54]

[50] Vgl. PushtoTest (2012)
[51] Vgl. Stoll, C. / Pommering, T. (2004), S. 12
[52] Vgl. ebenda, S. 13
[53] Vgl. PushtoTest (2012)
[54] Vgl. Schimrazki, O. (2009), S. 25

4.3.4 Funktionalität

Der TestMaker bietet grundsätzlich vier unterschiedliche Möglichkeiten um Testfälle zu generieren:

1. Skripte können über die Firefox-Erweiterung „TestGen4Web“ und „Silenium IDE“ erzeugt werden. „TestGen4Web“ ermöglicht über die intuitiv verständlichen Symbole eines Kassettenrekorders die Möglichkeit Testfälle im Browser aufzuzeichnen. Klicks auf Buttons und Checkboxen werden mittels XML Path Language (XPath) Ausdruck identifiziert. Dabei handelt es sich um eine Abfragesprache, die Teile eines XML Dokuments adressiert. Es spielt dabei keine Rolle, welches Protokoll (http oder https) zur Übertragung der Kommunikationsdaten verwendet wird, da die Interaktion mit der Seite direkt über XPath umgesetzt wird. Nachteilig daran ist, dass Interaktionen mit dynamisch generierten Inhalten unmöglich sind, da diese Änderungen am HTML Code nicht erfasst werden können.[55]
2. Bei der „Silenium IDE“ handelt es sich um eine Entwicklungsumgebung für Seleniumtests, die speziell für Webanwendungen entwickelt wurde. Sie wurde speziell für den Browser Mozilla Firefox entwickelt, weshalb die Aufnahme von Tests folglich browserabhängig ist. Bei Aktivierung des Aufzeichenmodus werden die Aktionen des Benutzers im Browser in Form eines sogenannten „Command-Target-Value Tripels“ registriert. Bei „Commands“ handelt es sich um Aktionen wie open, click, type oder select. „Targets“ sind Identifikatoren, die angeben, auf welchem Element die „Commands“ ausgeführt werden sollen. „Values“ beschreiben den Text oder den Wert, der beispielsweise über eine Eingabeformular eingegeben werden soll.[56]
3. Über das soapUI können Testfälle für Webservices generiert werden. Über den Script Wizard können außerdem JUnits-Tests eingebunden werden.
4. In früheren Versionen des TestMakers gab es neben den bereits aufgeführten Möglichkeiten zur Erzeugung von Skripten einen http-Rekorder. Dieser wird jedoch in der aktuellen Version nicht mehr unterstützt.

TestMaker bietet die Möglichkeit einzelne Testfälle parallel auszuführen. Sie können dabei als funktionale Tests auf nur einem Testknoten ausgeführt werden, oder als Lasttests auf mehreren Knoten einer Anwendung. Auf welchen Teilen einer Anwendung ein Test ausgeführt wird, muss der Benutzer des TestMakers in einer XML-Datei, der sogenannten „TestScenario“- Datei, definieren. Die Instanzen einer Software / Anwendung, auf der die

[55] Vgl. Schimrazki, O. (2009), S. 16 f.
[56] Vgl. ebenda, S. 18

Tests ausgeführt werden, können dabei sowohl lokal liegen, als auch über mehrere Maschinen verteilt sein. Wo die einzelnen zu testenden Instanzen der Anwendung liegen wird ebenfalls in der „TestScenario“-Datei definiert.[57] Dadurch ermöglicht der TestMaker auch das Testen von verteilten Anwendungen.

Bei der Ausführung eines Tests werden zunächst die benötigten Daten auf die zu testenden Knoten der Anwendung übertragen. Mithilfe der sogenannten „Data Protection Library“ werden die Tests in der nativen Sprache bzw. dem nativen Protokoll des zu testenden Systems abgespielt und die Resultate in einer Log-Datei gespeichert. Während der Ausführung von Tests überwacht der Monitor des TestMakers die Ressourcenauslastung der Anwendung und erstellt entsprechende Statistiken. Durch die anschließende Korrelation der Beobachtungsdaten mit den Testfalloperationen unterstützt der TestMaker dabei, mögliche Flaschenhälse in der getesteten Anwendung zu identifizieren. Einen Überblick über den Datenaustausch während des Ablauf eines Tests liefert Abb. 6.[58]

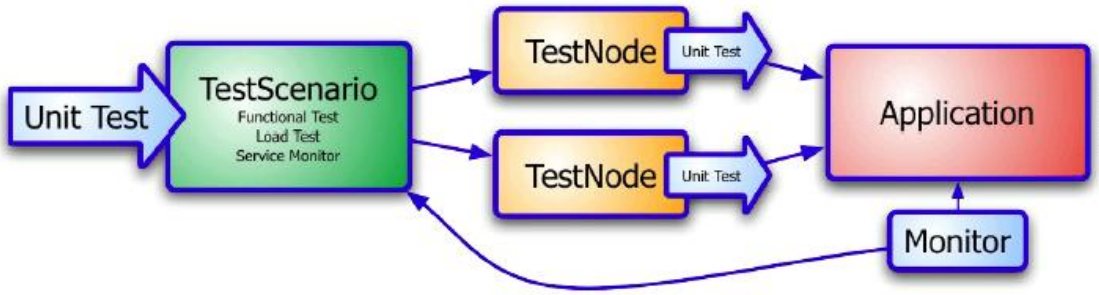


Abb. 6: Datenaustauschmodell des PushToTest TestMaker[59]

Während des Testverlaufs stehen dem Benutzer eine Vielzahl an Statistiken zur Verfügung. Die GUI des TestMakers liefert einen Überblick über die mittleren Transkationen pro Sekunde, die Auslastung der CPU, des Hauptspeichers sowie des Netzwerks für alle Teilsysteme, die in einen Test involviert sind. Dazu muss der Benutzer zuvor den Monitor des TestMaker auf allen relevanten Teilsystemen installiert haben. Die unterschiedlichen Werte zur Auslastung der Anwendung und der darunter liegenden Systeme können mit der Anzahl an parallelen, virtuellen Benutzern verglichen werden. In der GUI des TestMakers werden darüber hinaus Informationen, wie der Teststatus und Fehlermeldungen für die involvierten Teilsysteme ausgegeben.[60]

[57] Vgl. Schimrazki, O. (2009), S. 25
[58] Vgl. ebenda, S. 25 f.
[59] Enthalten in: ebenda, S. 26
[60] Vgl. ebenda, S. 26

4.3.5 Sonstiges

In Zusammenarbeit mit SQAForums.com bietet PushtoTest den Benutzern des TestMaker ein kostenloses Forum mit einer Online-Community an. Auf der Webseite steht außerdem ein recht ausführlicher User Guide für die Version 6 des Programms zur Verfügung, über den man außerdem Zugriff auf eine Vielzahl von Tutorials hat. Das Support Angebot wird durch einen Community-Blog abgerundet, über den die Benutzer über neuste Entwicklungen und Projekte informiert werden.[61]

Neben diesen kostenlosen Support-Möglichkeiten bietet PushToTest den kommerziellen Nutzern des TestMaker ein umfangreiches Service-Angebot gegen eine entsprechende Bezahlung.

[61] Vgl. PushtoTest (2012)

Vergleichskriterien	TestMaker
1. Bedienbarkeit	
Einfache Installation	✓
Intuitive Bedienbarkeit	!
Graphische Oberfläche (GUI)	✓
Schnelle Einarbeitung	✓
2. Technologie	
Zugrundeliegende Technologien	Java
Scriptingsprache	Java, Jython Python, Ruby, Groovy und JavaScript.
Plattformunabhängigkeit	✓
Betriebsysteme	
Windows	✓
Linux	✓
Unix	✓
Testbare Protokolle	
HTTP	✓
HTTPS	✓
POP3	✓
JDBC	✗
FTP	✗
Ajax	✓
SSL	✓
Parallele Nutzerverwaltung (Multithreading)	✓
Verteilte Anwendung	✓
3. Funktionalität	
Testbare Funktionalitäten	
Webservices(XML)	✓
Webanwendung	✓
Datenbanken	✗
Middleware	✗
Testfallgenerierung & Scenariounterstützung	
Manuelle Testfallgenerierung	✓
Recording von Testfällen	!
Integrierte Testfallgenerierung	✓
Client-seitiges Testing	✓
Server-seitiges Testing	✓
Parametrisierung	✓
Komplexität der Szenarien	✓
Auswertung&Genauigkeit der Testergebnisse	
Graphische Ausgabe der Testergebnisse	✓
Speicherung in Log Files	✓
Exportierbarkeit in Spreadsheettools	!
Format der Auswertung	xls
4. Sonstiges	
Support	✓
Dokumentation	✓
Weiterentwicklung	✓
Externer Support	✓
Aktuellste Version	Version 6.5 (Juni 2012)
Community	SQAForums.com

Tabelle 10: Bewertung des „TestMaker“ anhand der Vergleichskriterien

4.4 FunkLoad

FunkLoad ist ein auf PyUnit basierendes Performance Framework, welches sich vorrangig dazu eignet, funktionale-, regressions- und Lasttests von Web Projekten durchzuführen.

FunkLoad ermöglicht es Benutzern Performancetests von Web Anwendungen durchzuführen, um detaillierte Lastauswertungen durchzuführen und Kapazitätsengpässe zu identifizieren. Darüberhinaus können Anwender Testabläufe erstellen, um sich wiederholende Anwendungsschritte permanent durchführen zu lassen.

Die Implementierung in Python Skript basiert auf dem pyUnit Framework. Sie ermöglicht es, komplexe Szenarien in bestehenden Anwendungen zu simulieren. Dies ist vergleichbar zu herkömmlichen Test-Umgebungen.[62] FunkLoad ist eine freie Software, welche unter der GNU GPL (General Public License) erhältlich ist. Die Inhalte des Programms können, gemäß der Bedingungen der GPL, frei weiterverteilt und abgeändert werden.[63] Dieses Tool steht grundsätzlich für sämtliche gängigen Betriebssysteme zur Verfügung (MS Windows, MacOS, Linux). Einige Funktionalitäten, wie beispielsweise der Monitoring Server oder der Credential Server, können lediglich unter Linux in vollem Umfang genutzt werden.

Der Aufbau von FunkLoad spezifischen Tests setzt sich aus einem herkömmlichen Unittest Skript und einer Konfigurationsdatei zusammen.

FunkLoad Test Unit emuliert einen voll funktionsfähigen Webbrowser, der sämtliche Leistungsaspekte testen und simulieren kann. Dies sind *get, post, put und delete* Befehle. Außerdem wird die Implementierung von Java Anwendungen und XML Inhalten unterstützt. Authentifizierungs-gesicherte Inhalte können über definierte Authentifizierungsregeln ebenfalls in die Tests mit eingebunden werden.

Aktuelle Versionen des FunkLoads unterstützen des Weiteren die Verwendung von Cookies und die verbreiteten Web Security Standards des HTTPS sowie SSL und Transport Layer Security (TLS) Verschlüsselungen.

Die genannten Protokolle können sowohl in funktionalen Tests, als auch in Lasttests verwendet werden. Funktionale Testabläufe können durch den so genannten „bench runner“ in Lasttests umgewandelt werden, um Performance- und Skalierungstests vorzunehmen. Die Simulation des Lasttests kann hierbei auf mehrere Maschinen verteilt werden.

[62] Vgl. Benoit Delbosc (2011)
[63] Vgl. ebenda

Die Bedienbarkeit der FunkLoad Testing Unit beschränkt sich auf Eingaben in die Konsole. Dies setzt erweiterte Kenntnisse des Benutzers in der Bedienung einer Konsole voraus. Die Erstellung von Testfällen als Python Skript erfordern an dieser Stelle ebenfalls weitere Kenntnisse, um Tests mit FunkLoad durchzuführen. Diese Aspekte führen dazu, dass FunkLoad sich lediglich für Nutzergruppen mit fortgeschrittenen Linux und Python Kenntnissen eignet.

Von den Entwicklern des FunkLoad wird kein Support angeboten. Benutzer finden lediglich Unterstützung in Foren und von der Community erstellten Anleitungen. Ein Angebot von Supportleistungen durch Drittanbieter konnte trotz intensiver Recherche ebenfalls nicht am Markt gefunden werden.

Die umfangreiche Berichterstellung des FunkLoads ermöglicht es, zu sämtlichen simulierten Szenarien Reports und Auswertungen vorzunehmen. Diese können je nach Art im HTML-, PDF- und XLS-Format ausgegeben werden.[64]

Folgendes Beispiel beschreibt die Erstellung eines in Python geschriebenen Test Skripts. Ausgangssituation ist die Erstellung eines einfachen Test Packets – mytest. Allerdings handelt es sich lediglich um exemplarische Fragmente eines üblichen Testscripts. Die Darstellung eines ganzen Tests wurde an dieser Stelle als nicht sinnvoll erachtet.

```
$ ls ./mytest
boostrap.py buildout.cfg setup.py src

$ ls ./mytest/src/mytest
__init__.py tests

$ ls ./mytest/src/mytest/tests
__init__.py test_Simple.py Simple.conf
```

Wie beschrieben bestehen FunkLoad spezifische Tests aus einem Testskript und einer Konfigurationsdatei (.cfg). Diese wird wie folgt erstellt.

```
[buildout]
develop = .
parts = FunkLoad

[FunkLoad]
recipe = collective.recipe.FunkLoad
url = http://www.testFunkLoad.de
eggs = mytest
```

[64] Vgl. Benoit Delbosc (2011)

Das Skript zeigt die generellen Parameter, welche im Test beinhaltet sind. So wird in diesem Beispiel die fiktive URL www.testFunkLoad.de getestet. Der Host spielt hierbei keine Rolle. In diesem Beispiel wird der Einfachheit halber auf den localhost referenziert.

```
[main]
title=Simple
label=Simple test for dhbw.OpenSourcePaper
description=Simple test scenario

[test_Simple]
description=Simple test scenario

[ftest]
log_to = console file
log_path = Simple-test.log
result_path = Simple-test.xml

[bench]
cycles =  5:15:30
duration = 100
startup_delay = 2
sleep_time = 2
cycle_time = 2
log_to = file
log_path = Simple-bench.log
result_path = Simple-bench.xml
sleep_time_min = 2
sleep_time_max = 2
```

Die relevanten Kriterien für dieses einfache Testszenario sind:

- **cycles = 5:15:30** - es werden drei (default) aufeinanderfolgenden Testzyklen für 5, 15 und 30 nebenläufige Benutzer durchgeführt.
- **duration = 100** – gibt die Dauer eines Testzykluses in Sekunden an.
- **startup_delay = 2** – Wartezeit zwischen aufeinanderfolgenden Threads für die Testzyklen
- **sleep_time = 2** – Wartezeit bis der eigentliche Tests startet (in Sekunden)
- **cycle_time = 2** – Wartezeit der Anwendungen zwischen den eigentlichen (hier 3) Testzyklen.

Im Testszenario werden anschließend diese Schritte ausgeführt (hier nicht als Skript aufgezeigt):

- Öffnen der http://www.testFunkLoad.de Seite (get /).
- Das entsprechende Login Formular wird geöffnet (get / login_form).
- Login Daten werden als (post /login_form) in das Login Formular eingetragen.
- Zugriff auf ein Dokument (document1) welches nur für eingeloggte Nutzer sichtbar ist (get/folder/document1).
- Durchführung eines Logouts.

Das gesamte Testkonstrukt wird über den „bootsrap“- Befehl zu einem startfähigen Paket erstellt und im Folgenden dann wie das eigentliche Testfile gestartet.

```
$ cd mytest
$ python bootstrap.py

$ ./bin/buildout

$ ./bin/FunkLoad bench -t mytest.tests.test_Simple
```

Im Anschluss wird ein Report erstellt und (in Form einer .png Grafik) ausgegeben.[65]

```
$ ls ./var/FunkLoad/reports/2009-11-11-19-40-20
test_Simple-20091111T194020

$ ls ./var/FunkLoad/reports/2009-11-11-19-40-20/test_Simple-20091111T194020
FunkLoad.css  index.html  index.rst  tests.data  tests.gplot  tests.png
```

[65] Vgl. Red Turtle (2012)

Vergleichskriterien	FunkLoad
1. Bedienbarkeit	
Einfache Installation	!
Intuitive Bedienbarkeit	✗
Graphische Oberfläche (GUI)	✗
Schnelle Einarbeitung	!
2. Technologie	
Zugrundeliegende Technologien	Python, pyUnit
Scriptingsprache	Python
Plattformunabhängigkeit	✗
Betriebsysteme	
Windows	!
Linux	✓
Unix	!
Testbare Protokolle	
HTTP	✓
HTTPS	✓
POP3	✗
JDBC	✗
FTP	✗
Ajax	✗
SSL	✓
Parallele Nutzerverwaltung (Multithreading)	✓
Verteilte Anwendung	✓
3. Funktionalität	
Testbare Funktionalitäten	
Webservices(XML)	!
Webanwendung	✓
Datenbanken	✗
Middleware	✗
Testfallgenerierung & Scenariounterstützung	
Manuelle Testfallgenerierung	✓
Recording von Testfällen	✓
Integrierte Testfallgenerierung	✗
Client-seitiges Testing	✓
Server-seitiges Testing	✓
Parametrisierung	✓
Komplexität der Szenarien	✓
Auswertung&Genauigkeit der Testergebnisse	
Graphische Ausgabe der Testergebnisse	✗
Speicherung in Log Files	!
Exportierbarkeit in Spreadsheettools	✓
Format der Auswertung	pdf, xls, html
4. Sonstiges	
Support	✓
Dokumentation	!
Weiterentwicklung	✓
Externer Support	✗
Aktuellste Version	16.1 (Juli 2011)
Community	Nuxeo

Tabelle 11: Bewertung des Tools „FunkLoad“ anhand der Vergleichskriterien

5 Methodik zur Bewertung der Tools

Um die unterschiedlichen Tools miteinander zu vergleichen, wurde ein Kriterienkatalog erstellt, anhand dessen die ausgewählten Tools bewertet werden. Der Katalog besteht aus den vier bereits evaluierten Oberkategorien:

- Bedienbarkeit
- Technologie
- Funktionalität
- Sonstiges

Der Katalog gibt einen Überblick über alle wichtigen Funktionen der unterschiedlichen Anbieter und die unterschiedlichen Möglichkeiten einen Last- bzw. Performancetest durchzuführen. Die Bewertung entlang dieses Kriterienkatalogs soll dem Leser dabei helfen, die richtige Wahl für die jeweiligen Anforderungen an ein Tool zu treffen. Bewertet wurden die einzelnen Produktmerkmale der Tools mit unterschiedlichen Leistungsmerkmalen:

- Ja / Sehr gut
- Eingeschränkt / mittelmäßig
- Nein

Innerhalb der vier genanten Oberkategorien erfolgte die Bewertung für jedes einzelne Kriterium durch die Vergabe von Leistungspunkten. Dabei ergibt ein Ja / Sehr gut 2 Leistungspunkte, eingeschränkt / mittelmäßig 1 Leistungspunkte und Nein 0 Leistungspunkte. Innerhalb der Oberkategorien sammlen die vier Anbieter / Lasttesttools Leistungspunkte, die jeweils zu einer Zwischensumme aufsummiert werden. Diese Zwischensumme wird anschließend mit dem festgelegten Gewichtungsfaktor für die jeweilige Oberkategorie multipliziert. Ein Beispiel: Das Produkt JMeter erhält in der Oberkategorie Bedienbarkeit bei allen Kriterien ein „sehr gut“ und damit je 2 Leistungspunkte. Da es innerhalb der Oberkategorie vier Bewertungskriterien gibt, kommt JMeter insgesamt auf 8 Leistungspunkte. Diese Anzahl an Leistungspunkten wird mit dem Faktor 15 multipliziert, da in der Vergleichsstudie davon ausgegangen wird, dass der Aspekt der Bedienbarkeit mit 15 % anteilig in die Gesamtbewertung einfließen soll. Dadurch ergeben sich für den JMeter in dieser Kategorie 120 Leistungspunkte, die in die Gesamtbewertung einfließen. Dasselbe Verfahren wird in den anderen drei Kategorien ebenso angewendet. Die Gewichtung wird dabei in den drei Kategorien Bedienbarkeit, Technologie und Sonstiges nur auf Ebene der Oberkategorie vorgenommen und innerhalb dieser nicht weiter aufgeteilt. Lediglich in der Kategorie Funktionalität wurde ein abweichendes Verfahren angewandt. In dieser Studie wird davon

ausgegangen, dass die Oberkategorie Funktionalität zu insgesamt 50 % in die Gesamtbewertung einfließen soll. Der Leser dieser Vergleichsstudie kann jedoch individuell festlegen, wie sich diese 50 % bzw. 50 Leistungspunkte auf die Unterkategorien „testbare Funktionalitäten“, „Testfallgenerierung & Szenariounterstützung“ und „Auswertung & Genauigkeit der Testergebnisse“ verteilen und auswirken. Je nach Anwendungsfall können dadurch die genannten Unterkategorien unterschiedlich gewichtet werden. Dies gibt dem Leser die Möglichkeit individuelle Schwerpunkte zu setzen, wenn es zum Beispiel wichtig ist, dass das gewünschte Lastttesttool besonders viele unterschiedliche Anwendungen (Webservices, Webanwendungen, Datenbanken und Middleware) testen kann, oder der Leser einen besonderen Schwerpunkt auf die Testfallgenerierung legt und dabei bestmöglich durch das Testtool unterstützt werden will.

Zur Bildung der Gesamtsumme für die vier unterschiedlichen Tools werden die gewichteten Zwischensummen aus den vier Oberkategorien für das jeweilige Testtingtool aufsummiert. Das Gesamtergebnis wird durch Teilen dieser Gesamtsumme durch 100 berechnet, da insgesamt 100 Leistungspunkte für die vier Oberkategorien vergeben werden.

Wichtig ist, dass die in der Vergleichsstudie dargestellte Bewertung und Gewichtung auf einer subjektiven Einschätzung basiert. Wenn andere Schwerpunkte gesetzt werden und einzelne Kriterien mit einer höheren bzw. geringeren Gewichtung angesetzt werden, ergibt sich unter Umständen ein anderes Endergebnis. Es ist explizit gewünscht, dass der Leser dieser Vergleichsstudie die Gewichtung der Kriterien an die individuellen Bedürfnisse anpassen kann und so das individuell am besten geeignete Tool auszuwählen.

6 Auswertung

Bei der Betrachtung der Tabelle lassen sich zwei klare Testsieger aus der Bewertung identifizieren: der JMeter als „Alleskönner“ und der TestMaker als der „Exot“ unter den ausgewählten Tools. In diesem Kapitel werden die einzelnen Produkte mit ihren Vor- und Nachteilen gegenübergestellt.

Alle vorgestellten Tools sind mit marginalen Einschränkungen einfach zu installieren, ermöglichen eine parallele Nutzerverwaltung und können verteilte Anwendungen testen.
FunkLoad das „Expertentool“ schneidet in der Gesamtbewertung am schlechtesten ab. Begründet wird dies durch die fehlende Unterstützung bei der Testfallgenerierung und die nicht vorhandene graphische Benutzeroberfläche. Nichtsdestotrotz weist dieses Tool auch einige Vorteile auf. Er eignet sich vor allem für Experten, die den Anspruch haben, hoch

komplexe Testfälle mit großem Ressourceneinsatz zu generieren, ohne dabei von Interfaces und Frameworks beschränkt zu werden. Aus diesem Grund hat der FunklLoad jedoch auch in der Hauptkategorie Bedienbarkeit (bewertet mit 30 Punkten) im Gegensatz zum JMeter (mit 120 Punkten), schlecht abgeschnitten. Anwender, die bereits Erfahrungen und gute Kenntnisse im Bereich Phytonprogrammierung besitzen und sich mit dem zu testenden System sehr genau auseinandergesetzt haben, finden mit dem FunkLoad ein sehr flexibles, performancestarkes und ressourcenschonendes Lasttesttool. Für Anwender ohne Erfahrungen mit der Konsole ist das Tool jedoch nur schwer nutzbar.

Der PushToTest TestMaker kann durch seinen Funktionsumfang sowie unterschiedliche Skriptingsprachen und die unterstützten Protokollen überzeugen. Allgemein gefasst schneidet er gleich gut wie der Grinder ab, bietet jedoch umfassende Supportmöglichkeiten, da er von dem Unternehmen PushToTest weiterentwickelt und unterstützt wird.
Diese Serviceleistungen lassen sich extern einkaufen und garantieren somit die reibungslose Implementierung und den Betrieb der Software. Jedoch ist festzuhalten, dass im Falle einer eigenständigen Kompilierung der Software ein erhöhter Aufwand und Expertenkenntnisse erforderlich sind. Eigens kompilierte Versionen werden nicht durch PushToTest unterstützt. Für den Support stehen lediglich die Open Source Community und eine umfangreiche Dokumentation zur Verfügung.

The Grinder landete bei unserem Test auf dem dritten Platz. Seine durchweg positiven Ergebnisse in allen Kategorien führten zu diesem Ergebnis. Genauso wie der JMeter lassen sich mit ihm viele Funktionen eines Systems testen. Er ist einfach zu installieren, unterstützt jedoch den Benutzer bei der Testfallgenerierung nicht durch eine benutzerfreundliche Oberfläche. Daher sind eine gewisse Einarbeitungszeit und Kenntnisse im Bereich Python- und Java-Programmierung zur Testfallgenerierung Voraussetzung für die Arbeit mit dem Grinder. Aus diesem Grund hat The Grinder im Bereich Benutzerfreundlichkeit mittelmäßig abgeschnitten. Seit der dritten Version gibt es die Möglichkeit Testfälle aufzuzeichnen, jedoch ist diese Funktion noch nicht so weiterentwickelt wie vergleichsweise beim JMeter. Des Weiteren bietet der Grinder nur clientseitiges Testing an. Da alle anderen Tools auch serverseitiges Testing anbieten, kann er in diesem Vergleich nicht mithalten. Trotzdem bietet dieses Tool für Testsystemkenner alle nötigen Funktionen und kann sehr hohe Lasten erzeugen, da er sowohl parallele Nutzerverteilung als auch Tests von verteilten Anwendungen unterstützt.

Testsieger wurde in dieser Vergleichsarbeit der JMeter als voll funktionsumfassendes und intuitiv bedienbares Lasttesttool. Er konnte in allen vier Hauptkategorien überzeugen. Vor

allem bei der Bedienbarkeit schlägt er alle anderen Tools. Auch bei der Testfallgenerierung bietet der JMeter umfangreiche und komfortable Möglichkeiten Testfälle zu erstellen. Jedoch lassen sich mit diesem Tool nur bedingt flexible Testfälle erstellen, da er ein reines Interface gesteuertes Testtool ist. Einige Quellen sprechen davon, dass die visuelle Auswertungen des JMeters zum Teil unklar und verwirrend sein können.[66] Es ist anzuraten den JMeter unabhängig vom Testsystem laufen zu lassen, da er durch die komplexen Interfaces einen hohen Ressourcenbedarf hat und somit die eigentlich zu testende Performance negativ beeinflussen kann. Der JMeter ist beliebig durch Zusatzfunktionen erweiterbar und ist somit ein gutes Beispiel für Modularität und Kundenfreundlichkeit. Die identifizierten Vorzüge dieses Tools wurden auch in vergleichbaren Arbeiten aufgeführt.

Im Allgemeinen lässt sich festhalten, dass die Open Source Lizenz den Firmen ermöglicht, jegliche Tools zum Testen von Lasten zu nutzen, unabhängig davon, ob das Tool intern oder für einen Kunden genutzt wird.
Im Bezug auf die Weiterentwicklung schneiden JMeter, The Grinder und der TestMaker alle sehr gut ab, da alle bereits in diesem Jahr aktualisiert wurden und somit weiterhin vollständig von der Open Source Community unterstützt werden. Das letzte Update für den FunkLoad wurde im letzten Jahr veröffentlicht, die Community Nuxeo unterstützt weiterhin dieses Tool, jedoch gibt es keine dedizierten Mailinglisten. Entwicklern stehen jedoch Mailinglisten, die Nuxeo direkt adressieren, zur Verfügung.

Der Grinder ist dem TestMaker sehr ähnlich. Beide sind sehr mächtige Tools, da Testszenarien mit in Jython geschriebenen Skripten generiert werden und somit alle Java Features genutzt werden können.

Abschließend lässt sich festhalten, dass alle Tools die Anforderungen der Anwendern auf unterschiedlichste Weise erfüllen können. Jedoch bieten die integrierten Möglichkeiten Testergebnisse auszuwerten und grafisch darzustellen bei allen Tools in unterschiedlicher Ausprägung nur bedingt die gewünschten Resultate bei der Auswertung der Testergebnisse. Dieser Mangel lässt sich jedoch bequem durch Plug-ins und Erweiterungen reduzieren, wie zum Beispiel mit „Composite Graph“ für den JMeter oder mit dem „Grinder Analyzer“ für den Grinder kompensieren.

Tabelle 12 zeigt einen Kurzüberblick über die Leistungen der Tools in den einzelnen Hauptkategorien und stellt deren erreichte Punktzahlen anhand der vorgenommen Einschätzung dar.

[66] Stoll, C. / Pommering, T. (2004), S. 9

	JMeter	The Grinder	TestMaker	Funk Load
1. Bedienbarkeit				
Bedienbarkeit Summe gewichtet	120	90	105	30
2. Technologie				
Technologie Summe gewichtet	575	550	550	350
3. Funktionalität				
Funktionalität Summe gewichtet	458	404	392	330
4. Sonstiges				
Sonstiges Summe gewichtet	70	70	80	50
Ergebnis (∑/100)	**12,23**	**11,14**	**11,27**	**7,6**

Tabelle 12: Überblick über die Gesamtergebnisse der evaluierten Tools

7 Fazit

Ähnlich wie andere Bewertungs- bzw. Vergleichsstudien basiert auch diese Studie auf subjektiven Einschätzungen und Entscheidungen. Sowohl die Auswahl der hier verglichenen Tools, als auch die der Bewertungskriterien und deren Gewichtung wurden aus der Perspektive der Verfasser vorgenommen. Für die Verwendung der Ergebnisse unserer Arbeit sollten die Anforderungen an das spätere Last- und Performancetesttool genau analysiert werden und auf Basis dieser Angaben eine Entscheidung getroffen werden, welches Performance-Testtool für den speziellen Anwendungsfall am besten geeignet ist. Zwar wurde der Apache JMeter in dieser Vergleichsstudie als Testsieger eingestuft und wird somit auch explizit empfohlen, jedoch könnte aufgrund individueller Anforderungen und unterschiedlicher Gewichtungen des Lesers ein anderes Tool für Performance- bzw. Lasttests besser geeignet sein. Alle vier vorgestellten Tools sind für diese Art von Tests gut geeignet und zu empfehlen. Sie entsprechen den im Vorfeld definierten Ansprüchen an Aktualität und erfüllen alle die technischen Voraussetzungen der parallelen Nutzerverwaltung und der verteilten Testumgebung.

Diese Arbeit soll ein möglichst genaues Bild der ausgewählten Tools, deren Eigenschaften, sowie Vor-und Nachteile geben und eine Entscheidungsbasis für den Leser darstellen. Es wurde zudem als wichtig erachtet, die Vorteile eines Open Source Produktes gegenüber kommerziellen Produkten darzustellen und zu verdeutlichen, dass für ein gutes und geeignetes Produkt nicht zwangsläufig Geld ausgegeben werden muss. Für einen späteren möglichen Support können zum einen die kostenlosen Supportmöglichkeiten in Form von Mailinglisten und Foren, oder die zum Teil angebotenen Dienste externer Unternehmen zur individuellen Anpassung der Systeme genutzt werden.

Anhang

Anhangverzeichnis

Anhang 1: Übersicht von Open Source Performace- und Lasttesttools

Beispielhafte Auswahl an Open Source Performace- und Lasttesttools

Allmon
Apache JMeter
benerator
CLIF is a Load Injection Framework
curl-loader
Database Opensource Test Suite (DOTS)
DBMonster
Deluge
Dieseltest:
Faban
FunkLoad
The Grinder
Hammerhead 2
Hammerora
httperf
JCrawler
Load Impact
OpenSTA
OpenWebLoad
Parallel-junit
Pylot
Seagull
PushToTest TestMaker
VisualVM
Web Application Load Simulator
WebLOAD
zapwireless

Tabelle 13: Übersicht von Open Source Lasttesttools[67]

[67] Mit Änderungen übernommen aus: Philips, J. (2010)

Anhang 2: Bewertungskriterien für Lasttesttools aus unterschiedlichen Quellen

Vergleich der Bewertungskriterien aus unterschiedlichen Quellen

Performancetest- Stolze, J.	Evaluation of Load/Stress tools for Web Applications testing - HSC	Open Source and Commercial Performance Testing Tools- Accenture	Leistungstests auf Basis von OpenSource Werkzeugen für das Content Repository Framework- Schimratzki, O.
Bedienbarkeit			
Technologie			
	Skript Sprache	Skript Sprache	Sprachen
unterstützte System und Protokolle	unterstützende Protokolle		java basierend
			Protokolle
	Systemvoraussetzungen		plattformaunabhängig
paralelle Nutzerzahl	Skalierbarkeit Nutzerzahlen		Multithreading/parallel Nutzerzahlen
verteilte Anwendung	verteilte Testfall unterstützung		verteilte Ausführung
Funktionalität			
Szenario Unterstützung		Komplexität der Testszenarien	
Testskript Erzeugung	Skript Erstellung	Rekorden von Testszenarien	
		manuelles skripten	
Monitoring der Clinte und server		Monitorng von Client und Server	
		Echtzeit Monitoring	
	Parametriesieung		
Analyse Möglichkeiten	Reports und Analyse	Analyse Möglichkeiten	auswertungsarten
		Visualiesierung von Testergebnissen	graphische Auswertung
Sonstiges			
	Support und Consuting		aktive Weiterentwicklung

Tabelle 14: Bewertungskriterien für Lasttesttools aus unterschiedlichen Quellen[68]

[68] Mit Änderungen übernommen aus: Stolze, J. (2008); Hughes Systique (2008), S. 5 f.; Accenture (2008), S. 10 f.; Schimrazki, O. (2009), S. 27

Quellenverzeichnisse

Literaturverzeichnis:

Bradtke, R (2008): ISTQB 100 Success Secrets, Emereo Publishing

Dempfle, M. (2007): Entwurf und Implementierung eines Frameworks für Profiling- und Performancemessungen mit Hilfe von aspektorientierter Programmierung (AOP), Nürnberg: Grin-Verlag

Hughes Systique (2008): Evaluation of Load / Stress tools for Web Applications testing, Whitepaper, May, Rockville: Hughes Systique Corporation

Klinger,J. /Schipfer, G. (2004), Open Source Webshops, IT Seminararbeit, Graz: Technische Universität

Lucca, G./ Fasolino, A. (2006), Testing Web-based applications: The state of the art and future trends, in: Computer Software and Applications Conference, 29 Jg.,Nr.2, S. 65-69

Schimratzki, O. (2009): Leistungstests auf Basis von Open Source Werkzeugen für das Content Repository Framework MyCore, Jena: Friedrich-Schiller-Universität

Stoll, C. / Pommering, T. (2004): Evaluation von Lasttesttools und Performance Studie, Seminararbeit

Weber, S. (2004): The Success of Open Source, USA: Harvard University Press

Verzeichnis der Internet- und Intranetquellen:

Accenture (2008):	Open Source and Commercial Performance Testing Tools, http://isqtinternational.com/Routhu%20Leelaram%20&%20Vinod%20Kumar%20Palla%20-%20Accenture.pdf, Abruf: 14.06.2012
Apache Software Foundation (2012):	Apache JMeter, http://JMeter.apache.org/, Abruf: 16.06.2012
Aston, P./ Fitzgerald, C. (2012):	The Grinder 3, http://grinder.sourceforge.net, Abruf: 29.06.2012
Benoit Delbosc (2011)	FunkLoad Documentation http://FunkLoad.nuxeo.org/intro.html, Abruf: 20.06.2012
Billens, A. (2007)	Ratgeber: Lasttests mit Open Source, http://www.computerwoche.de/heftarchiv/2007/03/1217471/, Abruf: 20.06.2012
Falk, R. u. a. (2009):	Lasttests von Webanwendungen mit The Grinder http://wiki.fhstralsund.de/index.php/Kategorie:Lasttests_von_Webanwendungen_mit_The_Grinder, Abruf: 20.06.2012
Inventage AG (2012):	Performance- und Lasttests, http://inventage.com/performance--und-lasttests.html, Abruf: 16.06.2012

Pansch, C. (2010) Open Source und kommerzielle Lösungen gegenübergestellt, http://www.christian-pansch.de/mein-wissen/rund-um-typo3-und-content-management-systeme/open-source-und-kommerzielle-loesungen-gegenuebergestellt, Abruf: 27.06.2012

Philips, J. (2010): 50 + Open Source Performance Testing Tools, http://www.jayphilips.com/2010/01/07/50-open-source-performance-testing-tools/, Abruf: 28.06.2012

PushToTest (2012): PushToTest TestMaker, http://www.pushtotest.com/ Abruf: 30.06.2012

Red Turtle (2012): Write FunkLoad Tests in a few minutes, http://blog.redturtle.it/redturtle/how-to-write-FunkLoad-test-in-few-minutes, Abruf: 28.06.2012

Stolze, J (2008): Performancetests und Bottleneck-Analyse in Multischichtarchitekturen, http://www.performance-test.de/, Abruf: 20.06.2012

The Apache Software Foundation (2012): User Manual, http://jakarta.apache.org/JMeter/usermanual/index.html, Abruf: 15.06.2012

Wikipedia (2012a): Lasttest (Computer), http://de.wikipedia.org/wiki/Lasttest_(Computer), Abruf: 16.06.2012

Wikipedia (2012b): Stress testing, http://en.wikipedia.org/wiki/Stress_testing, Abruf: 16.06.2012