DHBW
Duale Hochschule
Baden-Württemberg
Stuttgart

KOS.content

01 | 2013

Ergebnisse der Untersuchungen des
Kompetenzzentrum Open Source der DHBW-Stuttgart

Frühjahr 2013
band.1

INHALT BAND.1

Inhalt __

Das Kompetenzzentrum Open Source (KOS)

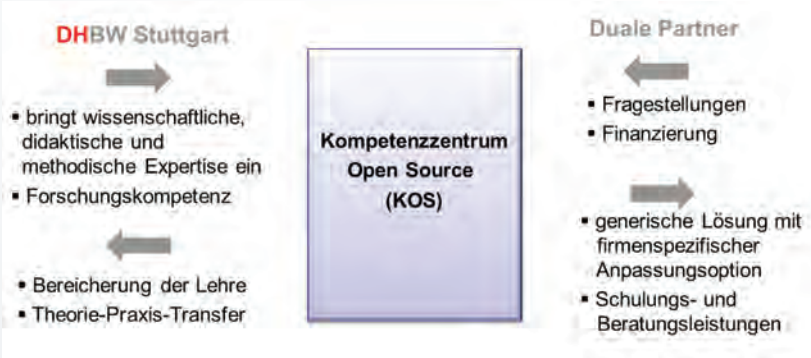
Ziel des Projektes

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.

Open Source Continuous Integration und Testing

Schriftliche Ausarbeitung
im Rahmen der Lehrveranstaltung „Projekt des 5. Semesters“
Kompetenzzentrum Open Source (KOS)

Vorgelegt von

Marco Berger, Janina Höchner,
Sarah Kieninger, Robert Krombholz

am 23.01.2012

Fakultät Wirtschaft
Studiengang Wirtschaftsinformatik
WWI2010I

Inhaltsverzeichnis

Abkürzungsverzeichnis

API	**A**pplication **P**rogramming **I**nterface
CIS	**C**ontinuous **I**ntegration **S**erver
CI	**C**ontinuous **I**ntegration
CVS	**C**oncurrent **V**ersions **S**ystem
IDE	**I**ntegrated **D**evelopment **E**nvironment
JDK	**J**ava **D**evelopment **K**it
LDAP	**L**ightweight **D**irectory **A**ccess **P**rotocol
POM	**P**roject **O**bject **M**odel
SVN	**S**ub**v**ersion
VCS	**V**ersion **C**ontrol **S**ystem
WAR	**W**eb **A**pplication A**r**chive

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einführung

1.1. Problemstellung

Heutzutage wird Software zunehmende von global verteilten Teams entwickelt, die aus mehreren Entwicklern bestehen. Obwohl bereits von vielen Entwicklerteams die Methodik der agilen Softwareentwicklung erfolgreich angewandt wird, gibt es bei der Integration der einzelnen Prozessschritte immer noch Lücken und Verbesserungsbedarf. Die einzelnen Werkzeuge zur Unterstützung der Softwareentwicklung und insbesondere zum Testen werden in vielen Projekten als Einzelsysteme verwendet, wie beispielsweise ein isoliertes Versionsverwaltungssystem zum Einfügen von Codeteilen in das Gesamtprojekt. Dies führt dazu, dass fehlerhafter Code leicht in den primären Entwicklungszweig integriert werden kann. Da die Integration der einzelnen Softwarebausteine häufig zu Fehlern und Problem führt, wird diese von den Entwicklern gerne möglichst weit hinausgezögert. Kurz vor einem Release kommt es häufig zu einem sogenannten „Big Bang“ also einem Fehler-Urknall, welches häufig zu kostenintensiven Nacharbeiten und Zeitverzögerungen führt. Dabei ist es offensichtlich, dass die relativen Kosten zur Fehlerbeseitigung unverhältnismäßig steigen, je später ein Fehler gefunden wird.

Um genau diesen „Big Bang“ zu vermeiden wird zunehmend das Konzept von Continuous Integration in der Softwareentwicklung angewandt. Eines der Prinzipien der agilen Softwareentwicklung ist es stets ein funktionales Produkt bereitzustellen. An genau dieses Prinzip soll in dieser Arbeit angeknüpft werden. Durch Automatisierung und vollständige Integration von Versionsverwaltung, Build-Prozessen, Unit-Testen, Quellcodeanalyse und ständige Auswertungen wird dieses Prinzip umgesetzt und gefördert. So kann bereits durch das Einchecken von Code eines Entwicklers ein hoch automatisierter Prozess angestoßen werden. Das Resultat eines solchen Prozess ist eine komplett gebaute Software sowie ausführliche Ergebnisse aus Tests und Quellcodeanalysen. Schon dieser kleine Ausschnitt zeigt, dass es sich um ein äußerst komplexes, aber auch gewinnbringendes und interessantes Konzept handelt.

1.2. Zielformulierung

Das Projekt wurde in Auftrag gegeben, um für ein Unternehmen das Konzept und die Hintergründe einer Continuous Integration Lösung darzustellen. Dies soll zum Einen anhand einer schriftlichen Ausarbeitung geschehen, zum anderen anhand eines Prototypens in die Praxis umgesetzt und erprobt werden. Der Hauptaspekt liegt dabei auf der praktischen Umsetzung.

Ziel der schriftlichen Ausarbeitung ist, das Konzept einer Continuous Integration (CI) Lösung, sowie die einzelnen Komponenten im Open Source Bereich vorzustellen und kritisch zu hinterfragen. Zur Übersicht der verfügbaren Open Source CI- Server und Test Werkzeuge soll eine

kurze Marktanalyse erstellt werden. Hierbei ist die Anforderungen des Unternehmens einen groben Überblick über die aktuellen Vertreter auf dem Open-Source Markt zu geben.

Bei der Erstellung eines Konzepts zum Aufbau einer Toollandschaft, soll neben der technischen Umsetzung auch das phasenweise Vorgehen zur Implementierung eines CI-Prozesses im Unternehmen eingegangen werden, insbesondere welche Veränderungen für die Entwickler entstehen.

Ein weiteres Ziel ist es den Mehrwert einer CI-Lösung, für die einzelnen Entwickler und das Unternehmen darzustellen. Es soll gezeigt werden, wie die Softwarequalität mit einer CI-Lösung sichergestellt und optimiert werden kann. Hierbei wird insbesondere der Aspekt des Softwaretestens in einer CI-Lösung beleuchtet. Es soll gezeigt werden, inwiefern sich die Testprozesse innerhalb einer CI-Lösung für den Entwickler verändern.

1.3. Anforderungen an das Projekt

Um das theoretische Konzept hinsichtlich der Umsetzbarkeit zu überprüfen, ist das hauptsächliche Ziel des Projekts, für das Auftrag gebende Unternehmen einen Prototyp zu erstellen. Hierbei ist bei der Auswahl der verschiedenen Komponenten auf die Anforderungen des Auftraggebers einzugehen.

Das Unternehmen zieht in Betracht seine Java Projekte zukünftig in einer Continuous Integration Lösung zu entwickeln. Hierbei möchte es sich nur auf Open Source Produkte stützten. Im Moment verwendet das Unternehmen das Versionskontrollsystem Subversion, das Build-Werkzeug Ant und das Testwerkzeug JUnit. Daraus ergibt sich die Anforderung eines CI-Systems das möglichst auf den bereits genutzten Werkzeugen aufbaut.

Die Anforderungen an die schriftliche Ausarbeitung bestehen darin dem Unternehmen eine Einführung in das Thema Continuous Integration zu geben und besonders das Vorgehen der Implementierung eines CI-Systems darzustellen - am besten anhand der praktischen Erfahrungen die während der Entwicklung des Prototypen gemacht wurden. Das Unternehmen ist daran interessiert zu erfahren, wie hoch der Aufwand einer Implementierung eines CI-Systems ist und welchen genauen Nutzen es dem Unternehmen stiftet.

1.4. Struktur der Arbeit

Im theoretischen Teil dieser Arbeit werden zunächst die Grundlagen von Continuous Integration Systemen eingeführt. Darunter finden sich auch die verschiedenen Komponenten, aus welchen eine CI Lösung aufgebaut ist. Die verschiedenen Anbieter von Continuous Integration Servern sowie von whitebox-Testframeworks und UI-Testframeworks im Open Source Bereich werden anhand ihrer Funktionalitäten miteinander verglichen und die vorteilhaftesten Tools für das Konzept ausgewählt.

Da das Testen von Software in CI-Lösungen eine besonders wichtige Rolle spielt, sind die Grundlagen des Testens ebenfalls kurz dargestellt. Die Vorteile der Testautomatisierung im Allgemeinen und insbesondere in einer CI-Lösung werden anhand von Prozessabläufen dargestellt. Es wird gezeigt wie alle Teststufen mithilfe eines CI-Systems automatisiert getestet werden können.

Im Anschluss wird das Vorgehen der Einführung einer CI-Lösung in einem Unternehmen zunächst theoretisch dargestellt und daraufhin anhand eines technischen Konzepts auch in der Praxis erläutert. Zuletzt wird der Einsatz eines CI-Systems hinsichtlich der Anwendung im Unternehmenskontext bewertet und ein Fazit für den Auftraggeber gezogen.

1.5. Methodik

Die Vorgehensweise zur Erstellung dieser Seminararbeit ergibt sich aus der Anforderungen an das Projekt, welche vor allem eine praktische Ausarbeitung vorsehen. Der theoretische Teil dieser Arbeit umfasst auch eine Literaturrecherche bei der vor allem Fachbücher, Magazine aber auch Internetartikel einbezogen wurden. Um die Vertreter der verschiedenen Werkzeuge anhand einer groben Marktanalyse miteinander zu vergleichen wurden Kriterien sowohl aus der Literatur als auch aus den Kundenanforderungen zusammengestellt. Alle Werkzeuge, die im praktischen Teil dieser Arbeit beschriebenen werden wurden während der Erstellung des Prototyps in Form praktischer Arbeit analysiert, bewertet und genutzt. So können insbesondere – aber nicht ausschließlich – in den Kapiteln 8-12 qualifizierte Aussagen über die Werkzeuge getroffen werden.

2. Einführung in Continuous Integration Systeme

Zur Einführung in die Thematik, wird zunächst das Konzept von Continuous Integration theoretisch vorgestellt. Dabei wird der Ablauf eines typischen CI-Prozess anhand eines Szenarios dargestellt und erläutert. Daraufhin werden die verschiedenen Komponenten eines CI- Systems kurz vorgestellt, und ein Überblick gegeben welche Vertreter des jeweiligen Werkzeuges sich zurzeit auf dem Markt befinden.

In der Softwareentwicklung ist unter Integration das zusammenfügen einzelner Softwaremodule zu einem Projekt zu verstehen. Während diese Aufgabe für einen einzelnen Entwickler noch relativ überschaubar ist, stellt sie für ein Entwicklerteam häufig einen großen Aufwand dar, sicherzustellen, dass die zusammengefügten Module fehlerfrei integrierbar sind.

Der Lösungsansatz für die oben ausgeführte Problemstellung ist ein agiles Entwicklungskonzept und nennt sich Continuous Integration (CI). Im Deutschen wird der Ansatz als kontinuierliche Integration bezeichnet. Es finden sich aber auch Bezeichnungen wie permanente oder fortlaufende Integration. Erfunden wurde das Konzept wahrscheinlich von mehreren Entwicklerteams unabhängig voneinander. Durch Kent Beck wurde permanente Integration im Rahmen von „Extreme Programming“ schließlich populär.[1] Nigthly Builds, also das nächtliche Durchführen eines Build-Prozesses, stellen eine vereinfachte Version von Continuous Integration dar.

Continuous Integration ist ein agiler Entwicklungsprozess, bei dem der Code der zu entwickelnden Software so häufig wie möglich integriert, kompiliert, installiert und getestet wird.[2] Hierbei wird immer nur ein kleiner Teil entwickelt und die Änderungen sofort in den Hauptzweig der Entwicklung integriert, sobald der Arbeitsschritt fehlerfrei integriert werden kann. Der Hauptbestandteil einer Continuous Integration Lösung ist, dass die Integration einer Änderung im Sourcecode automatisch einen neuen Build-Prozess anstößt. Um diesen Schritt ohne gewaltigen Mehraufwand für den Entwickler zu gewährleisten, wird durch die Kombination verschiedener Open-Source Werkzeuge ein Continuous Integration System aufgebaut. Die vier zentralen Komponente eines solchen Systems sind ein zentrales Repository beziehungsweise ein Versionskontrollverwaltungssystem, ein Continuous Integration Server, ein Build-System sowie automatisierte UnitTests und optional auch automatische Codeanalysen. Hierbei ist die Nutzung der verschiedenen Werkzeuge optional. Bevor diese Komponente im nächsten Abschnitt näher erläutert werden, soll zunächst der typische Ablauf eines Continuous Integration

[1] Vgl.Wiest, S.(2011), S. 13
[2] Vgl. Wagner, S./ Handte, M.(2012), S. 70

Prozesses anhand eines Szenarios dargestellt werden. Abbildung 1 stellt schematisch den typischen Aufbau eines CI-Systems dar.

Der Ausgangspunkt für die Verwendung eines CI-Systems ist ein Projekt, das mit CI arbeiten soll und dessen Quellcode mit einem Version Control System (VCS) verwaltet wird.[3] Subversion[4] ist ein Beispiel für ein gängiges VCS, diese werden im Laufe dieses Kapitels noch detaillierter beschrieben.

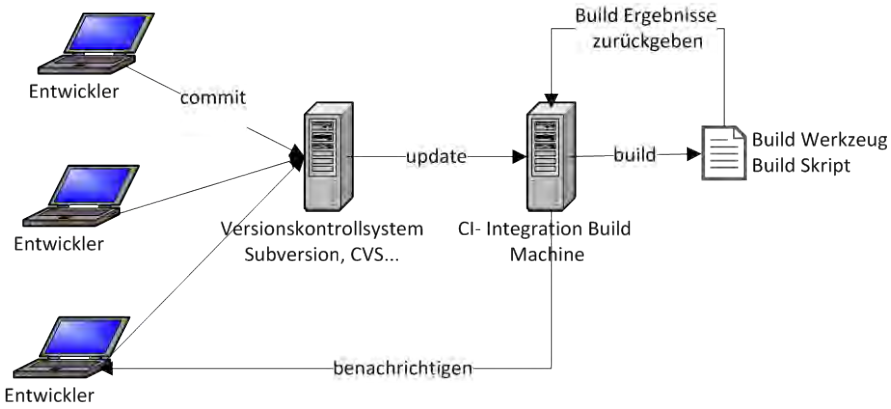


Abbildung 1: Continuous Integration Ablauf

Zunächst wird in dem CI-Server konfiguriert, wo sich das Projekt im Versionskontrollsystem befindet und wie sich der CI-Server gegenüber dem VCS authentifiziert.[5] Daraufhin erzeugt der CI-Server eine lokale Kopie des Projekts. Die Entwickler übergeben ihre Änderungen mindestens einmal am Tag in das VCS. Der CI-Server überprüft regelmäßig, ob im VCS Änderungen vorliegen. Ist dies der Fall, wird automatisch die lokale Datei auf dem CI-Server durch ein Update aktualisiert und automatisch ein Build angestoßen.[6] Die Definition des Build kann je nach Projekt variieren, beinhaltet aber üblicherweise das Kompilieren des Sourcescodes sowie das Durchführen automatisierter Tests. Diese Ausführung übernimmt allerdings nicht der CI-Server sondern ein Build-Tool wie Ant oder Maven. Der Einsatz eines Build Werkzeuges setzt voraus, dass der Bauvorgang eines Projektes komplett automatisiert wurde und die Build spezifischen Dateien ebenfalls im VCS verwaltet werden.[7]

---

[3] Vgl. Feustel, B./ Schluff, S. (2012)
[4] http://subversion.apache.org/
[5] Vgl. Feustel, B./ Schluff, S. (2012)
[6] Vgl. Ebenda
[7] Vgl. Feustel, B./ Schluff, S. (2012)

Der Build kann entweder erfolgreich durchlaufen, vom Build Tool mit einer Warnung versehen werden oder fehlschlagen - zum Beispiel bei Abbruch des Compilers oder bei Überschreiten einer bestimmter Anzahl fehlgeschlagener Tests. [8] Je nach Endstatuts wird er vom CI-Server klassifiziert und die Ergebnisse des Builds werden in Form von Berichten auf dem CI-Server gespeichert und sind über die CI-Server Webseite aufrufbar. Da der CI-Server Zugriff auf das VCS hat, kann er ermitteln welcher Entwickler den letzten CI-Build ausgelöst hat und informiert den Entwickler über E-Mail oder andere Kommunikationswege über das Build-Ergebnis.

Im Folgenden werden die einzelnen Komponenten eines CI-Systems vorgestellt und erläutert. Dabei wird bei der Komponente der Continuous Integration Servern auch eine kurze Marktanalyse durchgeführt.

2.1. Versionsverwaltung

Das Versionskontrollsystem (VCS) stellt die zentrale Schnittstelle für die Zusammenarbeit von mehreren Entwicklern dar.[9] Die Nutzung eines VCS ist Vorrausetzung und damit auch Ausgangspunkt für die Nutzung eines CI-Systems. Durch die Nutzung eines Versionskontrollsystems ist über den gesamten Entwicklungszeitrum nachvollziehbar welche Änderungen zu welchem Zeitpunkt stattgefunden haben und von welchem Entwickler sie hinzugefügt wurden. Durch die zentrale Führung des Projektstandes wird eine konfliktfreie Entwicklung im Team ermöglicht.[10] Im Unternehmensumfeld haben sich vor allem die Versionsverwaltungstools *Subversion und CVS (*Concurrent Versions System) durchgesetzt. Das Quellcode-Repository befindet sich auf einem zentralen Server, was den Nachteil mit sich bringt, dass Änderungen nur eingefügt werden können wenn der Server auch online ist.

Neben den oben genannten zentralen Versionsverwaltungssystemen, besteht auch ein Ansatz der dezentralen Versionsverwaltung. Dezentrale Versionsverwaltungssysteme wie *Git*[11] erweitern den Freiraum der Entwickler in Gestaltung und Zusammenarbeit, allerdings mit dem Nachteil dass sich die Komplexität in der Bedienung des Tools enorm erhöht.[12] Der elementare Unterschied zu einem Quellcode-Repository auf einem zentralen Server besteht darin, dass jeder Entwickler eine Kopie (clone), aus einem remote liegenden Repository, auf seinen eigenen lokalen Rechner ziehen kann. Somit können Commits durchgeführt werden auch wenn keine Verbindung zu dem Server besteht - was einen großen Vorteil für dezentrale Versionsverwal-

[8] Vgl. Ebenda
[9] Vgl. Wickner, B /Müller, B. (2012), S. 299
[10] Vgl. Ebenda S. 300
[11] http://git-scm.com/
[12] Vgl. Wickner, B /Müller, B. (2012), S. 300

tung bringt.[13] Des Weiteren können die Entwickler untereinander Änderungen austauschen und synchronisieren.[14]

Aus dem einführendem Gespräch mit einem Vertreter des Unternehmens ist bekannt geworden, dass bereits das VCS Subversion eingesetzt wird. Die große Mehrheit der Softwareprojekte des Unternehmens ist in einem Subversion VCS abgelegt. Es bedeutete einen erheblichen Aufwand die Projekte in ein anderes VCS zu übertragen.

Wie später in Kapitel 8.1. erläutert werden wird, muss auf dem CI-Server lediglich eine URL zum Verweis auf das VCS hinterlegt werden. Somit ist es auch für die Darstellung und Erarbeitung des Konzeptes von Continuous Integration unerheblich, ob es sich um ein bestimmtes VCS handelt.

2.2. Continuous Integration Server (Marktanalyse)

Der Continuous Integration Server ist eine Anwendung, die bei Änderung des Quellcodes in der Versionsverwaltung einen automatischen Build des gesamten Projekts auslöst. In dieser Anwendung lässt sich eine Vielzahl von Softwareprojekten verwalten. Dabei stellt der Continuous Integration Server die zentrale Schnittstelle zur Verwaltung der Projekte und der in der Umgebung des Servers integrierten Werkzeuge dar. Die Abbildung 2 zeigt exemplarisch die Oberfläche des CI-Servers Jenkins mit den darin angelegten Jobs (hierarchische Elemente zur Verwaltung von verschiedenen Projekten).

Bei der Wahl eines Continuous Integration Servers sollte auf verschiedene Kriterien geachtet werden. Hierbei sollte vor allem zwischen funktionalen und nicht-funktionalen Kriterien unterschieden werden. Aus den Anforderungen des Auftrag gebenden Unternehmens ergibt sich ein besonderer Fokus auf die funktionalen Kriterien:

- Kompatibilität mit der vorhandenen Infrastruktur
- Funktionsumfang und Erweiterbarkeit
- Optimierung für bestimmte Szenarien

[13] Vgl. Gentz, E. (2012)
[14] Vgl. Ebenda

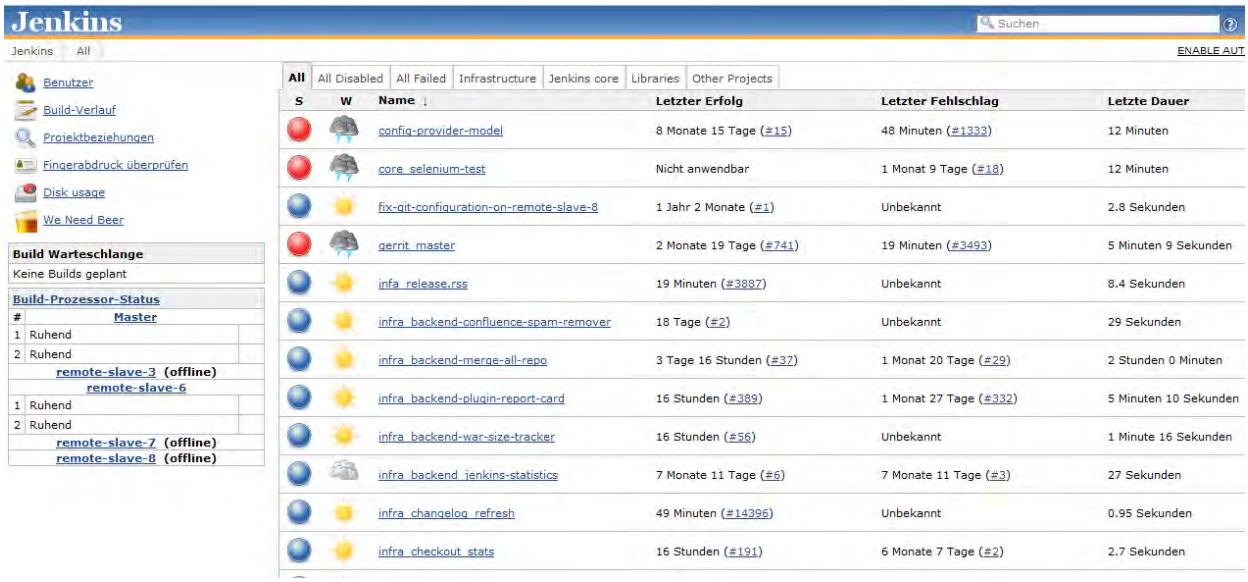


Abbildung 2: Jenkins Oberfläche

Im Folgenden werden die verschiedenen CI-Server anhand ihrer Funktionalitäten miteinander verglichen. Im Sinne der Kompatibilität mit der vorhandenen Infrastruktur sollte vor allem darauf geachtet werden, dass eine Interoperabilität mit einem vorhanden VCS oder einem etabliertem Build-Tool gewährleistet ist.[15] So macht es zum Beispiel keinen Sinn ein Tool zu wählen welches das bestehende Versionskontrollsystem nicht unterstützt und man zunächst ein Plugin entwickeln müsste. Das gleiche gilt für die Unterstützung eines Build-Tools. Hierzu wird aufgezeigt welche Werkzeuge von dem jeweiligen CI-Server unterstützt werden, welche Grundfunktionen ein CI-Server mit sich bringt und inwiefern sich diese von dritter Seite anhand von Plugins funktional erweitern lassen.[16] Ein weiteres Feature stellen die Kommunikationswege dar, die der CI-Server nutzen kann, um den Entwickler über den Verlauf seines Builds zu informieren. Hierbei können z.B. E-Mail aber auch proaktive Kommunikationswege wie Instant-Messenger oder RSS-Feeds zum Einsatz kommen. Die Erweiterbarkeit der Tools bzw. die Anzahl an bereits bestehenden Plugins können ebenfalls für einen Produktverglich in Betracht gezogen werden. Darüber hinaus werden auch nicht-funktionale Kriterien wie zum Beispiel die Einfachheit der Installation, die Bedienbarkeit und die Oberflächengestaltung in Betracht gezogen und im Laufe des Kapitels ebenfalls bewertet.

Die Auswahl an CI-Servern auf den Markt ist sehr groß und vor allem sehr vielfältig. Eine Vergleichsmatrix mit den dreißig bekanntesten Vertretern von CI-Servern wurde von der Firma Thoughtworks entwickelt und kann auf deren Website eingesehen werden.[17] Allerdings ver-

[15] Vgl. Feustel, B./ Schluff, S. (2012)
[16] Vgl. Ebenda
[17] http://confluence.public.thoughtworks.org/display/CC/CI+Feature+Matrix

gleicht diese Matrix CI-Server für alle Programmiersprachen und beinhaltet sowohl CI-Server aus dem Open Source als auch aus dem proprietärem Bereich.

Anhand folgender funktionalen Kriterien konnte eine Vielzahl an CI-Servern ausgeschlossen werden:

- CI-Server, die nicht für Java Projekte geeignet sind.
- CI-Server, die nicht als Open Source Lizenz verfügbar sind.
- CI-Server, die Apache Ant nicht unterstützen und diese Funktion auch nicht via ein Plugin System erweiterbar ist.
- Produkte, die nicht mit dem Versionsverwaltungssystem Subversion kompatibel sind.

Nach Anwendung der Kriterien musste die Auswahl ein weiteres Mal anhand des Bekanntheitsgrades des CI-Servers eingeschränkt werden. Dies ergibt sich aus den Anforderungen des Auftrag gebenden Unternehmens.

Im Folgenden werden die CI-Server Jenkins, Hudson, Continuum und Cruise Control kurz beschrieben und anhand einer Übersichtstabelle sowie einem Ranking miteinander verglichen.

In der Java Software-Entwicklung hat sich **Jenkins**[18] welches früher unter dem Namen Hudson bekannt war, sehr erfolgreich auf dem Markt etabliert und wird von einer großen und aktiven Community unterstützt und weiterentwickelt.[19] Dabei unterliegt Jenkins einer Plugin-basierten Architektur. Dies bedeutet, dass Jenkins nicht in die Anwendung selbst und Plugins getrennt ist, sondern vollständig aus diesen gebaut wird. Jenkins ist als Fork aus einer Abspaltung von dem CI-Server Hudson hervorgegangen. Anfang 2011 kam es zum Streit zwischen der Entwickler Community und Oracle, welches auf Namensrechte bestand und die Migration des Servers auf die Hosting Plattform Github verhindern wollte. Mittlerweile hat Oracle das mit Sun übernommene Open-Source Projekt Hudson an die Eclipse Foundation übergeben.

Aufgrund der Tatsache, dass Jenkins und Hudson auf dem gleichen Code basieren und seit der Spaltung keine bedeutenden neuen Features hinzugekommen sind, sind sich die beiden Tools sehr ähnlich. Einen aufwendigen Installationsprozess gibt es bei **Jenkins** beziehungsweise Hudson nicht. Native Pakete für eine Installation auf einem Unix/Linux System sowie auf Windows stehen auf der Jenkins Website bereit. Alternativ wird Jenkins mit einem integrierten Servlet-Container (Windston) ausgeliefert, was eine Standalone-Ausführung ermöglicht.[20] Zum Starten des Tools muss lediglich die Webarchive (.war) Datei von der Projektwebseite herun-

[18] http://jenkins-ci.org/
[19] Vgl. Wickner, B./ Müller, B.(2012), S. 302
[20] Vgl. Wickner, B./ Müller, B.(2012), S. 303

tergeladen werden und über java jar- jenkins.war gestartet werden. Alternativ kann auch ein Servlet-Container wie Tomcat oder Glassfish genutzt werden. Jenkins verfügt über eine deutschsprachige Web-Oberfläche und ist über diese komplett bedienbar. Es sind keinerlei Konsoleneingaben nötig. Dies gilt auch für die Installation von Plugins oder Versionsupdates. Beim Anlegen eines Jobs können eine Vielzahl von Komponenten konfiguriert werden. Unter anderem lässt sich das Repository über eine URL einbinden, um den zu bauenden Quellcode auszuchecken.

Jenkins unterstützt bereits in der Standard Konfiguration eine Reihe von Testwerkzeugen. Weitere Werkzeuge, wie zum Beispiel Sonar als Werkzeug für Code Coverage Analysen, lassen sich über die Jenkins Plugins nachinstallieren.

Apache Continuum[21] ist ein weiteres Open-Source Produkt welches für die Verwendung mit dem Build-Tool *Maven* konzipiert und in dessen Rahmen von Apache weiterentwickelt wurde.[22] Wie bei Jenkins zeichnen sich die Funktionalitäten durch eine einfache Installation und Konfiguration aus, allerdings müssen bei der Installation Benutzerkonten angelegt werden sowie ein Arbeitsverzeichnis. Apache Continuum unterstützt eine Breite von Versionskontrollsystemen darunter auch CVS, Subversion und Mercruial. Da Continuum auf Maven basiert, wird mit Projekten gearbeitet, die in sogenannten Project Object Models (POM)- Dateien gespeichert sind. Dies ist eine Konfigurationsdatei, die die Schritte eines Builds bestimmt. Zur Verwaltung von Projekten können frei definierte Projektgruppen angelegt werden. Daraufhin wird ein Projekt eingerichtet indem man, im Idealfall, nur die SVN-URL zur Maven POM-Datei angibt oder alternativ eine pom.xml hoch lädt. Ein weiteres Feature von Apache Continuum ist die Unterstützung sogenannter Build-Queues. Dies ist bei Jenkins allerdings auch aufgrund des *Build Pipeline Plugins* möglich. Zusammenfassend zeichnet sich Apache Continuum vor allem durch die hervorragende Integration mit Maven2 und Maven SCM sowie durch ein umfassendes Management der Benutzerrechte und Möglichkeit der Einteilung von Benutzer in Projektgruppen aus. Allerdings fehlt die Möglichkeit Plugins einzubinden. Somit ist z.B. die Integration von Sonar um einiges aufwendiger als bei Jenkins. Abbildung XYZ zeigt den Fokus auf Projekte beziehungsweise die Einteilung in Projektgruppen bei Continuum und den Status der Vorgänge innerhalb der Projektgruppe.

[21] http://continuum.apache.org/
[22] Vgl. Schumacher, M. /Kaman, T. (2008), S. 53

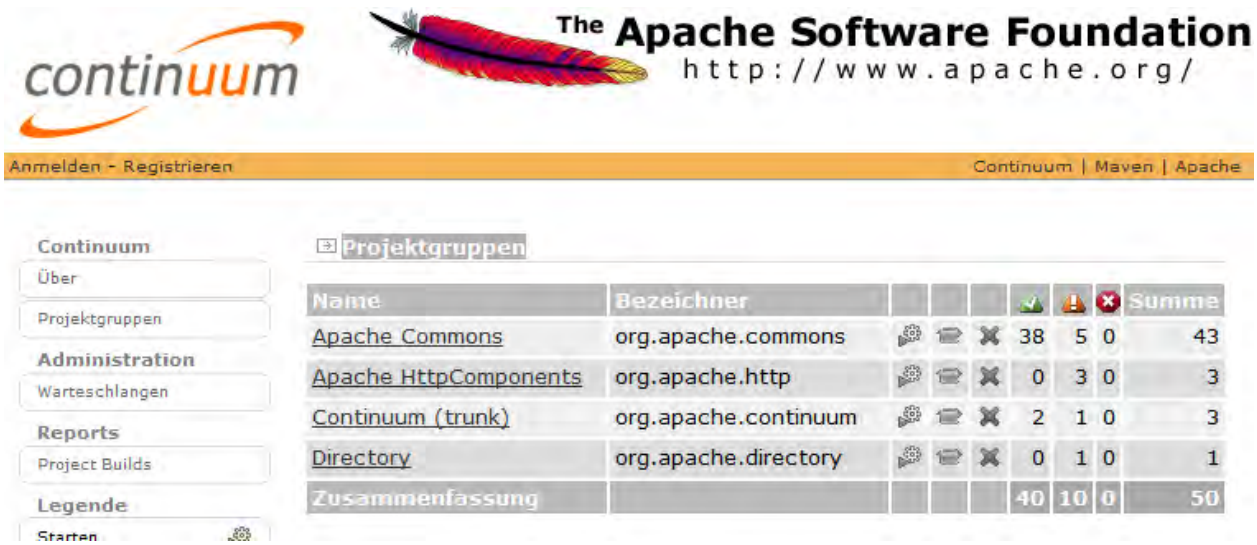


Abbildung 3: Projektgruppen Übersicht in Continuum

Cruise Control[23] war eines der ersten CIS auf dem Markt. Es wird somit auch häufig als Urvater der CI-Server bezeichnet. Cruise Control stellt sowohl ein CI-Tool, als auch ein erweiterbares CI-Framework dar, welches ursprünglich von der Firma Thoughtworks entwickelt wurde, heute aber als Open Source Projekt „unter einer BSD-ähnlichen Lizenz vertrieben wird“.[24] Der Hauptbestandteil des CI-Servers stellt ein Prozess dar, der in einer Schleife („the loop“) Änderungen im CVS feststellt, Builds auslöst und über den Ausgang dieser Builds benachrichtigt. Allerdings steht dem Nutzer, anders als bei Jenkins und Hudson, keine Web-Oberfläche zur Konfiguration zur Verfügung sondern zwei webbasierte GUIs mit einer klassischen Ansicht und einem Dashboard.[25]

Die folgende Übersichtstabelle gibt Auskunft über die Projektleitung beziehungsweise den Anbieter des jeweiligen Open Source CI-Server, die Lizenzform sowie die Versionsnummer des aktuellen Release, anhand welchem die CI-Server miteinander verglichen wurden. Des Weiteren werden die verschiedenen Funktionen und Werkzeuge, die der CI-Server unterstützt, gegenübergestellt.

[23] http://cruisecontrol.sourceforge.net/
[24] Vgl. Schumacher, M. /Kaman, T. (2008), S. 52
[25] Wiest, S.(2011), S. 71

Tabelle 1: CI-Server Funktionalitäten im Vergleich

	Jenkins	Hudson	Continuum	Cruise Control
Projektleitung	Jenkins CI community	Eclipse Foundation Projekt	Apache- Projekt	Cruise Control Projekt Team
Lizenz	Creative Commons Attribution Share-Alike &MIT-Lizenz	MIT Lizenz. Die Eclipse Version ist EPL lizenziert (Eclipse Public license)	Apache 2.0	BSD-style Lizenz
Release	1.492	2.2.1	1.3.8	2.8.4
Benutzermanagement	Plugin (DB basiert)	Plugin (DB basiert)	Eigene DB, LDAP	
Benachrichtigungen	Mail, RSS-Feeds, Instant Messenger, Twitter, Plugins für IDEs & Browser, Desktop-Gadgets	Mail, RSS-Feeds, Instant Messenger, Twitter, Plugins für IDEs & Browser, Desktop-Gadgets	Mail, Jabber and Google Talk, MSN, IRC, report deployment with wagon	E-Mail, http, FTP, sFTP
Build-Tools	Ant, Maven, Maven 2, Maven 3, Gant, Groovy, MSBuild, NAnt, Phing, rake, ruby	Ant, Maven, Maven 2, Maven 3, Gant, Groovy, MSBuild, NAnt, Phing, rake, ruby	Maven 1 and 2, ANT, shell scripts	u.a ANT, Maven, Maven2, nAnt, rake, phing, shellskript
VCS	Subversion, CVS, Git, Mercurial, Perforce	Subversion, CVS, Mercurial, Perforce	CVS, Subversion, Git, Clearcase, Perforce, Starteam, Visual Source Safe, CM Synergy, Bazaar, Mercurial	u.a CVS, Subversion, AccuRev, Git, AlienBrain ClearCase,Git, Mercurial, P4, Plastic SCM, PVCS, VisualWorks Store, etc.
Plug- ins	JUnit und TestNG schon eingebaut. Aktuell 636 Plugins verfügbar	JUnit und TestNG schon eingebaut. Etwa 400 Plugins verfügbar	nein	Plugin Architektur

Anhand der Übersichtstabelle zeigt sich, dass die miteinander verglichenen CI-Server alle sehr gut mit den wichtigsten Vertretern der verschiedenen CI-Werkzeuge kompatibel sind. Darunter die Build-Werkzeuge Ant und Maven sowie die Versionskontrollsysteme Subversion, CVS und Git. Da die Anforderungen des Auftrag gebenden Unternehmens sich auch auf diese Werkzeuge beschränken, ist aufgrund der Übersichtstabelle noch kein Favorit feststellbar. Bei Apaches Continuum ist bereits zu erkennen, dass es sich aufgrund der fehlenden Plugin-Struktur von den anderen CI-Servern unterscheidet. Die Kommunikationswege sind bei allen CI-Servern sehr umfangreich, wobei sich Cruise Control eher auf die traditionellen Emails beschränkt. Die Übersicht zeigt auch, dass sich Jenkins und Hudson im Bereich der Funktionalitäten noch nicht sehr weit auseinander entwickelt haben. Lediglich die Anzahl an vorhandenen Plugins ist im Jahr nach der Spaltung bei Jenkins rapide gestiegen. Das kann mit der sehr aktiven Community begründet werden. Besonders bei der Entwicklung des Prototyps hat sich gezeigt, dass die Größe und Aktivität der Community eine entscheidende Rolle bei der Einführung und Integration eines CI-Servers spielt. So sind viele potentielle Probleme schon erkannt, dokumentiert oder auch bereits behoben.

Aufgrund der Tatsache, dass alle CI-Server die grundlegenden Anforderungen gleichermaßen erfüllen, wird im Folgenden ein Ranking durchgeführt, welches auch nicht-funktionale Kriterien wie die Bedienbarkeit des CI-Severs in Betracht zieht.

2.2.1. Ranking

Die Kriterien stammen zum Teil aus den Anforderungen des Auftrag-gebenden Unternehmens, für welches der Funktionsumfang eines CI-Servers aber auch der Installationsaufwand und der Gesamteindruck von Bedeutung ist. Die Bewertung der einzelnen Kriterien erfolgt aufgrund von Recherche und praktischen Erfahrungen mit den einzelnen CI-Servern. Besonders mit Jenkins haben die Autoren sehr viele praktische Erfahrungen gesammelt. Continuum bietet eine Live-Demo auf seiner Webseite an, durch welche die Autoren sich ein gutes Bild über die Funktionalitäten und die Bedienbarkeit des Tools machen konnten. Im Folgenden werden die ausgewählten Kriterien kurz näher erläutert:

Funktionsumfang: Dieses Kriterium bezieht sich auf die vorherige Übersichtstabelle 1 und bewertet wie viele CI-Werkzeuge von dem CI-Server unterstützt werden. Bei einer Unterstützung von allen gängigen CI-Werkzeugen wie Ant, Maven, CVS, SVN, etc. wurde volle Punktzahl vergeben. Hierzu zählt auch die Integration von Test Werkzeugen wie zum Beispiel JUnit oder TestNG.

Oberfläche: Je nach Übersichtlichkeit der Oberfläche eines CI-Servers gestaltet sich die Bedienung eines CI-Servers. Eine übersichtliche Oberfläche mindert die Einarbeitungszeit für den Entwickler.
Zunächst wird in Betracht gezogen wie die Oberfläche technisch umgesetzt ist, das heißt es wird bewertet ob der CI-Server über eine GUI oder ein Webinterface zu bedienen ist oder womöglich Konsoleneingaben nötig sind. Auch die verfügbaren Sprachen der Weboberflächen gehen in die Bewertung ein. Da die Analyse für ein deutsches Unternehmen durchgeführt wird, wird eine deutschsprachige Oberfläche positiv gewertet.
Bei Bewertung der Oberfläche wurde die Punktzahl generell auf 5 Punkte gesetzt, Abzug gab es im Falle dass keine GUI oder Webinterface vorhanden ist. Auch ein unattraktive, unübersichtliche Optik führt zu Abzügen. Darüber hinaus wird auch die Notwendigkeit von Konsoleneingaben mit einem Abzug gewertet.
Installationsaufwand: Für die Einführung eines CI-Prozesses in das Unternehmen sollte die Hürde des Installationsaufwandes möglichst gering sein. Ein CI-Server sollte ohne komplexe Installationsroutinen installierbar sein. Im Besten Fall führt ein Wizard den Nutzer durch eine Routine. Zur Beurteilung der Installationsroutine wurden die Installationsanleitungen herangezogen.

Konfigurationsaufwand: Nach Installation muss der CI-Server einmalig konfiguriert werden. Neben dem CI-Server und den verschiedenen Werkzeugen die in das System integriert sind, müssen aber die Build /Test und Deployment Prozesses konfiguriert werden.

Dieses Kriterium ist besonders wichtig, da es die meiste Zeit der Implementierung in Anspruch nimmt und somit einen sehr hohen Aufwand darstellt. Zudem birgt die Konfiguration des CI-Servers ein sehr hohes Fehlerpotential. Je mehr Werkzeuge in ein CI-System integriert werden umso höher ist der Konfigurationsaufwand da diese wiederum aufeinander abgestimmt werden müssen.

Zukunftssicherheit: Führt das Unternehmen einen CI-Server ein, so möchte es sicher gehen, dass dieser auch innerhalb der nächsten Jahre weiterentwickelt wird und im Notfall ein Support bereit steht, beziehungsweise ein aktive Community an die man sich mit Problemen wenden kann. Die Community kann anhand der Aktivitäten auf der Hosting Platform Github bewertet werden. Bei einer sehr aktiven Community sind viele „Commits“ zu beobachten, darüber hinaus werden regelmäßig neue Versionen des CI-Servers bereitgestellt.

Gesamteindruck: Bei diesem Kriterium wird die Internetpräsenz des CI-Servers im Allgemeinen in Betracht genommen, Referenzen zu bekannten Firmen sind ebenso wichtig wie eine gute Dokumentation anhand von Screenshots. Funktionierende Links zählen dabei ebenso zum Gesamteindruck.

Gewichtung der Kriterien:

Die folgende Tabelle zeigt die einzelnen Anforderungen mit ihren Gewichtungen und eine kurze Erklärung für die Gewichtung. Die Gewichtung geht von 1 – 5. Dabei spielt 1 die kleinste Rolle und 5 spiegelt eine wichtige Anforderungen dar.

Tabelle 2: Gewichtung der Kriterien an CI-Server

Anforderung	Gewichtung	Begründung
1. Funktionsumfang	5	Sehr wichtig für Kompatibilität mit der bestehenden Infrastruktur des Unternehmens. Die Interoperabilität der verschiedenen Werkzeuge ist von Bedeutung
2. Oberfläche	3	Die Weboberfläche des CI-Servers beeinflusst maßgeblich die Usability des Produkts. Sind Konsoleneingaben erforderlich so wird meist Expertenwissen benötigt. Je besser die Oberfläche gestaltet ist umso weniger Einarbeitungszeit wird benötigt.
3. Installationsaufwand	2	Mit einem geringen Installationsaufwand verkleinert sich die Barriere der Einführung eines CI-Servers, es ist aber unwahrscheinlich dass eine Installation mehrfach durchgeführt wird
4. Konfigurationsaufwand	5	Sehr entscheidend für die Implementation da der Konfigurationsaufwand die meiste Zeit in Anspruch nimmt
5. Zukunftssicherheit	2	Wird die Entwicklung eines Tools nicht weitergeführt so muss das Unternehmen langfristig umsteigen
6. Gesamteindruck	3	Ein guter Gesamteindruck hilft den Entwicklern das Tool anzunehmen. Eine gute Dokumentation hilft Probleme zu beseitigen

Vergleich der Ergebnisse:

Tabelle 3: Ranking der CI-Server

Anforderungen	Jenkins/Hudson	Continuum	Cruise Control	Ergebnis
1. Funktionsumfang	Sehr große Vielfalt an Plugins welche Aufgrund der Plugin Architektur sehr leicht	Besondere Ausrichtung auf Maven 2. Das Tool wird ausgeliefert mit einem eige-	CI-Server und Framework, Live-Build-Überwachung über Weboberflä-	

	einzubinden sind. JUnit ist bereits in der Standardversion enthalten.[26]	nen Jetty Web Server und einer integrierten Datenbank. Plugins sind schwierig zu integrieren. Ein umfassende Rechtemanagement und Unterteilung in Projektgruppen möglich[27]	chen, fehlendes Rechte und Rollensystem[28]	
	5 Punkte (*5)[29]	**3 Punkte (*5)**	**4 Punkte (*5)**	**25|15|20[30]**
2. Oberfläche	Deutschsprachige Weboberfläche, keine Konsoleneingabe nötig. Sehr gute Darstellung von Build Ergebnissen inklusive History.[31]	Dashboard	webbasierte GUIs klassische Ansicht-Dashboard, Konsoleneingaben	
	5 Punkte (*3)	**5 Punkte (*3)**	**4 Punkte (*3)**	**15|15|12**
3. Installationsaufwand	Sehr einfache Installation, versch. Plattformen, Native Pakete, WAR Datei	Benutzerkonten und Arbeitsverzeichnis muss bei Installation direkt angelegt werden	Binary-Distribution herunterladen, Beispielprojekt mitgeliefert.	
	4 Punkte (*2)	**3 Punkte (*2)**	**4 Punkte (*2)**	**8|6|8**
4. Konfigurationsaufwand	Es wird sowohl eine grafische Benutzeroberfläche zur Konfiguration angeboten als auch eine serverseitg gespeicherte XML-Konfigurations Datei.[32] Jenkins/Hudson besitzt eine gute Dokumentation, dies erleichtert die Konfigu-	Es steht eine vorkonfigurierte Seite zur Verfügung, die bereits mit default Werten gefüllt ist. Auch das Design kann konfiguriert werden, beispielsweise mit einem Firmenlogo versehen werden.In Maven 2 können ebenfalls Konfigurationen in der POM Datei vorge-	Sehr hoher Konfigurationsaufwand.[34] Lässt sich nicht über die Oberfläche konfigurieren. Die Konfiguration erfolgt ausschließlich über Änderungen an einer zentralen XML-Datei.[35] Allerdings kann man ein “CruiseControl Configu-	

[26] Eigene Erfahrung der Autoren
[27] Vgl. Brinkman, S. (2013)
[28] Wiest, S. (2011), S. 72
[29] Der Faktor in der Klammer stellt die Gewichtung dar und wird multipliziert
[30] Gewichtete Bewertung
[31] Eigene Erfahrung der Autoren
[32] Wiest, S. (2011), S. 60

	ration erheblich. Default Werte sind teilweise schon eingestellt und es erscheinen Tipps auf der Weboberfläche.[33]	nommen werden, die Continuum auslesen und in das Projekt übernehmen kann.	ration tool" herunterladen. Konfiguration erfolgt dann über eine Swing-Oberfläche Benutzereingaben werden nicht automatisch validiert, keine geführte Konfigurationsmöglichkeit	
	5 Punkte (*5)	**5 Punkte (*5)**	**3 Punkte (*5)**	**25|25|15**
5.Zukunftssicherheit	(Jenkins) Sehr aktive Community inklusive Hauptentwickler von Hudson, wöchentliche Releases	Aktive Community, viel Entwicklungspotenzial. Der Einsatz von Maven-Projekten verbreitet sich immer mehr.	Viele Erfahrungswerte aber Entwickler bei CruiseControl.NET. Es zeigt sich eine deutlich abklingende Weiterentwicklung, große Versionssprünge nicht mehr zu erwarten[36]	
	5 Punkte (*2)	**4 Punkte (*2)**	**3 Punkte (*2)**	**10|8|6**
6.Gesamteindruck	Auf der Jenkins Seite sind viele Tutorials noch auf die Hudson Seite verlinkt. Das Design ist sehr ansprechend und übersichtlich, die Bedienbarkeit ist sehr gut.[37] In einer Umfrage innerhalb der Community zeichnet sich eine sehr hohe Zufriedenheit der Nutzer ab.[38]	Sehr stark auf Maven ausgerichtet. Ein Einsatz macht nur Sinn wenn man Projektgestützt arbeitet d.h. Maven 2 nutzt (POM Dateien). Die Oberfläche ist sehr aufgeräumt. Eine Demo ist ohne herunterladen verfügbar.	Das Design ist etwas veraltet und unübersichtlich. Links auf der Startseite sind falsch verlinkt.[39] Kaum Aktionen innerhalb des Tools möglich. Passive Darstellung des Fortschritts.[40]	
	5 Punkte (*3)	**4 Punkte (*3)**	**3 Punkte (*3)**	**15|12|9**

[34] Vgl. Wiest, S. (2011), S. 71
[35] Vgl. Ebenda, S. 71)
[33] Eigene Erfahrung der Autoren
[36] Vgl. Wiest, S. (2011), S. 71
[37] Eigene Erfahrung der Autoren
[38] Vgl.w.a (2013)
[39] Eigene Erfahrung der Autoren
[40] Vgl. Wiest, S. (2011), S. 71

Gesamt		**98|81|70**[41]

Bewertung der Ergebnisse:

Der Vergleich der verschiedenen CI-Server zeigt ein relativ klares Ergebnis für Jenkins beziehungsweise Hudson. Obwohl diese beiden Tools gemeinsam bewertet wurden ist auf den Unterschied aufmerksam zu machen, dass Jenkins eindeutig die aktivere Community besitzt und somit der Grad der Weiterentwicklungen von Jenkins höher sein sollte als der von Hudson. Somit geht Jenkins als Favorit aus dem Vergleich hervor.

Das liegt vor allem an der einfachen Installation, der Nutzung über die Weboberfläche und das allgemeine sehr überzeugende Auftreten in punkto Benutzerfreundlichkeit, Dokumentation und Internetpräsenz. Allerdings zeigt das Bewertungssystem nicht den Zweck für den der CI-Server eingesetzt werden soll. Ist beispielsweise ein Entwicklerteam auf Maven 2-Projekte spezialisiert so macht ein Einsatz von Continuum durchaus Sinn. Da aus der Anforderungsanalyse des Unternehmens eine Nutzung von Ant hervorgeht, kann Continuum in diesem Vergleich für seine Maven Unterstützung nicht punkten. Die Betrachtung von Cruise Control hat ergeben, dass dieeses sich nur auf die wichtigsten CI-Schritte Schritte: „Änderungen erkennen, Projekt bauen und Team benachrichtigen„ beschränkt.[42] Wohingegen Jenkins und Continuum einen weit größeren Funktionsumfang besitzt. Auch in der Praxis hat sich gezeigt, dass die Integration verschiedener anderer Komponenten in Jenkins sehr einfach und flexibel gestaltet ist. Für alle gängigen Build-, Test- und Analysewerkzeuge gibt es Plugins, welche die Einbindung dieser Werkzeuge ermöglichen. Die anschließende Konfiguration der Komponenten ist gut dokumentiert und bereits in der Jenkins Oberfläche durch Tooltipps und vordefinierte Standardwerte unterstützt. Hinzu kommt eine große Anzahl von Plugins, die mit zusätzlichen Funktionen viele Abläufe vereinfachen. So können zum Beispiel Backups automatisiert angelegt oder verschiedenste Benutzerverwaltung-Werkzeuge wie LDAP angebunden werden.

Während Jenkins als eine Art „aktive Schaltzentrale“[43] für Entwickler zur Verfügung steht, ist die Oberfläche von Cruise Control sehr passiv aufgebaut und erlaubt lediglich das manuelle auslösen von Builds, beschränkt sich ansonsten allerdings nur auf das Anzeigen des Fort-

[41] Addierte Punktzahl des jeweiligen CI-Servers
[42] Vgl. Wiest, S. (2011), S. 72
[43] Vgl. Ebenda, S. 70

schritts eines solchen.[44] Auch die Konfiguration von Cruise Control gestaltet sich um einiges schwieriger als die der anderen CI-.Servern, wie sich aus dem Ranking ergibt.

Zusammenfassen kann man sagen, dass von einer Neueinführung von Cruise Control abgeraten werden sollte, da dieses nicht mit den Funktionalitäten und der Einfachheit der anderen CI-Server mithalten kann. Continuum ist eine durchaus attraktive Alternative zu Jenkins und Hudson, allerdings nur, wenn die sonstige Entwicklung auch auf Maven Projekten basiert.

2.3. Build-Werkzeuge

Der „Build-Prozess“ beinhaltet alle Aktivitäten, die für die Produktion und Bereitstellung von lauffähiger Software notwendig sind wie z.B. das Kompilieren des Gesamtsystems und das Ausführen von Tests mit der erstellten Software.[45] Jede Entwicklungsumgebung verfügt über die Möglichkeit des direkten Kompilieren, Testen und Ausführen der Software. Allerdings geht hierbei die Portabilität des Projektes verloren, da die Konfigurationen an die Entwicklungsumgebung gebunden sind. Zudem ist es für einen Continuous Integration Prozess erforderlich, dass der gesamt Erstellungsprozess einer Software automatisierbar ist. Um die Bindung von der Entwicklungsumgebung zu lösen, werden spezielle Build-Werkzeuge eingesetzt, die eine Automatisierung der Builds ermöglichen.[46] Im Java Umfeld sind als bekannteste Vertreter die Werkzeuge Apache Ant[47] und dessen Nachfolger Apache Maven[48] zu nennen. Bei einem Einsatz von Ant ist der größte Vorteil, dass Build Prozesse sehr fein konfigurierbar sind. Allerdings führt dies auch zu einem großen Aufwand bei der Erstellung von Buildskripten.[49] Den unkomplizierteren Ansatz bietet Maven mit einer festen Projektstruktur und dem Konzept *Konvention über Konfiguration*. Dies bedeutet, dass für jedes Projekt der gleiche Build Life Cycle verwendet wird, welcher aus verschiedenen Phasen zusammengesetzt ist. Individuelle Projektstrukturen sind in einer XML-Datei, dem sogenannten Project Object Model (POM), deklariert. Durch die verwendete Konvention wird die Konfiguration eines Builds sehr stark minimiert. Im Moment ist die Verbreitung von Ant als Build Werkzeug noch höher. Jedoch wird Maven vermehrt in Verbindung mit Testing-Frameworks, die ihm Rahmen von Maven-Projekten konzipiert werden, genutzt.

---

[44] Vgl. Wiest, S. (2011), S. 70
[45] Vgl. it-agile GmbH, (2013)
[46] Vgl. Wickner, B./Müller, B. (2012), S. 300
[47] http://ant.apache.org/
[48] http://maven.apache.org/
[49] Vgl. Wickner, B./Müller, B. (2012), S. 301

3. Testen in einem CI System

Nach einer Studie von Electric Cloud in Zusammenarbeit mit Osterman Research im Jahr 2010 sind die meisten Softwarefehler auf schlechte Testprozesse und mangelhafte Infrastruktur zurück zu führen. [50]

Für viele Entwickler ist der Testprozess eine Qual und aus ihrer Sicht eine Verschwendung ihrer Ressourcen. 85% aller Entwickler geben an, dass die schlechten Testprozesse der Hauptgrund sind, warum fehlerhafte Software ausgeliefert wurde.

Weitere Ergebnisse der Studie sind: [51]

- Nur 12% der befragten Softwareentwickler Teams testen vollautomatisiert. 10% testen nur manuell.
- 46% der Entwickler gaben an nicht genügend Zeit für das Testen zu haben.
- 45 % der Studienteilnehmer gaben, dass der Schaden durch Softwarefehler mehr als 200.000 $ beträgt.

Obwohl die Ergebnisse unbefriedigend sind, zeigen sie auch die wirklichen Probleme des Testens für die Softwareentwickler und die jeweiligen Abteilungen. Vor allem die schlecht organisierten und zeitraubenden Prozesse behindern die Entwickler bei ihrer Arbeit. Daher sollten die Testprozesse so einfach und so zeitschonend wie möglich gestaltet sein.

Dieses Ziel kann mit einer Continuous Integration Lösung erreicht werden. Die einzelnen, oft unabhängigen Testprozesse werden vom CI-Server angestoßen und bearbeitet. Der Tester beziehungsweise der Entwickler gibt durch einen Commit den Anstoß zu einem Build-Prozess und bekommt nach dem Durchlauf eines Builds das Testergebnis.

In dem folgenden Kapitel geht es um das Testen in einem Continuous Integration System, als auch um die Ermittlung von Softwaremetriken. Dabei werden zunächst die Grundlagen des Softwaretestens im Bereich Unit Tests und Systemtest vorgestellt, um ein Grundverständnis des Testens zu vermitteln. Anschließend werden jeweils zwei Open Source Softwarewerkzeuge analysiert und miteinander verglichen. Zusätzlich werden die Ziele und der richtige Einsatz von Softwaremetriken diskutiert und belegt. Darüber hinaus wird ein Open Source Softwareprodukt zur Ermittlung dieser Metriken vorgestellt.

---

[50] Electric Cloud (2010)
[51] Electric Cloud (2010)

3.1. Grundlagen des Softwaretestens

Die Motivation des Softwaretestens ist es nicht nur Fehler im Softwarecode zu finden, sondern diese möglichst frühzeitig zu erkennen. Je später ein Fehler im Softwarelebenszyklus gefunden wird, desto aufwendiger und teurer ist die Beseitigung des Fehlers. Abbildung 4 zeigt die Ergebnisse einer Studie der Software Quality Systems Group zu Fehlerkosten in IT Projekten. Daher ist es wichtig, so zeitnah wie möglich nach der Fertigstellung des Programmcodes diesen zu testen und bei Fehlern zu überarbeiten.

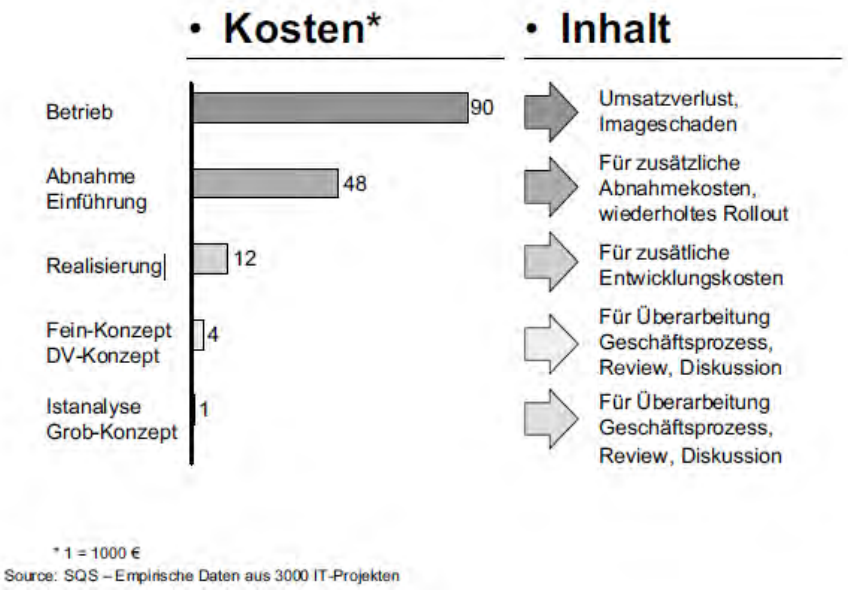


Abbildung 4: Relative Kosten der Fehlerbeseitigung[52]

Beim Softwaretest wird zum einem zwischen der jeweiligen Teststufe als auch der Testmethode unterschieden.

Bei den Testmethoden gibt es zum einen **Black-Box-Tests** welche den Quellcode nicht berücksichtigen sondern die Funktionsweise der Software.[53] Es wird getestet ob die Anwendung gemäß den Anforderungen funktioniert. Black-Box-Tests werden häufig in Verbindung mit Funktionstests, Akzeptanztests oder Abnahmetests angewandt. [54] Black-Box-Tests basieren damit nur auf den Systemspezifikationen.[55]

[52] Vgl. Majer, D. (2008), S.38
[53] Vgl. Kenneth, M.A. (2009)
[54] Vgl. Offutt, J. / Ammann, P. (2008) S. 21
[55] Vgl. Sommerville, I. (2006), S. 549

Bei den **White-Box-Tests** wird der Quellcode der Software berücksichtigt. So werden einzelne Komponenten unabhängig voneinander getestet.[56] Es wird anhand der Quellcodes getestet, ob diese den Anforderungen entsprechen. Häufig angewandt wird dies bei den Modultests oder Integrationstests. [57]

Es ist wichtig zu verstehen, dass sich diese beiden Testmethoden nicht ausschließen sondern ergänzen. Bei kleineren Softwareprojekten ist zwar häufig das White-Box-Test Verfahren ausreichend, jedoch gewinnt mit zunehmender Komplexität das Black-Box-Test Verfahren mehr und mehr an Bedeutung. In der Endlösung dieser Arbeit werden beide Methoden angewandt.

Beide Testmethoden sind für das Testen in einer Continuous Integration Lösung sehr wichtig. Bei einem Continuous Integration System geht es nicht nur darum alle Codeteile möglichst schnell einzuchecken und zu „deployen“, sondern auch die Codeteile automatisch zu testen und das Testergebnis zu veröffentlichen. Es ist möglich beide Testmethoden in solch einem System zu integrieren. White-Box-Tests werden häufig in Form von Unit Tests implementiert und Black-Box-Tests als Oberflächentests.

Die Teststufen werden im Folgenden dargestellt. Bei einem Unit Test oder auch Modultest genannt wird die jeweilige Funktion oder Methode getestet. Damit wird die Funktionalität gewährleistet. Diese werden hauptsächlich vom Entwickler selbst getestet. Bei einer testgetriebenen Entwicklung werden zuerst die Testfälle geschrieben. Danach wird solange implementiert, bis die ausgeführte Testfälle als erfolgreich durchlaufen.

Der Integrationstest testet die Komptabilität der einzelnen Module miteinander. Allerdings ist die Abgrenzung zwischen Unit Test und Integrationstest oft ungenau. In der Literatur gibt es unterschiedliche Abgrenzungen und Definitionen. Das Testen eine Datenbankverbindung, welche über eine Klasse im Quellcode aufgerufen wird kann beispielsweise auch ein Integrationstest sein, da hierbei die Klasse an sich und die Datenbank kommunizieren. In dieser Seminararbeit wird das Testen verschiedener Klassen miteinander ebenfalls als Integrationstest definiert. [58]

Bei dem Systemtest wird das System als Ganzes getestet. Dabei wird vor allem darauf geachtet, dass das System den zu Beginn definierten Anforderungen und Qualitätsmerkmalen entspricht. Der Systemtest sollte in einer produktionsnahen Umgebung getestet werden. Beim

[56] Vgl. Ebenda S. 549
[57] Vgl. Offutt, J. / Ammann, P. (2008) S. 21
[58] Vgl. Cleff, T. (2010) S. 19

Abnahmetest, welcher meist vom Kunden durchgeführt wird, wird das System in der produktiven Umgebung mit Echtdaten getestet. [59]

Ein Continuous Integration System deckt bestenfalls alle verschiedenen Teststufen ab. Somit kann gewährleistet werden, dass alle Codeteile sofort und in jeder Teststufe getestet werden. Das bedeutet, dass ein Unit Test, Integrationstest und der Systemtest automatisiert angestoßen und ausgeführt werden. Das Ziel ist es den Fehler so zeitnah wie möglich zu finden - am besten direkt nach dem der Entwickler den betreffenden Code entwickelt und veröffentlicht hat. In der vorgestellten CI-Lösung werden alle Teststufen berücksichtigt.

3.2. Marktanalyse der einzelnen Testwerkzeuge

Im Folgenden werden jeweils zwei verschiedene Testwerkzeuge für White-Box-Tests und Black-Box-Tests vorgestellt. Bei allen vorgestellten Werkzeugen handelt es sich um Open-Source Produkte für die Java Entwicklung. Um ein genauen Vergleich zwischen den Programmen ziehen zu können, sind die Informationen nicht nur aus der Literatur entnommen, sondern wurden durch praktische Anwendung erworben. Als Entwicklungsumgebung wurde dabei Eclipse verwendet.

3.2.1. Modultest

Im Bereich der Unit Tests gibt es zwei große Vertreter. Dies ist das sehr bekannte JUnit und TestNG.

JUnit ist ein Testframework, welches es ermöglicht automatisierte Modultests zu entwickeln. Dadurch ist es möglich Methoden auf Klassenebene zu testen.[60] Entwickelt wurde JUnit von Erich Gamma und Kent Beck.[61] JUnit ist sehr einfach in Eclipse oder auch in Netbeans zu integrieren.[62] Um Methoden mit JUnit zu testen muss ein Test Code programmiert werden, welche sich in einer eigenen Klasse befindet. JUnit bietet einige nützliche Funktionen, die diese Programmierung erleichtern. Der Entwickler wählt die zu testenden Klasse mit den jeweiligen Methoden aus und JUnit implementiert automatisch die passenden Methodenrümpfe. JUnit arbeitet mit Annotationen und Zusicherungen. Die wichtigste Annotation ist @Test, dadurch wird die Testmethode gekennzeichnet.[63] Weitere wichtige Annotationen sind @Ignore mit der sich Klassen oder Methoden ignorieren lassen und damit nicht getestet werden. Weiterhin ist es möglich mit der Annotation @BeforeClass Funktionalitäten aufzurufen, die vor dem Testen

[59] Vgl. Ebenda, S. 19
[60] Vgl. Dömer, S. (2010)
[61] Vgl. JUnit Sourceforge (2013)
[62] Eigene Erfahrungen der Autorin
[63] Vgl. Westphal, F. (2005)

der einzelnen Methoden aufgerufen werden sollen, beispielweise der Aufbau einer Datenbankverbindung. Zusicherungen werden benötigt um die Methode auf bestimmte Werte zu testen, beispielsweise kann mit der Zusicherung assertTrue getestet werden, ob die Aussage wahr ist oder mit assertEquals ob sie einem bestimmten Wert entspricht.[64]

Damit bietet JUnit viele verschiedene Funktionalitäten, um die Testfälle gemäß den Anforderungen anzupassen.

TestNG ist ebenfalls ein Test-Framework zum Durchführen von Unit- und Integrationstests. Es orientiert sich sehr stark an dem JUnit Framework, weicht jedoch in einigen Punkten ab. Beispielsweise werden bei TestNG Parameter und Integrationstest in einer XML Datei angegeben.[65] Obwohl TestNG erst seit 2004 auf dem Markt ist und daher weniger bekannt, gibt es für die gängigen Entwicklungsumgebungen, Build Automatisierungsanbieter und Metrik Werkzeuge einfache Plugins. Immer mehr Entwicklerteams benutzen TestNG in ihren Testprozessen, da es insgesamt sehr flexibel ist und inhaltlich mehr Funktionalitäten bietet als der bekannte Wettbewerber JUnit. TestNG arbeitet ebenfalls mit Annotationen und Zusicherungen. Diese sind den Annotationen und Zusicherungen von JUnit sehr ähnlich und werden daher nicht nochmals ausgeführt.[66]

3.2.1.1. Vergleich von Werkzeugen zum Modultesten

Anforderungen für den Vergleich

Im Folgenden werden die Anforderungen an ein Unittestwerkzeug definiert und erklärt. Die Anforderungen stammen zum Einem aus den wissenschaftlichen Recherchen als auch aus den eigenen Erfahrungen. Um die Testwerkzeuge besser beurteilen zu können, wurden sie selber praktisch vom Projektteam ausprobiert und einfache Testfälle implementiert.

1. Einfache Implementierung in die Entwicklungsumgebung

Da Unit Tests vom Entwickler lokal getestet werden, sollte die Integration dieses Testwerkzeugs in die vorhandene Entwicklungsumgebung wie beispielsweise Eclipse oder Netbeans unkompliziert sein. Das Bedienen des Testwerkzeugs sollte benutzerfreundlich und klar ersichtlich sein. Da das Testen der einzelnen Units Grundbestandteil eines Testprozesses sein sollte, ist es sehr wichtig, dass das Testwerkzeug kompatibel zu anderen Entwicklungswerkzeugen ist. Beispielsweise zu Build Automatisierungswerkzeugen oder CI- Servern.

---

[64] Vgl. Beck, K. (2005), S. 26
[65] Vgl. Jammy (2010)
[66] Vgl. Mkyong (2009)

2. Bereitgestellte Annotation

Das Testen soll für den Entwickler möglichst einfach und unkompliziert gestaltet sein. Dabei helfen Annotationen, mit denen der Entwickler angeben kann, ob beispielsweise bestimmte Ereignisse vor Beginn oder nach Beendigung eines Testes ausgeführt werden sollen. Die Annotationen sollten klar verständlich sein. Zwei wichtige Ereignisse sind beispielsweise das Ignorieren eines Testfalls und der Timeout.[67] Mit dem Timeout hat der Entwickler die Möglichkeit einen Test abzubrechen und als Fehler zu deklarieren, wenn die angegebene Zeit überschritten wurde. Das Abfangen von Exceptiones ist ebenfalls mit den bereitgestellten Annotationen möglich.

3. Suite Test – Das Testen verschiedener Klassen in einem Testdurchgang

Neben den einzelnen Methoden in einer Klasse, sollte es ebenfalls möglich sein mehrere Testfälle in einem Testdurchgang zusammen zu setzen. Dabei kann vor allem die Komptabilität dieser getestet werden. [68] Zudem ist es Ressourcensparen mehr viele Testfälle in einem Testdurchgang testen zu können.

4. Testen von Exceptions

Neben den normalen Methoden ist es wichtig implementierte Exceptions zu testen. Diese werden unter bestimmten Voraussetzungen aufgerufen. Dabei wird zwischen den erwarteten und den unerwarteten Exceptions unterschieden. Erwartete Exceptions können beispielsweise eine NullPointerException sein. Problematischer sind die unerwarteten Exceptions. Diese müssen dennoch bei der Programmierung des Testcodes beachtet werden, da sonst der Test fehl schlägt sofern die Exceptions nicht aufgefangen wird.

5. Parametrisierung

Parametrisierung spielt für die Testautomatisierung eine sehr große Rolle. Die verschiedenen Testfälle sollen oftmals mit verschiedenen Testdaten getestet werden. Um zu vermeiden, dass die Testdaten vor jedem Testdurchlauf manuell verändert werden müssen, ist eine Parametrisierung der Testfälle notwendig. Nur so können die Testfälle automatisiert mit Werten hinterlegt werden.

6. Weiterentwicklung des Testframeworks

Keine Software ist perfekt, auch kein Testwerkzeug. Daher sollte das Feedback der Nutzer oder eventuelle Bugs entsprechend aufgenommen und aktualisiert werden. Zudem entwickelt

[67] Vgl. Mkyong (2009)
[68] Vgl. Mkyong (2009)

sich die Programmiersprache Java ebenfalls weiter, daher ist es wichtig zeitnahe Veränderungen aktiv einzubauen.

7. Anzahl der Mitglieder in einer Community

Eine aktive Community kann bei Fragen und Problemen sehr hilfreich für den Entwickler sein. In Blogs oder Foren kann dieser nach ähnlichen Problemen und Lösungsvorschlägen suchen[69]. Tutorials können zudem bei Schwierigkeiten schnell und kostenlos helfen.

Prioritäten

Um die zwei Testwerkzeuge besser miteinander zu vergleichen, werden die einzelnen Anforderungen mit Prioritäten gewichtet. Die Priorisierung stammt zum einem aus wissenschaftlichen Recherchen, als auch aus eigenen Erfahrungen des Projektteams. Dazu wurden die Testwerkzeuge praktisch ausprobiert und insbesondere mit JUnit einige Testfälle implementiert.

Die folgende Tabelle zeigt die einzelnen Anforderungen mit ihren Gewichtungen und eine kurze Erklärung für die Gewichtung. Die Gewichtung geht von 1 – 5. Dabei spielt 1 die kleinste Rolle und 5 stellt eine wichtige Anforderung dar.

Tabelle 4: Gewichtung der Kriterien für Testwerkzeuge

Anforderung	Gewichtung	Begründung
Implementierung	3	Für den Entwickler ist es zwar einfacher, wenn das Plugin für das jeweilige Testprogramm vorimplementiert ist oder leicht zu installieren ist, aber hierbei handelt es sich um ein einmaliges Problem. Ist dieses einmal behoben, tritt es im Verlauf des Projektes kaum bzw. gar nicht mehr auf.
Annotationen	4	Annotationen sind absolut maßgebend für das Erstellen von Unit Tests. Sie deklarieren die Testfälle und werden benötigt, um weitere wichtige Konfigurationen zu implementieren.[70]

[69] Eigene Erfahrung der Autorin
[70] Eigene Erfahrungen der Autorin

Suite Test	4	Suite Test sind wichtig, um verschiedenen Testklassen miteinander zu testen. Dadurch kann die Funktionalität in Abhängigkeit von anderen Testklassen ermittelt werden.[71]
Exceptions Test	3	Das Testen von erwarteten Exceptions und das Abfangen von unerwarteten Exceptions sind sehr wichtig für das Erstellen von Testfällen. Ohne diese Funktionalität können Testfälle fehlschlagen, obwohl sie den Anforderungen entsprechen.
Parametrisierung	5	Die Hinterlegung von Testdaten für die einzelnen Testfälle ist ein Grundbaustein der Testautomatisierung. Nur mit Parametrisierung ist es möglich einen Testfall mit verschiedenen Werten automatisiert zu testen.
Weiterentwicklung	5	Die Weiterentwicklung eines Testwerkzeugs ist sehr wichtig, da vor allem bei CI-System vorrausschauend und langfristig mit den Testwerkzeugen geplant werden muss. Veränderungen in der Programmiersprache Java müssen bei der Weiterentwicklung berücksichtigt werden. Ebenfalls zu berücksichtigen ist das Feedback der Nutzer und eventuelle Bugs.
Community	3	Eine aktive Community kann dem Entwickler sehr hilfreich sein. Oftmals sind dort auftretende Probleme bereits diskutiert worden und können bei der Lösung des Problems helfen.

Ergebnis des Vergleichs

Die folgende Tabelle zeigt das Ergebnis des Vergleichs zwischen **JUnit** und **TestNG** anhand der definierten Kriterien. Die Ergebnisse stammen aus den eigenen praktischen Erfahrungen des Projektteams und den wissenschaftlichen Recherchen. Die jeweiligen Punkte sind mit der Gewichtung multipliziert und ergeben das Endergebnis für die Anforderungen. Zusammenfassend werden die einzelnen Teilergebnisse addiert.

Tabelle 5: Ranking der Testwerkzeuge

Anforderung	JUnit	TestNG	Ergebnis

[71] Vgl. Mkyong (2009)

1. Implementierung	Oftmals ist JUnit bereits in Eclipse oder Netbeans vor konfiguriert und kann sofort benutzt werden.	Die Implementierung von TestNG in Eclipse oder Netbeans ist sehr gut, leider ist es häufig nicht vorkonfiguriert.	
	5 Punkte (*3)	**4 Punkte (*3)**	**15 | 12**
2. Annotationen	JUnit bietet eine übersichtliche Auswahl an Annotationen. Dabei sind alle wichtigen Annotationen vorhanden, wie beispielsweise das Ignorieren von Testfällen mit @Ignore[72] oder das Timeout[73]. Die Benutzung der Annotationen ist sehr einfach und selbsterklärend**.**	TestNG bietet eine Vielzahl an Annotationen. Die Grundlegenden Annotationen sind dabei identisch zu den Annotationen von JUnit. Nur in wenigen Ausnahmen unterscheiden sich diese. Allerdings bietet TestNG noch weitere Annotationen und damit weitere Funktionalitäten wie zum Beispiel die Gruppierung von Testfällen.[74]	
	4 Punkte (*4)	**5 Punkte (*4)**	**16 | 20**
3. Suite Test	JUnit bietet die Möglichkeiten von Suite Tests. Diese Implementierung ist sehr einfach unproblematisch.[75]	Der Suite Test ist mit TestNG sehr einfach möglich. Allerdings wird hierbei mit XML gearbeitet. Ebenfalls ist das Gruppieren von Testfällen möglich. Damit können gezielt nur die Testfälle einer jeweiligen Gruppe getestet werden.[76]	
	4 Punkte (*4)	**5 Punkte (*4)**	**16 | 20**

[72] Vgl. Ullenboom, C. (2010)
[73] Vgl. Mkyong (2009)
[74] Vgl. TestNG (2013)
[75] Vgl. Tutorials.de (2007)
[76] Vgl. TestNG (2013)

4.Exceptions Test	Mit JUnit ist eine Implementierung von Exceptions sehr einfach und unproblematisch. Somit ist das dasTesten von erwarteten und unerwarteten Exceptions ist ohne Probleme möglich. [77]	Mit TestNG ist das Testen von erwarteten Exceptions und unerwarteten Exceptions ebenfalls einfach und ohne Probleme möglich.[78]	
	5 Punkte (*3)	**5 Punkte (*3)**	**15 | 15**
5. Parametrisierung	Mit JUnit ist eine Parametrisierung möglich der Testfälle möglich. Die benötigen Annotationen sind mit @RunWith und @Parameters. Allerdings benötigt es etwas Einarbeitungszeit, um die Programmierung dieser Parameter zu verstehen.[79] Wenn diese aber verstanden sind, ist es sogar möglich Daten aus einem File oder einer Datenbank zu lesen. [80]	TestNG bietet ebenfalls eine Parametrisierung der Testfälle an, allerdings ist dies nur mit XML Dateien möglich. Der klare Nachteil ist, dass der Entwickler die XML Dateien schreiben und anlegen muss. Dadurch können aber gleiche Testdaten und damit Parameter für verschiedene Testfälle angewandt werden. Die Parameterdaten können so auch von anderen Abteilungen als einfach dem Entwickler übergeben werden. Mit dieser Methode ist es allerdings nicht mögliche komplexe Datentypen als Parameter zu verwenden. Bei diesen Datentypen muss die Annotation @DataProvider implementiert werden.[81]	
	4 Punkte (*5)	**5 Punkte (*5)**	**20 | 25**
7. Weiterentwick-	Stand November 2012 ist die Version JUnit 4.11 das neuste	Stand November 2012 ist die neuste Version 6.8. Die-	

[77] Vgl.Vogel, L. (2012)
[78] Vgl. Konigsberg, R.(2007)
[79] Eigene Erfahrung der Autorin
[80] Vgl. JUnit Sourceforge (2013)
[81] Vgl. Kaczanowski, T. (2012), S.44

lung	Release. Die vierte Version ist im Jahr 2006 erschienen, davor gab es sieben Jahre keine neue Version des Testframeworks. Mit dem Release von Java 8, wird eine neue Version erwartet.	se ist ebenfalls im November erschienen. Es erscheinen häufig neue Releases.	
	4 Punkte (*5)	**5 Punkte (*5)**	**20 | 25**
8. Community	Yahoo Gruppe als Community mit über 31.000 Mitgliedern. Da es als de-facto Standard in der Softwareentwicklung eingesetzt wird, gibt es unzählige Foren und Hilfestellungen	Obwohl TestNG immer mehr eingesetzt wird, gibt es noch wenige Communities. Insbesondere im deutschsprachigen Umfeld ist TestNG noch relativ unbekannt.	
	5 Punkte (*5)	**3 Punkte (*3)**	**15 | 9**
gesamt			**117| 126**

Evaluation des Ergebnisses

Obwohl TestNG mit 126 Punkten gegenüber JUnit mit 117 der klare Favorit ist, sollten zusätzliche Aspekte betrachtet werden. Diese lassen sich nicht in Zahlen ausdrücken und umfassen z.B. die Erfahrung, die Entwickler bereits mit JUnit gesammelt haben. Zwar ist TestNG in vielen Punkten ähnlich wie JUnit, jedoch weist es gerade bei der Handhabung mit XML Daten wesentliche Unterschiede auf. Im dem Falle, dass Entwickler bisher mit JUnit keine Probleme hatten oder es keinen Bedarf gibt für bessere Parametrisierung, Group Testing oder Dependency Tests wird es schwierig sein, diese von einem neuem Test Framework zu überzeugen. Zwar gibt es viele Tutorials wie bisheriger JUnit Test Code einfach in TestNG Code umgewandelt werden kann, trotzdem müssen sich die Entwickler und Testingenieure zunächst mit XML vertraut machen.

Daher gilt es hier das Verhältnis von Kosten und Nutzen zu beachten. Zwar sind beides Open Source Programme, dennoch sollte die Umstellung nicht unterschätzt werden. Doch am wichtigsten ist es mit den betreffenden Personen zu reden, sie sind die Schlüsselfigur im Testprozess. Die Testwerkzeuge sollten den Entwickler bei seiner Arbeit unterstützen und diese nicht zusätzlich erschweren.

Vor diesem Hintergrund soll beachtet werden, dass das Unternehmen bereits zum JUnit für Modultests verwendet. Hinzu kommt, dass gängige CI-Server sowohl JUnit als auch TestNG integrieren können. So gibt es also auch aus dieser Perspektive keinen Grund von dem bisher verwendeten JUnit Abstand zu nehmen.

3.2.2. Das Testen von Weboberflächen

Das Testen von Weboberflächen gehört zur Kategorie des Black-Box-Testens. Hierbei sieht der Tester nicht den Quellcode der zu testenden Software sondern nur die grafische Oberfläche. Im Folgenden wird gezeigt wie solch ein Testwerkzeug arbeitet, welche Vorteile das Testen von Weboberflächen hat und welche aktuellen Open Source Anbieter auf dem Markt vertreten sind.

Die meisten Testwerkzeuge arbeiten nach Capture / Replay Verfahren. Diese besteht aus den drei Schritten: [82]

- Aufnahme des Testvorgangs
- Bearbeitung und Anpassung des Testvorgangs
- Automatisiertes Abspielen des Testvorgangs

Bei der Aufnahme des Testvorgangs werden alle einzelnen Schritte des Testers aufgenommen und automatisch in der jeweiligen Skriptsprache geschrieben und gespeichert. Jedes Klicken oder Eintragen von Daten wird aufgezeichnet. Jedes Objekt, wie zum Beispiel ein Button oder ein Textfeld, bekommt automatisch einen eindeutigen Namen zugewiesen. Bei der Bearbeitung und Anpassung des Testvorgangs werden die Testfälle falls notwendig parametrisiert und Checkpoints implementiert. Checkpoints werden benötigt, um zu validieren, ob der Testfall erfolgreich war oder nicht. Dabei werden die Testergebnisse mit dem Soll Wert verglichen.

Beim automatisierten Abspielen des Testvorgangs beginnt das eigentliche Testen. Für eine Reihe von Werten wird die Applikation auf ihre Anforderungen getestet. Falls ein Test nicht erfolgreich durchläuft bieten viele Anbieter spezielle Reports an, welche zeigen an welchen Schritt und mit welchen Werten es zum Fehler kam.

Vorteile vom Testen von Weboberflächen

Bevor es zum Testen von Weboberflächen kommt, wurde der Quellcode bestenfalls mehrmals mit Unit Tests und Integrationstest getestet. Beim Testen von Weboberflächen wird jedoch nicht nur getestet, ob die Funktionalität den Anforderungen entspricht, sondern auch, ob dieses

[82] Vgl. Grechenig, T. et al. (2009), S. 33

entsprechend angezeigt wird. Wichtig zu verstehen ist, dass mit einem Oberflächentest nicht nur die eigentliche Oberfläche getestet wird, sondern das ganze System.[83]

3.2.2.1. Marktanalyse

Die folgende Marktanalyse fokussiert sich auf die Vertreter von Testwerkzeugen für Webapplikationen im Open Source Bereich.

Da der Hauptaspekt dieser Seminararbeit nicht die Marktanalyse von Testwerkzeugen ist sind nur zwei bekannteste Testwerkzeuge miteinander verglichen. Für eine bessere Vergleichbarkeit wurden die jeweiligen Anforderungen priorisiert.

Selenium

Selenium ist das bekannteste Open Source Testwerkzeug zum Testen von Webapplikationen. Es basiert auf HTML und Java Script und ist von der Firma ThoughtWorks im Jahre 2004 entwickelt worden.[84] Es kann direkt im Browser (vorzugsweise Firefox) genutzt werden und simuliert die Browsereingaben. Selenium besteht aus den folgenden Komponenten:[85]

Selenium IDE: Hierbei handelt es sich um ein Firefox Plugin, mit dem einfache Testfälle nach dem Capture/Replay Verfahren aufgenommen und ausgeführt werden können.

Selenium RC: Mit dem Selenium RC ist es möglich die Testfälle in unterschiedlichen Browsern ausgeführt werden. Außerdem können selbst entwickelte Skripts ausgeführt werden.

Selenium Grid: Mit dem Selenium Grid ist es möglich selbst geschriebene Selenium Skripts gleichzeitig in verschiedenen Browsern und auf verteilten Systemen zu testen.

Dabei wird unterschieden wie komplex die zu testenden Anwendungen sind und welche Funktionen benötigt werden. In dieser Arbeit steht das Testen von Webapplikationen nicht im Mittelpunkt, daher wird die einfachste Version das Selenium IDE benutzt.[86] Für den späteren Vergleich mit Canoo WebTest werden jedoch alle Komponenten von Selenium betrachtet.

Selenium verbindet Einfachheit und Flexibilität in einem. Durch das einfache Capture/Replay Verfahren können Testfälle einfach und schnell aufgenommen werden. Das Testskript wird in einer Selenium eigenen Sprache geschrieben. Es unterstützt alle bekannten Programmiersprachen wie .Net, Perl, Python, Ruby und natürlich Java[87]. Der Support ist bei Selenium ebenfalls

---

[83] Scholz, F. (o.J.)
[84] http://seleniumhq.org/
[85] Vgl. Rukes, C. / Weich, W. / Neuhaus, D. (2010)
[86] Vgl. Selenium (2013)
[87] Vgl. Selenium (2013)

ein besonderes Merkmal. Durch die Popularität gibt es viele Communitys, die dem Entwickler helfen können.

Canoo WebTest[88]

Canoo WebTest ist ebenfalls ein Open Source Testautomatisierungswerkzeug zum Testen von Webapplikationen. Es ist im Gegensatz zu Selenium nicht so weit verbreitet. Canoo WebTest zeichnet sich besonders durch den Support für den Anwender aus. Obwohl es Open Source ist, stehen zahlreiche Hilfen für den Anwender zur Verfügung.[89] Ein weiteres besonderes Merkmal ist die einfache Benutzung von Canoo WebTest. Die Syntax mit der Testfälle geschrieben sind, ist sehr einfach zu verstehen und zu erlernen. [90] Die Geschwindigkeit mit der die Testfälle ausgeführt werden ist ebenfalls sehr gut. Das wird dadurch erreicht, dass Canoo WebTest bei dem Testen der Webapplikation keine CSS Dateien runter lädt. Das Test Reporting zeigt ebenfalls klar die fehlerhaften Testschritte und eignet sich sehr gut, um die Fehler schnell zu erkennen. Ein weiterer Vorteil ist, dass Canoo Webtest reines Java ist, damit kann es überall ausgeführt werden. Die einzige Voraussetzung ist eine installierte Java Development Kit (JDK). Durch Konfigurationen und Erweiterungen lässt sich Canoo WebTest individuell den jeweiligen Anforderungen anpassen.

Da die Canoo WebTest Skripts als Ant Skripts aufgebaut sind, eignet es sich besonders gut für eine Integration in ein CI-System, wenn Ant als Build Automatisierungstool verwendet wird. Es unterstützt neben Java die Programmiersprachen PHP, ASP, .Net. und weitere.

Um mit der Webapplikation zu kommunizieren wird die HtmlUnit API verwendet. HtmlUnit ist ein Java Tool, welches das Verhalten des Browser simuliert.[91] Das zeigt wiederrum auch den großen Nachteil von Canoo WebTest. Es ist lediglich möglich zu testen, wie die Browser Firefox und der Internet Explorer mit der Webapplikation umgehen.

3.2.2.2. Vergleich von Werkzeugen zum Testen von Weboberflächen

Anforderungen für den Vergleich

Die folgenden Anforderungen stammen zum einen aus wissenschaftlichen Quellen, als auch aus eigenen Erfahrungen des Projektteams. Dabei sind die Erfahrungen und Kenntnisse nicht nur auf dieses Projekt zurück zu führen, sondern auch aus anderen Projekten und Hausarbeiten in denen es um Testautomatisierung mit grafischen Oberflächen ging. Insbesondere mit

[88] http://webtest.canoo.com/webtest/manual/WebTestHome.html
[89] Vgl Canoo WebTest (2013)
[90] Vgl. Ebenda
[91] Vgl. Ebenda

dem nicht Open Source Produkt HP Quicktest konnten vielen Erfahrungen gesammelt und in diese Arbeit miteingebracht werden.

1. Parametrisierung

Wie auch bei den Unit Tests sind Parameter sehr wichtig für Testautomatisierung. Ohne eine geeignete Auswahl an Testdaten ist keine sinnvolle Testautomatisierung möglich. Der Mehrwert von Testautomatisierung wird zudem erst erreicht, wenn die Testfälle wiederholt für viele verschiedene Szenarien getestet werden kann.[92]

2. Anpassen des Testfalls (Checkpoints, Synchronisation)

Nachdem ein Testfall aufgenommen wurde, ist er noch lange nicht zum Testen geeignet. Es müssen oftmals viele Anpassungen vorgenommen werden. Beispielsweise müssen Checkpoints eingefügt werden. Außerdem sind Synchronisationen wichtig, damit das Testobjekt richtig getestet werden kann.[93] Checkpoints sind notwendig, um zu deklarieren wann ein Testfall erfolgreich ist und wann eben nicht.[94] Es dient dazu den Soll-Zustand eines Objektes zu definieren. Das kann beispielsweise ein bestimmter Wert in einem Textfeld sein. Die Synchronisierung ist notwendig, um Verzögerungen in den Testvorgang zu implementieren. Diese Verzögerungen können beispielsweise durch Ladezeiten oder Zugriffe auf externe Quellen entstehen. Ohne eine implementierte Synchronisation im Testwerkzeug kann der Test fehlschlagen.[95]

3. Installation und Bedienbarkeit

Um die Hürde des Testens von Weboberflächen so gering wie möglich zu halten, ist es wichtig alle Schritte so einfach wie möglich zu gestalten. Je einfacher ein Testwerkzeug bei der Installation und besonders bei der Bedienung ist, desto eher wird es in den Entwicklungsprozess aufgenommen und vor allem von den Entwicklern und Testern angenommen und angewandt. Zudem soll Testautomatisierung den Testprozess beschleunigen, daher muss das Testwerkzeug mit wenigen Schritten zum gewünschten Ergebnis führen.

4. Testen von verschiedenen Browsern

Jeder Entwickler von Weboberflächen kennt das Problem der verschiedenen Browser. In jedem Browser sieht die Webseite anders aus und agiert anders.[96] Daher ist es wichtig die Oberflächen in möglichst viele verschiedenen Browsern zu prüfen, um ihre Funktionalität ausgiebig zu testen.

---

[92] Vgl. Ghazali, F.(2005)
[93] Vgl. Schuster, G. (2012)
[94] Vgl. Taentzer (2012)
[95] Vgl. Scholz, F. (o.J.)
[96] Eigene Erfahrungen der Autorin

5. Darstellung der Testergebnisse

Besonders interessant werden Testreports wenn ein Testdurchlauf nicht erfolgreich war. Dabei ist es wichtig so detailliert wie möglich zu erfahren, an welchem Schritt mit welchen Werten der Test erfolglos war.[97] Die Testreporte sollten so übersichtlich wie möglich gestaltet sein, um einen schnellen Blick auf die wichtigen Ergebnisse zu liefern.

6. Integration in den Entwicklungsprozess

Um den Testprozess weiter zu automatisieren ist es wichtig alle Testwerkzeuge gemeinsam in einen Entwicklungsprozess zu integrieren. Dies gilt besonders hinsichtlich dem Ziel das Testwerkzeug in ein CI-System zu integrieren.

7. Dokumentation und Support

Die Entwickler und Tester müssen sich in das Testwerkzeug einlernen und einlesen. Daher ist eine gute Dokumentation und Support durch beispielsweise Wikis oder Foren ein sehr wichtiger Punkt.

Prioritäten

Wie bereits bei dem Vergleich von JUnit und TestNG, werden die einzelnen die Gewichtungen zu den jeweiligen Anforderungen beziffert und begründet. Dabei geht die Gewichtung von 1 – 5, absteigend in der Wichtigkeit. Bei der Gewichtung und Begründung der jeweiligen Anforderungen spielt vor allem die eigenen Erfahrungen eine Rolle. Durch andere Projekte im Bereich des Testens konnte praktisches Wissen erlangt werden.

Tabelle 6: Gewichtung der Kriterien für Weboberflächentest

Anforderung	Gewichtung	Begründung
Parametrisierung	**4**	Ohne Parametrisierung ist keine Testautomatisierung vorstellbar. Testfälle sollten möglichst mit verschiedenen Werten getestet werden, um ein optimales Ergebnis zu ermitteln.[98] Das Testwerkzeug sollte es ermöglichen die Parametrisierung der Testfälle unkompliziert zu gestalten. Das Testwerkzeug sollte auch hinsichtlich der Implementierung von externen Datenquellen wie Excel oder HTML

[97] Vgl. Scholz, F. (o.J.)
[98] Vgl. Ghazali, F.(2005)

		untersucht werden.
Anpassung des Testfalls	**4**	Nach der Aufnahme eines Testfalls in der Capture beziehungsweise Record Phase muss dieser oftmals noch weiter angepasst und konfiguriert werden. Beispiele für solche Anpassungen sind Synchronisierungen oder Check Points. Ohne die Möglichkeit den Testfall individuell anpassen zu können, ist es nicht möglich die Testfälle so zu konfigurieren, dass sie den Anforderungen gerecht werden. Die Bearbeitung der Testfälle sollte daher unkompliziert und einfach durchzuführen sein.[99]
Installation und Bedienbarkeit	**5**	Ein großes Problem für den Entwickler beziehungsweise den Tester ist es, wenn das Testwerkzeug in seiner Bedienung zu umständlich und komplex ist. Die Installation und die erstmalige Konfiguration des Testwerkzeugs sollte ebenfalls leicht und verständlich sein. Dadurch werden auch die Einstiegshürden verringert. [100]
Testen von verschiedenen Browsern	**4**	Es ist wichtig die Weboberflächen mit verschiedenen Browsern zu testen, insbesondere wenn zu erwarteten ist, dass unterschiedlichste Besucher/Nutzer auf die Seite zugreifen werden.
Testreports	**4**	Um schnell zu erkennen warum ein Test fehlschlug sind übersichtliche Testreports wichtig.[101] Alle einzelnen Testschritte sollten dargestellt werden, um zu erkennen welcher Testschritt noch fehlerhaft ist. Testreports können zudem archiviert werden und später mit anderen Testfällen verglichen werden.
Integration in den Entwicklungsprozess	**5**	Die Integration des Testwerkzeugs ist mit das wichtigste Kriterium in dieser Marktanalyse. Ziel ist es ein CI-System aufzubauen. Daher ist es wichtig, dass alle Komponenten möglichst unkompliziert integriert werden können. Nur bei einer unkomplizierten Eingliederung in die Prozesse wird das Tool langfristig effektiv genutzt werden.
Dokumentation und Support	**5**	Mit geeigneter Dokumentation und Support lässt sich zum einen das Testwerkzeug häufig besser bedienen, zum anderen können Anfangsprobleme

[99] Vgl. Scholz, F. (o.J.)
[100] Eigene Erfahrungen Sarah Kieninger
[101] Vgl. Ghazali, F.(2005)

		schneller gelöst werden. Ein schneller und hilfreicher Support beschleunigt und erleichtert die Arbeit des Entwicklers.

Ergebnis des Vergleichs

Die folgende Tabelle zeigt das Ergebnis des Vergleichs zwischen **Selenium** und **Canoo WebTest** anhand der definierten Kriterien. Die Ergebnisse stammen aus den eigenen praktischen Erfahrungen des Projektteams und den wissenschaftlichen Recherchen. Die jeweiligen Punkte sind mit der Gewichtung multipliziert und ergeben das Endergebnis für die Anforderungen. Zusammenfassend werden die einzelnen Teilergebnisse addiert.

Tabelle 7: Ranking der Werkzeuge für Oberflächentests

Anforderungen	**Selenium**	**Canoo WebTest**	**Ergebnis**
Parametrisierung	Eine Parametrisierung ist nicht mit der einfachsten Version Selenium IDE möglich. Mit den anderen Komponenten ist es ohne Probleme möglich. Mit Java und SQL können die Testfälle parametrisiert werden.[102]	Die Parametrisierung ist mit Canoo WebTest einfach zu implementieren. Testdaten können auch mit Excel Tabellen ausgelesen und verwendet werden.	
	3 Punkte (*4)	**4 Punkte (*4)**	**12 | 16**
2. Anpassung des Testfalls	Die Anpassung des Testfalls ist mit Selenium ohne Probleme möglich. Bei Selenium IDE können die Check Points und Synchronisierung per Mausklick eingefügt werden. Bei den anderen Selenium Komponenten müssen diese programmiert werden.[103] Somit können die Testfäl-	Canoo WebTest bietet eine sehr große Auswahl an verschiedenen Konfigurationsmöglichkeiten wie beispielsweise Check Points und Synchronisierung. Allerdings müssen diese wie auch bei Selenium immer programmiert werden und sich nicht so einfach einzufügen wie bei Selenium IDE.	

[102] Vgl. Selenium (2013)
[103] Vgl. Selenium (2013)

	le unkompliziert, je nach den Bedürfnissen angepasst werden**.**		
	4 Punkte (*4)	**3 Punkte (*4)**	**16 | 12**
3. Installation und Bedienbarkeit	Die Installation und die Bedienbarkeit von Selenium IDE sind sehr einfach. [104] Bei der einfachsten Version muss nur ein Plugin installiert werden. Die Oberfläche von Selenium IDE ist selbst erklärend und benötigt kaum Einarbeitungszeit.[105] Allerdings ist der erste Überblick über die einzelnen Selenium Komponenten kompliziert.	Die Installation von Canoo WebTest ist schwieriger als von Selenium. Die Dokumentation für die Installation ist ebenfalls sehr gering. Die Implementierung von Testfällen, sowie die allgemeine Bedienung mit dem Programm sind allerdings sehr einfach. [106]	
	3 Punkte (*5)	**3 Punkte (*5)**	**15 | 15**
4. Testen von verschiedenen Browsern	Die Testdurchführung des Testfalls ist mit Selenium RC Server ohne Probleme möglich. Mit der einfachen Version Selenium IDE ist allerdings nur ein Testen im Firefox möglich.	Canoo WebTest arbeitet mit HtmlUnit.[107] Dieser simuliert den Browser und kommuniziert so mit der Webapplikation. Derzeit ist es damit nur möglich zu testen wie der Firefox und der Internet Explorer mit der Webapplikation kommunizieren.	
	3 Punkte (*4)	**1 Punkt (*4)**	**12 | 4**
5. Testreports	Die Standardversion von Selenium IDE bietet keine automatisierten Testreports. Allerdings ist es möglich diese mit einem zusätzlichen Plugin zu integrieren. Soll bei einer CI-Lösung	Beim Canoo WebTest ist die Möglichkeit zur Erstellung von Testreports standardmäßig integriert. Es ist ebenfalls möglich diese in einer CI-Lösung an den CI-Server zu übermitteln.	

[104] Eigene Erfahrungen der Autorin
[105] Eigene Erfahrungen der Autorin
[106] Vgl. Webtest Canoo (2013)
[107] Vgl. Ebenda

	die Ergebnisse an den CI-Server gesendet werden, ist dies ebenfalls über Plugins möglich		
	3 Punkte (*4)	**4 Punkte (*4)**	**12 | 16**
6. Integration in den Entwicklungsprozess	Aufgrund des hohen Bekanntheitsgrades gibt es für jeden bekannten CI-Server Plugins für Selenium. Ebenfalls gibt es für Eclipse Plugins. Die Integration in ein CI-System von Selenium IDE ist sehr einfach.[108]	Da Canoo Webtest auf der Struktur von Ant aufbaut ist es überhaupt kein Problem, Canoo WebTest mit Ant zu synchronisieren. Auch bei den anderen Softwaretools gibt es viele Plugins.	
	4 Punkte (*5)	**5 Punkte (*5)**	**20 | 25**
7. Dokumentation und Support	Die Dokumentation von Selenium ist sehr gut. Durch die weite Verbreitung gibt es auch viele Communities die Hilfestellungen geben können. Aufgrund des Bekanntheitsgrades gibt es auch einige Drittanbieter die Support für Selenium anbieten.	Obwohl Canoo WebTest einen geringeren Bekanntheitsgrad hat ist die Dokumentation sehr detailliert und hilfreich für den Entwickler.	
	5 Punkte (*5)	**5 Punkte (*5)**	**25 | 25**
Endergebnis			**112| 113**

Das Ergebnis zeigt, dass Canoo Webtest - obwohl es so unbekannt ist - durchaus mit dem berühmten Vertreter Selenium aufnehmen kann. Daher sollte insbesondere bei neuen Softwareprojekten Canoo Webtest durchaus in Erwägung gezogen werden. Allerdings gilt auch hier wieder, es kommt auf die entsprechenden Anforderungen und vor allem auf die Entwickler an. Das Ergebnis zeigt ebenfalls, dass die Grundfunktionalitäten wie Parametrisierung und die Anpassung des Testfalls von beiden Testwerkzeugen gleichermaßen gut angeboten werden. Ein großer Nachteil von Selenium ist, dass es auf den ersten Blick aufgrund der verschiedenen Komponenten unübersichtlich und sehr komplex wirken kann. Der Entwickler muss sich zu-

[108] Eigene Erfahrungen der Autorin.

nächst mit der Aufteilung der jeweiligen Komponenten auseinander setzen und das passende für ihn finden. Andererseits kann der Entwickler so verhindern, ein Testwerkzeug zu integrieren, welches die eigentlichen Anforderungen weit übersteigt.

Die Dokumentation zur Bedienbarkeit ist bei beiden Testwerkzeugen sehr umfassend und leicht zugänglich. Hinsichtlich der Dokumentation zur Installation und erstmaligen Konfiguration schwächelt Canoo Webtest ein wenig.

Bezüglich der Integration von den Testwerkzeugen in den Entwicklungsprozess und dabei insbesondere in einem CI-System sind beide Vertreter sehr empfehlenswert. Aus eigenen Erfahrungen konnte erkannt werden, wie einfach die Konfiguration mit Selenium in ein bereits existierendes CI-System ist.

Der große Unterschied zwischen Canoo Webtest und Selenium ist das Testen der Testfälle in verschiedenen Browsern. Aufgrund der Architektur vom Canoo WebTest ist das Testen mit verschiedenen Browsern sehr eingeschränkt. Obwohl dieser Aspekt als großer Nachteil gesehen werden kann, muss bedacht werden, ob das jeweilige Unternehmen beziehungsweise Projekt diese Anforderung überhaupt benötigt. Soll eine Webapplikation getestet werden, welche nur intern über den Standardbrowser aufgerufen wird, muss nicht die Funktionalität von anderen Browsern getestet werden.

Die Tabelle zeigt nicht alle Faktoren und Aspekte, wonach zwischen Canoo WebTest und Selenium unterschieden werden sollte. Es ist ebenfalls zu beachten, welche Vorkenntnisse in der jeweiligen Skriptsprache vorhanden sind. Canoo WebTest zeichnet sich dabei durch eine sehr leicht zu erlernende Skriptsprache aus. Weiter ist zu bedenken, ob verschiedene Testfälle gleichzeitig getestet werden sollen. Dies ist wiederrum mit Selenium Grid möglich.

Zusammenfassend sollten wie bereits erwähnt, besonders die Anforderungen und die Bedürfnisse des Entwickler beziehungsweise des Teams beachtet werden. Ein Testwerkzeug soll den Testprozess und die betreffenden Personen unterstützen und nicht ihre Arbeit erschweren.

4. Metriken der Softwarequalität

„A methodology for establishing quality requirements and identifying, implementing, analyzing and validating the process and product software quality metrics is defined. The methodology spans the entire software life-cycle."[109]

Softwaremetriken sind Maßzahlen, die dafür geeignet sind Qualitätsmerkmale von Software zu messen. Das Ziel von Softwaremetriken ist es Software vergleichbar zu machen und die Qualität zu verbessern.[110] Es soll helfen Trends und Tendenzen eines Softwareprodukts frühzeitig zu erkennen und entsprechende Entscheidungen zu treffen. Außerdem sind Softwaremetriken hilfreich, um zu erkennen wie der aktuelle Projektstatus ist.[111]

Es gibt verschiedene Arten von Metriken wie Prozessmetrik oder Aufwandsmetrik. In dem folgenden Abschnitt geht es jedoch um die Produktmetrik, dabei ist das Produkt die zu entwickelnde Software.

Die Produktmetriken unterteilen sich weiter in folgenden Metriken:[112]

- Umfang (Lines of Code, Number of Classes)
- Komplexität des Codes
- Kopplung & Abstraktion (Schnittstellen, Vererbungstiefe)
- Lesbarkeit
- Laufzeiten
- Fehleranfälligkeit

Umfangsmetriken

Bei den Umfangsmetriken gibt es unterschiedliche Ansätze, welche Zeilen des Quellcodes, beziehungsweise welche Anzahlen von Klasse oder Methoden gezählt werden sollen. Dabei wird beispielsweise unterschieden zwischen allen Zeilen, ausführbaren Zeilen oder kommentierten Zeilen. Bei dieser Art von Metrik ergeben sich jedoch einige Probleme. Es kann keine Aussage über die Funktionalität oder die Komplexität der Software getroffen werden. Außerdem ist es schwierig Projekte mit verschiedenen Programmiersprachen zu vergleichen. Ein

[109] Vgl. IEEE Standards Association (1998)
[110] Vgl. Pichler, M. (2009)
[111] Vgl. Kreissl, H. (2004)
[112] Vgl. Pichler, M. (2009)

weiterer negativer Aspekt ist zudem, dass Entwickler bei dem Ziel möglichst eine hohe Umfangsmetrik zu erreichen viel mit Redundanzen arbeitet um die Zahl künstlich zu erhöhen. [113]

Komplexitätsmetriken

Neben den Umfangsmetriken kann zusätzlich die Komplexität des Softwarecodes gemessen werden. Als Maß für Komplexitätsstrukturen wird die Anzahl von Kontrollstrukturen oder Verzweigungen als Berechnungsgrundlage herangezogen.[114]

Ein Beispiel für eine Komplexitätsmetrik ist die Zyklomatische Komplexität von McCabe. Der Grundgedanke hinter dieser Berechnung ist, dass ein Softwarecode umso komplexer ist, je mehr Verzweigungen er hat. Da es mehr Möglichkeiten gibt, wie das Programm weiterläuft. Bei einfachen Verzweigungen wie *if* gibt es dabei nur zwei Zweige, das heißt zwei Möglichkeiten. Bei Case und Switch Verzweigungen können dies deutlich mehr sein. Die Formel zur Berechnung dieser Metrik ist in der Abbildung 5 abgebildet.[115]

Zyklomatische Komplexität =	
Anzahl der Verzweigungen	(if)
+ Anzahl der Schleifen	(for, while, repeat)
+ je: (Anzahl der Zweige –1)	(case-, switch-verzweigungen

Abbildung 5: Abbildung xy: Darstellung der Zyklomatische Komplexitätsformel[116]

Der klare Vorteil dieser Formel ist, dass im Gegensatz zu den *lines of Code* hier ein direkter Vergleich von verschiedenen Programmiersprachen gezogen werden kann.

Je höher der Wert dieser Metrik ist, desto schwieriger ist es den entsprechenden Code zu testen und zu verstehen. Nach McCabe ist eine Zyklomatische Komplexität unter 10 niedrig, zwischen 10 und 20 mittel, zwischen 20 – 50 hoch und über 50 undurchschaubar.[117]

4.1. Sonar als Werkzeug zum Messen von Metriken

Als eins der bekanntesten Tools für die Code Analyse und für die Zusammenstellung eines Dashboards von Metriken ist Sonar[118]. Sonar eignet sich sehr gut für eine Continuous Integra-

[113] Vgl. Pichler, M. (2009)
[114] Vgl. Pichler, M. (2009)
[115] Vgl. Schneider, K. (2012), S. 66
[116] Vgl. Ebenda S. 66
[117] Vgl. Schneider, K. (2012), S. 66

tion Lösung, da es über zahlreiche Plugins für die wichtigsten Vertreter von CI-Servern verfügt. Zum Anderem ist es über eine Weboberfläche zu steuern und somit zentral für alle zugänglich und einsehbar. Obwohl Sonar mit vielen vorgefertigte Metriken und Coderegeln vorimplementiert ist, hat der Anwender die Möglichkeit Sonar seinen Bedürfnissen und Anforderungen anzupassen. Diese Regeln umfassen in der Voreinstellung gängige Best Practices wie das Setzen von Klammern, Benennungsregeln bis hin zur komplexen Erkennung von Fehlermustern. Es ist außerdem möglich, die Ergebnisse der Unit Tests sowie den Anteil, des von diesen Tests abgedeckten, Quellcodes auf dem Dashboard anzeigen zu lassen. Damit ist es möglich ein zentrales Dashboard mit allen wichtigen Metriken und Ergebnissen zu Verfügung zu stellen. Ebenfalls ist es möglich die Zeitliche Abhängigkeit und den Projektstatus mit den Metriken zu verbinden. Dadurch kann analysiert werden, welche Metriken sich in welchen Zeiträumen verändert haben und welche Trends zu erkennen sind. Ausgehend von jeder einzelnen Metrik, die auf dem Dashboard dargestellt ist, ist es möglich bis auf des Detaillevel des ausgewerteten Quellcodes tiefer zu gehen. So können leicht potenzielle Fehlerquellen identifiziert und deren Entwicklung über die Analysen der letzten Tests zurückverfolgt werden.

Abbildung 6 zeigt das Dashboard von Sonar auf der höchsten Abstraktionsebene. Die abgebildete Auswertung betrifft ein angewähltes Softwareprojekt. Hier sind die Standardmetriken wie Anzahl der Zeilen an Quellcode, Anzahl der Klassen, Anzahl der Kommentarzeilen, die Quellcodeabdeckung der Unittests und der Grad der Einhaltung der hinterlegten Regeln zu erkennen. Zu jeder Metrik wird die Veränderungsrate in Relation zu der letzten Codeanalyse angezeigt.

[118] http://www.sonarsource.org/

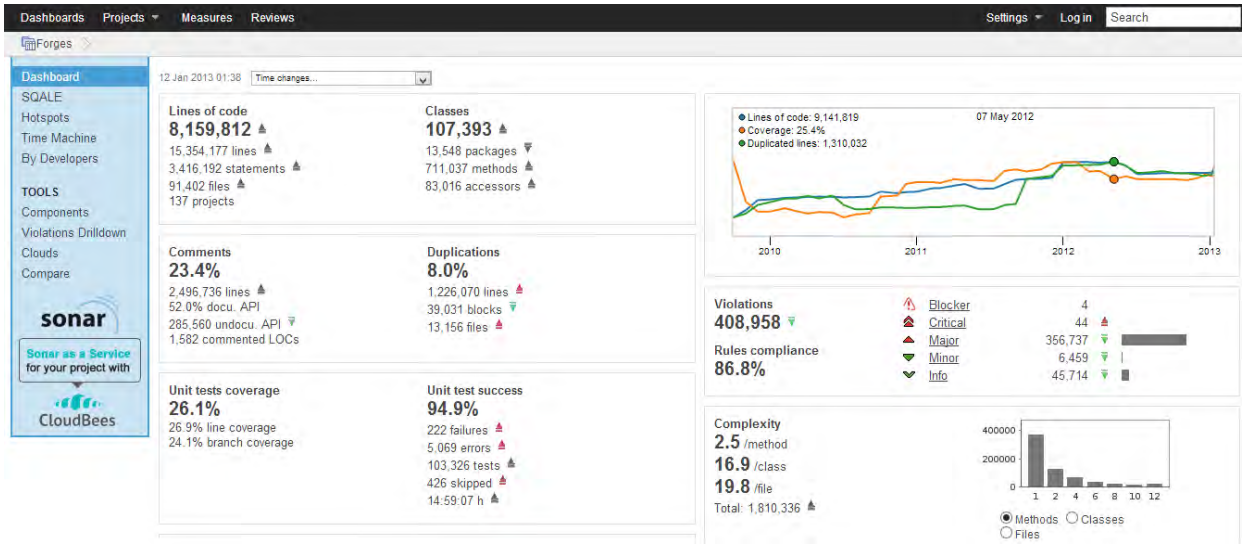


Abbildung 6: Dashboard von Sonar[119]

4.2. Weitere Coding Rule Engines

Im Umfeld der Softwareentwicklung sind bei vielen Entwicklern weitere Coding Rule Engines bekannt. Diese werden meist als Plugins in die IDE eingefügt und analysieren den geschriebenen Quellcode anhand des lokalen Arbeitsbereiches des Entwicklers. Die bekanntesten Vertreter sind hier CheckStyle, PMD und Findbugs. Diese drei Vertreter decken unterschiedliche Coding Regeln ab:

Checkstyle deckt hauptsächlich Konventionen ab. Dabei wird unter Anderem überprüft, ob public Methoden mit javadoc dokumentiert sind, Benennungsregeln eingehalten wurden oder der Quellcode in einem konsistenten Format geschrieben wurde.

PMD dient der Erkennung von bad prctices. Dies sind potentielle Fehlerquellen wie zum Beispiel direkte Implementierung anstelle der Verwendung von Interfaces, zu komplexe Methoden, ungenutzter Quellcode oder das catchen von Exceptions ohne diese zu behandeln.

Findbugs hilft bei der Identifizierung von nicht offensichtlichen, potentiellen Fehlern wie Deadlocks oder das Zurückgeben von Referenzen an mutable Objekte.

Die Coding Regeln, die von den drei oben genannten Werkzeugen abgedeckt werden finden sich bereits in der Standardkonfiguration von Sonar wieder und sind in die Metriken eingebunden. Viele andere Coding Rules Engines können per Plugin in die Analyse von Sonar eingebunden werden.

---

[119] Vgl. Sonar (2012)

4.3. Richtiger Einsatz von Softwaremetriken

Wie bereits beschrieben gibt es eine Vielzahl von verschiedenen Softwaremetriken. Allerdings sind dies alles nur Zahlen, die ohne ein genaues Ziel oder einen Vergleich eine sehr geringe Aussagekraft haben. Es müssen Grenz- und Referenzwerte vorliegen, um die ermittelten Ergebnisse einteilen zu können. Aus der Vielzahl an verschiedenen Softwaremetriken sollte allerdings nicht wahllos gewählt werden. Zunächst sollte das eigentliche Ziel des Projektes klar sein. Die Metriken sollten dann entsprechend ausgewählt werden. Daraus kann ein sogenannter Metrikplan erstellt werden, der angibt welche Metriken wie und wie oft ermittelt werden sollen und welche Konsequenzen bei unbefriedigenden Ergebnis entstehen. Alle ausgewählten Metriken sollten übersichtlich auf einem Dashboard präsentiert sein. In regelmäßigen Abständen sind die Metriken zu ermitteln und dem Management mitgeteilt werden.

Das Management sollte sich jedoch nicht nur auf Softwaremetriken beschränken, um die Arbeit der Entwickler zu bewerten. Der Vergleich von den Metriken ermöglicht zwar eine Transparenz, allerdings nicht ohne Konsequenzen. Durch diesen direkten Vergleich können sich Entwickler unter Druck gesetzt fühlen und die Metriken für ihren Vorteil beschönigen. Das wiederum zerstört den Wert dieser Metriken und macht sie nutzlos.

5. Veränderung der Testprozesse in einem CI System

Um die Vorteile und den Mehrwert für den Entwickler und für die Softwareentwicklung darzustellen sind im Folgenden die verschiedenen Testprozesse für den jeweiligen Automatisierungsgrad des Testvorgangs aufgezeigt. Die Testprozesse sind in die drei Bereiche Unit Tests, Integrationstests und Systemtest unterteilt.

5.1. Manuelles Testen

Der Testprozess für das manuelle Testverfahren ist in der Abbildung 7 zu erkennen. Die einzelnen Schritte werden im Folgenden erläutert.

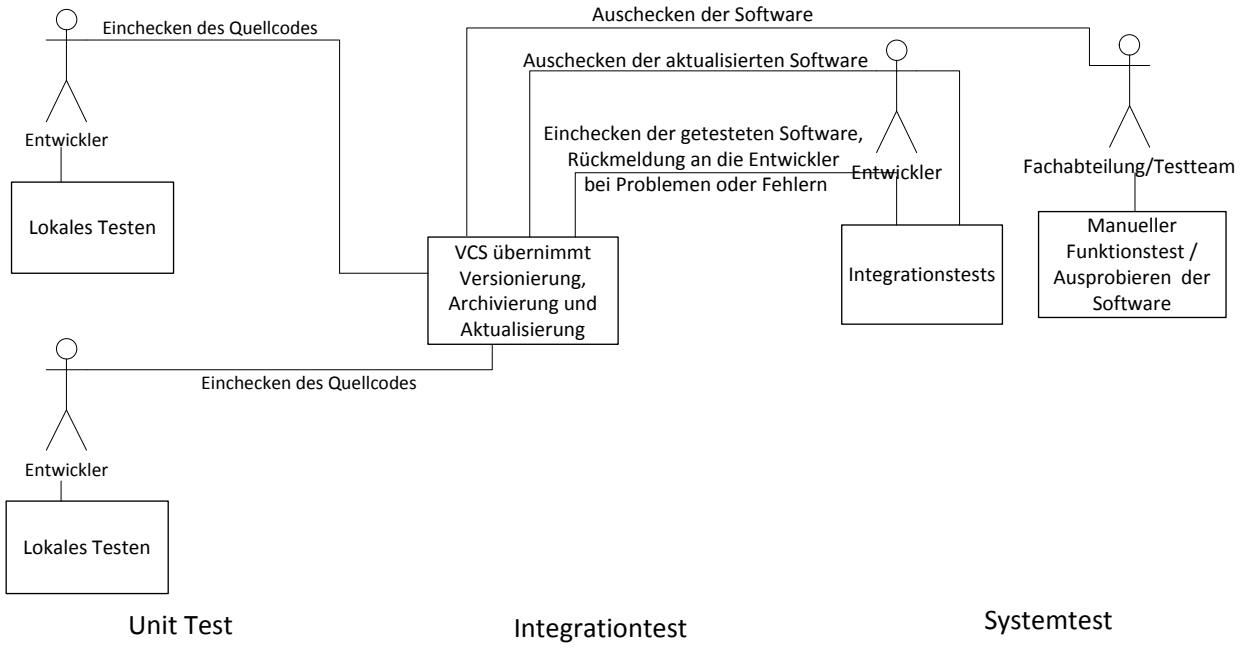


Abbildung 7: Testprozess für das manuelle Testverfahren

Beim komplett manuellen Testen wird zunächst der Quellcode beim Entwickler lokal getestet. Dabei ist zu unterscheiden, ob der Entwickler komplett manuell testet, indem einzelne Funktionen aufgerufen und mit Daten getestet werden oder, ob ein Automatisierungstool eingesetzt wird. In den meisten Fällen werden Unit Tests mit Hilfe von einem Automatisierungstool getestet. Um diese Tools nutzen zu können, muss der Entwickler vorher jedoch einen Test Code passend zum Testfall programmieren. Dieser ist notwendig, um zu definieren welche Methoden mit welchen Daten getestet werden sollen und welche Ergebnisse zu erwarten sind. Test Code wird häufiger weiser parallel zum Testfall programmiert.

Automatisierungstools sind sehr häufig einfach mit der eingesetzten Entwicklungsumgebungen wie zum Beispiel Eclipse oder Netbeans zu integrieren. Beim Kompilieren des Softwarestücks wird der Test Code automatisch ausgeführt und somit die einzelnen Module getestet. Obwohl diese Art und Weise zunächst sehr aufwendig erscheint bringt sie doch eindeutige, zeitliche Vorteile bei der späteren Weiterentwicklung.

Der Test Code muss häufig nur einmalig geschrieben und implementiert werden. Bei späteren Veränderungen am Quellcode des Softwarestücks können die Funktionen sofort bezüglich ihrer Funktionalität getestet werden. Allerdings ist dies nur möglich, wenn die Schnittstellen des Test Codes zum Testfall bestehen bleiben.

Es ist auch möglich erst den Test Code zu schreiben und dann den eigentlichen Programmcode zu entwickeln. Diese Idee nennt man Test-Driven Development und ist in der agilen Softwareentwicklung sehr verbreitet. Damit ist es möglich Anforderungen und Spezifikationen auf

Plausibilität zu überprüfen. Zudem erhält der Entwickler noch schneller Feedback zu seinem programmierten Code.
Die Unit Tests sind ebenfalls Voraussetzung für die Integrationstest. Nur wenn diese implementiert sind kann getestet werden, ob die Funktionalitäten auch noch nach der Integration mehrerer Module funktionieren.
Werden Veränderungen in einem Codeabschnitt in das Versionsverwaltungstool geladen, so sollte von einem Entwickler das komplette Programm getestet werden, um sicherzustellen, dass alle Funktionalitäten den Anforderungen entsprechen. Das Versionsverwaltungstool übernimmt die Aufgabe des Verwaltens und der Versionierung der Codeabschnitte.
Wichtig dabei ist, dass das Einchecken in das Versionsverwaltungswerkzeug einen konsequenten und durchgehenden Prozess darstellt. Problematisch wird es, wenn der Entwickler seine Codeabschnitte nur lokal speichert und nicht ins Versionsverwaltungswerkzeug eincheckt. Dadurch kann es zu einem kompletten Chaos kommen, welches viel Zeit benötigt um die Versionen wieder auf den aktuellen Stand zu bringen.
Sobald die Software einen gewissen Entwicklungsgrad hat, sollte die Fachabteilung oder das Testteam das Programmstück mit Hilfe von Oberflächentests gemäß den Anforderungen überprüfen. Dieser Teil entspricht dem Black-Box-Testverfahren. Die Fachabteilung hat somit keinen Einblick in den Quellcode der Software, sondern überprüft ausschließlich über jeweilige Schnittstellen die funktionalen Eigenschaften. Die Entwickler sollten schnellst möglich Rückmeldung bekommen, um Fehler zu beheben und daraus folgende Änderungswünsche schnell umzusetzen.
Es gibt kein zentrales Dashboard, welches dem Management den aktuellen Projektstatus zeigt. Zudem ist es nicht ersichtlich wie viele Testfälle bereits getestet wurden.

Fazit

Beim manuellen Testen gibt es keinen einheitlichen, automatisierten Testprozess. Oftmals wird auch der Prozess zur Versionsverwaltung nicht beachtet. Viele Informationen müssen manuell von einer Person zur Anderen kommuniziert werden. Dadurch wird unnötig viel Zeit verschwendet und Ressourcen werden nicht richtig eingesetzt. Der Entwickler bekommt erst sehr verspätet Feedback zu seinem Ergebnis. Wegen dem komplizierten und nicht einheitlichen Testprozess wird der Entwickler nicht dazu ermutigt seinen Softwarecode zu integrieren und zu testen. Außerdem hat das Management keine Möglichkeit den aktuellen Projektstatus einzuschätzen, da es kein zentrales Dashboard gibt, welches Beispielsweise die bereits getesteten Testfälle anzeigt.

5.2. Testautomatisierung

Testautomatisiert im Zusammenhang der vorgestellten Testprozesse bedeutet, dass zwar die einzelnen Teststufen automatisiert getestet werden, jedoch noch kein automatisierter Testprozess vorhanden ist. Im Vergleich zum manuellen Testen werden also die Unit Tests, Integrationstests und der Systemtests von Testwerkzeugen unterstützt automatisiert getestet, jedoch ist die Kommunikation zwischen den einzelnen Werkzeugen noch nicht automatisiert. Zudem gibt es noch kein zentrales Dashboard, welches dem Management einen Überblick über das Projekt gibt. Obwohl Testautomatisierung sehr viele Vorteile bringt, ist es immer noch nicht in allen Unternehmen etabliert. Mit Testautomatisierung ist es möglich die Testabdeckung zu erhöhen, die Kosten zu reduzieren und den Testdurchlauf zu beschleunigen. Daneben verringert es auch die Fehler, die durch die manuelle Nutzung eines Testwerkzeugs durch den Menschen entstehen. [120] Dadurch kann nicht nur die Qualität der Software erhöht werden, sondern auch die Effektivität des Testens.

Auf dem Markt gibt es unzählig viele Anbieter für Testautomatisierungswerkzeuge. Allein im Bereich Open Source gibt es über 112 verschiedene Testautomatisierungswerkzeuge.[121] Beispiele hierfür sind Selenium, Dogtail, Expect, Frankenstein und noch viele andere.

Die Abbildung 8 zeigt wie ein Testprozess mit unterstützenden Testautomatisierungswerkzeugen aussehen kann.

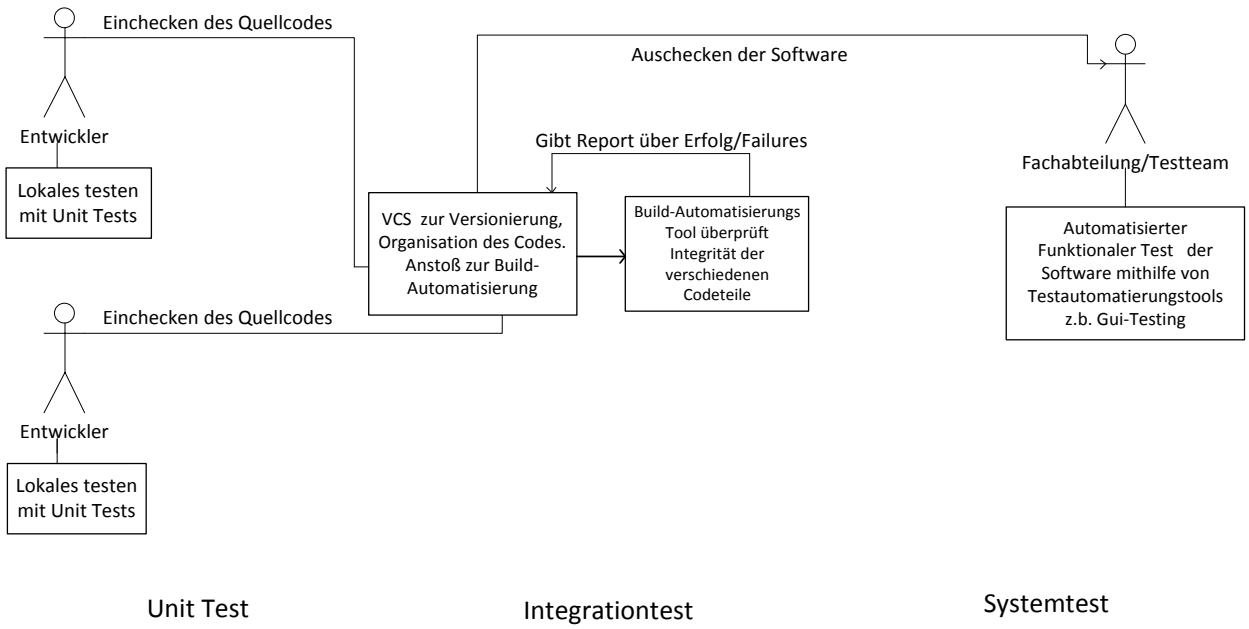


Abbildung 8: Testprozess in bei Testautomatisierung

[120] Vgl. Bommer, C. / Spindler, M. / Barr, V. (2008), S. 258
[121] Vgl. Opensourcetesting.org (2013)

Der Entwickler entwickelt den jeweiligen Quellcode lokal auf seinem Rechner beziehungsweise eingecheckt im Versionsverwaltungswerkzeug. Dieser Code wird ebenfalls zentral mit Hilfe von Unit Tests vom Entwickler getestet. Das VCS ist dafür zuständig die Codeteile zu versionieren, organisieren und das Build-Automatisierungstool wie zum Beispiel Maven oder Ant anzustoßen. Die Aufgabe des Build-Werkzeugs ist es die einzelnen Codebausteine zusammen zu bauen. Dadurch kann die Integrität der einzelnen Codeteile überprüft werden. Das Ziel ist es zu überprüfen, ob die Codeteile auch in Abhängigkeit von einander gemäß den Anforderungen entsprechen.
Bei der Testautomatisierung gibt noch keinen einheitlichen Testprozess. Die verschiedenen Werkzeuge können noch nicht miteinander kommunizieren. Dadurch gibt es noch keine Synchronisierung zwischen dem VCS beziehungsweise dem Build-Automatisierungstool und einem Systemtestwerkzeug gibt, muss dieses manuell vom Entwickler oder Tester angestoßen werden. Dazu wird die Software von dem VCS ausgecheckt und das jeweilige Testwerkzeug aktiviert. Die Ergebnisse müssen wieder manuell dem Entwickler mitgeteilt werden.

Fazit

Der Prozess zeigt, dass Testautomatisierung in den einzelnen Teststufen zwar bereits verwendet wird. Allerdings gibt es noch keine einheitliche, automatisierte Prozesskette die alle Testautomatisierungswerkzeugen integriert. Dadurch müssen die einzelnen Testschritte manuell vom Entwickler oder Tester angestoßen werden, was wiederum viel Zeit in Anspruch nimmt. Außerdem gibt es immer noch keinen zentralen Punkt, an dem der aktuellen Projektstatus eingesehen werden kann. Es ist nicht möglich zu erkennen, welche Funktionen bereits implementiert und erfolgreich getestet worden sind. Das Management kann dadurch bei Probleme keine geeigneten Maßnahmen einleiten oder das Entwicklerteam unterstützen.

5.3. Testen in einer Continuous Integration Lösung

Im Vergleich zu den anderen bereits vorgestellten Testprozessen ist der große Unterschied in einem CI-System, dass nicht nur die einzelnen Testschritte automatisiert getestet werden, sondern auch automatisiert angestoßen. Durch gibt es einen einheitlichen, strukturierten und vor allem automatisierten Testprozess mit einer Werkzeugs Suite. Abbildung 9 zeigt den Testprozess in solch einer Lösung.

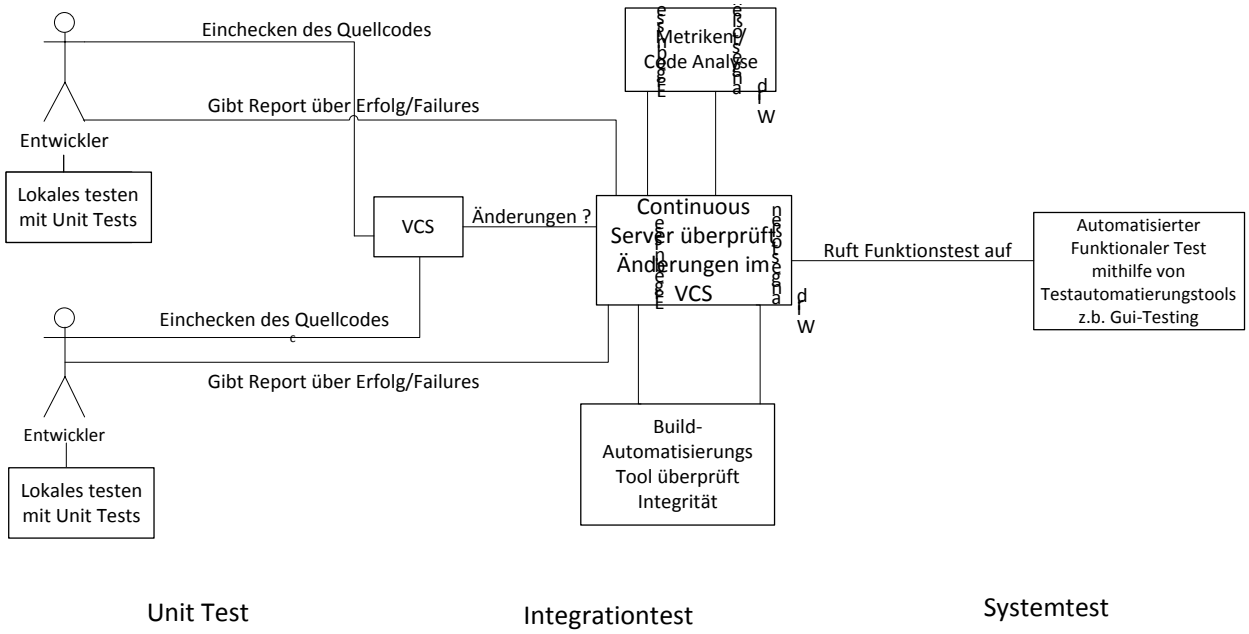


Abbildung 9: Testprozess in einer CI-Lösung

Damit ist der einzige Schritt der vom Entwickler manuell getätigt werden muss, das Einchecken des jeweiligen Programmcodes. Das Versionsverwaltungsprogramm übernimmt weiterhin alle bereits genannten Aufgaben wie die Organisation und Verwaltung der Codeabschnitte. Der Continuous Integration Server wie zum Beispiel Jenkins oder Hudson erkennt die Änderung im VCS und stößt daraufhin sogenannte Jobs an. Jobs sind hierbei Aufgaben die vom jeweiligen Werkzeug zu erfüllen sind. Es ist auch möglich VCS so zu konfigurieren, dass es selbst dem CI-Server anstößt sobald Änderungen erkannt wurden. Der Continuous Integration Server ist das Herzstück dieser Lösung. Dadurch wird eine zentrale Orchestrierung der einzelnen Werkzeuge erreicht. Die zentrale Orchestrierung ermöglicht wiederrum eine übergeordneter Struktur und ermöglicht dadurch einen einheitlichen Testprozess. Beispielsweise stößt er den Build Prozess an, um die Integrität der Codeteile zu überprüfen. Zudem kann optional eine Code Analyse oder auch Metriken implementiert werden, beispielsweise mit Sonar. Ein Systemstest, beispielsweise in Form eines Oberflächentests, kann ebenfalls vom CI-Server angestoßen werden. Die Ergebnisse werden an die CI-Server zurück geliefert und ermöglichen einen einheitlichen Testprozess. Die Ergebnisse können mit einem Dashboard übersichtlich dargestellt werden. Dadurch ist es dem Management möglich den aktuellen Projekt Status auf einem Blick zu erfassen und Trends abzuleiten. Entspricht der aktuelle Projektstatus nicht dem gewünschten Zielvorgaben können geeignete Maßnahmen eingesetzt werden. Je nach Konfiguration sendet der CI-Server automatisch Nachrichten in Form von Emails an den jeweiligen Entwickler, um ein schnelles und übersichtliches Feedback zu gewährleisten. Der klare Vorteil ist die Arbeitserleichterung des Entwicklers. Es müssen keine Absprachen mit anderen Entwicklern

stattfinden, das spart Zeit und Aufwand. Dadurch, dass der Aufwand sehr gering ist für den Entwickler, ist dieser eher dazu geneigt den Prozess zu akzeptieren und zu leben.

Fazit

Der große Unterschied des Testprozesses in einer CI-Lösung im Vergleich zur Testautomatisierung ist der einheitliche und automatisierte Testprozess. Somit werden alle Teststufen in diesen Prozess involviert. Unterstützt wird dieser Prozess durch die verschiedenen Komponenten. Besonders der CI-Server dient als zentrale Verwaltung der verschiedenen Werkzeuge. Die einzelnen Werkzeuge kommunizieren automatisch miteinander und es ist keine Kommunikation seitens der Entwickler oder Tester notwendig. Durch diesen einheitlichen und für den Entwickler einfachen Prozess hat der Entwickler einen sehr geringen Aufwand. Dadurch wird der Prozess vom Entwickler akzeptiert und vor allem gelebt. Durch den CI-Server kann der Entwickler ein schnelles Feedback zu seinem geschriebenen Softwarecode bekommen. Ein zentrales Dashboard ermöglicht dem Management einen Überblick über den Projektstatus zu bekommen. Es ist übersichtlich dargestellt welche Funktionalitäten bereits entwickelt und erfolgreich getestet worden.

5.4. Evaluation der verschiedenen Testprozesse

Bei dem Vergleich der verschiedenen Testprozesse ist klar zu erkennen, dass der Aufwand für den Entwickler beziehungsweise für den Tester deutlich abnimmt in einer CI-Umgebung. Die Kommunikation und der Informationsfluss erfolgt immer mehr zwischen den einzelnen Werkzeugen und ohne den Mensch als Schnittstelle zu benötigen. Die klaren Vorteile sind ein schnellerer, ein kostengünstigerer (da die Ressource Mensch weniger gebraucht wird) und ein fehlerfreierer Testprozess. Somit können die in Kapitel 3 angeführten Probleme die das Testen von Software so schwierig machen, größtenteils gelöst werden. Der Entwickler kann sich dadurch mehr auf seine eigentliche Arbeit - das Entwickeln und dem Testen - fokussieren.

Ebenfalls ist zu erkennen, dass jede Testautomatisierung bereits Vorteile für das Entwicklerteam und für das Unternehmen bringt. Es muss nicht gleich eine komplette CI-Lösung sein, um dem Unternehmen beziehungsweise dem Entwicklerteam einen Mehrwert zu schaffen. Jeder kleine hin zur Automatisierung dieser Prozesse bringt Vorteile für das Unternehmen. Die Vorteile sind beispielsweise Kostenreduktion, Qualitätsverbesserung und Entlastung des Entwicklers.

Das CI-System kann somit als Weiterentwicklung des Testautomatisierungsprozess gesehen werden. Bedingung ist hier, dass es zur jeder Zeit ein lauffähiges Produkt geben muss.

Durch den zentralen CI-Server in Verbindung mit den Metriken ist es möglich dem Management ein zentrales Dashboard zu geben. Das Management erhält damit eine Übersicht über den aktuellen Projektstatus und kann bei Fehlentwicklungen geeigneter Maßnahmen zur Ver-

fügung stellen. Die genauen Auswirkungen auf die Prozesse durch eine CI-Lösung ist im Kapitel 9.2 zu entnehmen.

6. Vorgehen bei Einführung von CI in ein Unternehmen

Im folgenden Kapitel soll die Einführung eines CI-Systems in ein Unternehmen theoretisch dargestellt werden. Die Einführung geschieht in mehreren eigenständigen Phasen, angelehnt an die Beschreibungen von Smart (2011). Das Unternehmen kann demnach auch nur bis zu einem bestimmen Grad den Weg der Continuous Integration gehen. Der Punkt an dem der Aufwand der Prozessumstellung den Nutzen durch CI übersteigt sollte hier als Ziel gewählt werden. Mit jeder neuen Phase kommt es auch zu einer Verbesserung der IT Infrastruktur. Von größerer Bedeutung sind allerdings die Umstellungen der Praktiken und Prozesse der Entwicklerteams.[122]

6.1. Phase Eins: kein CI Server

In der ersten Phase setzt das Unternehmen noch keinen zentralen CI-Server ein sondern führt den Build Prozess manuell aus. Ant Skripte werden beispielsweise auf der Maschine eines Entwicklers ausgeführt. Ein zentrales Versionsverwaltungssystem wird vom Unternehmen genutzt, allerdings pflegen die Entwickler ihre Änderungen nur unregelmäßig ein. Folglich ist der Code im VCS nicht immer aktuell. Kurz vor einem Release kommt es dann häufig zu dem gefürchteten „Big-Bang“, wenn alle Entwickler ihren entwickelten Teilcode gleichzeitig integrieren.

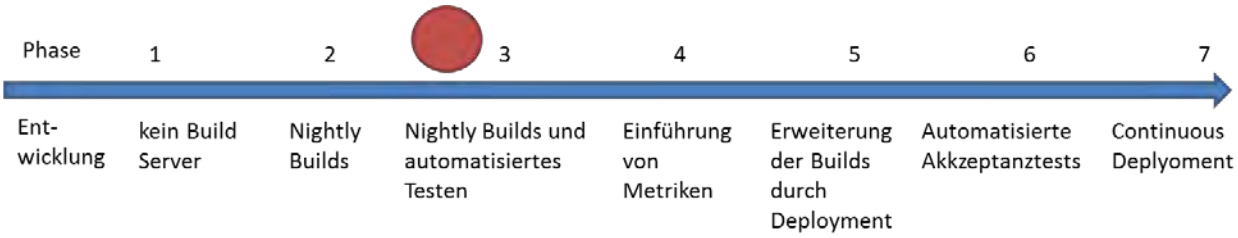


Abbildung 10: Phasenweise Entwicklung und Standpunkt des Auftrag gebenden Unternehmens

[122] Vgl.Smart, J.(2011), S. 5

6.2. Phase Zwei: Nightly Builds

In Phase Zwei wird ein CI-Server eingesetzt, der in regelmäßigen Abständen, automatisch Builds auslöst. In den meisten Fällen läuft der Build-Prozess über Nacht, sogenannte „Nightly Builds“. Bisher muss der Build-Prozess allerdings noch keine zuverlässigen Unit-Tests beinhalten. Die Entwickler fügen ihren Code nun regelmäßig in das VCS ein. Nach Einschätzung des Projektteams befindet sich das Auftrag gebende Unternehmen genau in dieser Phase bzw. in einer Phase die sich zwischen Phase 2 und Phase 3 befindet. In Abbildung 10 ist der Standpunkt des Auftrag gebenden Unternehmens eingezeichnet.

6.3. Phase Drei: Nightly Builds und Automatisiertes Testen

In der dritten Phase wird zum ersten Mal das Konzept von Continuous Integration tatsächlich angewandt. Der nächtliche Build Prozess beinhaltet nun neben dem Kompilieren auch automatische Unit Tests und Testfälle. Zur Orchestrierung der verschiedenen Werkzeuge wird ein Continuous Integration Server wie beispielsweise Jenkins eingesetzt. Der CI-Server ist so konfiguriert, dass er regelmäßig das Versionskontrollsystem auf Änderungen überprüft. Ist dies der Fall, so wird automatisch ein Build-Prozess angestoßen. Durch die Ergebnisse der automatisierten Unit Test und Testfälle lassen sich fehlgeschlagene Builds frühzeitig erkennen und reparieren. Die Entwickler können aufgrund des CI-Servers nachvollziehen, welche Änderung einen bestimmten Build-Prozess ausgelöst hat. Damit können Fehler schnell gefunden und eindeutiger zugeordnet werden[123]. Über Integrationsfehler während des Build-Prozesses informiert der CI-Server entweder via E-Mail, oder über andere Kommunikationsmethoden wie z.B. Twitter oder Instant Messenger.

6.4. Phase Vier: Einführung von Metriken

In Phase Vier werden Metriken zur Sicherung der Codequalität und der Codeabdeckung eingeführt. Mit den Metriken kann die Effektivität und Relevanz der Tests und des jeweiligen Codes sichergestellt und somit die Levels der Softwarequalität und der durchgeführten Testpraktiken hoch gehalten werden[124].

Um verlässliche Aussagen über den Zustand des Softwareprojekts und die Qualität des Programmcodes treffen zu können, muss der CI-Server Messgrößen wie den Umfang der Codebasis und die Stabilität des Entwicklungsprozesses ermitteln. Das können moderne CI-Server nur teilweise selbst, weswegen ein Entwickler hierfür normalerweise über Plugins auf weitere Tools zurückgreifen muss.

---

[123] Vgl.Smart, J.(2011), S. 5
[124] Vgl.Smart, J.(2011), S.5

Dadurch können verschiedene Softwareprojekte verglichen werden und der aktuelle Projektstatus besser erkannt werden. Potenzielle Risiken lassen sich als Trends erkennen und es kann entsprechend gegengesteuert werden.

6.5. Phase Fünf: Erweiterung der Builds durch Deployment

In der fünften Phase werden die Praktiken des Test-gesteuerten Entwicklungsprozesses weiter verbessert. Darunter versteht man, dass die Anwendung nicht nur kompiliert und getestet wird, sondern nach fehlerfreiem Test direkt auf einem Applikationsserver automatisch deployed wird. Dadurch sind einerseits End-To-End Tests der Anwendung als auch andererseits Performance Tests durchführbar.

6.6. Phase Sechs: Automatisierte Akkzeptanztests

In Phase Sechs werden letztendlich auch die Akzeptanztest, also Tests des ganzen Systems auf Anwendbarkeit und Systemfunktionalität, automatisiert. Hierfür wird die Anwendung vom Build Server automatisch in eine dafür vorgesehene Testumgebung „deployed“. Es wird sozusagen ein Prototyp inklusive aller Tests erstellt, um die Anforderungen des Kunden mit der entwickelten Software abzugleichen und bei negativen Abweichungen schnell gegen zu wirken zu können[125]. Dies geschieht entweder nach eingegangenen Änderungen im VCS oder anhand von Nightly-Builds.

6.7. Phase Sieben: Continuous Delivery

Mit diesen sechs Phasen ist das eigentliche Konzept der Continuous Integration vollständig abgedeckt. Allerding geht die Entwicklung auch bei CI weiter und es entstehen immer wieder neue Ideen und Verbesserungen im Bereich Softwareentwicklung. Ein Thema ist dabei z.B. das Konzept der Continuous Delivery welches die siebte und letzte Phase der Einführung von Continuous Integration ins Unternehmen darstellt. Aufgrund der Tatsache dass die Einführung dieser Phase für das Unternehmen sehr unwahrscheinlich ist soll hier nur genannt werden, dass bei diesem Ansatz eine Deployment Pipeline zum Einsatz kommt, die aus mehreren Stages besteht, die jeweils einem Build-Prozess entsprechen. Jede erfolgreich durchlaufene Stage erhöht die Gewissheit dass eine Änderung zu einer neuen funktionieren Version der Software führen wird.[126]

[125] Vgl.Ebenda, S.5
[126] Vgl. Feustel, B./ Schluff, S. (2012)

6.8. Zusammenspiel von Prozessen, Menschen und Werkzeugen

Der vorherige Abschnitt zeigt die Theorie der Einführung eines Continuous Integration Systems in ein Unternehmen. Dabei müssen neben der Einführung von verschiedenen Werkzeugen auch neue Prozesse eingeführt oder bestehende Prozesse geändert werden sowie die Entwickler und Tester mit den Neuerungen vertraut gemacht werden. Aus den drei Faktoren Werkzeuge, Menschen und Prozesse in einem CI-System, ergibt sich für die Autoren ein Dreieck welches in Abbildung 12 dargestellt ist.

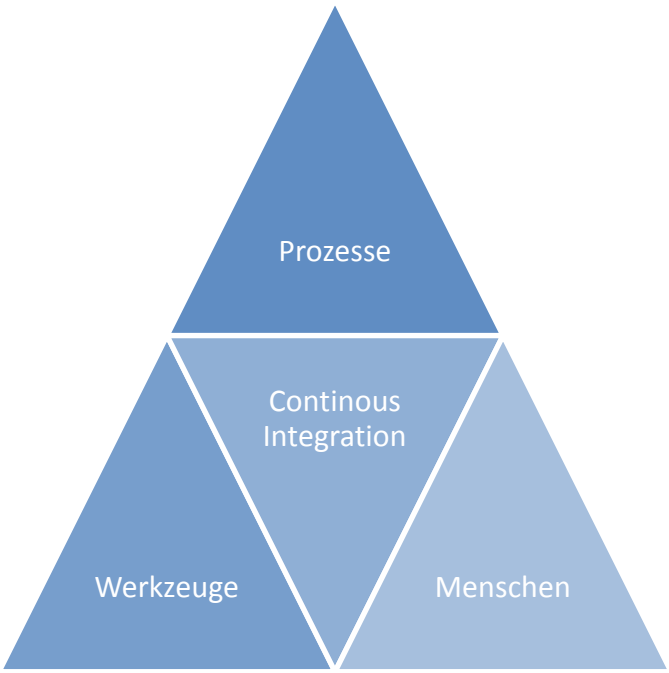


Abbildung 11: Das Dreieck von Continuous Integration

Es zeigt, dass alle drei Faktoren ineinander übergehen müssen. Dabei bildet die bestehende IT-Infrastruktur und die eingeführten CI-Werkzeuge den Ausganspunkt für neue Prozesse. Bevor also kein geeigneter Prozess für die gesamte Entwicklung inklusive aller seiner Konsequenzen konzipiert worden ist, wird das System nicht als solches funktionieren. Genauso ist es bei den Menschen, diese müssen den Ablauf und die Technik verstehen und akzeptieren. Die Vorteile für das Projekt und der Mehrwert für den einzelnen Entwickler müssen klar erkennbar sein. Prozesse und Werkzeuge erzeugen nur einen Mehrwert, wenn diese von den Menschen gelebt werden.

7. Konzept für eine Continuous Integration Infrastruktur

Im folgenden Kapitel werden die Erfahrungen und Schritte bei dem Aufbau einer CI-Umgebung beschreiben. Dazu haben die Autoren eine CI-Umgebung exemplarisch aufgebaut und das

Vorgehen in dem folgenden Kapitel dokumentiert. Insbesondere werden auf die verwendeten Werkzeuge und die Prozesse innerhalb der CI-Umgebung eingegangen. Dem zugrunde liegt der im Kapitel 1 erwähnte Prototyp, welcher an das Unternehmen übergeben wurde.

7.1. Aufbau einer Continuous Integration-Umgebung

Der Aufbau einer Continuous Integration-Umgebung umfasst eine Vielzahl unterschiedlicher Komponenten, die in der Softwareentwicklung genutzt werden. Der Prozess der in der CI-Umgebung abgebildet wird umfasst das Schreiben von Code, das Konfigurationsmanagement, das Buildmanagement sowie das Testen der Anwendungen und das Deployment. Das bringt es mit sich, dass aus jeden dieser Bereiche verschiedene Werkzeuge in die CI-Umgebung integriert sind.

Die im Folgenden vorgestellte Umgebung umfasst diese Komponenten, welche auch in der Abbildung 12 wiederzufinden sind:

- Apache Tomcat als Applikationsserver
- Subversion als Versionskontrollsystem
- Jenkins als Continuous Integration-Server
- Ant als Buildmanagement-Werkzeug
- JUnit als Unittest-Werkzeug
- Sonar und Selenium als Qualitätsmanagement- und Testwerkzeug
- Eclipse IDE mit entsprechenden Plugins
- Eine simple Java-Webapplikation als Demonstrationsapplikation

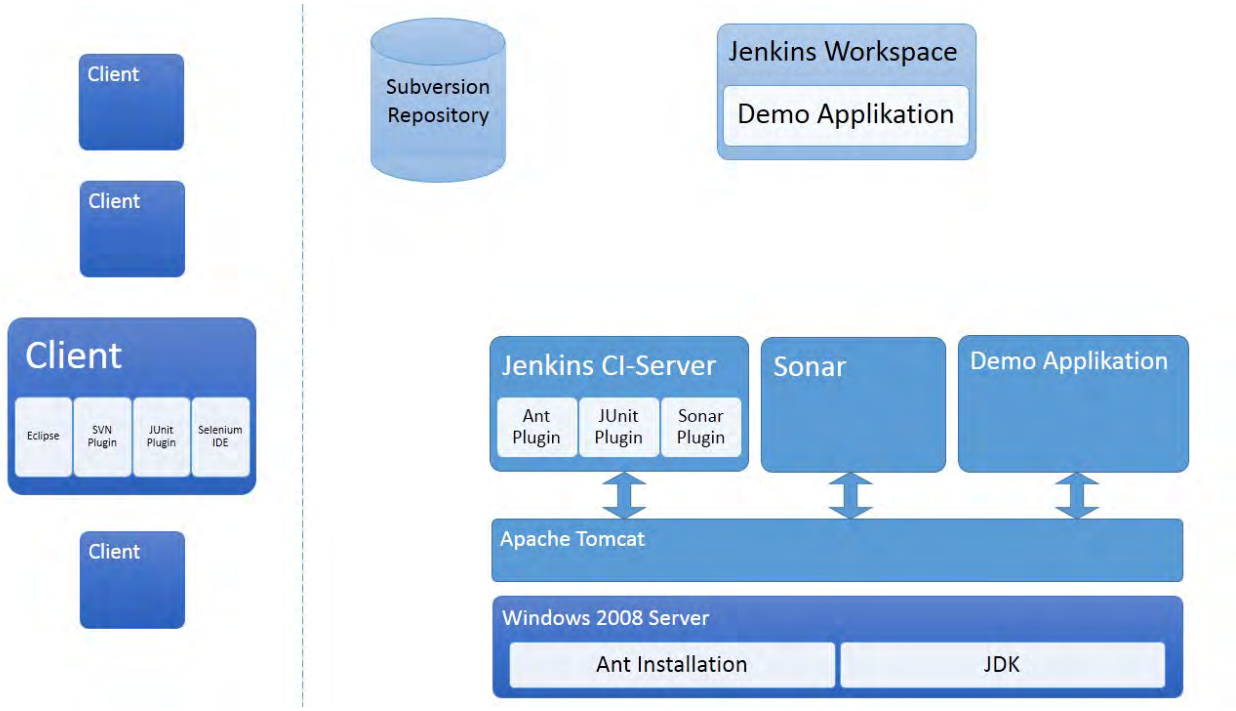


Abbildung 12: Komponenten in der CI-Umgebung

In der hier beschriebenen Umgebung gibt es einen *Server* und einen *Client*. Der Applikationsserver ist auf dem Server installiert und wird dort betrieben. Sowohl Jenkins als auch die Demonstrationsapplikation sind auf diesem Applikationsserver bereitgestellt. Näheres zu der Applikation wird unter 8.1.7. erläutert. Ebenfalls auf dem Server installiert ist Subversion, Ant und Sonar. Der Client ist ausgestattet mit einer IDE und greift auf das Subversion Repository sowie den Jenkins CI-Server zu.

7.1.1. Apache Tomcat als Applikationsserver

Apache Tomcat[127] als Applikationsserver übernimmt in der hier beschriebenen CI-Umgebung zwei Aufgaben. Zum einen dient der Applikationsserver als Middlewarekomponente zum Betreiben des Continuous Integration-Servers Jenkins. Außerdem wird die Demonstrationsapplikation auf dem Applikationsserver aufgespielt. In dem zugrundeliegenden Aufbau wird Sonar als Web Application Archive auf dem Applikationsserver bereitgestellt. Daneben ist es auch möglich Sonar als Standalone Anwendung zu betreiben.

Die Installation von Apache Tomcat ist einfach. Nach dem Download von der offiziellen Apache Website (www.apache.org) kann das heruntergeladene Archiv entpackt werden. Durch das Starten der Stratup.bat-Datei im Verzeichnis *bin* des Tomcats erfolgt der Start des Servers.

Das Deployment von, als .war-Dateien, gepackten Webapplikationen kann über drei Varianten durchgeführt werden. Zum einen kann über die Administrationsoberfläche im Browser ein Deployment per Knopfdruck durchgeführt werden. Darüber hinaus kann auf ein Administrationsinterface in der Kommandozeile zugegriffen werden. Dazu sollte ein Nutzer mit den entsprechenden Rechten angelegt sein. Zuletzt ist es auch möglich durch das Ablegen von .war-Dateien im Verzeichnis *webapps* ein Deployment durchzuführen. Daraufhin startet eine automatisches Deployment, welches die alte Version undeployed und die neu hinzugefügt Applikation deployed. Dies passiert ohne weiteren Benutzereingriff und darüber hinaus im laufenden Betrieb. Die zuletzt beschriebene Variante wurde aufgrund der Einfachheit in der hier beschriebenen CI-Umgebung angewendet.

Es ist darauf zu achten, dass der Applikationsserver bzw. die darauf betriebenen Applikationen über eine öffentliche URL zu erreichen ist. Dies gilt insbesondere für die Clients, die mit der CI-Umgebung arbeiten sollen. Das (Firmen-)Netzwerk und die Positionierung des Applikationsservers darin sind dementsprechend zu gestalten.

---

[127] http://tomcat.apache.org

Austauschbarkeit des Applikationsserver

Grundsätzlich ist es stellt es kein Problem dar, den Apache Tomcat Applikationsserver durch andere Produkte auszutauschen. Sowohl Jenkins als auch Sonar werden als Web Application Archive auf dem Applikationsserver bereitgestellt und können auch mit anderen Applikationsservern verwendet werden. Außerdem speichert Jenkins alle Daten in einem eigenen Workspace im Benutzerverzeichnis des Servers. Nach dem Austauschen des Applikationsservers greift Jenkins wieder auf dieses Archiv zu. So gehen auch Daten wie alte Builds, Testergebnisse oder Logdateien nicht verloren.

Bei einem Austausch muss allerdings der Deployment-Prozess angepasst werden. Dieser ist in der hier beschriebenen CI-Umgebung über das Ant Buildskript realisiert. Dieses Skript muss dem zum Austausch gewählten Produkt entsprechend angepasst werden. Auch hier besticht Jenkins durch eine große Auswahl an Plugins, welche das Deployment auf den gängigsten Applikationsservern wie Glassfish, JBoss oder IBM WebSphere unterstützen. So kann zum Beispiel bei einem Deployment auf dem IBM WebSphere ein Plugin (wsadmin-Plugin) genutzt werden, um die erstellten .war-Dateien über wenige Kommandozeilenbefehle zu deployen.

7.1.2. Subversion als Versionsverwaltungswerkzeug

Subversion[128] kann als Open Source Werkzeug kostenfrei von der Apache Website heruntergeladen werden. Das Initiieren eines neuen Repositorys kann sowohl über die Kommandozeile als auch eine Vielzahl von frei verfügbaren, grafischen Administrationsapplikationen erfolgen. In der hier beschriebenen Umgebung wurde das Repository auf dem Server initiiert. Dies entspricht der Betriebsweise bei dem Unternehmen und wurde nachgebildet. Dazu wurde das grafische Werkzeug VisualSVN benutzt, welches nach der Eingabe einer Bezeichnung das Repository samt der gängigen Verzeichnisstruktur anlegt. Nach dem Anlegen des Repository wird die Demonstrationsapplikation in das *trunk* Verzeichnis importiert. Außerdem sollten je nach Nutzungsart entsprechend viele Benutzer und deren Zugriffsrechte definiert werden.

Auch hier gilt, dass das Repository über eine URL öffentlich erreichbar sein sollte. Wobei sich hier der Ausdruck öffentlich auf die entsprechenden Bereiche im (Firmen-)Netzwerk bezieht.

Austauschbarkeit des Versionsverwaltungswerkzeugs

Soll das Versionsverwaltungswerkzeugs durch ein anderes ersetzt werden, sind zwei Aspekte zu bedenken**.** Dies ist zum einen die Konfiguration des CI-Servers und zum anderen die bereits vorhandenen Strukturen und Arbeitsweisen.

---

[128] http://subversion.apache.org/

Als Schnittstelle zwischen VCS und CI-Server dient die öffentliche URL, wobei die Kommunikation mit dem VCS über das HTTPS Protokoll getätigt wird. Die Konfiguration des CI-Servers sieht vor, dass die öffentliche URL des Repositorys und die entsprechenden Anmeldedaten angegeben werden. Diese müssten bei einem Austausch geändert werden. Das erfordert einen sehr geringen Aufwand. Hinzu kommt, dass bei dieser Konfiguration nicht von Belangen ist, ob es sich um ein zentrales oder dezentrales Versionsverwaltungswerkzeugs (vergleiche Kapitel 2.1.1.) handelt.

Viel ausschlaggebender ist das bereits im Unternehmen verwendete Versionsverwaltungswerkzeug und die damit zusammenhängenden Arbeitsweisen. So kann unter Umständen ein Austausch zur Folge haben, dass die IDEs der Entwickler angepasst und die Strukturen des Entwicklungsprojekte geändert werden müssen. Dies bedeutete bei großer Anzahl von Entwicklungsprojekten einen entsprechend großen Aufwand. Hier müssen also Nutzen und Aufwand gegeneinander abgewogen werden.

7.1.3. Jenkins als Continuous Integration-Server

Der Countinuous Integration-Server ist der Mittelpunkt der CI-Umgebung. Er stellt die Schnittstelle zwischen den einzelnen Komponenten dar und ist für die Automatisierung und Ausführung der einzelnen Schritte zuständig. Jenkins löst den Build, die Tests und deren Auswertung aus.

Die Installation ist denkbar einfach. Nach dem Herunterladen von der offiziellen Website[129] kann Jenkins als Stand Alone Applikation über eine .bat-Datei gestartet werden. Außerdem ist es möglich eine .war-Datei herunterzuladen und diese auf dem Applikationsserver bereitzustellen. Letzteres wurde in der zugrundeliegenden Umgebung getan. Auch hier ist darauf zu achten, dass Jenkins über eine URL *öffentlich* erreichbar ist.

Jenkins kann mit zahlreichen Plugins ausgestattet werden, um die Integration der einzelnen Komponenten der CI-Umgebung zu erleichtern. Diese Plugins dienen sowohl der Integration der Komponenten in Jenkins selbst als auch in den in Jenkins angelegten Jobs. Die Jenkins Jobs dienen der Verwaltung und Strukturierung von Projekten. So kann zum Beispiel pro Applikation ein Job angelegt werden. Ebenfalls ist es denkbar einzelne Schritte wie den Build oder das Testen in Jenkins Jobs zu strukturieren. In der hier beschriebenen CI-Umgebung wurde ein Jenkins Job für die Verwaltung der Demonstrationsapplikation angelegt.

[129] www.jenkins-ci.org

7.1.3.1. Konfiguration von Jenkins

Über die Plugin-Verwaltung von Jenkins wurden zunächst alle benötigten Plugins installiert. Zur Einbindung des Subversion Repositorys wird das SVN Plugin benötigt. Außerdem werden für die Verwendung Ant und Sonar ein Plugin installiert. Die verwendeten Plugins sind der folgenden Liste zu entnehmen:

- *Ant*
- *Jenkins Sonar Plugin*
- *Jenkins Subversion Plugin*

Nach der Installation der Plugins können diese in der Jenkins Konfiguration angepasst werden. Für Ant, Sonar und Sonar Runner wird eine Installationsmethode angegeben. Jenkins kann die Installation über einen Download von der offiziellen Website, eine zur Verfügung gestelltes Archiv oder einen Kommandozeilenbefehl durchführen. Nach der Konfiguration der Installationsvariante installiert Jenkins die Werkzeuge innerhalb der nächsten Minuten.

7.1.3.2. Konfiguration des Jenkins Jobs

Der, in der hier betrachteten CI-Umgebung, angelegte Jenkins Job umfasst eine Vielzahl von Teilschritten. Der Jenkins Job fragt im Pull-Modus im 5-minütlichen Takt das Repository nach Veränderungen ab. Das heißt, dass alle 5 Minuten das VCS nach Updates und Commits durchsucht wird. Wird eine Veränderung erkannt, so löst Jenkins einen Build und Tests aus. Anschließend werden die Auswertung der Testergebnisse sowie eine Codeanalyse durch Sonar angestoßen. Die Zeitspanne sollte natürlich auf die Situation und den Umfang des Build-Prozesses im Unternehmen angepasst werden.

In der Konfiguration des Jobs wird Subversion als Versionsverwaltungswerkzeug mit dessen URL und einem Benutzer hinterlegt. Das Pull-Verhalten regelt inwiefern und wann Builds ausgelöst werden. Außerdem wird der Build-Prozess konfiguriert. Dazu wird Ant als Build-Werkzeug festgelegt, die Pfad zu dem Buildscript sowie die zu involvierenden targets angegeben. Zusätzlich wird der Sonar Runner benötigt. Dieser führt die eigentliche Analyse des Codes durch und sendet die Ergebnisse der Analyse an Sonar. Als Post-Build-Schritt wird die Involvierung des Sonar Runners konfiguriert. Dazu wird die Properties-Datei des Sonar Runners in der Verzeichnisstruktur der Demonstrationsapplikation angegeben. Mit diesen Properties wird die Analyse des Sonar Runners von Jenkins ausgelöst. Zusätzlich wird der Pfad, unter dem die JUnit Testergebnisse abgelegt werden, angegeben.

Austauschbarkeit des Continuous Integration Servers

Das Austauschen des CI-Servers würde einen sehr großen Eingriff in die CI-Umgebung darstellen. Der CI-Server ist der Mittelpunkt des Systems und gewährleistet die Integration die in die Umgebung eingebundenen Werkzeuge. Bei einem Austausch müssten alle getätigten Konfigurationen wiederholt werden. Diese Konfigurationen stellen den größten Aufwand bei der Einrichtung eines CI-Systems dar. Außerdem ginge bei einem Austausch die Build-Historie und alle damit zusammenhängenden Daten verloren.

Außerdem ist hier nochmal auf die Marktanalyse im Kapitel 2.1. zu verweisen. Die Alternativen zu Jenkins sind Hudson, Continuum und Cruise Control. Dabei bietet Hudson einen ähnlichen Funktionsumfang, aber eine kleinere Community als Jenkins. Continuum und Cruise Control entsprechen nicht den Anforderungen des Unternehmens. Daher ist von einem Austausch abzuraten.

7.1.4. Ant als Buildmanagement-Werkzeug

Apache Ant wird als Werkzeug zum Build der Demonstrationsapplikation eingesetzt. In der CI-Umgebung muss beachtet werden, dass Ant auf dem Server ausgeführt wird. Unter anderen Targets enthält das Buildscript in der hier beschriebenen Umgebung ein Target für das Deployment der gebauten Applikation. Im Rahmen der lokalen Entwicklung in der IDE des Clients wird dieses Taget oft nicht genutzt oder ist im Falle der Nutzung an die lokale Umgebung des Entwicklers angepasst. Daher muss das Buildscript so angepasst werden, dass es in der Umgebung des Servers ausgeführt wird. Es empfiehlt sich besonders bei Pfaden mit Umgebungsvariablen des Systems zu arbeiten. Hart programmierte und absolute Pfade können hier eine Fehlerquelle darstellen.

Wie unter 1.3.2. beschrieben, müssen die auszuführenden Targets in der Konfiguration des entsprechenden Jenkins Jobs hinterlegt sein. So ist in der betrachteten CI-Umgebung eine build.xml Datei im Wurzelverzeichnis der Demonstrationsapplikation hinterlegt. Eines der Targets ist so definiert, dass es einen Säuberung des Build-Verzeichnisses, eine Kompilierung, das Unit-Testen der kompilierten Klassen sowie das Deployment auf dem Applikationsserver durchführt. Dieses Target ist in dem Jenkins Job definiert.

Die vollständige build.xml Datei kann im Anhang 1 näher betrachtet werden.

7.1.5. Sonar und Selenium als Qualitätsmanagement- und Testwerkzeuge

Sonar als Qualitätsmanagementwerkzeug

Als Qualitätsmanagement-Werkzeug wird in der beschriebenen CI-Umgebung Sonar eingesetzt. Sonar steht auf der offiziellen Website zum Download bereit. Nach dem Download kann Sonar als Standalone Applikation - eingebunden in eine Java Virtual Machine als Wrapper - gestartet werden. Außerdem ist es möglich über eine mitgelieferte Batchdatei eine .war-Datei zu erstellen. Diese wurde in dem hier beschriebenen Beispiel auf dem Applikationsserver bereitgestellt und ist so über eine öffentliche URL erreichbar. Sonar kann mit einer mitgelieferten Derby-Datenbank in der eigenen Laufzeit oder mit einer extern, über JDBC angebundene, Datenbank betrieben werden. Hier wurde die Derby-Datenbank gewählt, da diese während der Laufzeit erzeugt wird und keine weitere Konfiguration benötigt. Es handelt sich dabei um die Java interne Datenbank, welche von der Java JDK mitgeliefert wird. Dabei ist darauf zu achten, dass der Java-Laufzeitumgebung des Applikationsservers genügend Speicherplatz zur Verfügung gestellt wird.

Um die Integration mit Jenkins zu gewährleisten, ist in Jenkins das Sonar-Plugin zu installieren. Das Plugin eröffnet die Möglichkeit Sonar selbst und den Sonar Runner von Jenkins installieren zu lassen. In der beschriebenen CI-Umgebung wird Sonar auf dem Applikationsserver bereitgestellt. Daher muss nur noch eine Sonar Runner Installation angelegt werden. Auch hier kann wieder zwischen einer automatischen Installation von der offiziellen Website, aus einem heruntergeladenen oder aus einem lokal gespeicherten Archiv gewählt werden. Hier sollte darauf geachtet werden, dass ein Archiv eine stabile Version mit sich bringt, während das Herunterladen sicherstellt, dass stets die aktuellste Version verwendet wird.

Der Sonar Runner übernimmt die Aufgabe der Analyse des Codes. So wird als ein Schritt des Buildprozesses in dem Jenkins Job eine Standalone Sonar Analyse konfiguriert. Das heißt, dass nach dem eigentlichen Build der Sonar Runner ausgeführt wird. Der Sonar Runner parst alle Dateien im Jenkins Workspace des Projektes und übermittelt die Ergebnisse an Sonar. In der Sonar Webanwendung werden diese Ergebnisse aufbereitet und dargestellt. Wichtig ist hierbei, dass die Struktur in der „Sonar-Runner.Properties“ Datei definiert ist. Anhand dieser Datei wird die Analyse des Sonar Runners gesteuert. So wird dort unteranderem der Pfad zu den Quelldateien, den Testklassen oder externen Archiven hinterlegt. Außerdem enthält diese Datei die URL der Sonar Webanwendung, um diese dem Sonar Runner bekannt zu machen. Die „Sonar-Runner.Properties“-Datei kann im Anhang 2 eingesehen werden.

Selenium als automatisiertes Oberflächentestwerkzeug

Selenium bietet die Möglichkeit die Bedienung von Weböberflachen zu automatisieren und daraus Testskripte zu entwickeln. Diese Testskripte können für das Testen von Oberflächen verwendet werden. Selenium greift dabei auf die WebDriver zurück, welche im Wesentlichen eine API zur programmatischen Einbindung von bestimmten Browser-Betriebssystem-Kombinationen darstellt. Über diese API können Bedienelemente angewählt, Texteingaben vorgenommen und Klicks getätigt werden. Die Skripte können manuell geschrieben werden oder durch ein Werkzeug erzeugt werden. Im hier beschriebenen Beispiel wurde letzter Variante gewählt, da sie keine Kenntnisse von der, von Selenium verwendeten Skriptsprache, nötig sind. Über das Selenium IDE-Plugin für Firefox können diese Skripte über eine Aufnahmefunktion während des Bedienens der Weboberfläche aufgenommen werden. Anschließend können diese Skripte als Javaklasse exportiert werden.

Die so erzeugten Javaklassen können als JUnit-Tests in den Build-Prozess eingebunden werden. Dabei können die benötigten Bibliotheken von der offiziellen Selenium Website heruntergeladen und in den Buildpath des entsprechenden Softwareprojekts aufgenommen werden. Darauf aufbauend wird nach dem automatisierten Deployment der Demonstrationsapplikation ein weiterer Build-Schritt gestartet. Dieser Schritt führt den Oberflächentest aus und gibt die Testergebnisse weiter an die Auswertungsmechanismen von Jenkins und Sonar.

Austauschbarkeit von Qualitätsmanagement- und Testwerkzeuge

Die Einbindung von Codeanalyse und Oberflächentests gehören nicht zu den expliziten Anforderungen des Unternehmens. Nichtdestotrotz haben die Autoren sich entschlossen, diese beiden Aspekte mit aufzuzeigen. Daher handelt es sich bei deren Einbindung nur um eine Empfehlung, um das Konzept von Continuous Integration über das Builden, Doplyment und Unittesten hinaus zu ergänzen. Es ist jedoch auch an dieser Stelle denkbar andere Oberflächentest- und Analysewerkzeuge einzusetzen. Aufgrund der Anforderungen an die Arbeit wurden von den Autoren keine anderen Werkzeuge analysiert, sodass hier keine Empfehlung zur Austauschbarkeit gegeben werden soll.

7.1.6. Eclipse IDE auf dem Client

Als Entwicklungsumgebung ist auf dem Client die Eclipse IDE installiert. Hier können je nach Präferenz auch andere Entwicklungsumgebungen verwendet werden. Zu beachten ist, dass eine Integration von Subversion und JUnit gewährleistet ist. In Eclipse ist diese Integration durch Plugins geboten. In der CI-Umgebung wird ein Eclipse mit dem Eclipse Subversive ausgestattet, um die Synchronisierung mit dem Repository auf dem Server aus der IDE heraus ausführen zu können. Abhängig von der verwendeten Eclipse Version muss eventuell noch ein

JUnit Plugin installiert werden. Bei allen neueren Eclipse Derivaten ist diese schon vorimplementiert.

Mit Hilfe des Subversive Plugins kann das Repository ausgecheckt werden, um so das Softwareprojekt in den lokalen Arbeitsbereich zu laden. In der beschriebenen CI-Umgebung wird die Demonstrationsapplikation als Java-Projekt in den lokalen Arbeitsbereich geladen, um dort die Entwicklung vorzunehmen. Vom Entwickler getätigte Änderungen werden nach dem Einchecken von Jenkins erkannt und der Jenkins Job wird automatisiert gestartet.

Austauschbarkeit der Entwicklungsumgebung

Die Entwicklungsumgebung ist prinzipiell austauschbar. Bei den verwendeten IDE sollte eine Unterstützung von JUinit und Suberversion gewährleistet sein. Dies kann sowohl über Plugins als auch über andere mögliche Integrationen erfolgen. Bei gängigen IDEs wie Netbeans, Visual Studio oder Eclipse Derivaten (IBM RAD/WID) ist dies gewährleistet. So kann die IDE je nach Präferenz des Entwicklers ausgetauscht werden.

7.1.7. Java Webapplikation zur Demonstration

Grundsätzlich können in einer CI-Umgebung verschiedenste Anwendungen eingebunden werden. Dabei gibt es keine Einschränkungen auf gewisse Plattformen oder Programmiersprachen. Die Community um Jenkins als CI-Server hält Plugins für die gängigsten Technologien wie zum Beispiel Maven2/3 oder TestNG bereit.

Bei der Demonstrationsapplikation handelt es sich um eine einfache Java Webanwendung. Sie besteht aus einer Java-Engine, die aufgrund von Benutzereingaben eine Grußformel generiert. Die Darstellung erfolgt mit Hilfe von Java Server Pages. Außerdem sind Unittests für die Java-Engine angelegt.

7.1.8. Umgebungsvariablen des Systems

Der Jenkins Server führt während der Laufzeit verschiedene Tasks mit Hilfe der Kommandozeile des unterliegenden Betriebssystems aus. So wird beispielsweise der hier beschriebene Ant-Script via Kommandozeilenbefehl ausgeführt. Hinzukommt die Einbindung von Sonar Runner über die Kommandozeile. So muss sichergestellt werden, dass alle eingebundenen Werkzeuge im Betriebssystem registriert sind. Hierzu sollten insbesondere die Umgebungsvariablen des Systems gesetzt sein. In der folgenden Tabelle sind einige exemplarische Umgebungsvariablen aufgeführt. Abhängig vom Aufbau der individuellen CI-Umgebung muss diese Liste angepasst werden:

Tabelle 8: Umgebungsvariablen des Systems

Werkzeug	Umgebungsvariable	Pfad
Java	JAVA_HOME	*C:\Pfad\zu\JDK*
Ant	ANT_HOME	*C:\Pfad\zu\Ant*
Sonar Runner	SONAR_RUNNER_HOME	*C:\Pfad\zu\SonnarRunner*

7.1.9. Freigaben der Ports

Wie unter mehreren Punkten oben erwähnt, sollten einige der Werkzeuge über eine öffentlich URL erreichbar sein. Um dies zu gewährleisten müssen die entsprechenden Ports auf den Server für eingehende Verbindungen freigeschaltet sein. Auch hier sind die einzelnen Ports abhängig von den, in die CI-Umgebung integrierten, Werkzeugen und der individuellen Gestaltung des unterliegenden Netzwerks. Im Folgenden eine List exemplarischer Portfreigaben. Auch hier besteht kein Gewähr auf Vollständigkeit.

Tabelle 9: Portfreigaben

Applikation	Port	Service
Jenkins, Subversion, Sonar	443	HTTPS
Fernwartung	3389	RDP
Websphere AS	8080	HTTP
Sonar Runner	9000	HTTP

7.1.10. Aufbau des Jenkins Workspace

Der Jenkins Workspace ist das Verzeichnis, in dem alle Daten von Jenkins abgelegt sind. Dieser befindet sich meist in dem Benutzerverzeichnis des Servers. Besonders hervorzuheben sind hier drei Unterverzeichnisse:

1. Plugins
- Hier sind alle installierten Plugins und dessen Daten abgelegt. Diese sind Beispielsweise Ant, CVS, verschiedene Deployer, Subversion oder Backup-Plugins
2. Tools

- Hier sind alle installierten Werkzeuge und dessen Daten abgelegt. Dies sind Beispielsweise: Sonar, Sonar Runner, JDK

3. Jobs

- Hier sind alle individuell angelegten Jobs abgelegt. Die jeweiligen Jobs wiederum enthalten ein Build- und ein Workspace-Unterverzeichnis.
- Im Build-Unterverzeichnis werden Information zu allen Build gespeichert. Dies sind Testergebnisse, Änderungen an der Softwarekonfiguration sowie Log-Dateien
- Im Workspace-Unterverzeichnis befindet sich eine lokale Kopie des Subversion Repositorys. Diese lokale Kopie wird für die Build-, Test- und Deployment-Vorgänge genutzt.

7.2. Prozesse in der Umgebung

In folgender Abbildung ist die unter 8.1. beschriebene CI-Umgebung dargestellt. Es werden die einzelnen Komponenten und Werkzeuge gezeigt. Außerdem sind die Prozesse, welche zwischen den Komponenten ablaufen, aufgezeigt.

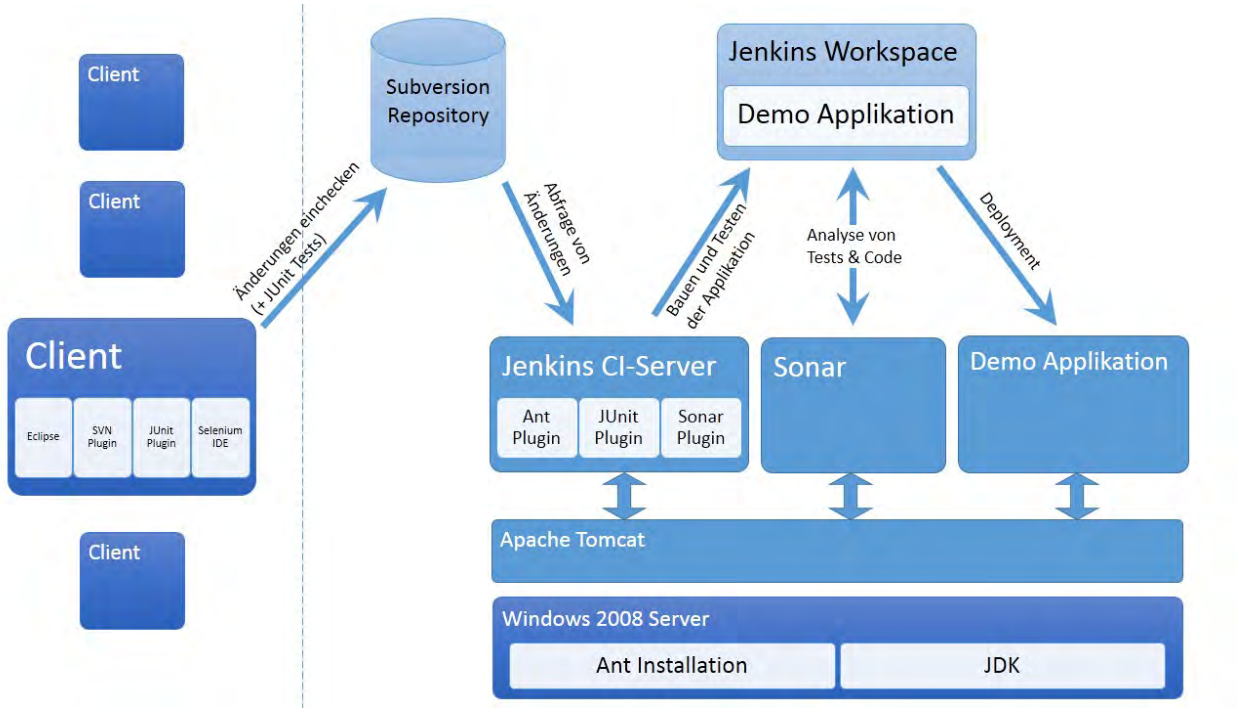


Abbildung 13: Komponenten und Prozesse in der CI-Umgebung

Wie oben zu erkennen ist, werden fünf wichtige Prozesse innerhalb der CI-Umgebung ausgeführt:

1. Einchecken von Änderungen an der Softwarekonfiguration
2. Abfragen von Änderungen am Repository
3. Bauen und Testen der Applikation
4. Bereitstellung der gebauten Applikation

7.2.1. Einchecken von Änderungen an der Softwarekonfiguration

In diesem Schritt werden Änderungen an dem Quellcode und in den einzelnen Softwarebestandteilen nach vollendeter Arbeit abgespeichert und muss anschließend mit dem Versionsverwaltungssystem synchronisiert werden. Dabei werden die Änderungen entweder “commited“, sprich im SVN Repository veröffentlicht, oder “updated“, sprich aus dem SVN Repository übernommen. Das Repository kann entweder lokal auf einem Server im Intranet bestehen oder auch im Internet in einer Cloud Umgebung. Dieser Schritt ist verständlicher Weise absolut notwendig um Versionierung zu betreiben, um Änderungen am Code nach zu vollziehen und zu verwalten. Ganz besonders in einer Entwicklungsumgebung mit vielen Entwicklern an unterschiedlichen Standorten ist solch ein Versionierungs-System essentiell.

7.2.2. Abfragen von Änderungen am Repository

Die Abfrage von Änderungen im Repository ist in zweierlei Sicht sehr wichtig.

Einerseits wird bei der Versionierung in einem Repository immer wieder Codechanges einzelner Entwickler veröffentlicht und müssen natürlich von anderen beteiligten Entwicklern übernommen werden um den aktuellen Stand der Softwarestruktur bei zu behalten und effektive entwickeln zu können.

Zweitens ist die Abfrage des Repositories wichtig um das Continuous Integration Tool, Jenkins, die Lokation und die Bestandteile der Software bereit zu stellen um den Build – Prozess durchlaufen zu lassen. Mit anderen Worten braucht das CI-System, welches auf einem Server läuft, eine Verlinkung zum Repository um aus den notwendigen Bestandteilen die Software zu bauen. Weiterhin können im Repository auch Test-Dateien liegen um die fertig kompilierten Klassen zu testen. Ebenso können auch bestimmte Konfigurationsdateien für Monitoringtools, wie Sonar, oder buildprozessspezifische Anweisungen, wie die build.xml, im Repository liegen. All diese Bestandteile und Konfigurationsdateien werden unbedingt benötigt um den Continuous Integration Prozess durchlaufen zu können. Daher ist das regelmäßige oder festgelegte Abfragen des Repositories im “Continuous“ Prinzip unabdingbar und kann in 4 Varianten seitens des CI-Systems geschehen:

- Manuelles Ausführen
- Fortlaufendes Ausführen wenn ein Build-Prozess beendet ist
- Zeitgesteuerte Ausführung des Build-Prozess
- Zeitgesteuerte Abfrage des SVN Repositories auf Veränderung mit ausgelösten Build-Prozess

7.2.3. Bauen und Testen der Applikation

Wie in der vorherigen Sektion beschrieben holt sich das Continuous Integration System alle benötigten Bestandteile der zu bauenden Software sowie die “Anleitung“ zum Bauen verfasst in einer build.xml Ant Script Datei. Dieses Ant Script beschreibt per Targets Schritt für Schritt den Vorgang des “Bauens“ der Software und gibt als Endprodukt eine *.war Datei die, wie in der folgenden Sektion beschrieben, dann genutzt wird um die Web-Anwendung auf einem Application-Server zu veröffentlichen. Nach dem Bauen der Web-Anwendung wird ein Test durchgeführt. Dieser Test wird mit Test-Klassen durchgeführt die speziell für die Web-Anwendung entworfen wurden und somit auch im Repository zur Verfügung stehen müssen. Die Tests werden dann durch JUnit am Back- sowie auch Frontend (Weboberfläche) durchgeführt. Die Tests werden dann zu einem Report zusammengefasst und stehen zu Auswertung bereit. Diese Auswertung kann dann entweder durch den Entwickler direkt passieren, durch Jenkins als grafische Ausgabe des Reports oder aber durch Sonar. Diese ganzen Prozessschritte sind im Build-Prozess vereint und definiert.

7.2.4. Bereitstellung der gebauten Applikation

Unter Bereitstellung der Web-Anwendung versteht man in einem Continuous Integration System das Deployment oder auch das Veröffentlichen. Im Klartext bedeutet das, das die erzeugte *.war Datei auf einem Applikations Server, in diesem Fall Apache Tomcat, veröffentlicht wird um Sie dann im Intranet oder Internet zugänglich zu machen. Im Falle Tomcat ist dieser Deployment Prozess sehr einfach gestaltet, da die *.war Datei lediglich in einem “webapp“ Ordner im Tomcat Verzeichnis hinterlegt werden muss und der Application Server diese Web-Anwendung automatisch veröffentlicht und zugänglich macht. Im Falle des Projektes wird die Web-Anwendung wie in den letzten 3 Sektionen beschrieben deployed. Das heißt beim Entwickler entwurfen und geschrieben, dann eingecheckt im Repository, durch die Continuous Integration Software Jenkins abgerufen mit allen zugehörigen Bestandteilen und Konfigurationdateien anhand der Vorgabe der build.xml Ant Script Datei gebaut. Anschließend mit jUnit komplett getestet und per Sonar ausgewertet und schlussendlich final automatisiert auf dem Tomcat Application Server veröffentlicht und zugänglich gemacht.

8. Evaluierung

Im den folgenden zwei Abschnitten soll - gestützt auf die vorhergegangenen Erläuterungen einerseits und die im Aufbau der Continuous Integration-Umgebung erworbenen Erfahrungen und Kenntnisse – eine Evaluierung des Konzeptes Continuous Integration erfolgen. Dazu wird der Einfluss von CI auf das Softwareprodukt selbst als auch auf die Entwicklungsprozesse zusammengefasst und bewertet. Vorab sei zu erwähnen, dass die Einrichtung einer CI-Umgebung einen großen Aufwand bedeutet. Dieser Aufwand umfasst vor allem viele Stunden Arbeitszeit für die Konfiguration der integrierten Komponenten. Nichtdestotrotz bringt die Einführung von Continuous Integration entscheidende Mehrwerte für das Unternehmen mit sich.

9.1. Continuous Integration verändert das Produkt

Der Quellcode der zu entwickelnden Software wird regelmäßig in kleinen Teilen eingecheckt, gebaut und getestet. So wird eine ständige Integration der Fortschritte eines Entwicklungsteam miteinander und auch im Kontext eines größeren Softwaresystems getestet und sichergestellt. Dieses Vorgehen bringt zwei wertvolle Aspekte mit sich.

Einerseits steht so ständig ein funktionales Produkt bereit. Dies sorgt dafür, dass das Risiko einer fehlschlagenden Integration – des „Big Bang“ – am Ende einer langwierigen Entwicklung minimiert werden kann. Außerdem bringt die ständige Integration eine ständige Testbarkeit mit sich. Das heißt, dass nicht bis zum Ende einer Entwicklung gewartet werden muss, um Tests durchzuführen. Dadurch wird es möglich Fehler früh zu erkennen und schon in den ersten Phasen der Entwicklung die Qualität des Produktes zu sichern.

Hinzu kommt, dass das Produkt in einer Umgebung gebaut wird, welche frei ist von Faktoren, die von der Entwicklungsumgebung des Entwicklers bestimmt werden. Der Build-Prozess wird auf einem oder mehreren entfernten Systemen ausgeführt. Dadurch wird es nötig, dass der Build-Prozess systemunabhängig gestaltet wird. Obwohl dies einen gewissen Mehraufwand darstellt, bringt es entscheidende Vorteile mit sich. Potentielle Fehlerquellen wie fehlende Einbindung von externen Bibliotheken, absolute Pfade oder von der IDE verwaltete Klassenpfade können schon früh identifiziert und beseitigt werden.

Es sei allerdings zu ergänzen, dass das Konzept von Continuous Integration kein Heilsbringer in den Aspekten der Softwarequalität ist. Das Produkt ist durch die Prinzipien von Continuous Integration nicht ausnahmslos frei von Fehlern. Schlechte Qualität von Quellcode oder Fehler in der Logik von Programmen können nicht durch Automatisierung von Prozessen ausgeschlossen werden. Hinzu kommt, dass eine erfolgreicher Build und positive Testmetriken ein Bild suggerieren können welches über eventuelle Schwächen hinwegtäuscht und den kritischen Umgang mit dem zu entwickelnden Code eher abschwächt.

9.2. Continuous Integration verändert die Prozesse

Neben dem Produkt an sich verändert die Einführung von CI vor allem auch die Prozesse in der Softwareentwicklung. Wie bereits in 9.1. erwähnt werden kleine Teile von Quellcode regelmäßig eingecheckt. Das hat zur Folge, dass der Entwickler stets funktionsfähigen Quellcode abliefern muss. Zusätzlich ist der einzucheckende Quellcode mit entsprechenden Testfällen zu versehen. Das heißt, dass hier ein Paradigmenwechsel zu einer test-intensiven Entwicklung erfolgen muss.

Das regelmäßige Testen und die Codeanalysen sind kurze Zeit nach dem Build-Prozess für den Entwickler sichtbar. So kann aufgrund der Test und Analysen (vergleiche Metriken in Kapitel 5.) eine weitere Iteration der Entwicklung angestoßen werden. Dadurch findet der Übergang zu einer agilen und flexiblen Entwicklung statt.

Hier ist zu beachten, dass der Übergang zu einer agilen und transparenten Entwicklung auf die Akzeptanz der Entwickler aufgebaut werden muss. Hierin liegt eine der großen Herausforderungen bei der Einführung von Continuous Integration. Es muss also von dem Management eine Atmosphäre geschaffen werden, in der dieser Wechsel positiv aufgenommen wird. Ist dies nicht der Fall, so werden die Prinzipien von CI schwer umzusetzen sein. Darüber hinaus muss klar sein, dass auch die Arbeitsergebnisse einzelner Entwickler sichtbar und messbar werden. Insbesondere sollten hier Vorkehrungen getroffen werden, um Probleme mit dem Betriebsrat und anderen Team-relevanten Problemen vorzubeugen.

Gelingt es diese Herausforderung erfolgreich zu meistern, so können weitere Mehrwerte realisiert werden. Neben der verbesserten Qualität kann durch die Automatisierung von Build-, Deployment- und Testprozessen eine enorme Zeitersparnis erreicht werden. Dies spiegelt sich nicht nur in den aufgewendeten Stunden der Entwickler, sondern auch in der verkürzten Zeit zur Fertigstellung des Produktes wieder.

Des Weiteren schöpfen auch Projektmanagementprozesse aus dem Verbesserungspotential der Continuous Integration-Einführung. So können durch eine gesteigerte Transparenz in den Bereichen der Zeit-, Ressourcen-, Budget- und Qualitätsplanung qualitativ hochwertigere Aussagen getroffen werden. Diese sorgen für fundiert Entscheidungen und ein verbessertes Risikomanagement. Die realisierten Potentiale bei der Einsparungen von Arbeitszeit, der frühen Erkennung und Beseitigung von Fehlern (vergleiche Kapitel 3.1. – relative Kosten der Fehlerbehebung) und verbesserten Managementprozessen zeigen sich schlussendlich in den reduzierten Kosten der Softwareentwicklung. Dies zeigte sich vor allem in den Erfahrungen bei der Erarbeitung des Prototyps.

9. Fazit und Empfehlung für das Unternehmen

Web-Anwendungen und Programme sind in der heutigen Zeit nicht mehr weg zu denken aus den verschiedenen Bereichen der Wirtschaft. Ob IT-Unternehmen, Warenhaus, Lebensmittel Unternehmen und im Falle dieses Projektes eventuell eine Versicherungen. Anwendungen sind in jedem Falle Werkzeuge die unabdingbar sind wenn es um organisatorische, aufklärende oder umsetzende Hilfsmittel geht. Um diese Unterstützungsfunktion zu gewährleisten ist die Qualität der Anwendung eines der wichtigsten Kriterien. Diese Qualität ist abhängig davon wie gut durchdacht und optimiert der Erstellungsprozess der Anwendung ist. Folglich, um die Qualität der Software zu optimieren, ist der Ansatzpunkt der Erstellungsprozess. Genau dieser Erstellungsprozess kann durch eine Continuous Integration System optimiert werden.

Ein Continuous Integration System ist ein Universalwerkzeug um die kontinuierliche Fertigstellung einer Anwendung in teilautomatisierte Prozesse zu strukturieren. Diese Umstrukturierung des Erstellungsprozesses mit Hilfe eines Continuous Integration Systems, wie umgesetzt in diesem Projekt, führt zu einem Prozessablauf der vom Erstellen von Codebestandteilen der Anwendung bis hin zu Integration in die bestehende Lösung reicht. Der Erstellungsprozess an sich ändert sich somit nicht, es bleiben die gleichen Prozessschritte die auch nicht automatisiert ablaufen können und in ihrer Individualität durchführbar sind wann immer benötigt oder gefordert. In diesem Projekt hat sich jedoch herausgestellt das diese Prozessschritte einem bestimmten Muster folgen und somit eine Optimierung durch eine Teilautomatisierung Aufwand in der Organisation des Erstellungsprozesses und Fehleranfälligkeit in der Umsetzung minimiert werden können.

So schließt sich nun der Kreis und es ist möglich die Qualität der Anwendung durch eine Optimierung im Integrationsprozess in einer Entwicklungsumgebung zu verbessern. Die notwendige Unterstützung für diese Optimierung bietet das Continuous Integration System, welche beispielhafte Umsetzung, Bestandteil dieses Projektes war.

Um die angestrebte Qualitätsverbesserung und Erstellungsprozessoptimierung der Anwendung zu gewährleisten ist demnach eine Teilautomatisierung der abhängigen Prozessschritte vor zu nehmen. Dieses Projekt gibt beispielhaft dafür das notwendige Wissen um, mit Anpassung an die unternehmenseigenen Entwicklungsumgebung, das Konzept der Continuous Integration als Hilfsmittel und Optimierungsmöglichkeit zu nutzen. Die Nutzung einer solchen Entwicklungsumgebung auf Basis von Continuous Integration ist somit eines der bekanntesten Best Practices und eine klare Empfehlung wert.

Abschließend ist also zu sagen das dieses Projekt beispielhaft aufzeigt wie ein Continuous Integration System aufgebaut und in eine Entwicklungsumgebung integriert wird. Es ist jedoch nur ein Teilschritt auf dem Weg zum angestrebten optimierten Entwicklungsprozess und setzt

nun die unternehmensspezifische Umsetzung voraus um eine Entwicklungsumgebung auf Basis des Continuous Integration und Deployment qualitätsoptimierend nutzen zu können. Dies ist unbedingt notwendig, um Business Guidelines ein zu halten, umzusetzen und das volle Potential des Systems ausschöpfen zu können. Dies liegt aber nun in der Hand des Unternehmens.

10. Anhang

10.1. Anhangverzeichnis

Anhang 1: Build Script

```
<?xml version="1.0"?>


<project name="hello.Tomcat" basedir="." default="usage">


	<tstamp/>


	<property name="src.dir"		value="src"/>
	<property name="build.dir"		value="build"/>
	<property name="classes.dir"		value="${build.dir}/classes"/>
	<property name="war.dir"		value="war"/>
	<property name="web-inf.dir"		value="${war.dir}/WEB-INF"/>
	<property name="lib.dir"		value="lib"/>
	<property name="result.dir"		value="build_result"/>
	<property name="jar.dir"		value="${build.dir}/jar"/>
	<property name="hello.pckg"		value="hello"/>
	<property name="tests.pckg"		value="Tests"/>
	<property name="report.dir"		value="junitreport"/>
	<property name="report-tstamp.dir"	value="junitreport/${DSTAMP}-${TSTAMP}"/>

	<property file="build.properties"/>


	<path id="classpath.build">
		<fileset dir="${lib.dir}" includes="**/*.jar"/>
	</path>
	<path id="classpath.test">
		<!-- <pathelement location="${classes.dir}"/>-->

		<fileset dir="${result.dir}" includes="**/*.jar"/>
		<fileset dir="${lib.dir}" includes="**/*.jar"/>
	</path>


	<target name="usage">
		<echo message=""/>
		<echo message="${name} build file"/>
		<echo message="------------------------------------"/>
		<echo message=""/>
		<echo message="Available targets are:"/>
		<echo message=""/>
		<echo message="clean 		--> Deletes build directory"/>
		<echo message="compile 		--> Compiles java-files in src di-
rectory"/>
		<echo message="jar 		--> Creates runnable jar from com-
piled classes"/>
		<echo message="junit 		--> Runs junit tests"/>
		<echo message="junitreport 	--> Reports on junit tests"/>
		<echo message="deploywar 		--> Deploy application as a WAR
file"/>
		<echo message="clean-build-deploy 	--> Cleans, compiles and deploys
application as a WAR file"/>
	</target>

	<target name="clean">
	  <delete dir="${build.dir}"/>
```

```
        <delete dir="${result.dir}"/>
        <delete dir="${web-inf.dir}/classes"/>
        <delete file="${jar.dir}/${hello.pckg}.jar"/>
    </target>

    <target name="compile">
            <mkdir dir="${classes.dir}"/>
            <javac          srcdir="${src.dir}"          destdir="${classes.dir}"
classpathref="classpath.build"/>
            <copy todir="${classes.dir}">
                  <fileset dir="${src.dir}" excludes="**/*.java"/>
            </copy>
      </target>

      <target name="jar" depends="compile">
            <mkdir dir="${result.dir}"/>
            <jar destfile="${result.dir}/result.jar" basedir="${classes.dir}"/>
      </target>

      <target name="junit" depends="jar">
            <mkdir dir="${report-tstamp.dir}"/>
            <junit printsummary="yes">
                  <classpath refid="classpath.test"/>

                  <formatter type="xml"/>

                  <!--    <test    name="Tests.TestJavaEngine"    todir="${report-
tstamp.dir}"/> -->

                  <batchtest todir="${report-tstamp.dir}">
                        <zipfileset       src="${result.dir}/result.jar"       in-
cludes="**/Test*.class"/>
                        <!--        <fileset        dir="${classes.dir}"        in-
cludes="**/Test*.class"/> -->
                  </batchtest>
            </junit>
      </target>

      <target name="junitreport" depends="junit">
            <junitreport todir="${report-tstamp.dir}">
                  <fileset dir="${report-tstamp.dir}" includes="TEST-*.xml"/>
                  <report todir="${report-tstamp.dir}"/>
            </junitreport>
      </target>

      <target name="war" depends="junitreport">
            <mkdir dir="${build.dir}/${war.dir}"/>
        <mkdir dir="${web-inf.dir}/classes/${hello.pckg}"/>
        <mkdir dir="${web-inf.dir}/classes/${tests.pckg}"/>
        <copy todir="${web-inf.dir}/classes/${hello.pckg}">
            <fileset dir="${classes.dir}/${hello.pckg}">
                  <include name="*.class"/>
            </fileset>
        </copy>
        <copy todir="${web-inf.dir}/classes/${tests.pckg}">
            <fileset dir="${classes.dir}/${tests.pckg}">
                  <include name="*.class"/>
            </fileset>
        </copy>
        <war    destfile="${build.dir}/${war.dir}/${war.name}.war"    webxml="${web-
inf.dir}/web.xml">
```

```
            <fileset dir="${war.dir}">
                <include name="**/*.*"/>
            </fileset>
        </war>
        <copy todir="." preservelastmodified="true">
            <fileset dir="${build.dir}/${war.dir}/">
                <include name="*.war"/>
            </fileset>
        </copy>
             <antcall target="clean"/>
      </target>

      <target name="deploywar" depends="war">
        <copy todir="${deploy.path}" preservelastmodified="true">
            <fileset dir=".">
                <include name="*.war"/>
            </fileset>
        </copy>
    </target>

    <target name="clean-build-deploy" depends="clean,deploywar"/>

</project>
```

Anhang 2: Sonar Properties

```
# required metadata

sonar.projectKey=my:helloTomcat
sonar.projectName=hello.Tomcat
sonar.projectVersion=1.0

# path to source directories (required)
sonar.sources=src

# path to test source directories (optional)
sonar.tests=src/Tests

# path to project binaries (optional), for example directory of Java bytecode
#sonar.binaries=binDir

# optional comma-separated list of paths to libraries. Only path to JAR file and
path to directory of classes are supported.
sonar.libraries=lib/*.jar

# Uncomment this line to analyse a project which is not a java project.
# The value of the property must be the key of the language.
sonar.language=java

#----- Default Sonar server
sonar.host.url=http://localhost:8080/sonar
#----- Global database settings
#sonar.jdbc.username=sonar
#sonar.jdbc.password=sonar
```

11. Quellenverzeichnis

11.1. Literaturverzeichnis

Beck, K. (2005): *JUnit kurz und gut.* Köln: O'Reilly.

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

Cleff, T. (2010): *Basiswissen Testen von Software.* Dortmund: W3L GmbH.

Grechenig, T. e. (2009): *Softwaretechnik: Mit Fallbeispielen aus realen Entwicklungsprojekten.* München: Pearson Studium.

Kaczanowski, T. (o.O.): *Practical Unit Testing with TestNG and Mockito.* 2012: Tomasz Kaczanowski.

Majer, D. (2008): *Unit Tests mit ABAP® Unit.* Heidelberg: dpunkt.Verlag.

Offutt, J. / Ammann, P. (2008): *Introduction to Software Testing .* Cambridge: Cambridge University Press.

Schneider, K. (2012): *Abenteuer Softwarequalität: Grundlagen und Verfahren für Qualitätssicherung und Qualitätsmanagement.* München: dpunkt.verlag GmbH.

Schumacher, M./Kaman, T. (2008): Continuous-Integration -Systeme. *java magazin, Volume 10*.

Smart, J. F. (2011): *Jenkins, the Definite Guide,* Sebastopol: O'Reilly Media.

Sommerville, I. (2006): *Software Engineering* (8 Ausg.). Bonn: Addison Wesley.

Wagner, S./Handte, M. (2012): Continuous Integration mit Git, Gerrit und Jenkins-Besser, öfter, mehr, In: *Eclipse Magazin*, Ausgabe 5.2012

Wickner, B./ Müller, B. (2012): Professionelle Software-Entwicklungs-Umgebungen in der Hochschulausbildung. In M. A. Goltz, *Proc. Informatik 2012.* Braunschweig.

Wiest, S. (2011): *Continuous Integration mit Hudson,* Heidelberg: dpunkt.verlag.

11.2. Internetquellen

Brinkman, S. (2013): *Die Qual der Wahl: Hudson/Jenkins vs. Continuum*. Von http://sascha.familie-brinkmann.net/die-qual-der-wahl-hudsonjenkins-vs-continuum, Abruf: 13.01.2013

Electric Cloud. (2010): *Survey Finds 58% of Software Bugs Result from Test Infrastructure and Process, Not Design Defects*. Von http://www.electric-cloud.com/news/2010-0602.php, Abruf: 13.01.2013

Dömer, S. (2010): *JUnit - Testing Framework für Komponententests*, http://saschadoemer.blogspot.de/2010/11/junit-testing-framework-fur.html, Abruf: 12.01.2013

Ghazali, F. (2005): *UNIVERSITI PUTRA MALAYSIA*. Retrieved from COMPARISON OF SOFTWARE TESTING TOOLS ON GUI,http://psasir.upm.edu.my/5191/1/FSKTM_2005_11.pdf, Abruf: 15.01.2013

Feustel, B./Schluff, S. (2012): Continuous-Integration-in-Zeiten-agiler-Programmierung *(Heise Developer)*, http://www.heise.de/developer/artikel/Continuous-Integration-in-Zeiten-agiler-Programmierung-1427092.html, 29. 11 2012

Gentz, E. (kein Datum). *Die-neue-Freiheit-bei-der-Versionskontrolle*. (Heise Developer) http://www.heise.de/developer/artikel/Die-neue-Freiheit-bei-der-Versionskontrolle-1224755.html?artikelseite=2, Abruf: 12. 01 2013

IEE Standards Association (1998): IEEE Standard for a Software Quality Metrics Methodology: http://standards.ieee.org/findstds/standard/1061-1998.html, Abruf: 12. 01 2013

it-agile GmbH. (2013). *Automatisierter Build-Prozess*. Von http://www.it-agile.de/build.html, Abruf: 15.01.2013

o.V. (2010): TestNG Tutorial and Example – Getting Started: http://www.asjava.com/testng/testng-tutorial-and-example-getting-started/, Abruf: 09. 01 2013

Kenneth, M. (2009): *Continuous Integration*. Retrieved from http://www.cs.colorado.edu/~kena/classes/5828/s09/lectures/19-testing.pdf, Abruf: 15.12. 2012

Konigsberg, R. (2007): *Blatherberg*. Retrieved from TestNG and Expected Exceptions: http://konigsberg.blogspot.de/2007/11/testng-and-expectedexceptions-ive.html, Abruf: 19.12. 2012

Kreissl, H. (2004): *Grundlagen der Informatik*. Retrieved from Kapitel 8 - Software-Metriken: http://www.kreissl.info/st_08.php, Abruf: 03.01. 2013

Mkyong. (2009): *Mkyong*. Retrieved from JUnit 4 Vs TestNG – Comparison: http://www.mkyong.com/unittest/junit-4-vs-testng-comparison/, Abruf: 15.12. 2013

Pichler, M. (2009): *Softwaremetriken verstehen und nutzen*. Retrieved from PHP Unconference Hamburg: http://manuel-pichler.de/stuff/slides/2009-09-12-phpunconf-softwaremetriken-verstehen-und-nutzen.pdf, Abruf: 15.12. 2012

Rukes, C. /Weich, W. (2010): *WinfWiki*. Retrieved from Analyse und Bewertung des Tools Selenium: http://winfwiki.wi-fom.de/index.php/Analyse_und_Bewertung_des_Tools_Selenium, Abruf: 19.12. 2012

Scholz, F. (o.J.): *Westfälische Wilhelms-Universität Münster*. Retrieved from Testen von Benutzeroberflächen: http://www.wi.uni-muenster.de/pi/lehre/ws0809/SeminarSE/ausarbeitungen/Ausarbeitung---Testen-von-Benutzeroberflchen-Fabian-Scholz.pdf, Abruf: 29.12. 2012

Schuster, G. (2012): Kriterien-fuer-GUI-Testwerkzeuge (*Heise Developer),* http://www.heise.de/developer/artikel/Kriterien-fuer-GUI-Testwerkzeuge-1569050.html?artikelseite=3, Abruf: 15.12. 2012

Taentzer. (2012): *Softwarequalität 2012*, http://www.mathematik.uni-marburg.de/~swt/ss12/sq/files/Folien120613.pdf, Abruf: 03.12. 2012

TestNG. (2013): *TestNG*. Retrieved from TestNG Documentation, http://testng.org/doc/documentation-main.html#groups-of-groups, Abruf: 15.12. 2012

w.V. (2007): *Tutorials.de User helfer User*. Retrieved from JUnit Test Suite erstellen: http://www.tutorials.de/java/298619-junit-test-suite-erstellen.html, Abruf: 12.12. 2012

Ullenboom, C. (2010):. *Tutego*. Retrieved from JUnit 4 Tutorial, Java-Tests mit JUnit: http://www.tutego.de/blog/javainsel/2010/04/junit-4-tutorial-java-tests-mit-junit/#, Abruf: 07.01. 2013

Vogel, L. (2012): *Vogella*. Retrieved from JUnit - Tutorial: , http://www.vogella.com/articles/JUnit/article.html, Abruf: 15.12. 2012

OpenSourceTesting.org (2013): OpensourceTesting.org Retrieved from

FunctionalTestin(2008): http://www.opensourcetesting.org/functional.php, Abruf: 03.12. 2012

o.V. (2013). Jenkins Community Survey Results,http://www.cloudbees.com/sites/default/files/documents/2012%20Jenkin%20Survey%20Results.pdf, Abruf: 18.01.2013

Westphal, F. (2005): *JUnit 4.0,* http://www.frankwestphal.de/JUnit4.0.html, Abruf: 15.12. 2012

Entwicklung eines End-to-End-Monitoring-Konzeptes für Webanwendungen am Beispiel von DokuWiki

Schriftliche Ausarbeitung

im Rahmen der Lehrveranstaltung „Projekt des 5. Semesters“

Kompetenzzentrum Open Source (KOS)

vorgelegt von

Fabian Flaig und Fabian Stellmacher

am 23.01.2013

Fakultät Wirtschaft

Studiengang Wirtschaftsinformatik

WWI2010I

Inhaltsverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
BS	Betriebssystem
BSD	Berkeley Software Distribution
CLI	Command-Line Interface
CPU	Central Processing Unit
CSS	Cascading Style Sheets
DNS	Domain Name System
E2E	End-to-End
FTP	File Transfer Protocol
GNU GPL	GNU General Public Licence
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IT	Informationstechnologie
JAMon	Java Application Monitor
JDK	Java Development Kit
NRPE	Nagios Remote Plugin Executor
OSS	Open Source Software
PDF	Portable Document Format
PHP	PHP: Hypertext Preprocessor
RC	Remote Control

SLES	Suse Linux Enterprise Server
WatiN	Web Application Testing in .Net
Watir	Web Application Testing in Ruby
XML	Extensible Markup Language

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Einführung in die Thematik

Mit wachsenden IT-Systemen und einem gesteigertem Stellenwert der IT innerhalb von Unternehmen, nimmt neben der Komplexität dieser auch der Wartungs- und Überwachungsaufwand über die letzten Jahre zu.[1] Systeme müssen häufig rund um die Uhr verfügbar sein, um geschäftskritische Prozesse aufrecht zu erhalten. Um Fehler und Ausfälle schnellstmöglich beheben zu können, wird zunehmend auf eine dauerhafte, automatisierte Überwachung, auch Monitoring genannt, gesetzt.

Dies kann in drei Schichten unterteilt werden: Zum einen befasst sich das reine Netzwerkmonitoring damit, die Netzwerkkomponenten, wie z.B. Hubs und Switche, zu überwachen und zu kontrollieren[2]. Fällt ein Teil der Hardware oder ein Dienst aus, wird dies umgehend gemeldet, damit sich um die Widerherstellung der Verfügbarkeit gekümmert werden kann. Die Aufgabe der Systemmonitoring-Schicht wiederum ist es, die Hardwarekomponenten, wie z.B. Server und deren Eigenschaften und Komponenten zu überwachen. So kann beispielsweise die CPU-Temperatur oder der noch verfügbare Speicherplatz kontrolliert werden. Die oberste Schicht bildet die Anwendungsschicht, also nur das Sicherstellen, dass die Anwendung funktionsfähig ist.

Eine Verbindung stellt das so genannte End-to-End-Monitoring (E2E-Monitoring) her, dessen Aufgabe es ist, nachzuweisen, dass eine Anwendung beim Nutzer ordnungsgemäß funktioniert[3]. Ein Problemfall kann zum Beispiel sein, dass bei einem reinen Netzwerkmonitoring alle Komponenten fehlerfrei funktionieren, aber dennoch der Service beim Endanwender nicht verfügbar ist und dies nicht rechtzeitig bemerkt wird. Andersherum könnte es auch passieren, dass eine geclusterte Anwendung durch das E2E-Monitoring System als verfügbar angezeigt wird, bei erhöhtem Zugriff aber schlechte Performance-Werte liefert oder gar ganz ausfällt.

Um diese beiden Schwachstellen zu vermeiden, soll in dieser Arbeit ein ganzheitliches Konzept für ein Unternehmen entwickelt werden, wie ein E2E-Moniotring umgesetzt werden kann, um eine dauerhafte und performante Verfügbarkeit zu gewährleisten und Dienstausfällen

[1] Vgl. Herrmann, W. (2010)
[2] Vgl. DHBW (2012)
[3] Vgl. ebenda

vorzubeugen. Anhand des Beispiels der Webanwendung „DokuWiki“ soll dieses Konzept als Prototyp vorgestellt werden.

1.2 Vorgehensweise

Im Rahmen dieser Arbeit werden zunächst die Grundlagen für die Entwicklung eines Lösungsansatzes erläutert. Mit Hilfe dieser und der Anforderungen des Unternehmens, wird dann eine Marktübersicht über mögliche Tools zur Ergänzung, des sich bereits im Einsatz befindlichen Systemüberwachungstools „Nagios“[4], erstellt. Hierzu gehören neben einer Auswahl näher zu betrachtender Tools auch Erläuterungen zu den Bewertungskriterien. Eine Kurzbeschreibung der Tools soll einen weiteren Einblick geben, bevor dann anhand des Kriterienkataloges eine Bewertung und Auswahl für die Umsetzung getroffen wird.

Anschließend soll die Entwicklung des Prototyps vorgestellt werden. Hierbei kommt es neben den Voraussetzungen vor allem auf die Architektur an und wie die Komponenten interagieren, um den ganzheitlichen Ansatz widerzuspiegeln.

Abschließend wird das Konzept und die Umsetzung kritisch betrachtet und ein Ausblick auf eine mögliche Nutzung bei dem Unternehmen gegeben.

[4] Nagios (2012a)

2 Grundlagen

Um ein einheitliches Verständnis zu schaffen, werden im Folgenden zunächst grundlegende Begriffe definiert und erläutert, die für die Entwicklung des Konzeptes nötig sind.

2.1 System- und Netzwerkmonitoring mit Nagios Core

Bevor die Software Nagios kurz vorgestellt wird, soll zunächst der Begriff Systemmonitoring betrachtet werden. Im Allgemeinen befasst sich das Monitoring mit der systematischen Beobachtung und/oder Überwachung von Prozessen unter Verwendung technischer Hilfsmittel. Um Schlussfolgerungen aus den Ergebnissen ziehen zu können, wird meistens auf eine umfassende Historie Wert gelegt, mit welcher Vergleiche gezogen werden können.[5]

System- und Netzwerkmonitoring beschäftigt sich demnach mit der Überwachung von Computernetzwerken, insbesondere deren Komponenten. Zu diesen zählt nicht nur Hardware, wie z.B. Server, Router oder Switche, sondern auch Netzwerkdienste, wie z.B. DNS- oder E-Mail-Dienste. [6] Die aktuellen Zustände der Komponenten werden dann durch ein Systemmonitoringprogramm visualisiert, welches zusätzlich Zustandsveränderungen per Benachrichtigung an die Verantwortlichen meldet. Unterschieden werden Monitoringprogramme in aktiv und passiv, was bedeutet, dass ein Programm entweder aktiv Pakete über das Netzwerk verschickt oder dass ein Programm passiv „zuhört“ wie Netzwerkkomponenten miteinander kommunizieren und dann die Ergebnisse verarbeitet.[7]

Das Tool Nagios Core beherrscht diese beiden Arten des Monitorings. Es ist ein Open Source Programm, welches unter der GNU General Public Licence (GNU GPL) verwendet werden darf und für Unix-ähnliche Betriebssysteme entwickelt wurde.[8]

Durch die kostenlose Verfügbarkeit, die hohe Anpassungs- und Erweiterbarkeit und die breite Unterstützung der Community, ist Nagios zu einem der meist verbreiteten Open Source Monitoring Tools geworden, mit schätzungsweise mehr als 1.000.000 Nutzern weltweit.[9] Es basiert auf der grundlegenden Unterscheidung der Konfiguration von Hosts, Services, Kommandos und Kontakten, wie in Abbildung 1 dargestellt wird. Ein Host ist ein im Netzwerk

[5] Vgl. Wikipedia (2012a)
[6] Vgl. Mitnacht, S. (2006), S.17
[7] Vgl. ebenda, S.17
[8] Vgl. Wikipedia (2012b)
[9] Vgl. Nagios (2012b)

vorhandener Computer oder Server, der Informationen oder Dienste bereitstellt. Dieser kann von Nagios über seine IP-Adresse identifiziert und so angesprochen werden. Die Dienste werden in Nagios Services genannt und beinhalten einerseits z.B. HTTP, FTP, usw. aber andererseits auch Eigenschaften des Hosts, wie Festplattenkapazität, CPU-Auslastung usw. Die Anweisungen, wie etwas von Nagios überwacht werden soll oder welche Benachrichtigung wann verschickt werden sollen, werden in so genannten Kommandos zusammengefasst. Löst eine Antwort bzw. eine ausbleibende Antwort ein Ereignis, und somit eine Benachrichtigung aus, wird diese dem zuständigen Kontakt, also beispielsweise einem IT-Verantwortlichen per E-Mail zugestellt, welcher sich dann mit dem Vorfall befassen kann.

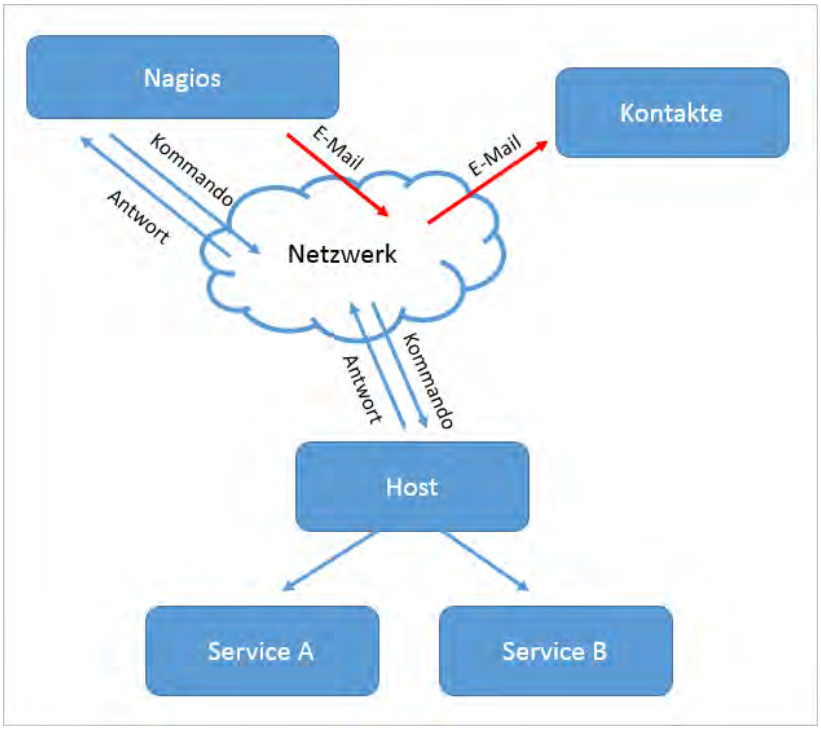
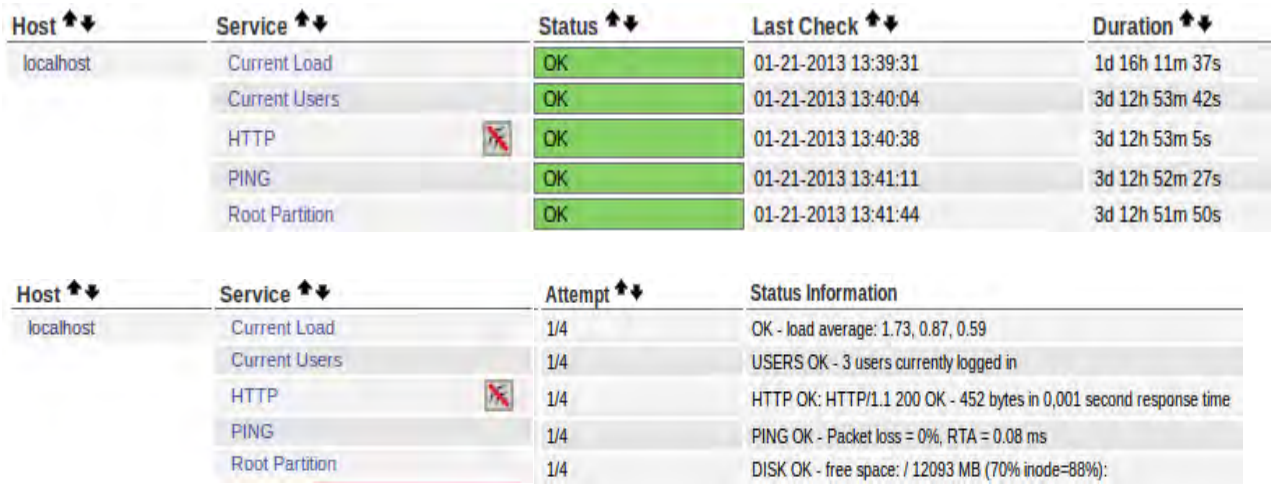


Abbildung 1: Zusammenhang zwischen Nagios-Komponenten

Getestete Services oder Hardwarekomponenten werden in Nagios üblicherweise mit einem Ampelsystem bestehend aus den Werten „OK“ in Grün, „Warning“ in Gelb oder „Critical“ in Rot angezeigt, wie in Abbildung 2 zu sehen ist. Den Hosts werden in dieser Tabelle ihre Services und die Ergebnisse des letzten Tests zugeordnet.



Host	Service	Status	Last Check	Duration
localhost	Current Load	OK	01-21-2013 13:39:31	1d 16h 11m 37s
	Current Users	OK	01-21-2013 13:40:04	3d 12h 53m 42s
	HTTP	OK	01-21-2013 13:40:38	3d 12h 53m 5s
	PING	OK	01-21-2013 13:41:11	3d 12h 52m 27s
	Root Partition	OK	01-21-2013 13:41:44	3d 12h 51m 50s

Host	Service	Attempt	Status Information
localhost	Current Load	1/4	OK - load average: 1.73, 0.87, 0.59
	Current Users	1/4	USERS OK - 3 users currently logged in
	HTTP	1/4	HTTP OK: HTTP/1.1 200 OK - 452 bytes in 0,001 second response time
	PING	1/4	PING OK - Packet loss = 0%, RTA = 0.08 ms
	Root Partition	1/4	DISK OK - free space: / 12093 MB (70% inode=88%):

Abbildung 2: Übersicht in Nagios

Die oben stehende Abbildung stellt diese Übersicht der Service-Tests für den Host namens „localhost“ dar. Im Feld Status wird der aktuelle Zustand des Services angezeigt, der durch das beschriebene Ampelsystem visualisiert wird. Zusätzlich zur Statusinformation wird zudem der Zeitpunkt der letzten Ausführung angegeben, sowie die Zeit, die der Host seit dem letzten Start in Betrieb ist (Duration). Ist ein Test nicht erfolgreich, wird der Status standardmäßig auf „Warning“ gesetzt. Erst nach dem vierten erfolglosen Versuch, den Test auszuführen, ist der Zustand „Critical“. Repräsentiert wird dies in der Spalte „Attempts“, in der momentan 1/4 angezeigt wird, weil alle Tests beim ersten Anlauf erfolgreich waren. In der letzten Spalte werden zusätzliche Informationen angezeigt, beispielsweise die Antwortzeit und die Anzahl der verloren gegangenen Pakete, wenn ein Ping ausgeführt wurde oder die freie Festplattenkapazität, wenn der Service Root Partition durchgeführt wird.

2.2 End-to-End Monitoring (E2E-Monitoring)

Im Gegensatz zum System- und Netzwerkmonitoring befasst sich das E2E-Monitoring nicht mit den einzelnen Komponenten eines Netzwerks, sondern mit dem Gesamtbild aller beteiligten Komponenten und der Überwachung, ob eine Anwendung beim Nutzer ordnungsgemäß funktioniert. Dazu wird überprüft, wie performant und in welcher Zeit ein Aufruf bearbeitet wird.[10] Dies liefert dem Administrator permanent Daten über die Performance und die Verfügbarkeit aus

[10] Vgl. DHBW (2012)

Sicht der Endanwender.[11] Die folgende Abbildung zeigt zusammenfassend die drei vom E2E-Monitoring überwachten Schichten. Sowohl die System-, als auch die Netzwerkschicht können von Nagios überwacht werden. In wie weit die Anwenderschicht hinzugefügt werden kann, um das Gesamtbild zu erhalten, soll Ziel dieser Arbeit sein.

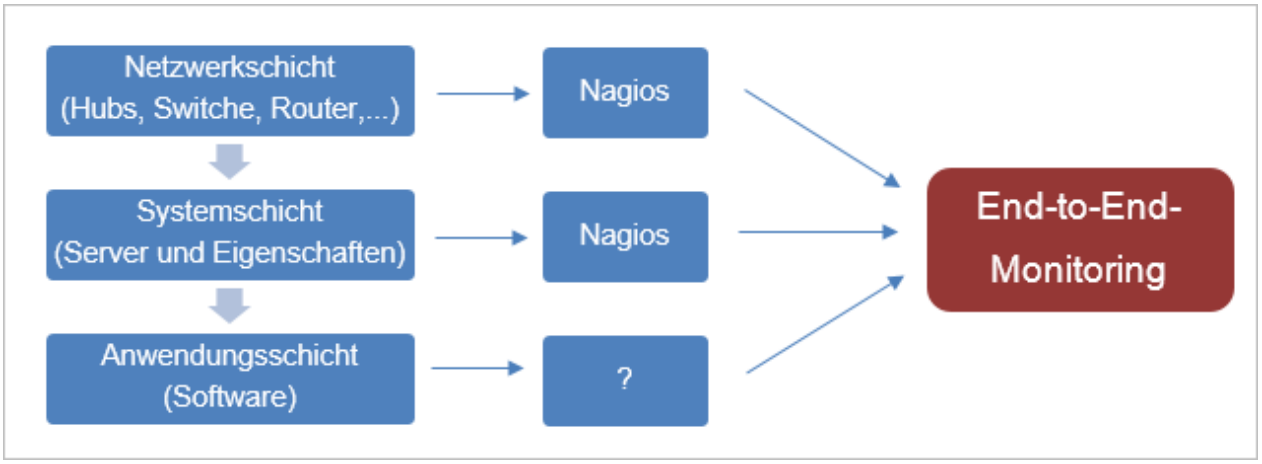


Abbildung 3: Schichten des E2E-Monitorings

Die Aussage aus dem E2E-Monitoring ist jedoch in geclusterten Umgebungen nur eingeschränkt nutzbar. In einem Cluster werden einzelne Hardwarekomponenten, wie Server oder Router, dupliziert, um bei einem Ausfall eines einzelnen Teils weiterhin die Funktionalität zu gewährleisten oder um in Zeiten von erhöhten Zugriffen eine hohe Verfügbarkeit zu garantieren. Da im Fall eines Hardwareausfalls, die Anfrage durch das Cluster-eigene Monitoring an einen anderen Server geleitet wird, der die Anfrage dann bearbeitet, liefert ein E2E-Monitoringsystem ein positives Feedback und ordnet die Anwendung als beim Endnutzer verfügbar ein. Problematisch kann es hier jedoch in Zeiten von hoher Auslastung zu Engpässen oder Ausfällen kommen.

Aus Zeitgründen widmet sich diese Arbeit einem E2E-Monitoring-Konzept für eine bestimmte Webanwendung, auch wenn weitere Anwendungen, wie z.B. E-Mail-Programme (Lotus Notes) ebenfalls mit Hilfe eines E2E-Monitoring überwacht werden können. In dieser Arbeit soll das E2E-Monitoring durch die Einführung von automatisierten Tests der Anwendungen durchgeführt werden. Neben dieser Möglichkeit wurden auch weitere Ansätze betrachtet, auf die an dieser Stelle kurz eingegangen werden soll.

[11] Vgl. Leutek (2013)

2.3 Testautomatisierung von Webanwendungen

Bevor näher auf den Zusammenhang zwischen Testautomatisierung und einem E2E-Monitoring eingegangen wird, soll zunächst ein einheitliches Verständnis zum Testen von Software und zur Testautomatisierung geschaffen werden.

Der IEEE-Standard 610 beschreibt das Testen im Rahmen des Software Engineering als einen Prozess, in welchem ein System oder eine Komponente unter bestimmten Bedingungen betrieben wird, dabei Beobachtungen notiert und dann bewertet werden.[12] Eingeschränkter ist die Sichtweise des IEEE-Standards 829, welcher das Testen als einen Prozess versteht, in welchem die Unterschiede zwischen dem Soll- und dem Ist-Zustand einer Software festgehalten und analysiert werden.[13] Der Testing-Prozess besteht also aus verschiedenen Aktivitäten, die die Software-Funktionen und –Eigenschaften mit den Anforderungen vergleichen sollen.

Die Testautomatisierung dient nun dazu, diese Aktivitäten automatisch ablaufen zu lassen, wodurch sich diverse Vorteile gegenüber einem manuellen Testen ergeben. Vor allem die Möglichkeit Tests zu geringen Kosten wiederholt ausführen zu lassen und die Anzahl der Fehler, die beim manuellen Testen entstehen können, beschreiben den größten Nutzen von automatisierten Tests.[14] Folglich ergibt sich hauptsächlich aus der hohen Testdichte (Coverage), aber auch durch die Fehlerreduktion bei der Testdurchführung eine gesteigerte Qualität der durch die Testfälle abgedeckten Software-Funktionen. Bei der Durchführung ist jedoch auch zu beachten, dass eine höhere Coverage Ressourcen benötigt und so die reguläre Verwendung beeinträchtigen kann.

Im Zusammenhang mit einem E2E-Monitoring wird diese Testautomatisierung dazu genutzt, konstruierte Testfälle ablaufen zu lassen und so den Endnutzer zu simulieren. Dafür wird eine Webanwendung, also eine Anwendung, welche auf dem Client-Server Prinzip beruht und über einen Webbrowser aufgerufen, auf ihre Funktionalität und Verfügbarkeit getestet. Dies bedeutet, dass das E2E-Monitoring System in festgelegten Zeitintervallen einen Testfall (z.B. einen JUnit-Testfall) aufruft, welcher dann aus Sicht des Users durchgeführt wird und abschließend Performance- und Verfügbarkeitswerte zurück an das Monitoring-System liefert, wo diese Daten aufbereitet und analysiert werden. Zu beachten ist dabei, dass jeder Testaufruf auch eine

[12] Vgl. Jonassen, A.M. (2008), S.XXV
[13] Vgl. ebenda, S.XXV
[14] Vgl. Bommer, C. / Spindler, M. / Barr, V. (2008), S. 258

Belastung für das System darstellt und eine zu häufige Durchführung der Tests, das System beeinträchtigen kann.

2.4 Open Source Software

Im Gegensatz zu proprietärer Software, also eine, die traditionell von einem Software-Anbieter verkauft wird, ist eine Open Source Software (OSS) mit einer von der Open Source Initiative konformen Lizenz ausgestattet. Der Definition nach müssen drei wesentliche Merkmale gelten, damit eine Lizenz von der Open Source Initiative anerkannt wird: Der Quellcode der Software ist öffentlich zugänglich und von jedem für seine Zwecke zu nutzen. Zudem ist es gestattet, OSS zu modifizieren, je nach Lizenz gibt es teilweise aber die Einschränkung, dass eine Veränderung wiederum den Entwicklern zur Verfügung gestellt werden muss.[15]

Getrieben wird die Entwicklung solcher Software von der so genannten Community, welche sowohl aus Privatleuten, als auch aus Unternehmen bestehen kann. Bekannte Beispiele für Open Source Software sind u.a. der Browser Firefox, bereitgestellt durch die Mozilla Foundation[16], das von der Open Handset Alliance / Google entwickelte Betriebssystem Android[17] oder das Betriebssystem Linux.

Eine ausführliche Definition zu Open Source Software und insbesondere zu Open Source Testautomatisierungssoftware kann in der Arbeit „Evaluation of Open Source Tools for Test Management and Test Automation“[18] nachgelesen werden.

[15] Vgl. Open Source Initiative (2013)
[16] Vgl. Mozilla Foundation (2013)
[17] Vgl. Android (2013)
[18] Vgl. Fleischer, D. et al (2012)

3 Marktübersicht zur Erweiterung von Nagios durch Tools zur Testautomatisierung von Webapplikationen

Die Erarbeitung einer Marktübersicht dient als Entscheidungshilfe, welches Tool für die Integration in die Systemmonitoring-Umgebung von Nagios in Frage kommt und inwiefern die Software den weiteren Anforderungen gerecht wird.

3.1 Auswahl der Produkte

Aufgrund der hohen Anzahl an verschiedenen Produkten im Rahmen des automatisierten Testens von Webanwendungen, muss zunächst eine Vorauswahl der Tools durchgeführt werden, welche dann im Folgenden näher betrachtet werden sollen. Erstes Auswahlkriterium war hierbei, dass es sich bei dem ausgewählten Tool um Open Source Software handeln sollte. Weitere Kriterien, wie Marktrelevanz, Funktionalität, Erwähnungen und Empfehlungen verschiedener Quellen, sowie Einschätzungen zur Umsetzbarkeit, bildeten dann die Grundlage für die Eingrenzung der Marktübersicht auf die folgenden sechs Testautomatisierungstools, welche in alphabetischer Reihenfolge vorgestellt werden:

- Canoo Test Web,
- JAMon,
- Selenium,
- WatiN (gesprochen „what-in“),
- Watir (gesprochen „water“) und
- WebInject.

Alle ausgewählten Tools dienen zum Aufzeichnen bzw. Programmieren von Benutzerinteraktion mit einer Webanwendung, auf deren Vor- und Nachteile soll in der jeweiligen Kurzvorstellung näher eingegangen werden. Die zusammenfassende Übersicht der Kurzvorstellungen findet sich im Kapitel 3.4.

3.2 Vorstellung der Testkriterien

Für die Marktübersicht wurde die Einteilung der Kriterien in drei Kategorien vorgenommen: **Technische Aspekte** umfassen Merkmale, wie z.B. Systemvoraussetzungen und die damit

verbundene Systeminteroperabilität, welche beschreibt, wie flexibel die Software in Hinblick auf verschiedene Betriebssysteme ist. Außerdem umfasst dieser Punkt die Kompatibilität mit Nagios.

Zusammenfassung der technischen Merkmale:

- Systeminteroperabilität
- Systemvoraussetzungen
- Kompatibilität mit Nagios

Neben den technischen Aspekten, werden weiterhin **funktionale Merkmale** betrachtet und beurteilt. Hierbei spielt einerseits die Installation und Konfiguration der Software eine Rolle, aber hauptsächlich die Komplexität der Bedienung, sowie der Erstellung der Test Cases. Alle Lösungen bieten eine manuelle Programmierung der Test Cases an, einige jedoch verknüpfen dies mit einem Aufnahmemodus („Capture and Replay“), sodass zunächst die Benutzersteuerung aufgenommen und dann manuell bearbeitet werden kann. Zudem hat der gebotene Funktionsumfang des Tools Einfluss auf die Bewertung der funktionalen Merkmale, genauso wie die unterstützten Programmiersprachen, bei denen nicht nur auf die Verbreitung der Programmiersprache, sondern auch die Breite der verschiedenen Sprachen berücksichtigt wird.

Zusammenfassung der funktionalen Merkmale:

- Installation und Konfiguration
- Komplexität
- Erstellung von Test Cases
- Funktionsumfang
- Unterstützte Programmiersprachen

Als dritte Hauptgruppe finden die so genannten **Open Source Merkmale** Einfluss auf die Gesamtwertung. Zu diesen Merkmalen zählen nicht nur die Art der Lizenz, unter welcher die Software genutzt und bearbeitet werden darf, sondern auch der Umfang und die Qualität der Dokumentation. Eng verknüpft damit ist das Support-Angebot, dessen Bewertung sich an der Verfügbarkeit von Foren, E-Mailing-Listen und Chats richtet. Auch ein möglicher kommerzieller Support wurde hier nicht ausgeschlossen, sondern positiv in die Bewertung aufgenommen. Abschließend wird in dieser Kategorie die Aktivität der Community bewertet. Betrachtet wird

dabei, wie häufig neue Versionen und Bugfixes veröffentlicht werden und wie die Beteiligung in Foren, Blogs usw. ausfällt.

Zusammenfassung der Open Source Merkmale:

- Lizenz
- Dokumentation
- Support
- Aktivität der Community

Jedes Tool wird pro Merkmal auf einer Skala von 0 bis 10 bewertet, wobei 0 das schlechteste und 10 das Beste zu erreichende Ergebnis ist. Die Ergebnisse der drei Kategorien werden dann gewichtet und verrechnet. Technische Aspekte erhalten dabei 30%, genauso wie die Open Source Merkmale, da diese beiden Kategorien auf das Umfeld der Software eingehen. Auf der einen Seite geben diese Auskunft über einen möglichen Einsatz, andererseits aber auch auf den Betrieb an sich. Wichtiger im Vergleich dazu wird die Funktionalität bewertet, weshalb diese mit 40% in die Wertung einfließt. Erfüllt eine Anwendung grundlegende Voraussetzungen des Kunden nicht, so findet hier bereits eine Abwertung statt.

3.3 Marktübersicht

3.3.1 Canoo WebTest

Das Open Source Tool Canoo WebTest[19] ist ein rein Java-basiertes Programm speziell für das automatisierte Testen von Webanwendungen. Es kann mit allen gängigen Betriebssystemen benutzt werden, auf denen das Java Development Kit (JDK) 5 (oder höher) sowie eine Version von Apache Ant (ab Version 1.7.0) installiert sind. Es zeichnet sich vor allem durch eine einfache Syntax aus, denn es unterstützt nicht nur die Erstellung von Testdateien in Groovy, sondern auch in XML.[20]

Neben einer umfangreichen Installations- und Konfigurationsanleitung bietet die Entwickler-Community von Canoo WebTest eine sehr ausführliche Dokumentation mit vielen hilfreichen Beispielen an. Weniger umfangreich sind hingegen die Supportmöglichkeiten, denn neben

[19] Canoo (2013a)
[20] Vgl. Canoo (2013b)

kommerzieller Unterstützung steht nur eine E-Mailing-Liste zur Verfügung. Die aktive Community sorgt dafür regelmäßig für Updates bzw. Bugfixes. Die letzte stabile Version wurde im August 2012 veröffentlicht.

Der Funktionsumfang des Tools beinhaltet eine umfassende Reportingfunktion, die sich vom Reporting anderer Tools positiv abhebt, da diese entweder keine oder nur über eine Erweiterung ein annähernd so umfassendes Reporting bieten.[21] Eine weitere Besonderheit ist die schnelle Ausführung von Testfällen, was einerseits durch die Ausführung in einer Java Virtual Machine und andererseits durch das Simulieren eines Browser, bei dem z.B. CSS-Elemente und Bilder außenvorgelassen werden und somit nicht geladen werden.[22] Doch hier besteht auch ein Nachteil: Da nur das Verhalten eines Browser wie Firefox oder Internet Explorer durch HTMLUnit simuliert wird, kann es Abweichungen zum Testergebnis bei der Ausführung von JavaScript in anderen Browsern geben. Canoo WebTest hebt sich außerdem dadurch ab, dass von einer Webseite aufgerufene PDF- oder Excel-Dateien in den Test integriert werden können, gleiches gilt auch für E-Mails, die durch die Ausführung der Webapplikation abgeschickt werden.[23]

Negativ zu bewerten ist hingegen, dass auf eine so genannte „Capture and Replay“-Funktion, also eine Aufnahme der Interaktion mit der Webapplikation und das Erstellen eines Testfalls in Form von Code, verzichtet wurde und auch bisher kein Plug-In für Nagios existiert.

Zusammenfassung Canoo WebTest:

- Benötigt XML- oder Groovy-Kenntnisse
- Reportingfunktion, Unterstützung von Excel- PDF- und E-Mail-Formaten
- Keine „Capture and Replay“-Funktion, kein Nagios-Plug-In
- Aktive Community
- Sehr gute Dokumentation, vielfältige Supportmöglichkeiten

[21] Vgl. Guillemot, M. (2007)
[22] Vgl. Canoo (2013b)
[23] Vgl. ebenda

3.3.2 Java Application Monitor (JAMon)

Das Java Application Programming Interface (API) JAMon[24] ist ebenfalls für alle gängigen Betriebssysteme geeignet, auf denen eine JDK-Version ab 1.4 laufen kann. Nach eigenen Angaben handelt es sich um ein einfaches Tool, in das man sich leicht einarbeiten und mittels Java Testfälle schreiben kann, andere Programmiersprachen werden von JAMon nicht unterstützt.[25]

Die Dokumentation ist eher einfach gehalten und weniger übersichtlich im Vergleich zu anderen bewerteten Dokumentationen, dafür kann über ein recht aktives Forum kommuniziert werden. Zur Installation gibt es keine näheren Angaben, ein Video auf YouTube präsentiert allerdings die ersten Schritte in JAMon. Das Tool bietet den geringsten Funktionsumfang aller betrachteten Lösungen, mit den grundlegendsten Funktionen zum Testen von Webapplikationen, aber ohne ein „Capture and Replay“, eine Reportingfunktion oder ein entwickeltes Nagios-Plug-In.

Zusammenfassung JAMon:

- Benötigt Java-Kenntnisse
- Beinhaltet nur grundlegendste Funktionen
- Keine „Capture and Replay“-Funktion, kein Nagios-Plug-In
- Aktives Forum
- Einfache, etwas unübersichtliche Dokumentation

3.3.3 Selenium

Die Produkte aus der Selenium Tool Suite[26] fallen besonders durch ihre Vielseitigkeit auf: Neben der Unterstützung aller Betriebssysteme mit einer Version der JDK 1.5 oder höher, können Tests in allen JavaScript-kompatiblen Browser durchgeführt werden. Zusätzlich besteht eine große Auswahl an Programmiersprachen (C#, Java, Perl, PHP, Python und Ruby), um Tests zu implementieren.[27]

[24] JAMon (2013)
[25] Vgl. ebenda
[26] Selenium HQ (2013a)
[27] Vgl. ebenda

Das Paket von Selenium umfasst vier Komponenten, von denen an dieser Stelle zwei näher betrachtet werden sollen: Das Firefox Add-On „Selenium IDE“ stellt die einfachste Möglichkeit dar, einen Testfall zu erstellen, indem die Interaktion im Browser aufgezeichnet wird und anschließend abgespielt werden kann. Die IDE kann auch dazu genutzt werden, den aufgenommenen Code im Nachhinein zu bearbeiten. [28] Die Erweiterung der „Selenium IDE“ nennt sich „Selenium Remote Control“ und stellt zusätzliche Kontrollstrukturen und Funktionen, wie die meisten Programmiersprachen ausführen können, zur Verfügung.[29]

Die Produkte „Selenium Grid“, welches für die simultane, parallele Ausführung von Tests gedacht ist, sowie „Selenium WebDriver“, welches „Selenium Remote Control“ (Selenium RC) vereinfacht und dynamische Webseiten besser unterstützt, werden an dieser Stelle nicht betrachtet, da diese noch nicht vollends ausgereift sind. Eine ausführliche Reportingfunktion steht in keiner Version zur Verfügung.[30]

Stattdessen kann beim Einsatz von Selenium auf eine sehr ausführliche Dokumentation gezählt werden, die sämtliche Beispiele in allen sechs unterstützten Programmiersprachen vorstellt. Zusätzlich zu den weiteren Supportmöglichkeiten durch ein Forum, einen Chat und einen Bug Tracker, kann auch auf umfassenden kommerziellen Support von verschiedenen Anbietern zurückgegriffen werden. Die Aktivität der Community ist sehr hoch, sowohl in Hinblick auf Diskussionen im Forum oder Artikel im Blog, als auch bezüglich Softwareupdates. Für Selenium steht bereits ein Plug-In über die Plattform Nagios Exchange bereit, welches durch Anpassungen auch für die Neuentwicklung „Selenium WebDriver“ nutzbar gemacht werden kann. [31]

Zusammenfassung Selenium:

- Unterstützung von: C#, Java, Perl, PHP, Python und Ruby
- Module: IDE („Capture and Replay“-Modus) und RC (Erweiterung um Kontrollstrukturen usw.
- Keine Reportingfunktion
- Nagios-Plug-In vorhanden
- Sehr gute Dokumentation, vielfältige Supportmöglichkeiten

[28] Vgl. Selenium HQ (2013b)
[29] Vgl. Selenium HQ (2013c)
[30] Vgl. Whichtestingtool (2013a)
[31] Vgl. Nagios Exchange (2013)

3.3.4 Web Application Testing In .Net (WatiN)

Bei WatiN[32] handelt es sich um ein einfach zu bedienendes und stabiles Testing Framework für .Net.[33] Die Installation erfolgt sehr einfach über die Plug-In Bibliothek von Microsoft Visual Studio, welches detailliert von den Entwicklern dokumentiert wurde. Aufgrund dessen ist WatiN aber recht eingeschränkt nutzbar, denn es steht bisher nur für Microsoft Windows Betriebssysteme zur Verfügung, auch wenn an einer Weiterentwicklung beispielsweise für MacOS inoffiziell gearbeitet wird.[34] Auch die Wahl des Browsers muss bei WatiN gut bedacht sein, denn es werden nur Versionen des Internet Explorers und von Firefox unterstützt.

Alle gängigen HTML-Elemente werden von WatiN unterstützt, insbesondere auch Pop-Ups und Benutzerdialoge.[35] Dabei kann jedoch nicht auf eine Identifikation der Elemente einer Webseite zurückgegriffen werden, wie bei anderen Tools. Stattdessen wird von den Entwicklern auf die Verwendung verschiedener Entwickler-Toolbars verwiesen, die diesen Zweck erfüllen können.[36] Ähnlich sieht es mit einer Aufnahmefunktion aus, die nur als Erweiterung des Grundpakets von Dritten zur zusätzlichen Installation bereitsteht. Auf eine Reportingfunktion der Testfälle verzichtet WatiN vollständig.[37]

Support kann über eine recht umfangreiche, aber leicht veraltete Dokumentation erhalten werden. Ebenso werden eine E-Mailing-List und ein Bug Tracker eingesetzt. Ist dieser Support nicht ausreichend, kann kommerzielle Unterstützung eingesetzt werden.[38] Die Aktivitäten der Community fallen im Vergleich zu den Communities der anderen Tools etwas geringer aus, d.h. es werden zwar regelmäßig neue Versionen veröffentlicht, die Zeitintervalle sind aber größer im Vergleich zu anderer Software. Auch ein Plug-In für Nagios ist noch nicht vorhanden.

---

[32] WatiN (2013a)
[33] Vgl. Whichtestingtool (2013b)
[34] Vgl. ebenda
[35] Vgl. WatiN (2013a)
[36] Vgl. Whichtestingtool (2013b)
[37] Vgl. ebenda
[38] Vgl. WatiN (2013b)

Zusammenfassung WatiN:

- Benötigt .Net-Kenntnisse
- Nur auf Windows-Betriebssystemen lauffähig
- Beinhaltet gängige Funktionen
- Keine „Capture and Replay“-Funktion, kein Nagios-Plug-In
- Leicht veraltete Dokumentation, Supportmöglichkeiten nicht ausreichend

3.3.5 Web Application Testing in Ruby (Watir)

Inspiration für WatiN ist in Watir[39] zu sehen, sodass grundsätzliche Ähnlichkeiten erkennbar sind, wenn auch Watir für Testfälle in Ruby programmiert wurde und somit weniger eingeschränkt bezüglich der unterstützten Betriebssysteme (Windows, MacOS und Linux) ist.[40] Die eigentliche Version von Watir ist zwar nur auf Tests im Internet Explorer ausgelegt, durch den Watir WebDriver können jedoch auch Tests in Firefox, Chrome, Safari und Opera oder mit einer Browsersimulation durch HTMLUnit durchgeführt werden.[41]

Besonderen Wert legt Watir neben einer stabilen und flexiblen Umgebung auf einen einfachen zu lesenden und zu wartenden Quellcode der Testfälle. Dies soll durch den Einsatz von Ruby garantiert werden.[42] Die Durchführung des Tests soll dem Benutzer nachempfunden sein und abschließend durch einen Vergleich mit Soll-Werten abgerundet werden. Beschränkungen, in welcher Programmiersprache eine zu testende Webapplikation geschrieben sein muss, gibt es keine. Über eine Aufnahmefunktion verfügt Watir genauso wenig wie über eine Reportingfunktion (über eine mögliche Erweiterung gibt die Dokumentation Auskunft) oder ein Nagios-Plug-In.

Auffallend bei Watir ist die sehr gute und ausführliche Dokumentation, welche sowohl eine schrittweise Anleitung zur Installation, als auch ein Tutorial für die ersten Schritte beinhaltet. Für

[39] Watir (2013a)
[40] Vgl. WatiN (2013a)
[41] Vgl. Watir Webdriver (2013)
[42] Vgl. Watir (2013a)

Supportanfragen steht eine sehr aktive Community zur Seite, mit der in einem Forum, per E-Mailing-List oder per Chat kommuniziert werden kann.[43]

Zusammenfassung Watir:

- Benötigt Ruby -Kenntnisse
- Durchführung dem Benutzer nachempfunden, einfache Umsetzung
- Keine Reportingfunktion
- Keine „Capture and Replay“-Funktion, kein Nagios-Plug-In
- Sehr ausführliche Dokumentation, sehr aktive Community

3.3.6 WebInject

Bei der letzten Software, die im Rahmen dieser Arbeit vorgestellt werden soll, handelt es sich um das in Perl-geschriebene WebInject[44]. Zur Ausführung dieses Tools bedarf es lediglich einem Betriebssystem mit einem Perl-Interpreter, sodass alle gängigen Betriebssysteme, genauso wie alle gängigen Browser in Betracht kommen.[45]

Eine Installation und Konfiguration gestaltet sich bei WebInject etwas komplexer, da hierzu auch die Dokumentation etwas schwächer ausfällt. Der Funktionsumfang ist vergleichsweise geringer, dennoch ist eine Reportingfunktion mit Ausgabe in HTML oder XML bereits integriert. Für eine graphische Auswertung der Testergebnisse besteht die Möglichkeit, WebInject mit einer anderen Open Source Lösung zu erweitern. Die Verknüpfung mit dieser wird genauso wie das mitgelieferte Nagios-Plug-In vom Hersteller bereitgestellt. Durch Nutzung von XML als Programmiersprache für die Testfall-Erstellung ist diese recht intuitiv und einfach gestaltet.[46]

Die ausführliche Dokumentation beinhaltet einige Beispiele, Erklärungen dazu sind jedoch meist recht kurz gehalten. Unterstützung findet der Nutzer in einem Forum oder im Blog des Entwicklers, welcher auch per E-Mail zu kontaktieren ist. Ein kommerzieller Support ist nicht vorhanden. Das letzte Update der Software ist bereits älter (Januar 2006).

[43] Vgl. Watir (2013a)
[44] WebInject (2013)
[45] Vgl. ebenda
[46] Vgl. ebenda

Zusammenfassung WebInject:

- Benötigt XML-Kenntnisse
- Beinhaltet gängige Funktionen und Reporting (Ausgabe in HTML oder XML)
- Nagios-Plug-In vorhanden
- Keine „Capture and Replay“-Funktion
- ausführliche Dokumentation, kaum Aktivitäten in der Community

3.4 Zusammenfassung der Eigenschaften der Tools und Auswahl eines Tools

Nach der Vorstellung der einzelnen Tools, soll in tabellarischer Form ein Überblick, indem die wesentlichen Punkte zusammengefasst werden, geschaffen werden. Zudem finden im Folgenden eine Auswertung und ein Vergleich zwischen den Produkten statt, um das am besten geeignete Produkt herauszufinden.

3.4.1 Technische Aspekte

Zu Beginn werden die Produkte anhand ihrer technischen Merkmale verglichen. In der Kategorie Systeminteroperabilität sind zwischen fünf Produkten keine gravierenden Unterschiede erkennbar, nur das Tool WatiN fällt hier auf, da es nur für Windows-Betriebssysteme geeignet ist. Aufgrund der Serverausstattung des Unternehmens führt dies zu einer Abwertung des Produktes, da die Server auf dem Betriebssystem Linux, Distribution Suse Linux Enterprise Server (SLES) in der Version 11 betrieben werden. Meistens wird zudem eine JDK oder ein bestimmter Browser vorausgesetzt. Positiv auffallend ist hier WebInject, welches auf jedem Betriebssystem ausgeführt werden kann, welches einen Perl-Interpreter besitzt und keine weiteren Systemvoraussetzungen erfüllt sein müssen. Alle vorgestellten Produkte verfügen über eine API, mit Hilfe derer sie in verschiedene Programme, wie z.B. auch Monitoring-Tools integriert werden können. Für die Integration in Nagios ist zu beachten, dass die Testfälle zwei Rückgabewerte liefern müssen. Auf der einen Seite benötigt Nagios einen Zahlenwert aus der Menge 0, 1, 2, 3, welche für das Ergebnis des Tests stehen („OK“, „Warning“, „Critical“, „Unknown“) und einen String, welcher zusätzliche Informationen zur Durchführung liefert.

Zusätzlich dazu bietet WebInject von Entwicklerseite ein Plug-In inklusive einer detaillierten Konfigurationsanleitung für WebInject selbst, aber auch für Nagios. Auch Selenium bietet eine vorgefertigte Erweiterung für Nagios an, welche von der Community bereitgestellt wird und auf der Nagios Exchange Plattform heruntergeladen werden kann. Allerdings ist hier die Anleitung weniger ausführlich. Die vier anderen Tools, die bisher kein Nagios-Plug-In besitzen, entfallen nicht aus der Wertung, da die Möglichkeit besteht, mit einfachen Mitteln dieses Plug-In selbst zu programmieren, umfassende Literatur dazu, steht z.B. in „Nagios: System- und Netzwerkmonitoring“ bereit. Da die Umsetzung zeitintensiver und daher auch kostenintensiver ist, werden die Produkte Selenium und WebInject aufgewertet.

Die nachfolgende Tabelle fasst alle Merkmale noch einmal zusammen:

Produkt	Technische Aspekte (30%)		
	System-interoperabilität	Weitere System-voraussetzungen	Kompatibilität mit Nagios
Canoo Web Test	alle gängigen BS	JDK 5+, Ant 1.7.0+	Programmierung eines Plug-Ins möglich
JAMon	alle gängigen BS	JDK 1.4+	Programmierung eines Plug-Ins möglich
Selenium	alle gängigen BS	JDK 1.5+, IDE: Firefox 3+	Plug-In von Dritten vorhanden
WatiN	Windows	.Net/Visual Studio, Firefox 2.0+ / IE 6+	Programmierung eines Plug-Ins möglich
Watir	Windows, MacOS, Ubuntu Linux	IE 6+, Ruby	Programmierung eines Plug-Ins möglich
WebInject	alle gängigen BS	Perl Interpreter	Plug-In von Entwicklern vorhanden

Tabelle 1: Zusammenfassung der technischen Merkmale

Insgesamt stechen bei den technischen Merkmalen vor allem WebInject und an zweiter Stelle Selenium heraus, nicht nur wegen den bereits vorhandenen Plug-Ins, sondern auch wegen der breiten Einsatzmöglichkeiten. Die Produkte Canoo WebTest, JAMon und Watir werden als fast

gleich auf bewertet, die Software WatiN schneidet hier am schlechtesten ab. Die folgende Abbildung soll diese Wertung verdeutlichen:

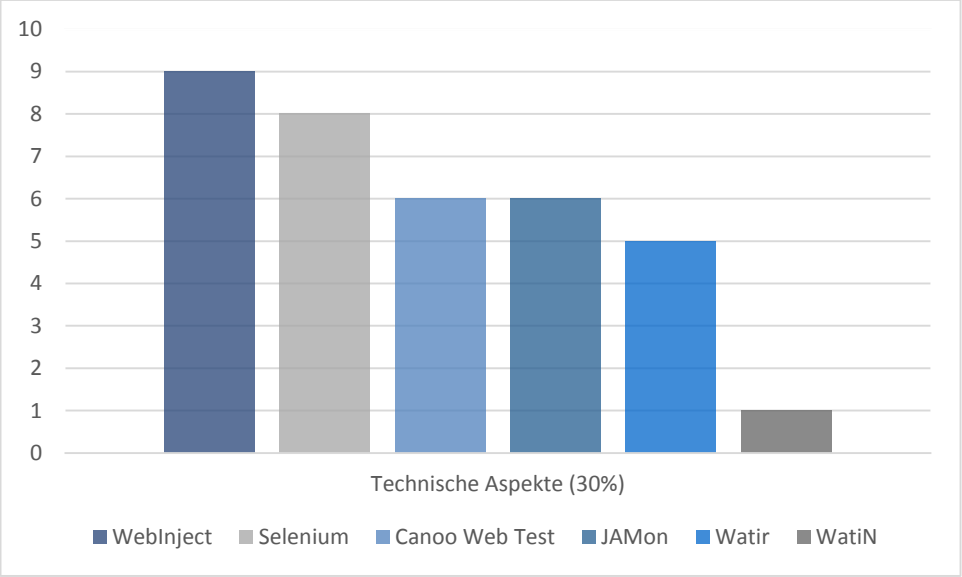


Abbildung 4: Ranking der Tools nach technischen Aspekten

3.4.2 Funktionale Aspekte

Im nächsten Schritt werden die funktionalen Aspekte der einzelnen Softwareprodukte gegeneinander abgewogen und verglichen. Bei der Bewertung der Installation und Konfiguration wurde betrachtet, wie die Installation abläuft, wie dies dokumentiert und umsetzbar ist. Da die Dokumentation von JAMon hierzu keinerlei Angaben und Hilfestellungen gibt, schneidet es in diesem Punkt am schlechtesten ab. Alle weiteren Produkte unterscheiden sich hier nur gering in der Komplexität, auch wenn es verschiedene Arten der Installation gibt. Anhand diverser Quellen, welche sich mit Tools zur Testautomatisierung auseinandergesetzt haben, wurde die Komplexität der Bedienbarkeit bewertet. Zudem wurde die Erklärung der ersten Schritte in der Dokumentation angeschaut und in das Ergebnis aufgenommen. Hierbei wurde JAMon auch aufgrund der knappen Beschreibungen als mittel eingeschätzt, genauso wie WatiN. Nur WebInject wurde hier als mittel bis schwierig eingestuft. Bei Selenium muss in dieser Kategorie zwischen der IDE und der RC-Version unterschieden werden. In der IDE ist es sehr einfach und selbsterklärend, wie mit der graphischen Oberfläche Testfälle aufgezeichnet und abgespielt

werden können. In Selenium RC ist die Durchführung komplexer, kann aber mit der IDE kombiniert werden, sodass dies schlussendlich den Schwierigkeitsgrad gering bis mittel ergibt.

Die Aufnahmefunktion ist ebenfalls ein entscheidender Unterschied zwischen Selenium und den anderen vorgestellten Tools, denn kein anderes beinhaltet diese Funktion im Grundpaket. Durch die Kombination von Selenium IDE und RC kann ein Testfall zunächst aufgenommen und anschließend um Kontrollstrukturen erweitert werden. Dies macht insgesamt die Testfallerstellung wesentlich einfacher als bei den Vergleichstools. Der weitere Funktionsumfang ist mit einer ausführlichen Reportingfunktion und der weiteren unterstützten Formate bei Canoo WebTest als sehr gut einzustufen. Unterdurchschnittliche Funktionen sind nur bei JAMon und WebInject aufgefallen. Selenium lässt sich je nach eingesetzter Programmiersprache um eine Reportingfunktion erweitern. Wird die Programmierung mit Java durchgeführt, kann durch den Einsatz von JUnit und JUnit-Report eine Historie über die Ergebnisse der Testfälle angelegt werden.

Um den Bereich abzuschließen, wurden noch die Programmiersprachen betrachtet, die von den Tools unterstützt werden und mit den Ressourcen des Unternehmens abgeglichen. Da Know-How in den Programmiersprachen XML, Java, Perl und PHP vorhanden ist, werden WatiN und Watir hier abgewertet, weil .Net und Ruby nicht zum aktuellen Wissensschatz gehören und eine Einarbeitung zusätzlichen Aufwand bedeuten würde. Zusammengefasst werden die funktionalen Merkmale in der folgenden Tabelle 2.

Insgesamt hat hier Selenium am besten abgeschnitten, nicht allein durch die sechs Programmiersprachen, die unterstützt werden, sondern auch durch den guten Funktionsumfang. Genau dieser ist auch ausschlaggebend dafür gewesen, dass Watir, WatiN und JAMon die hinteren Plätze belegen, wie die folgende Abbildung verdeutlicht.

Produkt	Funktionale Aspekte (40%)				
	Installation und Konfiguration	Komplexität	Erstellung von Test Cases	Funktions-umfang	unterstützte Programmier-sprachen
Canoo Web Test	sehr gute Installations-anweisungen, einfach	gering	einfache Syntax, keine "Capture and Replay" Funktion	Reporting-funktion vorhanden, Browser wird nur simuliert (Probleme mit JavaScript), auch für Tests von E-Mail-, Excel- und PDF-Inhalten	XML, Groovy
JAMon	keine Dokumentation	mittel	keine "Capture and Replay" Funktion	Performance Monitoring, keine Historie	Java
Selenium	**IDE:** sehr einfach als Browser Add-on, **RC:** Kommandozeile	gering - mittel	einfach; "Capture and Replay" möglich in IDE und Export nach RC	Reporting-funktion nicht vorhanden; RC erweitert IDE um Kontroll-strukturen	C#, Java, Perl, PHP, Python, Ruby
WatiN	sehr einfach, zwei Plug-Ins in Visual Studio nötig	mittel	"Capture and Replay" mit Erweiterung möglich	für alle HTML Elemente, auch Popups und Dialoge, keine besonderen Extras	.Net
Watir	Kommandozeile, Schritt-für-Schritt Anleitung	gering	keine "Capture and Replay" Funktion	für alle User-Interaktionen und Kontrolle der Ergebnisse	Ruby
WebInject	Kommandozeile	mittel-schwierig	kein "Capture and Replay", mit XML aber recht intuitiv	Funktions-umfang eingeschränkt, Reports in HTML and XML	XML

Tabelle 2: Zusammenfassung der funktionalen Merkmale

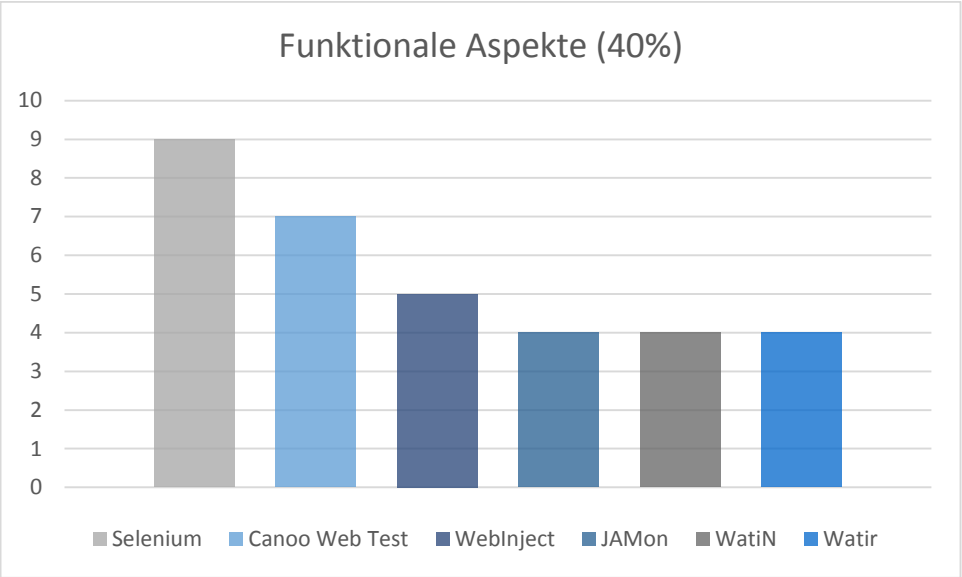


Abbildung 5: Ranking der Tools nach funktionalen Aspekten

3.4.3 Open Source Aspekte

Als letzter Punkt werden die Produkte nun anhand ihrer Open Source Merkmale eingeordnet. Die Lizenzen unterscheiden sich nicht wesentlich von einander, denn alle drei Lizenzen (Apache-Licence, BSD und GNU GPL) unterliegen der Free Software Vereinbarungen. Einziger Unterschied ist die Copyleft-Bestimmung der GNU GPL, welche eine Verbreitung mit einer anderen Lizenz als der ursprünglichen untersagt. Dies wird aber nicht als Anlass für eine bessere bzw. schlechtere Bewertung genommen, da es bei Testmanagement-Software unkritisch ist, ob die modifizierte Version unter derselben Lizenz steht.

Dokumentations- und Supportqualität sind besonders bei Selenium und Canoo WebTest überdurchschnittlich gut und ermöglichen einen leichten Einstieg. Auch Watir und WatiN sind in dieser Kategorie sehr gut, im Gegensatz zu JAMon, bei welchem wenig Support und nur eine einfache Dokumentation zur Verfügung stehen. Abschließender Punkt dieser Kategorie ist die Aktivität der Community, welche bei Canoo WebTest, Selenium und Watir sehr hoch ist, was nicht nur durch die Häufigkeit der Veröffentlichung neuer Versionen der Software widergespiegelt wird, sondern auch durch die Beteiligung im Forum. Vor allem WebInject schneidet in dieser Kategorie schlecht ab, da die aktuelle Downloaddatei über sechs Jahre alt ist. Die aktuelle Version von JAMon ist zwar neuer, doch die Aktivität im Forum ist deutlich geringer

als bei den Vergleichsprodukten. Insgesamt wird deshalb Selenium zusammen mit Watir als führende Software in diesem Bereich eingeordnet, vor Canoo WebTest und WatiN.

Um die Open Source Merkmale zusammenzufassen, findet sich im Anschluss die vollständige Übersichtstabelle:

Produkt	Open Source Aspekte (30%)			
	Lizenz	Dokumentation	Support	Aktivität der Community
Canoo Web Test	Apache-Lizenz 2.0	sehr gut strukturierte Dokumentation mit vielen Beispielen	E-Mailing Lists, kommerzieller Support möglich	hoch, letzte Änderung August 2012
JAMon	adaptierte BSD Lizenz	einfache Dokumentation, keine FAQs	Forum, E-Mail Support	mittel, letzte große Änderung Juli 2011
Selenium	Apache-Lizenz 2.0	sehr ausführliche und übersichtliche Dokumentation, Beispiele für verschieden Programmiersprachen wählbar	Forum, Chat, Bug Tracker, viele Anbieter von kommerziellem Support	sehr hoch, letzte Änderung Dezember 2012
WatiN	Apache-Lizenz 2.0	sehr ausführliche Dokumentation, jedoch weniger übersichtlich, viele Beispiele, einige FAQs	E-Mailing Lists, kommerzieller Support möglich	gering, letzte Änderung April 2011
Watir	BSD Lizenz	sehr ausführliches und über-sichtliches Wiki, FAQs, sehr gutes Tutorial	Forum, E-Mailing Lists, Chat	sehr hoch, aktive Diskussionen und Beteiligungen
WebInject	GNU GPL	ausführliche Dokumentation mit einigen Beispielen, wenige FAQs	Forum, E-Mail-Kontakt	sehr gering, letzte Änderung Januar 2006

Tabelle 3: Zusammenfassung der Open Source Merkmale

Insgesamt wird deshalb Selenium zusammen mit Watir als führende Software in diesem Bereich eingeordnet, vor Canoo WebTest und WatiN. Die genauen Bewertungen können der nachstehenden Abbildung entnommen werden.

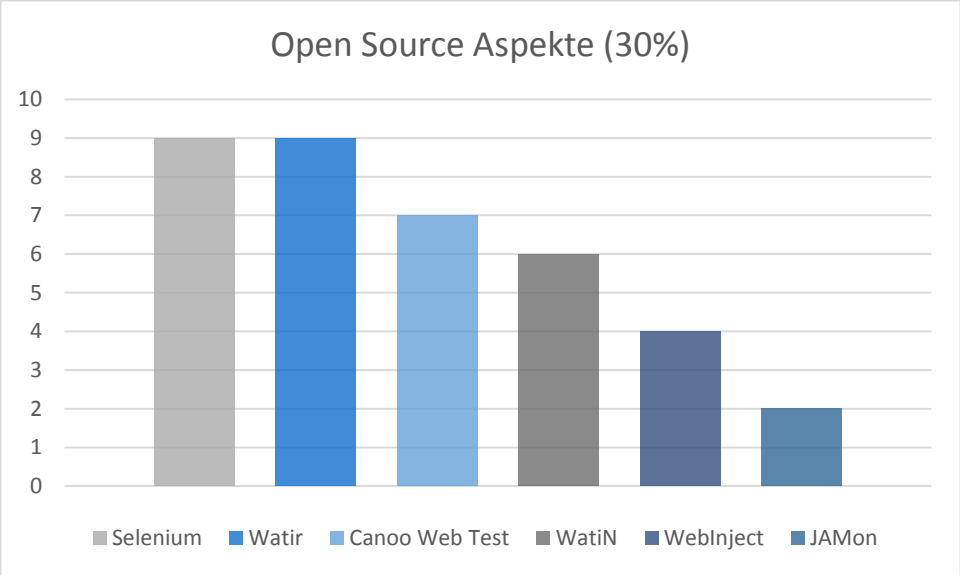


Abbildung 6:Ranking der Tools nach Open Source Aspekten

3.4.4 Auswahl

Die folgende Tabelle 4 bietet eine Zusammenfassung der Benotung zu den einzelnen Aspekten und die finale Gesamtbewertung, durch welche die Wahl auf Selenium gefallen ist, da hier über alle Kategorien hinweg, die beste Leistung zu erwarten ist.

Produkt	Technische Aspekte (30%)	Funktionale Aspekte (40%)	Open Source Aspekte (30%)	Gesamt-bewertung
Selenium	8	**9**	**9**	**8,7**
Canoo Web Test	6	7	7	**6,7**
WebInject	**9**	5	4	**5,9**
Watir	5	4	9	**5,8**
JAMon	6	4	2	**4,0**
WatiN	1	4	6	**3,7**

Tabelle 4: Bewertung der Produkte

Das E2E-Monitoring in dieser Arbeit soll also aus der Kombination vom System- und Netzwerkmonitoringtools Nagios und der Testautomatisierungssoftware Selenium umgesetzt werden. Wie die folgende Abbildung 7 zeigt, stellt Selenium im Schichtenmodell die Funktionsfähigkeit die Anwendungsschicht sicher.

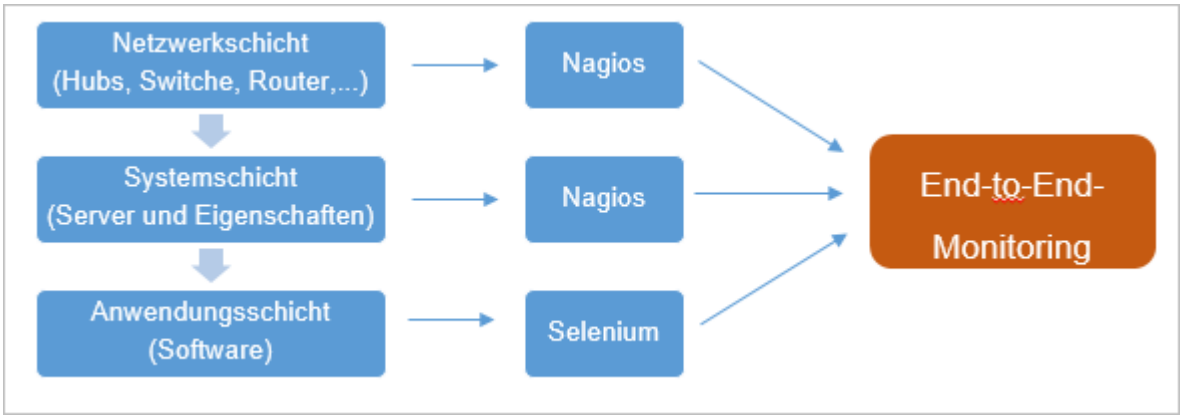


Abbildung 7: Schichten des E2E-Monitorings

4 Dokumentation zur Umsetzung eines End-to-End-Monitorings

Nach der Herleitung des Konzeptes des E2E-Monitorings soll nun die Erstellung eines Prototyps beschrieben werden. Zunächst werden die Anforderungen seitens des Unternehmens vorgestellt, bevor dann auf die Architektur und die Kommunikation der einzelnen Komponenten der technischen Umsetzung eingegangen wird.

4.1 Situation und Anforderungen des Unternehmens

Im Unternehmensumfeld wird Nagios als System- und Netzwerkmonitoringtool eingesetzt. Sowohl auf dem Nagios-Server, als auch auf dem Server, auf dem die zu betrachtende Webanwendung „DokuWiki“ betrieben wird, befindet sich ein Linux-Betriebssystem in der Distribution Suse Linux Enterprise Server (SLES) 11. Im Rahmen dieses Monitorings wird die Webanwendung „DokuWiki“ mit Hilfe des Nagios Plug-Ins „check_http“ überwacht.

Das Plug-In versucht während der Ausführung mit dem Host von „DokuWiki“ eine HTTP-Verbindung aufzubauen, um so festzustellen, ob die Anwendung erreichbar ist. Ist der Verbindungsaufbau erfolgreich, wird in Nagios der Status „OK“ angezeigt, während Timeouts und Verbindungsfehler zum Status „CRITICAL“ führen. Eine „Warning“ wird im Falle eines anderen Fehlers ausgegeben. Diese Umsetzung kann jedoch noch nicht als E2E-Monitoring betrachtet werden, da lediglich eine HTTP-Verbindung zum Server aufgebaut wird, aber nicht getestet wird, ob die Anwendung tatsächlich beim Endanwender ordnungsgemäß funktioniert und wie Last- und Performancewerte aussehen.

Aus diesem Grund wurde die Software Selenium ausgewählt, um mit dem „check_selenium“-Plug-In das Monitoring der Anwendung zu verbessern. Die dazu benötigten Kenntnisse sind im Unternehmen vorhanden, dies bedeutet, dass sowohl einerseits die Konfiguration des Nagios-Plug-Ins keine Probleme bereiten sollte und andererseits die Nutzung von Selenium sehr intuitiv und mit verschiedenen Programmiersprachen möglich ist. Das Plug-In selbst wird als Java-Klasse bereitgestellt, auch hierfür sind Ressourcen im Unternehmen vorhanden.

4.2 Voraussetzungen

Angenommen wird, dass Nagios bereits konfiguriert ist und sich im Einsatz befindet. Auf der Nagios-Plattform „Exchange“, auf der alle verfügbaren Plug-Ins zum Download bereit stehen, kann dann das Plug-In heruntergeladen werden. Zusätzlich liegt dem Downloadpaket die JUnit Library bei, die für die Ausführung des Plug-Ins benötigt wird. Zur Erstellung der Testfälle wird bei der Umsetzung des Prototyps die Selenium IDE verwendet, um einen einfachen Login-Prozess aufzuzeichnen. Selenium RC wird für die Erstellung der Testfall-Datei nach der Aufzeichnung durch die IDE benötigt, ebenso für deren Ausführung. Der Test kann nun auf dem Nagios-Server selbst oder mit Hilfe eines entfernten Plug-In-Aufrufs auf entfernten Computern durchgeführt werden. Das Unternehmen kann so testen, ob DokuWiki auch von einer anderen Niederlassung aus erreichbar ist. Für den Prototyp wird der Test auf demselben Server ausgeführt, in der nachfolgenden Dokumentation soll aber auch auf das entfernte Testen auf einem anderen Computer vorgestellt werden.

Voraussetzung für einen entfernten Plug-In-Aufruf, ist die Verwendung des Nagios Remote Plugin Executor (NRPE). Auf dem Nagios-Server wird der „check_nrpe“ eingesetzt, welche die Kommunikation mit dem auf dem entfernten Host installierten NRPE übernimmt, welches wiederum das Plug-In „check_selenium“ ausführen kann. Die folgenden Abbildungen sollen die unterschiedliche Funktionsweise noch einmal grafisch darstellen.

Soll der Testfall auf dem Nagios-Server ausgeführt werden, so kann Nagios direkt das Plug-In aufrufen und ausführen, wie Abbildung 8 verdeutlicht.

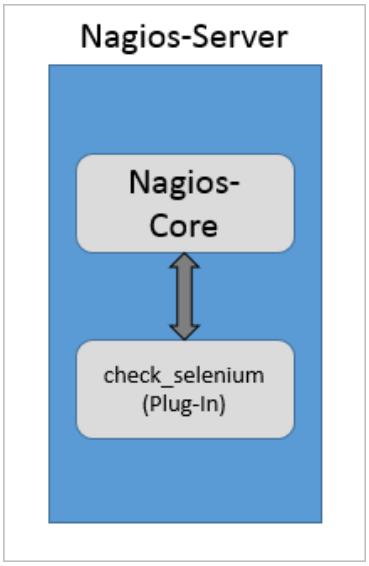


Abbildung 8: Lokale Ausführung des "check_selenium"

Im Gegensatz dazu, bildet die untere Abbildung 9 den entfernten Aufruf ab: Nagios ruft statt dem „check_selenium“ Plug-In, das „check_nrpe“ Plug-In auf, welches sich mit dem NRPE des entfernten Computers verbindet. Das NRPE kann dann das „check_selenium“ ausführen und liefert seinerseits auch über den NRPE die Testergebnisse zurück an Nagios.

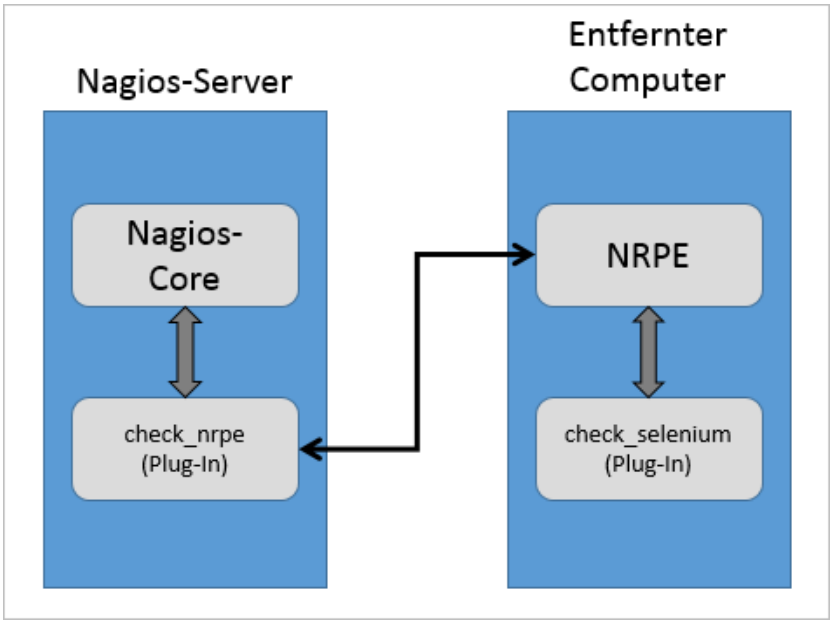


Abbildung 9: Entfernter Aufruf von "check_selenium"

4.3 Funktionsweise des Plug-Ins „check_selenium“

Das Plug-In besteht im Wesentlichen aus zwei Komponenten, die für die Integration in Nagios benötigt werden. Zum einen besteht es aus einem ausführbaren Skript und zum anderen aus einer Java-Klasse, in der die Testausführung und die Rückgabewerte für Nagios definiert sind.

Zunächst kontaktiert Nagios den zuständigen Host, in dem Fall der Computer, auf welchem der Testfall durchgeführt werden sol. Der zugehörige Service wird anschließend ausgewählt und aufgerufen. In einem Service in Nagios ist das auszuführende Kommando definiert, welches wiederum den Pfad für das Skript und die Definition der Parameter enthält. Das Skript besteht aus verschiedenen Pfadangaben, u.a. zur installierten Java-Version oder zum Selenium Server. Am wichtigsten ist jedoch die Angabe des Pfades zur Java-Klasse „CallSeleniumTest.java“, deren Quellcode im Anhang gefunden werden kann. Diese Klasse dient dazu, mit der JUnit Library den Testfall auszuführen und das Ergebnis der Ausführung zu verarbeiten. Dies

bedeutet konkret, dass das Ergebnis in eine Form gebracht werden muss, die für Nagios verständlich ist. Erreicht wird dies dadurch, dass eine der Zahlen 0 („OK“), 1 („Warning“), 2 („Critical“) oder 3 („Unknown“) zurückgegeben wird. Zusätzlich muss eine Textzeile ausgegeben, die im Erfolgsfall den Testnamen und die Ausführungszeit in Millisekunden enthält. Kann der Test nicht vollständig durchlaufen oder tritt ein Fehler auf, wird neben dem entsprechenden Zahlencode eine kurze Fehlermeldung im Textfeld von Nagios ausgegeben. Zur Veranschaulichung soll die folgende Abbildung 10 dienen.

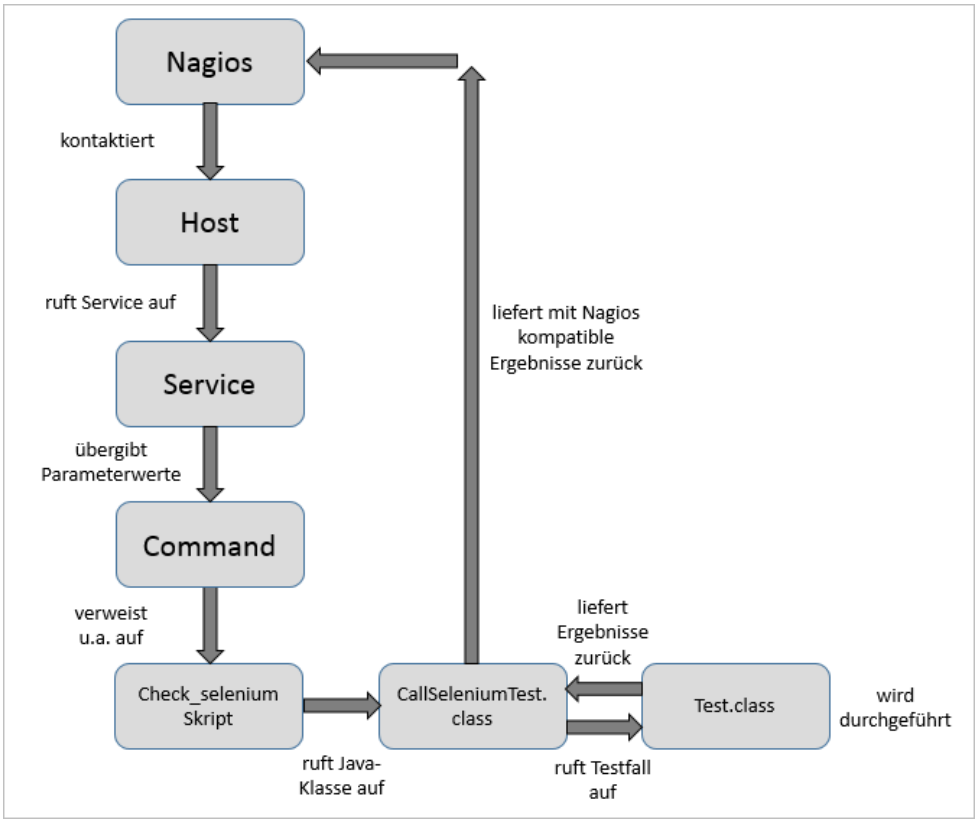


Abbildung 10: Ausführung eines Tests

4.4 Umsetzung des Prototyps

4.4.1 Eingesetzte Software

Die Dokumentation der Umsetzung bezieht sich auf folgende Versionen der oben genannten Software:

Software	Version	Release Datum
Ubuntu	12.10	23.04.2012
Apache Webserver	2.2.22	06.11.2012
Firefox	18.0.1	18.01.2013
Nagios Core	3.4.4	30.11.2012
DokuWiki	Adora Belle	13.10.2012
Selenium IDE	1.10.0	20.12.2012
Selenium Server (ehemals Selenium RC)	2.28.0	11.12.2012
JUnit Library	4.8.1	08.12.2009
Plug-In „check_selenium“	N/A	16.11.2012
Commons CLI	1.2	16.03.2009

Tabelle 5: Eingesetzte Software-Versionen

Der Prototyp basiert auf einem Linux-Betriebssystem in der Ubuntu Distribution 12.10. Hier werden zunächst ein Webserver (Apache Webserver) und ein Browser (Firefox) benötigt, die für die spätere Benutzeroberfläche in Nagios und für die Ausführung von DokuWiki genutzt werden. Die Monitoringumgebung, in die das E2E-Monitoring integriert werden soll, besteht aus der aktuellsten Version von Nagios Core, welche durch das Plug-In „check_selenium“ erweitert wird.

Für die vereinfachte Testfall-Erstellung wird das Firefox-Add-On „Selenium IDE“ verwendet, welches die „Capture and Replay“-Funktion von Selenium beinhaltet. Um einen erstellten Testfall als Java-Datei zu exportieren, müssen zusätzlich die JUnit Library und der Selenium Server vorhanden sein. Letzterer wird ebenfalls zum kompilieren der Java-Datei genutzt.

Nach der Installation und dem Herunterladen des Download-Pakets, müssen die beinhalteten Dateien entpackt und in die Ordnerstruktur von Nagios integriert werden. Als API wird auf das Commons CLI von Apache gesetzt. Dieses Paket und das Paket der JUnit Library werden direkt mit dem Plug-In-Download heruntergeladen und sind deshalb nicht die neusten Versionen. Die Skript-Datei muss im Nagios Ordner „libexec“ abgespeichert werden, die Java-Klasse des Plug-

Ins hingegen bleibt im bisherigen Pfad des Paketes. Außerdem müssen im selben Ordner die Verzeichnisse /com/example/tests angelegt werden, indem die kompilierten Java-Testfälle standardmäßig abgelegt werden.

Die bereits erwähnte Skript-Datei muss als letztes noch an die eigene Ordnerstruktur angepasst werden, indem die enthaltenen Pfade eingetragen werden, wie im unten stehenden Code gezeigt wird:

```
JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
DIR=/usr/local/nagios/libexec

CLASSPATH=${DIR}/selenium-java-client-driver.jar:${DIR}/junit-
4.8.1.jar:${DIR}/commons-cli-1.2.jar:${DIR}/com/example/tests/.:${DIR}

${JAVA_HOME}/bin/java -cp ${CLASSPATH} info.devopsabyss.CallSeleniumTest $@
```

Nach dem Setzen der Java-Pfade, werden die Pfade vom Selenium Server, von der JUnit Library, der API und dem Testverzeichnis angegeben, die im Beispielfall im selben Ordner abgelegt wurden. Als letztes wird in der Datei der Speicherort der Java-Klasse des Plug-Ins angegeben.

4.4.2 Testfall erstellen

Als Testfall können beliebige Szenarien ausgedacht und umgesetzt werden, um die Erreichbarkeit und Funktionalität von DokuWiki sicherzustellen. Um das Beispiel einfach zu halten, soll der Testfall die Startseite von DokuWiki, danach das Login-Fenster aufgerufen werden und dort einen Benutzernamen und das zugehörige Passwort eintragen. Anschließend soll die Anmeldung und schließlich der Logout-Vorgang durchgeführt werden.

In der folgenden Abbildung wird die Oberfläche der Selenium IDE gezeigt, in der durch Klick auf den roten Startbutton, die Aufnahme gestartet und beendet werden kann. Darunter werden die genutzten Befehle angezeigt und zur manuellen Bearbeitung bereitgestellt.

Das Beispiel liefert folgenden Code für die einzelnen Testschritte:

```
selenium.open("http://localhost/dokuwiki/doku.php?id=start");
selenium.click("link=Login");
selenium.waitForPageToLoad("30000");
selenium.type("id=focus__this", "nagios");
selenium.type("p", "nagios");
selenium.click("css=fieldset > input.button");
```

```
        selenium.waitForPageToLoad("30000");
        selenium.click("link=Logout");
        selenium.waitForPageToLoad("30000");
}
```

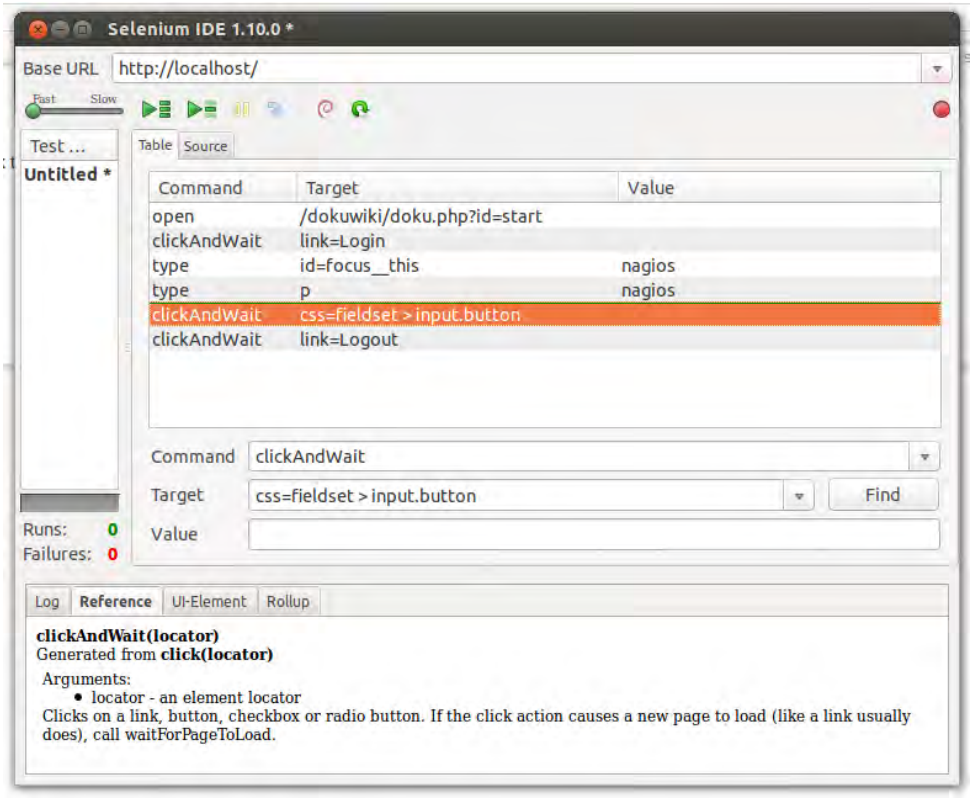


Abbildung 11: Ansicht der Selenium IDE

Zunächst öffnet Selenium die DokuWiki Startseite und klickt den Link zur Login-Seite an. Danach soll gewartet werden, bist die Seite geladen ist, bevor das Feld mit der ID „focus__this“ den Benutzernamen, im Beispiel lautet dieser „nagios“ als Wert zugewiesen bekommt. Im Feld p darunter soll das Passwort, welches ebenfalls „nagios“ lautet eingegeben werden (dies kann aufgrund der Konfiguration von DokuWiki nur mit dem Namen und nicht mit der ID angesprochen werden). Nachdem der Inputbutton gedrückt wurde, wartet Selenium erneut das Laden der Seite im Browser ab. In der neuen Seite wird dann der Link zum Logout betätigt und abgewartet, ob die Logout-Seite komplett geladen wird, bevor der Test beendet wird.

Dieser Code muss dann mit dem Selenium Server und unter Verwendung der JUnit Library als Java-Datei exportiert werden (File/Export/JUnit 4 Remote Control). Zum Starten des Selenium Servers kann folgender Befehl in der Kommandozeile genutzt werden:

java -jar /*Pfad*/ selenium-server-standalone-2.28.0.jar.

Der daraus resultierende Java-Quellcode (check_login.java) muss dem Beispiel aus der Anleitung zum Plug-In entsprechend angepasst werden (besonders die Schreibweise des Methodennamen muss beachtet werden) und sieht wie folgt aus:

```
package com.example.tests;

import com.thoughtworks.selenium.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.*;
import java.util.regex.Pattern;

public class check_login extends SeleneseTestCase{
//    private Selenium selenium;

      @Before
      public void setUp() throws Exception {
            setUp("http://localhost/", "*firefox");
      }

@Test
public void testcheck_login() throws Exception {
      selenium.open("http://localhost/dokuwiki/doku.php?id=start");
      selenium.click("link=Login");
      selenium.waitForPageToLoad("30000");
      selenium.type("id=focus__this", "nagios");
      selenium.type("p", "nagios");
      selenium.click("css=fieldset > input.button");
      selenium.waitForPageToLoad("30000");
      selenium.click("link=Logout");
      selenium.waitForPageToLoad("30000");
}

      @After
      public void tearDown() throws Exception {
            selenium.stop();
      }
}
```

Die Anpassung beinhaltet im Wesentlichen die Vererbung von der Klasse SeleneseTestCase, sowie eine angepasst setUp-Methode. Die Methoden testcheck_login() und tearDown() bleiben unverändert. Im nächsten Schritt wird die bearbeitete Datei mit dem Selenium Server kompiliert werden. Dazu wird der folgende Befehl in die Kommandozeile eingegeben:

javac -cp ./selenium-server-standalone-2.28.0.jar */Pfad/Dateiname*

und die kompilierte Datei check_login.class im Nagios-Verzeichnis libexec/com/example/tests abgespeichert.

4.4.3 Ausführung des Testfalles

Um einen Testfall in Nagios auszuführen, muss ein Kommando angelegt werden, welches auf das zugehörige Skript verweist. Angelegt wird das Kommando in der command.cfg Datei von Nagios. Für den Prototyp und das „check_selenium“-Plug-In bedeutet dies:

```
define command{
  command_name    local_check_selenium
  command_line    $USER1$/check_selenium -c $ARG1$ -b $ARG2$ -t $ARG3$
}
```

Mit Hilfe des Namens kann ein Kommando eindeutig identifiziert werden. Die Command Line dient wiederum dazu, das Skript und die zugehörigen Parameter zu benennen, die für den Ablauf notwendig sind. Im Beispiel soll also unter dem Namen „local_check_selenium“ das Skript „check_selenium“ mit dem Parameter c, der für den Speicherort des Skriptes steht, den Parametern b und t, für die Startseite des Tests und den Browser, in dem der Test ablaufen soll.

Dieses Kommando wird im Service „DokuWiki“, welches lokal über den Localhost ausgeführt werden soll, implementiert und mit den Parameterwerten aufgerufen.

```
define service{
  use                     local-service
  host_name               localhost
  service_description     DokuWiki
  check_command           local_check_selenium!com.example.tests.check_login!
                          http://www.google.com!*firefox
  normale_check_interval  5
  retry_check_interval    1
}
```

Nach der erfolgreichen Konfiguration des Kommandos und des Services, müssen der Selenium- und der Nagios-Server neugestartet werden, bevor der Service in der Nagios-Übersicht angezeigt wird.

Der eingebundene Service „DokuWiki“ wird in der gewohnten Übersicht unter dem Host „localhost“ angezeigt, wie die Abbildung 12 zeigt. Bei den zusätzlichen Statusinformationen werden der zugehörige Testfall und dessen Pfad genannt.

Host	Service	Status	Last Check	Duration
localhost	Current Load	OK	01-22-2013 10:25:10	2d 12h 54m 56s
	Current Users	OK	01-22-2013 10:25:43	4d 9h 37m 1s
	DokuWiki	OK	01-22-2013 10:27:32	0d 0h 0m 23s
	HTTP	OK	01-22-2013 10:26:17	4d 9h 36m 24s

Host	Service	Attempt	Status Information
localhost	Current Load	1/4	OK - load average: 1.23, 0.75, 0.42
	Current Users	1/4	USERS OK - 3 users currently logged in
	DokuWiki	1/4	OK - com.example.tests.check_login Test passed
	HTTP	1/4	HTTP OK: HTTP/1.1 200 OK - 452 bytes in 0.001 second response time

Abbildung 12: Nagios-Übersicht mit dem Service "DokuWiki"

Für weitere Informationen kann, wie in Nagios üblich, der Servicename angeklickt werden. Hier werden nun Details, wie z.B. Ausführungszeit unter dem Punkt Performance Data oder in der Zeile „Next Scheduled Check“ der nächste Testzeitpunkt, angezeigt (siehe Abbildung 13).

Service State Information

Current Status:	OK (for 0d 0h 0m 59s)
Status Information:	OK - com.example.tests.check_login Test passed
Performance Data:	ExecTime=3470ms;;;;
Current Attempt:	1/4 (HARD state)
Last Check Time:	01-22-2013 10:27:32
Check Type:	ACTIVE
Check Latency / Duration:	0.145 / 14.772 seconds
Next Scheduled Check:	01-22-2013 10:32:32
Last State Change:	01-22-2013 10:27:32
Last Notification:	N/A (notification 0)
Is This Service Flapping?	NO (0.00% state change)
In Scheduled Downtime?	NO
Last Update:	01-22-2013 10:28:29 (0d 0h 0m 2s ago)

Active Checks:	ENABLED
Passive Checks:	ENABLED
Obsessing:	ENABLED
Notifications:	ENABLED
Event Handler:	ENABLED
Flap Detection:	ENABLED

Abbildung 13: Detailansicht des "DokuWiki"-Services

5 Fazit

Ziel dieser Arbeit war es basierend auf dem System- und Netzwerkmonitoring-Tool Nagios einen E2E-Monitoring-Ansatz zu erarbeiten, mit dessen Hilfe Webanwendungen überwacht werden können. Da Nagios Core nur zwei der drei Schichten des E2E-Monitorings abdeckt, wurde nach einer Möglichkeit gesucht, die fehlende Schicht der Anwendungen abzudecken. Mit Hilfe einer Testautomatisierungs-Software soll diese Lücke geschlossen werden, da diese den Endnutzer simulieren kann und somit die Erreichbarkeit, Performance und Funktionalität der Anwendung testen kann.

Um die bestgeeignete Lösung zu finden, wurden sechs Produkte ausgewählt, die in einer Marktübersicht näher beleuchtet werden sollten. Kriterien aus den drei Kategorien technische, funktionale und Open Source Merkmale wurde dazu betrachtet und die Produkte anhand dieser verglichen. Dabei wurde auch schon auf die Anforderungen des Unternehmens eingegangen, sodass die Gewichtung der Kriterien und die Bewertung der Produkte genau betrachtet und ggf. individuell angepasst werden sollten.

Im Vergleich zwischen den Tools hat sich, nach Meinung der Autoren, Selenium herausgestellt, welches für die Umsetzung des Prototyps verwendet werden sollte. Dieser basiert auf Nagios und benutzt das Plug-In „check_selenium“, um in Selenium erstellte Testfälle in das Monitoring durch Nagios zu integrieren und somit den E2E-Monitoringansatz umzusetzen. Die Beschreibung der Umsetzung im Prototyp war ebenfalls der Bestandteil dieser Arbeit.

Abschließend soll diese und der Prototyp selber noch einmal kritisch betrachtet werden. Auf der einen Seite hatte die Umsetzung mit Hilfe des Plug-Ins den großen Vorteil, dass keine eigene Schnittstellte zwischen den Anwendungen programmiert werden musste. Andererseits brachte die Umsetzung einige Schwierigkeiten mit sich, da die Installations- und Konfigurationsanleitung zum Plug-In sehr knapp gehalten ist und so einige Fragen offen blieben. Zudem beruft sich das Plug-In auf teilweise veraltete Software, sodass die Umsetzung mit den neuen Software-Versionen mehr Zeit als erwartet in Anspruch genommen hat.

Ein Positiver Aspekt der Umsetzung in dieser Form ist aber die leichte Anpassbarkeit und die damit verbundene Flexibilität. Sind Nagios, das Plug-In und Selenium erst einmal eingerichtet und aufeinander abgestimmt, ist es sehr einfach möglich beliebige Webanwendungen in das E2E-Monitoring einzubinden. Hierzu muss lediglich ein neuer Testfall aufgezeichnet und ein neuer Service in Nagios definiert werden. Die üblichen Nagios-Funktionalitäten, wie z.B. Häufigkeiten der Tests oder Einstellen der Schwellenwerte runden das Bild ab. Zudem können

auch entfernte Tests, wie in Nagios üblich, mit Hilfe des Nagios Remote Plugin Executor ausgeführt werden, sodass kein neuer Aufwand entsteht, falls das NRPE bereits in einem anderen Zusammenhang zum Einsatz kommt.

Anhang

Quelltext: CallSeleniumTest.java

```
package info.devopsabyss;

/*
* This is a nagios plugin to integrate Selenium Test Cases into Nagios.
* Copyright (C) 2010 Christian Zunker (devops.abyss@googlemail.com)
*
* This program is free software; you can redistribute it and/or
* modify it under the terms of the GNU General Public License
* as published by the Free Software Foundation; either version 2
* of the License, or (at your option) any later version.
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program; if not, write to the Free Software
* Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301,
* USA.
*/

import org.apache.commons.cli.BasicParser;
import org.apache.commons.cli.CommandLine;
import org.apache.commons.cli.CommandLineParser;
import org.apache.commons.cli.HelpFormatter;
import org.apache.commons.cli.Option;
import org.apache.commons.cli.Options;
import org.apache.commons.cli.ParseException;
import org.apache.commons.cli.UnrecognizedOptionException;
import org.junit.runner.JUnitCore;
import org.junit.runner.Result;

public class CallSeleniumTest {

private final int NAGIOS_RC_OK = 0;
private final int NAGIOS_RC_WARNING = 1;
private final int NAGIOS_RC_CRITICAL = 2;
private final int NAGIOS_RC_UNKNOWN = 3;

private final String NAGIOS_TEXT_OK = "OK";
private final String NAGIOS_TEXT_WARNING = "WARNING";
private final String NAGIOS_TEXT_CRITICAL = "CRITICAL";
private final String NAGIOS_TEXT_UNKNOWN = "UNKNOWN";
```

```
private Options options = null;

//TODO: compile java files only when no class file found. this way the user
does not have to compile the sources.
// what is when i get compile errors? => return NAGIOS_UNKNOWN?

private Result runJUnitTest(String className) throws ClassNotFoundException {
	Class<?> seleniumTestClass = Class.forName(className);
	return new JUnitCore().run(seleniumTestClass);
}

public static void main(String[] args) throws Exception {
	// TODO: Selenium Server host and port as parameters

	CallSeleniumTest seTest = new CallSeleniumTest();

	Option optionclass = new Option("c", "class", true, "full classname of
testcase (required) e.g. \"com.example.tests.GoogleSeleniumTestCase\"");

	//optiontype.setRequired(true);
	Option optionverbose = new Option("v", "verbose", false, "show a lot of
information (useful in case of problems)");

	Option optionhelp = new Option("h", "help", false, "show this help
screen");

	Option optionNagios3 = new Option("3", "nagios3", false, "in case of a
test failure, print a multiline message in nagios 3 format");

	seTest.options = new Options();
	seTest.options.addOption(optionclass);
	seTest.options.addOption(optionverbose);
	seTest.options.addOption(optionhelp);
	seTest.options.addOption(optionNagios3);

	CommandLineParser parser = new BasicParser();
	CommandLine cmd = null;

	// TODO: verify baseURL
	// TODO: is there a possibility to verify classname?

	String output = seTest.NAGIOS_TEXT_UNKNOWN + " - Upps |";
	int nagios_rc = seTest.NAGIOS_RC_UNKNOWN;

	try {
		cmd = parser.parse(seTest.options, args);
		// has to be checked manually, otherwise you can't access the help
		//message without specifying correct parameters
		if (cmd.hasOption("h") || cmd.getOptionValue("c") == null) {
```

```
                usage(seTest.options);
                System.exit(nagios_rc);
            }

            Result result = seTest.runJUnitTest(cmd.getOptionValue("c"));
            if (result.wasSuccessful()) {
                output = seTest.NAGIOS_TEXT_OK + " - " +
cmd.getOptionValue("c")+" Tests passed | ExecTime="+result.getRunTime()+"ms";
                nagios_rc = seTest.NAGIOS_RC_OK;
            }
            else {
                String failureMessage = result.getFailures().toString();
                output = seTest.NAGIOS_TEXT_CRITICAL+" -
"+cmd.getOptionValue("c");
                if (cmd.hasOption("3")) {
                    output += " Test Failures | ExecTime="
+result.getRunTime()+"ms\n"+ failureMessage;
                }
                else {
                    output += " Test Failures: "+
withoutNewlines(failureMessage) + " | ExecTime=" + result.getRunTime() + "ms";
                }
                nagios_rc = seTest.NAGIOS_RC_CRITICAL;
            }
        }
        catch (UnrecognizedOptionException ex) {
            output = seTest.NAGIOS_TEXT_UNKNOWN + " - " + "Parameter problems:
" + messageWithoutNewlines(ex) + " |";
            nagios_rc = seTest.NAGIOS_RC_UNKNOWN;
            usage(seTest.options);
        }
        catch (ParseException ex) {
            output = seTest.NAGIOS_TEXT_UNKNOWN + " - " + "Parameter problems:
" + messageWithoutNewlines(ex) + " |";
            nagios_rc = seTest.NAGIOS_RC_UNKNOWN;
            usage(seTest.options);
        }
        catch (NoClassDefFoundError ex) {
            printStackTraceWhenVerbose(cmd, ex);
            output = seTest.NAGIOS_TEXT_UNKNOWN + " - " +
cmd.getOptionValue("c") + ": " + messageWithoutNewlines(ex) + " |";
            nagios_rc = seTest.NAGIOS_RC_UNKNOWN;
        }
        catch (ClassNotFoundException ex) {
            output = seTest.NAGIOS_TEXT_UNKNOWN + " - " +
cmd.getOptionValue("c") + ": Testcase class " + messageWithoutNewlines(ex) + "
not found! |";
            nagios_rc = seTest.NAGIOS_RC_UNKNOWN;
        }
        catch (Exception ex) {
```

```
            printStackTraceWhenVerbose(cmd, ex);
            output = seTest.NAGIOS_TEXT_CRITICAL + " - " +
cmd.getOptionValue("c") + ": " + messageWithoutNewlines(ex) + " |";
            nagios_rc = seTest.NAGIOS_RC_CRITICAL;
      }
      finally {
            System.out.println(output);
            System.exit(nagios_rc);
      }
}

private static String messageWithoutNewlines(final Throwable ex) {
      return withoutNewlines(ex.getMessage());
}

private static String withoutNewlines(final String message) {
      return message.replaceAll("\n", " ")
      .replaceAll(" ", " ")
      .replaceAll(" ", " ")
      .replaceAll(" ", " ");
}

private static void printStackTraceWhenVerbose(final CommandLine cmd, final
Throwable ex) {
      if (cmd.hasOption("v")) {
            ex.printStackTrace();
      }
}

private static void usage(Options options) {
      HelpFormatter formatter = new HelpFormatter();
      formatter.printHelp("check_selenium", options);
      System.out.println("");
      System.out.println("This version of check_selenium was tested with:");
      System.out.println(" - selenium server 2.20.0");
      System.out.println(" - selenium ide 1.7.2");
      System.out.println(" - test case exported as JUnit 4 (Webdriver)");
      System.out.println("");
      System.out.println("Some example calls:");
      System.out.println(" ./check_selenium -c
\"com.example.tests.GoogleSeleniumWebdriverTestCase\"");
      System.out.println(" ./check_selenium --class
\"com.example.tests.GoogleSeleniumWebdriverTestCase\"");
}
}
```

Quellenverzeichnisse

Literaturverzeichnis

Bommer, S. / Spindler, M. / Barr, V. (2008)	Software-Wartung: Grundlagen, Management und Wartungstechniken, o.O.: Dpunkt.Verlag GmbH
DHBW (2012)	Projekt: Open Source Monitoring von Anwendungen, November, Stuttgart
Fleischer, D. et al (2012)	Open Source Seminar Paper: Evaluation of Open Source Tools for Test Management and Test Automation, Stuttgart: DHBW Stuttgart
Jonassen, A.M. (2008)	Guide to Advanced Software Testing, o.O.: Artech House
Mitnacht, S. (2006)	Netzwerkmonitoring in einer dynamischen Umgebung, Koblenz: Universität Koblenz Fachbereich Informatik

Verzeichnis der Internet - Quellen

Android (2013)	Android Community, http://source.android.com/community/index.html, letzter Abruf: 03.01.2013
Canoo (2013a)	Canoo WebTest, http://webtest.canoo.com/, letzter Abruf: 04.01.2013
Canoo (2013b)	Manual, http://webtest.canoo.com/webtest/manual/WebTestHome.html, letzter Abruf: 04.01.2013

Guillemot, M. (2007)	WebTest vs Selenium, http://mguillem.wordpress.com/2007/10/29/webtest-vs-selenium-webtest-wins-13-5/, letzter Abruf: 04.01.2013
Herrmann, W. (2010)	Bedeutung der IT im Unternehmen, http://www.computerwoche.de/a/so-planen-it-entscheider-2010,1930207,2, letzter Abruf: 01.12.2012
JAMon (2013)	JAMon (Java Application Monitor) Users Guide, http://jamonapi.sourceforge.net/, letzter Abruf: 05.01.2013
Leutek (2013)	End-to-End-Monitoring: http://www.leutek.de/loesungen/end-to-end-monitoring.html, letzter Abruf 02.01.2013
Mozilla Foundation (2013)	Über uns, https://www.mozilla.org/de/about/, letzter Abruf 03.01.2013
Nagios (2012a)	Nagios, http://www.nagios.org/, letzter Abruf: 30.12.2012
Nagios (2012b)	Nagios: Customer and Users, http://nagios.com/users, letzter Abruf: 30.12.2012
Nagios Exchange (2013)	Nagios Exchange - Check_selenium, http://exchange.nagios.org/directory/Plugins/Websites,-Forms-and-Transactions/check_selenium/details, letzter Abruf: 20.01.2013)
Open Source Initiative (2013)	The Open Source Definition, http://opensource.org/osd, letzter Abruf 02.01.2013
Selenium HQ (2013a)	Selenium Web Browser Automation, http://seleniumhq.org/, letzter Abruf 05.01.2013

Selenium HQ (2013b)	Selenium IDE, http://seleniumhq.org/projects/ide/, letzter Abruf: 05.01.2013
Selenium HQ (2013c)	Selenium RC, http://seleniumhq.org/projects/remote-control/, letzter Abruf: 05.01.2013
WatiN (2013a)	WatiN, http://watin.org/, letzter Abruf: 05.01.2013
WatiN (2013b)	WatiN Support, http://watin.org/support/, letzter Abruf: 05.01.2013
Watir (2013a)	Watir, http://watir.com/, letzter Abruf: 05.01.2013
Watir (2013b)	Watir Support, http://watir.com/support/, letzter Abruf: 05.01.2013
Watir Webdriver (2013)	Watir Webdriver, http://watirwebdriver.com/, letzter Abruf 06.01.2013
WebInject (2013)	WebInject, http://webinject.org/, letzter Abruf: 06.01.2013
Whichtestingtool (2013a)	Tool Review Selenium, http://www.whichtestingtool.com/reviews/9-selenium.html, letzter Abruf: 05.01.2013
Whichtestingtool (2013b)	Tool Review WatiN, http://www.whichtestingtool.com/reviews/11-watin.html, letzter Abruf: 05.01.2013
Wikipedia (2012a)	Network Monitoring, http://en.wikipedia.org/wiki/Network_monitoring, letzter Abruf: 28.12.2012

Wikipedia (2012b) Nagios, http://de.wikipedia.org/wiki/Nagios, letzter Abruf: 28.12.2012

Darstellung und Gegenüberstellung der JEE-Anwendungsserver IBM WebSphere und JBoss, sowie eine Nutzwertanalyse in Bezug auf einen möglichen Einsatz in dem auftraggebenden Unternehmen

Schriftliche Ausarbeitung
im Rahmen der Lehrveranstaltung „Projekt des 5. Semesters“
Kompetenzzentrum Open Source (KOS)

Vorgelegt von

Philipp Brüßler, Timothy Dörr,
Sascha Jörg, Julian Löbbers

am 22.01.2013

Fakultät Wirtschaft
Studiengang Wirtschaftsinformatik
Kurs WWI-2010V

Inhaltsverzeichnis

Abkürzungsverzeichnis

a.G.	= **a**uf **G**egenseitigkeit
AG	= **A**ktien**g**esellschaft
API	= **A**pplication **P**rogramming **I**nterface
AS	= **A**nwendungs**s**erver
bKV	= **B**etriebliche **K**ranken**v**ersicherung
CE	= **C**ommunity **E**dition
CPU	= **C**entral **P**rocessing **U**nit
DHBW	= **D**uale **H**ochschule **B**aden-**W**ürttemberg
DMS	= **D**okumenten**m**anagement**s**ystem
DRS	= **D**aten**r**eplikations**s**ervice
EE	= **E**nterprise **E**dition
EJB	= **E**nterprise **J**ava**B**ean
ESB	= **E**nterprise **S**ervice **B**us
GEICO	= **G**overnment **E**mployees **I**nsurance **Co**mpany
GmbH	= **G**esellschaft **m**it **b**eschränkter **H**aftung
GPL	= **G**eneral **P**ublic **L**icense
GUI	= **G**raphical **U**ser **I**nterface
HAM	= **H**igh **a**vailability **m**anager

IBM	= **I**nternational **B**usiness **M**achines Corporation
HTTP	= **H**yper**t**ext **T**ransfer **P**rotocol
HTML	= **H**yper**t**ext **M**arkup **L**anguage
Inc	= **Inc**orporated/**Inc**orporée
IT	= **I**nformations**t**echnologie
JAAS	= **J**ava **A**uthentication and **A**uthorization **S**ervice
JASP	= **J**Boss **A**uthorized **S**ervice **P**artner
JBoss AS	= **J**Boss **A**pplication **S**erver
JDBC	= **J**ava **D**ata**b**ase **C**onnectivity
JDK	= **J**ava **D**evelopment **K**it
JEE	= **J**ava **E**nterprise **E**dition
JMX	= **J**ava **M**anagement **E**xtensions
JNDI	= **J**ava **N**aming and **D**irectory **I**nterface
JON	= **J**Boss **O**perating **N**etwork
JSF	= **J**ava**S**erver **F**aces
JSP	= **J**ava **S**erver **P**ages
JVM	= **J**ava **V**irtual **M**achine
KOS	= **K**ompetenzzentrum **O**pen **S**ource
LGPL	= **L**esser **G**eneral **P**ublic **L**icense

Log4J	= **Log4J**ava
mbH	= **m**it **b**eschränkter **H**aftung
NiLS	= **N**eues **i**ntegriertes **L**eistungs**s**ystem
NW	= **N**etz**w**erk
RAD	= **R**ational **A**pplication **D**eveloper
RMI	= **R**emote **M**ethod **I**nvocation
RZ	= **R**echen**z**entrum
SLA	= **S**ervice-**L**evel-**A**greement
SR	= **S**ervice **R**equest
S&S	= **S**ubscription and **S**upport
TCO	= **T**otal **C**ost of **O**wnership
TEMS	= **T**ivoli **E**nterprise **M**onitoring **S**erver
TEPS	= **T**ivoli **E**nterprise **P**ortal **S**erver
TPV	= **T**ivoli **P**erformance **V**iewer
URL	= **U**niform **R**esource **L**ocator
USD	= **US**-**D**ollar
WAS	= **W**ebSphere **A**pplication **S**erver
WID	= **W**ebSphere **I**ntegration **D**eveloper
WPA	= **W**arehouse **P**roxy **A**gent

WTS	= **W**indows **T**erminal **S**erver
XML	= **E**xtensible **M**arkup **L**anguage

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Motivation

Im Rahmen des Studiums zum Bachelor of Science der Wirtschaftsinformatik an der Dualen Hochschule Baden Württemberg (DHBW) Stuttgart werden im 5. Semester Studienarbeiten zu Themengebieten aus dem Kompetenzzentrum Open Source (KOS) erarbeitet. Das KOS wird von den dualen Partnern finanziell unterstützt. Die möglichen Themengebiete für die Studienarbeiten werden von diesen Partnern zur Ausarbeitung bereitgestellt. Die Studenten erhalten durch diese Projektarbeit die Möglichkeit zur Erweiterung von Wissen und die Gewinnung von ersten Erfahrungen in Kundengesprächen.

Die vorliegende Aufgabenstellung der Studienarbeit, die von dem auftraggebenden Unternehmen gestellt wurde, befasst sich mit der Thematik der JEE-Anwendungsserver. Die Grundlage der Untersuchung ist der kommerzielle WebSphere Application Server (WAS), welcher vom IT-Unternehmen IBM vertrieben wird, und der JBoss Application Server (JBoss AS), welcher ein Open-Source-Produkt darstellt und vom Softwarehersteller Red Hat angeboten wird. Das Ziel dieser Studienarbeit, welches eine reine theoretische Betrachtung darstellt, ist die Analyse dieser beiden Produkte anhand vorgegebener Kriterien.

Die aktuelle Situation zeigt, dass sowohl mittelständische Unternehmen als auch große Konzerne als Basis Ihrer Webapplikationen kommerzielle JEE-Applikationsserver einsetzen. Seit Anfang der Jahrtausendwende setzten sich in verschiedensten Bereichen der Informatik vermehrt sogenannte Open-Source-Produkte durch. Dies geschieht auch in dem Bereich der JEE-Anwendungsserver, in dem sich der JBoss AS erfolgreich am Markt platziert.

1.2 Aufbau der Arbeit

Die Studienarbeit lässt sich vereinfacht in einen theoretischen und einen praktischen Teil untergliedern. Zuallererst werden die Grundvoraussetzungen für eine Analyse der beiden JEE-Applikationsserver geschaffen. Dazu werden theoretische Aspekte, die grundlegenden Charakter für die Bearbeitung der Aufgabenstellung aufweisen, bearbeitet und vorgestellt. Aufbauend auf dieser Darstellung werden die beiden bereits erwähnten JEE-Anwendungsserver vorgestellt und anhand festgelegter Kriterien untersucht. Auf Grund der Tatsache, dass der WAS bereits bei dem auftraggebenden Unternehmen im Einsatz ist, werden firmeninterne Besonderheiten in der Darstellung berücksichtigt.

Bezug nehmend auf die vorangegangene Vorstellung wird eine direkte Gegenüberstellung der beiden Produkte erfolgen. Darüber hinaus gilt es spezifische Besonderheiten weiter zu vertiefen.

Die Gegenüberstellung hat nicht nur den Zweck eines direkten Vergleiches, sondern bildet zusätzlich die Grundlage für eine folgende Nutzwertanalyse. Die Nutzwertanalyse trägt alle gefunden Aspekte des Vergleichs zusammen, damit diese direkt analysiert werden können. Dadurch soll gewährleistet werden, dass insbesondere das auftraggebende Unternehmen einen schnellen und nachvollziehbaren Überblick der Ergebnisse erhält. Anschließend wird eine Kostenanalyse auf Basis von externen Studienarbeiten und Expertengesprächen aufgestellt, bevor eine abschließende Zusammenfassung der Studienarbeit stattfindet und eine Handlungsempfehlung aufgezeigt wird.

1.3 Zielsetzung

Das Ziel dieser vorliegenden Studienarbeit ist, wie bereits erwähnt, die Untersuchung der beiden JEE-Anwendungsserver nach vorgegebenen Analysekriterien, die in einem Kundengespräch mit dem auftraggebenden Unternehmen definiert wurden. [1] Die Aufgabenstellung besagt diesbezüglich, dass vorallem die Besonderheiten eines kommerziellen Produktes gegenüber einem Open-Source-Produkt dargestellt werden sollen. Des Weiteren soll die Funktionsvielfalt hinsichtlich der Administration, der Erweiterbarkeit und der Verfügbarkeit beider Produkte erörtert und zusammengetragen werden. Zudem soll dargestellt werden, ob die bestehende Systemlandschaft mit Hilfe von Open-Source-Mitteln nachgestellt werden kann. Der Fokus liegt dabei auf der Abbildung der Systemlandschaft des auftraggebenden Unternehmens mit Hilfe des Open-Source-Produktes JBoss AS.

1.4 Wertbeitrag

Der Wertbeitrag der Studienarbeit orientiert sich an der Aufgabenstellung die beiden JEE-Applikationsserver auf sachlicher Ebene vorzustellen und zu analysieren. Im Umfeld dieser beiden Produkte existiert eine Vielzahl an Untersuchungen, die die Vorteile des jeweiligen Produkts gegenüber dem Konkurrenzprodukt erläutern und belegen. Diese Untersuchungen werden in der Regel aber finanziell von den jeweiligen Unternehmen unterstützt oder selbst von diesen veröffentlicht. Unabhängige Studien, die fundiert die beiden Produkte vorstellen und vergleichen sind nicht verfügbar. Genau hier versucht die Projektgruppe ihren Fokus und Wertbeitrag zu legen.

[1] Siehe dazu: Anhang 1

Die Untersuchung setzt sich kritisch mit den wissenschaftlichen Quellen auseinander und versucht auf sachlicher Ebene, fernab von Marketingversprechungen die beiden Produkte vorzustellen und zu vergleichen. Die Studienarbeit wird weder von IBM noch von Red Hat finanziell unterstützt, sodass die Unabhängigkeit der Untersuchung gewährleistet wird.

Wie bereits erwähnt, wurde diese Ausarbeitung speziell auf die Bedürfnisse des auftraggebenden Unternehmens abgestimmt. Es wurde nicht nur der Bezug zum Kunden hergestellt sondern auch, bei der Ergebnisfestsetzung, auf die Bedürfnisse des Kunden eingegangen. Dieser erhält mit der Nutzwertanalyse das Ergebnis der Untersuchung kompakt, übersichtlich und dank graphischer Darstellung leicht nachvollziehbar.

2 JEE-Anwendungsserver

In diesem Kapitel sollen grundlegende Themenaspekte, die für die Bearbeitung der Aufgabenstellung von elementarer Bedeutung sind, vorgestellt und erläutert werden. Zunächst findet eine Spezifikation eines JEE-Anwendungsserver, unter anderem mit Hilfe eines Beispiels, statt, bevor dessen Architektur erörtert wird. Anschließend werden die Begrifflichkeiten Skalierbarkeit, Logging, Monitoring, Deployment und Clustering beschrieben, da in den späterfolgenden Kapiteln diese Eigenschaften behandelt werden.

2.1 Beschreibung

Ein JEE-Anwendungsserver, auch Applikationsserver genannt, ist ein Server in einem Client-Server-Netzwerk, auf dem Anwendungsprogramme installiert sind, auf die ein Client zugreifen kann. [2] An dieser Stelle werden exemplarisch einige Aufgaben eines JEE-Anwendungsserver aufgelistet: [3]

- Transaktionsverwaltung
- Bereitstellung von Datenbankanbindungen
- Verwaltung der Sicherheitseinstellungen
- Prozessverwaltung

Die Funktionsweise eines JEE-Anwendungsservers kann beispielsweise an einem Einkaufzentrum verdeutlicht werden. Ein Einkaufszentrum beherbergt in der Regel mehrere Geschäfte. Jedes einzelne Geschäft, innerhalb des Einkaufszentrums, beschäftigt sich ausschließlich mit der Präsentation und dem Verkauf der eigenen Waren. Die übrigen Aufgaben, wie unter anderem die Bereitstellung von Parkplätzen, der Einsatz von Sicherheitskräften oder die Durchführung von Säuberungen, werden vom Einkaufszentrum organisiert. Das Einkaufszentrum stellt somit Dienstleistungen zur Verfügung. In diesem Beispiel repräsentiert das Einkaufszentrum den JEE-Anwendungsserver und das Geschäft die Anwendung, da diese ebenfalls von den Dienstleistungen eines Applikationsservers profitiert.

[2] Vgl. DATACOM Buchverlag GmbH (2012b)
[3] Vgl. Siedersleben, J. (2003) , S.139

2.2 Architektur

Grundlage einer jeden Anwendung ist die Technologie auf der sie gegründet ist. Dies gilt auch für webbasierte Anwendungen, die vom Anwender über einen Standard-Browser, wie zum Beispiel dem Internet Explorer oder dem Firefox, aufgerufen und ausgeführt werden. [4] Zur Auswahl der entsprechenden Technologie muss grundsätzlich eine Auswahl einer Architektur für die Anwendung getroffen werden. Im Folgenden werden die architektonischen und technologischen Grundlagen, anhand der Schichtenarchitektur einer webbasierten Anwendung aufgezeigt.

Für die Darstellung der architektonischen Grundlage einer Anwendung wird die Schichtenarchitektur, auch als Tier-Architektur bezeichnet, erläutert. Diese unterteilt anhand der Anzahl der einzelnen Schichten, die Architekturkategorien. [5]

2-Schichten-Architektur

Die älteste Architektur ist die 2-Schichten-Architektur, die eine Trennung in Client und Datenbankserver vorsieht. Ein Vertreter dieser Architektur ist das Client-Server-Modell. [6] Der Client, der die erste Schicht darstellt, regelt die Präsentations- und Geschäftslogik. Der Datenbankserver dient zur Speicherung von Unternehmensdaten und stellt die zweite Schicht dar. Diese wird ebenfalls als Datenhaltungsschicht bezeichnet. Abbildung 1 veranschaulicht die 2-Schichten-Architektur.

Abb. 1: 2-Schichten-Architektur

[4] Vgl. Langer, T./Reiberg, D. (2006), S. 3
[5] Vgl. DATACOM Buchverlag GmbH
[6] Vgl. Leander, R. (2000), S. 3-4

3-Schichten-Architektur

Ein weiteres Architekturmodell ist die 3-Schichten-Architektur. In dieser Architektur wird die Präsentationslogik von der Geschäftslogik getrennt und auf einem eigenen Server, zum Beispiel einem Applikationsserver, implementiert. Die neuhinzugefügte Schicht besitzt die Funktionalität der Verwaltung und Steuerung der Verbindungen zwischen Client und Datenbankserver. Dies bietet den Vorteil, dass eine wesentlich größere Anzahl von Clients auf die Anwendung zugreifen kann. [7] Abbildung 2 zeigt die 3-Schichten-Architektur.

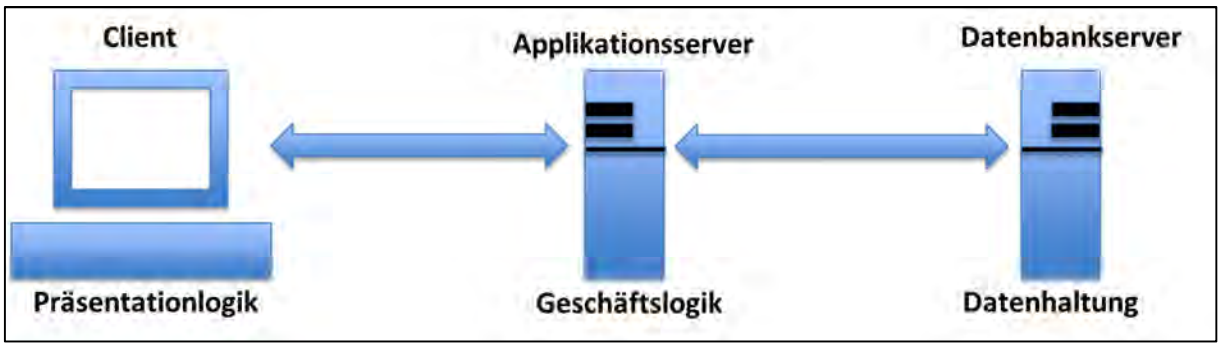


Abb. 2: 3-Schichten-Architektur

Eine Anwendung, die auf dem Prinzip der 3-Schichten-Architektur oder 4-Schichten-Architektur, die im Folgenden beschrieben wird, basiert, ermöglicht eine erleichtertere Weiterentwicklung oder Neuanpassung der Umgebung. Zum Beispiel kann die Einbindung einer Datenbank ohne eine Anpassung der gesamten Anwendung vorgenommen werden und die Präsentationslogik kann den Bedürfnissen der verschiedensten Präsentationstypen, wie zum Beispiel Laptop oder Smartphone, angepasst werden.

Die Technologiebasis für einen Applikationsserver und die darauf liegenden Anwendungen basieren auf der Plattform Java Enterprise Edition (JEE), welche früher als J2EE bezeichnet wurde. [8] Die JEE besitzt Spezifikationen und Programmierschnittstellen, die zur transaktionsbasierten Ausführung von mehrschichtigen Unternehmens- und Webanwendungen nützen. [9, 10] Weitere Informationen zu dem Thema JEE Version 5, da der eigesetzte WAS bei dem auftraggebenden Unternehmen auf dieser Grundlage beruht, finden Sie in der Literatur von Sun Microsystems mit dem Titel „Java Platform, Enterprise Edition 5“. [11]

[7] Vgl. DATACOM Buchverlag GmbH (2012d)
[8] Vgl. Robinson, S. (2011), S.15
[9] Vgl. Stark, T. (2010), S.19
[10] Vgl. Fischer, S./Koschel, A./ Wagner, G. (2006)
[11] Vgl. Sun Microsystems, Inc. (2006)

4-Schichten-Architektur

Basierend auf der 3-Schichten-Architektur existiert die 4-Schichten-Architektur. Diese unterscheidet sich von der bereits vorgestellten 3-Schichten-Architektur durch die Trennung in die Visualisierungsschicht, die dem Client zugeordnet ist, und der Präsentationslogik, die zum Beispiel in einem Webserver geregelt ist. Diese Trennung ermöglicht den Einsatz eines herstellerunabhängigen Webservers, der mittels eines Protokolls, wie zum Beispiel dem Hyper-Text Transfer Protocol (HTTP), mit dem Client kommuniziert. Das Protokoll HTTP ermöglicht den Datentransfer zwischen Client und Webserver. [12, 13] Der Webbrowser unterstützt zur Darstellung die Auszeichnungssprachen HTML, XML sowie JavaScript zur Anzeige dynamischer Inhalte. Die benötigten Standards der JEE für die Präsentationslogik auf dem Webserver heißen Java Servlets und Java Server Pages (JSP). [14, 15]

Um die Kommunikation von Schicht 2, dem Webserver, zu Schicht 3, dem JEE-Applikationsserver, zu ermöglichen, bietet JEE zum Beispiel das Protokoll Remote Method Invocation (RMI) an. Das RMI beinhaltet den Mechanismus zum Zugriff von einer Anwendung auf entfernte Dienste, um somit die Datenerstellung und die Kommunikationssteuerung zu ermöglichen.[16, 17] Des Weiteren wird dem JEE-Applikationsserver das herstellerunabhängiges Komponentenmodell Enterprise JavaBean (EJB) als JEE-Standard vorgeben.[18]

Für die Datenhaltungsschicht, die Schicht 4, gibt es keine Eigenimplementierung. Jede relationale Datenbank, die einen kompatiblen Java Database Connectivity (JDBC) Treiber bereitstellt, kann eingesetzt werden, um über dieses Protokoll mit dem Applikationsserver zu kommunizieren. Weitere technische Informationen zu dem Thema JEE finden Sie in der technischen Dokumentation der JEE von dem Unternehmen Oracle. [19] Abbildung 3 veranschaulicht die gerade beschriebene 4-Schichten-Architektur.

[12] Vgl. Kioskea Deutschland (2013)
[13] Vgl. Stark, T. (2010), S.39
[14] Vgl. javer-server.net (2007)
[15] Siehe dazu: java-server.net (2007)
[16] Vgl. Adler, S. (o.J.), S.1
[17] Vgl. Bögel, S. (2005)
[18] Vgl. Siedersleben, J. (2003), S.137
[19] Vgl. Oracale (2013)

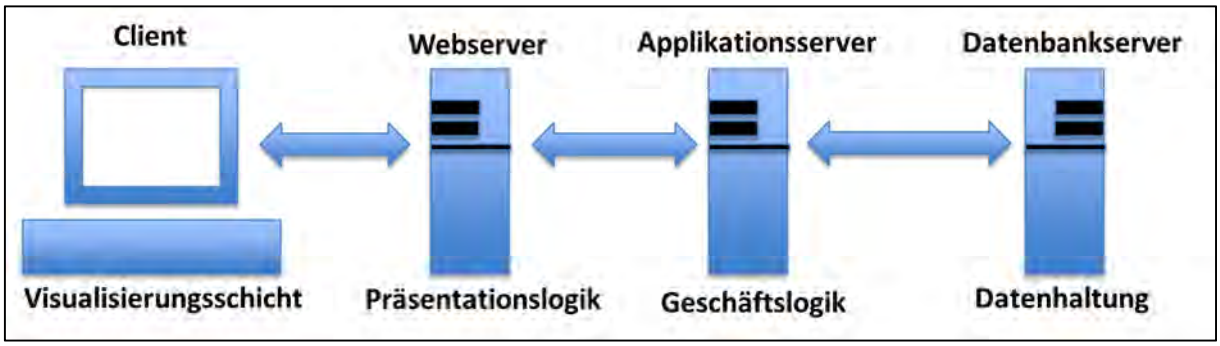


Abb. 3: 4-Schichten-Architektur

2.3 Definition von Eigenschaften

An dieser Stelle werden, wie bereits erwähnt, ausgewählte Eigenschaften definiert, die in dieser Studienarbeit unter anderem von Bedeutung sind. Diese Kriterien sind Bestandteile der Grundlage für die Vorstellung des WAS und des JBoss AS.

Skalierbarkeit

Die Skalierbarkeit ist ein Maß für die Anpassungsfähigkeit zum Beispiel eines Servers, eines Dienstes oder einer Anwendung an wachsende Leistungsanforderungen. [20] Man unterscheidet zwischen vertikaler und horizontaler Skalierung.

Bei der vertikalen Skalierung werden die Komponenten des Systems durch leistungsfähigere Hardware-Komponenten ersetzt oder ergänzt. [21] Traditionelle Beispiele sind unter anderem die Erweiterung des Speichers und der Einbau einer leistungsfähigeren CPU. Der Vorteil dieser Skalierung ist, dass in der Regel keine softwareseitige Anpassung vorgenommen werden muss. Der Nachteil dieser Skalierung ist, dass die Kosten stark überproportional ansteigen. Dies hat zur Folge, dass entweder die finanzielle Leistungsgrenze schnell erreicht ist oder keine leistungsfähigere Hardware-Komponente auf dem Markt verfügbar ist. [22]

Bei der horizontalen Skalierung werden die Aufgaben des Systems zum Beispiel auf mehrere Server verteilt. [23] Dies bietet den Vorteil, dass diese Art der Skalierung meist kostengünstiger ist. Ein Nachteil dieser Skalierung ist, dass zunächst die Voraussetzung einer Parallelbearbeitung der Aufgaben des Systems geschaffen werden muss, wie zum Beispiel durch softwareseitige Unterstützung. [24]

[20] Vgl. Microsoft Corporation (2012)
[21] Vgl. Vogel, O. u.a. (2009), S. 516
[22] Vgl. Fowler, M. (2003), S.23
[23] Vgl. Vogel, O. u.a. (2009), S. 516
[24] Vgl. Fowler, M. (2003), S.23

Die horizontale Skalierung gegenüber der vertikalen Skalierung ist meist weniger Effizient, da ein Teil der Leistungsfähigkeit und der eingesetzten Ressourcen durch die Notwendigkeit der Koordination wegfällt. [25]

Logging

Unter dem Begriff Logging versteht man das Protokollieren von Informationen, Daten und Prozessen in einen Speicher. Zu einem späteren Zeitpunkt lassen sich diese abgelegten Informationen, Daten und Prozesse, zum Beispiel zur Systemwiederherstellung nach einem Ausfall, verwenden. Mit Hilfe eines Application-Programming-Interface (API) für das Logging lassen sich beispielsweise Meldungen in eine XML-Datei oder in eine Datenbank schreiben.[26] Die API ist eine Programmierschnittstelle von Anwendungen, die eine indirekte Kommunikation von einem Softwareentwickler zu einer Hardware-Komponente, beispielsweise Daten auf der Festplatte, ermöglicht. [27] Im Anhang 2 befindet sich eine Tabelle, die die Unterscheidung der Meldungskategorien eines Logs darstellt.

Das Thema Logging ist auch im Bereich der aspektorientierten Programmierung anzutreffen. Die aspektorientierte Programmierung ist ein Programmierstil bei dem eine klare Trennung von übergreifenden Zusammenhängen in einzelne Aspekte, die einen expliziten Verantwortungsbereich definieren, vorgenommen wird. Ziel der aspektorientierte Programmierung ist die Steigerung der Modularität von objektorientierten Programmen. [28] Weitere Informationen zur aspektorientierte Programmierung sind in der Seminararbeit „Aspektorientierte Programmierung“ von Herrn Buss zu finden. [29]

Monitoring

Unter dem Begriff Monitoring versteht man die Beobachtung von Objekten, zum Beispiel technische Komponenten in Bezug auf ihre Stabilität, Effizienz, Produktivität und Funktionalität. Auswertungen dieser Beobachtungen dienen für zukünftige Strategien als Entscheidungshilfen. [30] Mit Hilfe eines zuvorkommenden, regelmäßigen und durchdachtem Monitoring des JEE-Anwendungsservers besteht die Möglichkeit, dass zum Beispiel Ausfallzeiten durch proaktive Überwachung verringert werden oder Kosten- und Kapazitätenplanungen durch Auswertungen von Daten identifiziert werden. [31]

[25] Vgl. ebenda, S.23
[26] Vgl. Galileo Press (2009)
[27] Vgl. TechTerms (2012)
[28] Vgl. DATACOM Buchverlag GmbH (2012c)
[29] Vgl. Buss, M. (2005)
[30] Vgl. SEO-united.de (o.J.)
[31] Vgl. Faustmann, H. /u.a. (2009), S.401

Deployment

Unter dem englischen Begriff Deployment, übersetzt Softwareverteilung, versteht man die Installation bzw. die Verteilung von Applikationen auf eine entsprechende Umgebung, wie zum Beispiel Produktions- oder Entwicklungsumgebung. Das Deployment kann manuell oder automatisch erfolgen. [32] Die automatisierte Softwareverteilung bietet gegenüber der manuellen Softwareverteilung einige Vorteile, jedoch sind auch Risiken mit dieser Vorgehensweise verbunden. Diese sind in der Tabelle 1 dargestellt.

Risiken	Nutzen
• Unerwartete Probleme beim Bootvorgang eines Clients • Firewall-Konfiguration verhindert automatische Softwareverteilung • Auftretende Fehler bleiben unerkannt	• Verringerung des Administrationsaufwands • Reduktion der notwendigen Personalressourcen • Einheitliche Installationen • Beschleunigung von Systemupdates • Beschleunigung von Systemwiederherstellungen • Installationsmöglichkeit außerhalb der Arbeitszeit • Kosteneinsparungen • Reduzierung der Fehleranfälligkeit

Tabelle 1: Automatisierte Softwareverteilung - Risiken und Nutzen [33, 34]

Clustering

Ein Unternehmen verfolgt das Ziel, die IT-Anwendungen rund um die Uhr auf einer anpassbaren und zuverlässigen Umgebung ohne Unterbrechungen und Schwierigkeiten betreiben zu können. Der Einsatz von Clusters ist vor allem in Bezug auf Skalierbarkeit, Ausfallsicherheit und Performance der Anwendungen von immer größerer Bedeutung. [35]

Clustering erlaubt, dass eine Anwendung auf mehreren parallelen Servern gleichzeitig laufen kann, obwohl der Anwender nur eine einzige Ansicht erhält. Zur Bearbeitung von Benutzeranfragen werden mehrere Server verwendet, um die Anwendung ausfallsicherer und die Datenbanknutzung effizienter zu gestalten. Zum Beispiel kann das Laden einer Anwendung beim Starten auf mehreren Servern ablaufen, was zu einer verbesserten Performance beiträgt. Des Weiteren werden bei einem Ausfall eines Servers die Funktionen von den übrigen Servern, die im Netzwerk verbunden sind, übernommen. Der Vorteil ist die erhöhte Ausfallsi-

[32] Vgl. Lee, R. (2008), S. 30
[33] Vgl. IT-Administrator (o.J.a)
[34] Vgl. IT-Administrator (o.J.b)
[35] Vgl. Stansberry, Brian (o.J.)

cherheit. [36] Der Nachteil ist jedoch, dass die Lizenzkosten steigen, da mehrere Lizenzen im Einsatz sind.

In einem Netzwerk kann es mehrere verschiedene Cluster geben. Ein Cluster besteht aus mehreren Knoten. Die folgende Abbildung 4 zeigt einen Client, der in Verbindung mit einem Cluster, in der Abbildung grün umrandet, steht. Diese Kommunikation findet über einen Management-Server und einen Load-Balancer statt. Der Management-Server überprüft, welche Server im Cluster aktiv sind und teilt dies dem Load-Balancer mit. Der Load-Balancer ist ein Lastverteiler und dient bei Anfragen eines Clients zur Bedienung der optimalsten Server-Performance. [37]

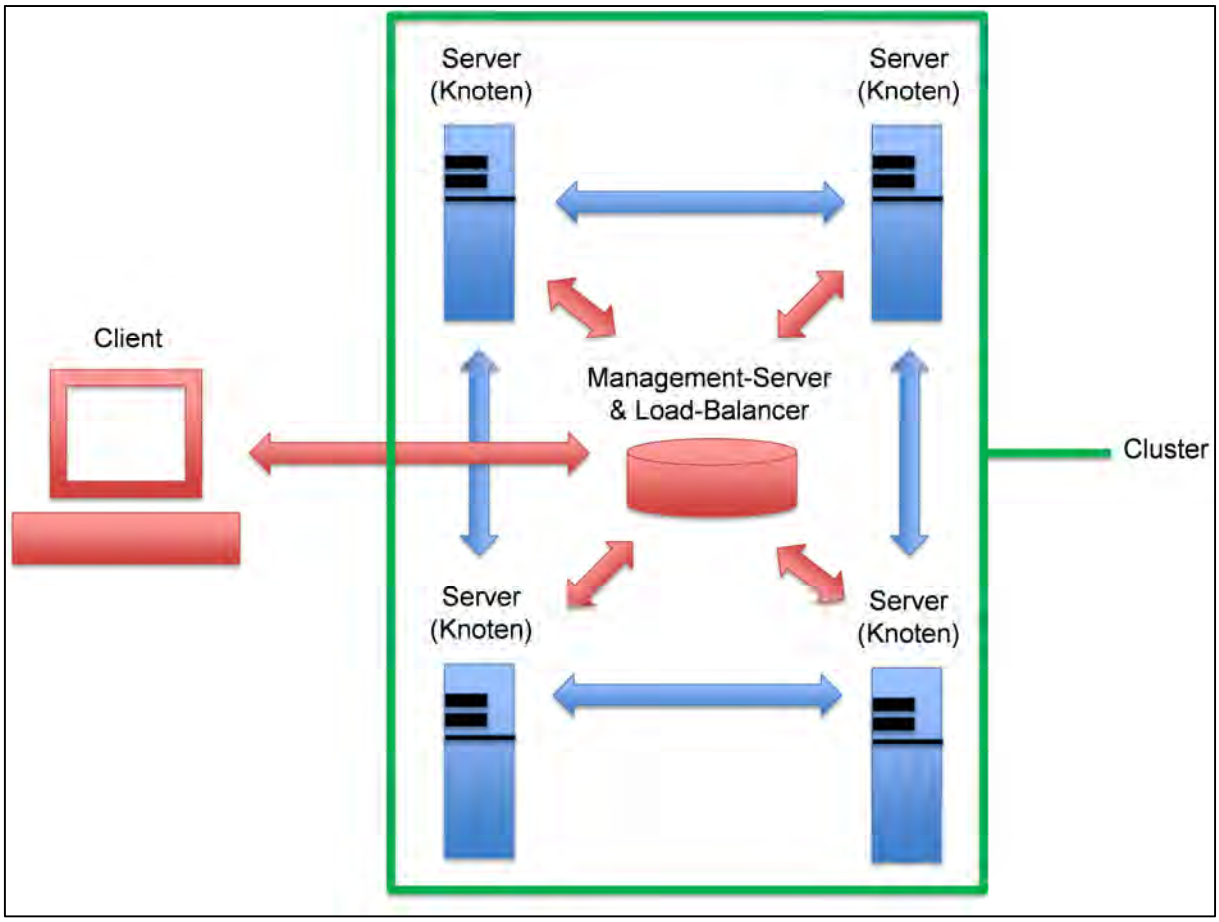


Abb. 4: Clustering

[36] Vgl. Stansberry, Brian (o.J.)
[37] Vgl. Faure, M. (2012)

3 IBM WebSphere AS

Der WAS ist eine kommerzielle Implementierung eines JEE-Anwendungsservers. [38] Erste Teile des Servers wurden bereits 1998 entwickelt. [39] Seit 2001 erfüllt der Anwendungsserver von IBM die Spezifikation eines JEE-Anwendungsservers. [40] Das auftraggebende Unternehmen arbeitet aktuell mit der Version 7.0. Auf diese Version wurde Mitte des Jahres von Version 6 migriert. Die Version 7 unterstützt die JEE 5 Spezifikation und arbeitet mit dem Java Development Kit (JDK) 6. [41] Die aktuellste Version des WAS ist die Version 8, die sich seit Juni 2011 auf dem Markt befindet. Für die Ausarbeitung wird die Version 7, die bei dem auftraggebenden Unternehmen im Betrieb ist, berücksichtigt.

IBM vertreibt die kommerzielle Variante des JEE-Anwendungsservers in einer Vielzahl von unterschiedlichen Produktvariationen und Umfang. [42] Es unterstützt zudem eine Vielzahl unterschiedlicher Systeme, wie zum Beispiel Windows, z/OS oder Solaris. [43] Das auftraggebende Unternehmen nutzt die Produktvariation WAS Network Deployment, die auf einem Windows Server 2008 betrieben wird. [44] Der Unterschied zu dem Standard WAS besteht darin, dass der WAS auf stand-alone Serverfunktionen begrenzt ist. [45] Das Paket Network Deployment bietet dem Nutzer Funktionalitäten für das verteilte Serverumfeld wie zum Beispiel einer zentralen Verwaltungseinheit und dem Workload Management an. [46]

3.1 Einsatz in der Anwendungslandschaft des auftraggebenden Unternehmens

Der WAS wird bei dem auftraggebenden Unternehmen für das Dokumentenmanagement (DMS), in das alle Dokumente der Kunden eingepflegt und verwaltet werden, und die Anwendung NiLS-GUI verwendet. NiLS ist das Leistungsabrechnungssystem des auftraggebenden Unternehmens Hier werden Versicherungsfälle dokumentiert, bearbeitet und beglichen. In der nachfolgenden Abbildung 5 ist die Architektur der NiLS-Anwendung abgebildet. Die Gesamte Architektur von DMS und NiLS kann im Anhang 6 nachgelesen werden.

[38] Vgl. IBM Deutschland GmbH (2012)
[39] Vgl. Sys-Con Media (2013)
[40] Vgl. Hodgson, J. u. a. (2001), S.3
[41] Vgl. Agopyan, A. (2009)
[42] Vgl. IBM Deutschland GmbH (2012b)
[43] Vgl. IBM Deutschland GmbH (2012c)
[44] Siehe dazu: Anhang 3
[45] Siehe dazu: Anhang 4
[46] Vgl. Agopyan, A. (2009), S.7 f.

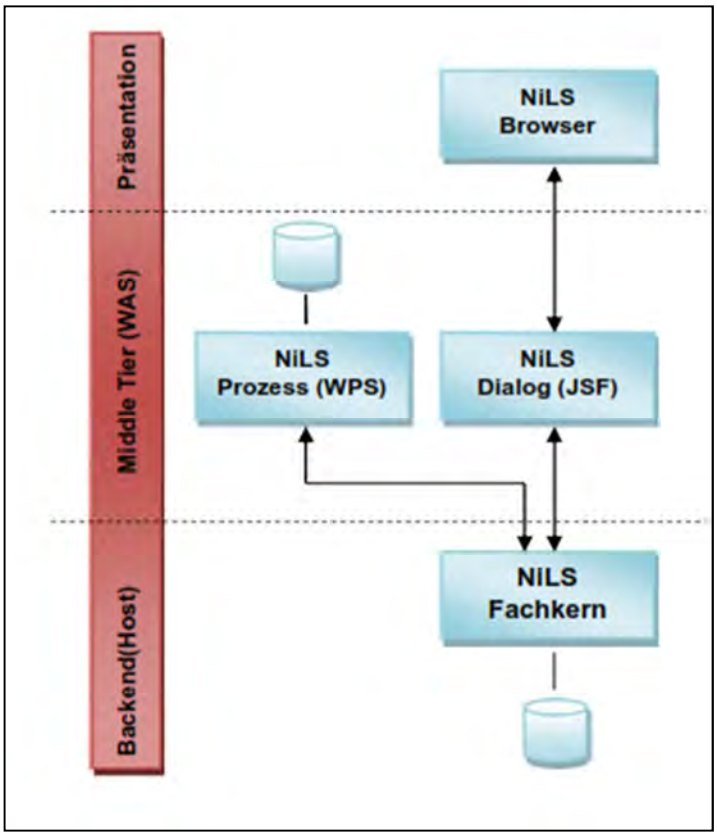


Abb. 5: NiLS-Architektur

Der Fachkern von NiLS wird mit Hilfe von Cobol-Modulen auf einem Großrechner betrieben. Das gesamte Graphical User Interface (GUI) befindet sich indessen auf dem WAS. Sie wird mittels JavaServer Faces (JSF) realisiert. Die JSF-Technologie stellt ein Programmiergerüst zur webbasierten Anwendungsentwicklung bereit. [47]

Darüber hinaus laufen das Bestandssystem AVIB, die GUI für die Online-Module und die bKV-GUI auf dem WAS. Die Online-Module bei dem auftraggebenden Unternehmen sind Module, bei denen Kunden bestimmte Versicherungen online abschließen können. Beispielsweise eine Auslandsreisekrankenversicherung. Der Kunde schließt den Vertrag über ein Modul auf der Webseite des auftraggebenden Unternehmens ab und die Daten gelangen so dunkelverarbeitet in das Bestandssystem AVIB. Unter Dunkelverarbeitung versteht man im Umfeld der IT die autarke und automatische Ver- und Bearbeitung eines Geschäftsprozess ohne menschliches Zutun.[48]

Sollten Probleme bei dieser Dunkelverarbeitung auftreten, kann mit Hilfe der Online-Modul-GUI die Daten und der Vorgang manuell bearbeitet werden. Die bKV-GUI ist für die Verwaltung der Themen rund um die betriebliche Krankenversicherung zuständig. Desweiteren

[47] Vgl. IRIAN Solutions Softwareentwicklungs- und Beratungsgesellschaft mbH (o. J.a)
[48] Vgl. Oletzky, T. (2012)

werden sämtliche Aktionen des auftraggebenden Unternehmens, beispielsweiße bei Kooperationen mit der Knappschaft, über den WAS gesteuert. In den nachfolgenden Kapiteln wird der WAS nach den vorgegebenen Kriterien untersucht. Das erste Oberkriterium das dabei untersucht wird ist die Verfügbarkeit.

3.2 Verfügbarkeit

Das Themengebiet Verfügbarkeit teilt sich in die Unterkriterien Deployment, Support, Ausfallsicherheit und Clustering, die nachfolgend untersucht werden. Bei der Beschreibung werden zuallererst die Besonderheiten des WAS dargestellt. Darüber hinaus werden auf die firmenspezifischen Besonderheiten des auftraggebenden Unternehmens eingegangen.

3.2.1 Deployment

Um eine Anwendung auf dem WAS zu deployen gibt es zwei Möglichkeiten. Eine Anwendung kann zum einen über eine Webanwendung, der sogenannten Integrated Solutions Console, deployed werden, als auch über das Wsadmin Tool, welche eine Kommandozeile darstellt. [49] Beide Tools werden im Kapitel 3.3.1 genauer erläutert.

Der WAS enthält in seiner Standardausführung auch die Lizenzen für eine eclipse-basierte Entwicklungsumgebung, dem Rational Application Developer (RAD). Damit können unter anderem statische und dynamische Codeanalysen sowie Performancetests durchgeführt werden. Für das auftraggebende Unternehmen ist dies aber eher uninteressant, da das Unternehmen das Produkt WebSphere Integration Developer (WID) von IBM einsetzt. Dies ist ebenfalls eine eclipse-basierte Entwicklungsumgebung. Die Funktionen des RAD können mithilfe des WID nahezu vollständig abgedeckt werden. [50]

Bei dem auftraggebenden Unternehmen funktioniert der Deploy-Vorgang wie folgt: Anwendungen werden über die webbasierte Administrationsoberfläche deployed. Die Server laufen dabei weiter. In der Abbildung 6 ist die Ebenenarchitektur des auftraggebenden Unternehmens abgebildet. Für die Entwicklungs-, Integrations- und Produktionsumgebung existieren verschiedene Deployment-Rechte.

[49] Vgl. IBM Deutschland GmbH (2012d)
[50] Siehe dazu auch: IBM Deutschland GmbH (2012e)

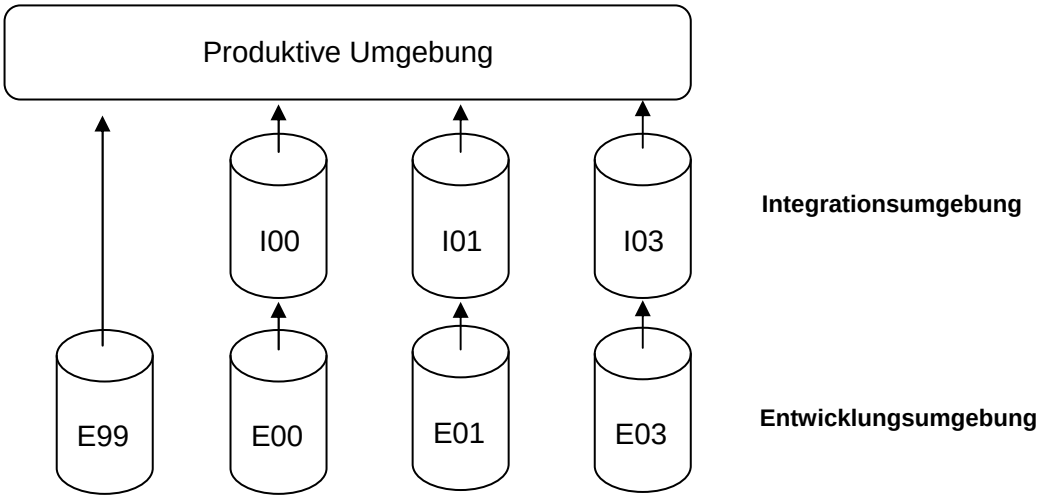


Abb. 6: Ebenenarchitektur des auftraggebenden Unternehmen

Der Deploymentvorgang von Lokal auf die Entwicklungsumgebung kann während des Betriebes erfolgen. Von der Entwicklungsumgebung auf die Integrationsumgebung erfolgt der Deploymentvorgang über das Rechenzentrum. Die E99-Entwicklungsumgebung ist ein sogenannter Hotfix-Kanal. Hier können Änderungen direkt auf die produktive Umgebung deployed werden. Eine spezielle Testumgebung existiert nicht. Stattdessen werden Tests auf der Entwicklungs- und Integrationsumgebung durchgeführt.

Für das produktive Umfeld gilt, dass der Deploymentvorgang am Wochenende durchgeführt wird. Die 8 produktiven Server werden abwechselnd, pro Tag jeweils ein Server, aktualisiert. Der Server auf dem aktuell deployed wird läuft weiter, wird aber nicht angesprochen. Das bedeutet, dass der Server während des Betriebs aktualisiert wird und weiter läuft, der Server aber nicht von Nutzern angesprochen werden kann.

3.2.2 Support

Da es sich bei dem WAS um einen kommerziellen JEE-Anwendungsserver handelt wird die Software inklusive Support und Wartung verkauft. Die Supportleistungen von IBM kann anhand der Abbildung 7 erläutert werden.

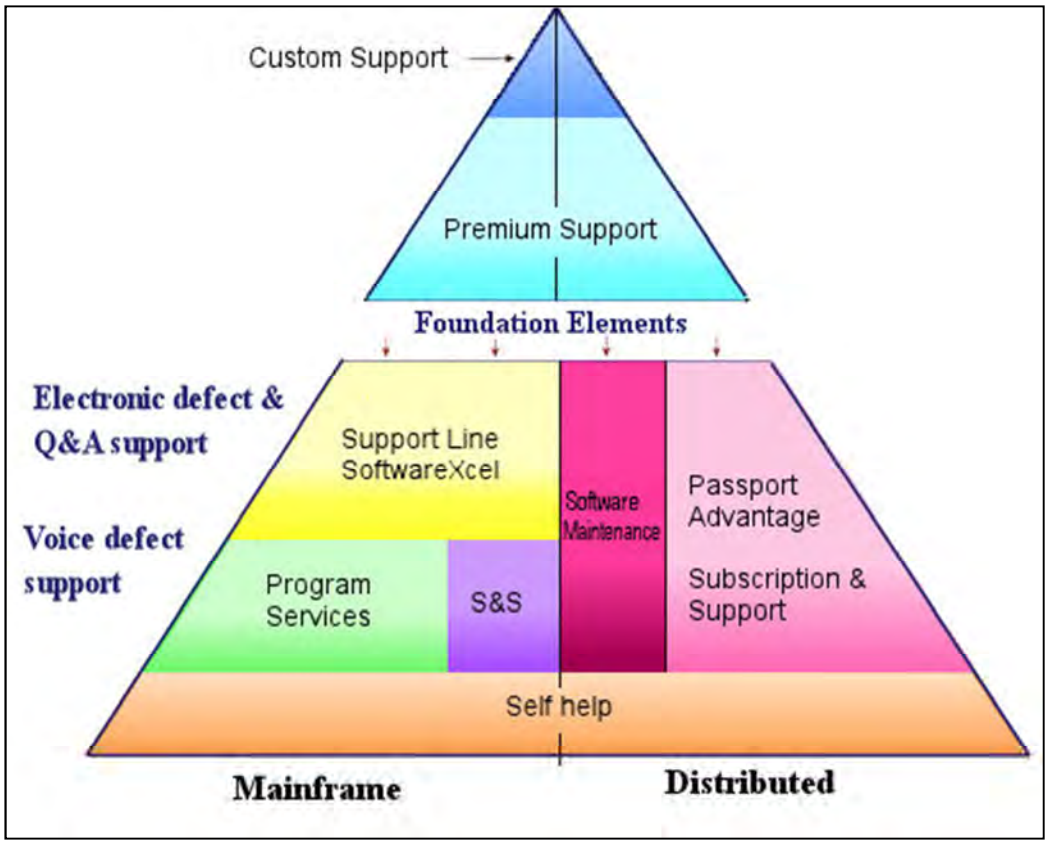


Abb. 7: Supportleistungen des Unternehmen IBM [51]

Im verteilten Umfeld bietet die IBM für den Supportbereich das Paket „IBM Software Subscription and Support“ (S&S) an, welches ein Bestandteil des Basissupports „IBM Passport-Advantage“ ist. Bei jeder neuen Produktlizensierung bietet dieses Paket einen einjährigen Zugang zu Produktupgrades und online bzw. telefonischen technischen Support. Das Paket wird jährlich erneuert. [52] Bei einem Erstkauf sind die S&S-Kosten im ersten Jahr im Software-Preis enthalten. [53]

Bei technischen Anfragen, sogenannte Service Requests (SR), gibt es vier verschiedene Prioritäten und zeitliche Ziele bezüglich der Antwortzeiten von IBM. Für Anfragen mit der Priorität 1 versucht IBM innerhalb von zwei Stunden zu antworten. Bei Anfragen mit der Priorität 2-4 nimmt sich IBM als Ziel, innerhalb von zwei Geschäftsstunden zu antworten. Die Einteilung in die verschiedenen Prioritätsstufen übernimmt der Kunde in Zusammenarbeit mit einem Support Analysten von IBM. [54]

[51] Vgl. IBM Deutschland GmbH (2012f)
[52] Vgl. IBM Passport Advantage and Passport Advantage Express (2012), S.1 ff.
[53] Vgl. IBM Deutschland GmbH (2012a)
[54] Siehe dazu: Anhang 5

Der Premium Support ist optional und bietet zusätzlich zu dem Standard Support einen personalisierten technischen Support, den Wissenstransfers von IBM zum Kunden sowie eine Unterstützung bei der Integration von IBM-Produkten in die Anwendungslandschaft. [55] Weiterhin erhält der Kunde einen Premium Service Manager als direkten Ansprechpartner, der als Vermittler zwischen dem Kunden und dem Support von IBM fungiert. [56]

Der Custom Support ist ebenfalls optional und bietet zusätzlich zum Premium Support einen speziell auf die Bedürfnisse des Kunden angepassten Support. Dieser Support hat eher proaktiven Charakter und versucht die Ausfallsicherheit des Gesamtsystems zu erhöhen. [57]

Die Höhe der Supportkosten des Basispakets bzw. des Premium oder Custom Support können nicht beziffert werden. IBM ist dazu weder telefonisch noch schriftlich bereit Angaben über die Höhe der Supportkosten zu machen. Zudem hängt die Höhe der Kosten von vielen unterschiedlichen Faktoren, wie zum Beispiel die Anzahl verschiedener IBM-Produkte und Lizenzen, die Länge der Kundenbindung oder die Höhe des Gesamtvoluminas, ab, sodass keine genauen Angaben gemacht werden können. Darüber hinaus ist auch das auftraggebende Unternehmen nicht bereit Angaben über die Höhe der Supportzahlungen zu machen.

Ein wichtiger Faktor der im Umfeld des Supports noch erwähnt werden muss ist, dass die Produkte von IBM einen sogenannte Support Lifecycle besitzen. In diesem Lebenszyklus ist geregelt, wie lange eine bestimmte Version eines Produktes gewartet wird. Sollte das Ende dieses Lebenszyklus erreicht werde, stoppt automatisch, sofern nicht spezielle Wartungsverträge einspringen, der Support für die Version eines Produktes.

Um wieder Support erhalten zu können, muss der Kunde die Version auf einen neueren Stand migrieren. Bei IBM wird aktuell die Regel „5 plus 3“ verfolgt, was jedem Produkt einem Standard Support von 5 Jahren gewährt, plus die Möglichkeit einen zusätzlichen Support von 3 Jahren hinzuzukaufen. [58] Der Support Lifecycle des Produktes endet nach Ablauf immer am 30. April oder am 30. September eines Jahres. [59]

[55] Vgl. IBM Deutschland GmbH (2012f), S. 8
[56] Siehe dazu: IBM Deutschland GmbH (2012g)
[57] Vgl. IBM Deutschland GmbH (2012f), S. 10
[58] Vgl. Hohberger, R. (2010), S. 13
[59] Vgl. IBM Deutschland GmbH (2012h)

3.2.3 Ausfallsicherheit

Die Ausfallsicherheit ist ein Kernthema rund um JEE-Anwendungsserver. Der Ausfall eines Servers kann verschiedenste Ursachen haben: fehlerhafte Bedienung, Hardware-Fehler, Auslastung der Serverkapazitäten oder Fehler in der Software. Idealerweise sollten diese Fehler ausgemerzt und vermieden werden. Falls ein Fehler dennoch eintritt, ist die Aufgabe eines JEE-Anwendungsservers, dass dieser schnellstmöglich den Fehler durch beispielsweise das Einspielen eines Backups behebt. Seit dem WAS Version 6 existiert dafür der „High availability manager“ (HAM) eine Komponente des WAS. Er bietet unteranderem folgende Hauptfunktionen an:

- **Singleton-Services Vermeidung**
  Ein Framework, das es ermöglicht sogenannte Singleton-Services hoch verfügbar zu machen. Singleton-Services sind Prozesse, die nur auf einem Knoten eines Clusters laufen. [60] Bei Ausfall des Knotens fällt demnach ein Singleton-Service-Prozess für das Gesamtsystem aus, welches einem Single-Point-Of-Failure entspricht. [61] Typische Singleton-Services, die das Framework des HAM verwenden, sind im WAS-Umfeld der Transaktionsmanager im Clusterverbund und der IBM Standart Messaging Provider, auch als Service-Integration-Bus bezeichnet.

- **Bulletin-Board**
  Das Bulletin-Board ist ein Mechanismus, der es den Servern und einzelnen Prozessen erlaubt untereinander Statusinformationen auszutauschen. Diese dienen primär zur Workloadsteuerung, was die Verbesserung der Verfügbarkeit zur Folge hat.

- **Datenreplikationsservice (DRS)**
  Der Datenreplikationsservice (DRS) wird verwendet, um HTTP-Sitzungsdaten, Stateful-EJB-Sitzungen und Informationen des dynamischen Caches zwischen Cluster-Membern zu replizieren. Die Replika der Speicher werden zwischen den Cluster-Membern übergeben. Somit ist eine direkte Betriebsübernahme innerhalb des Clusterverbunds möglich. [62]

Für die Ausfallsicherheit sind bei dem auftraggebenden Unternehmen insbesondere die Bereiche Rechenzentrum (RZ) und Netzwerk (NW) zuständig. Speziell für kritische Ausfälle

[60] Vgl. IBM Deutschland GmbH (2012i)
[61] Vgl. Agopyan, A. (2009), S.23
[62] Vgl. IBM Corporation (2005), S.1 ff.

wurde in dem auftraggebenden Unternehmen eine schnelle Eingriffstruppe bestehend aus Gruppen- und Bereichsleiter gegründet. Bei Ausfällen kann so schnell reagiert werden.

Sollte ein Server im Betrieb ausfallen oder erst gar nicht hochfahren können trotzdem alle Nutzer arbeiten. Denn die Nutzer wählen sich nicht direkt auf die Server ein, sondern gehen auf die Adresse des Load-Balancers. Dieser Load-Balancer verteilt die Nutzeranfragen auf die verfügbaren Server. Sollte ein Server nicht hochgefahren sein, erkennt der Load-Balancer, dass dieser Server nicht aktiv ist und verteilt die Nutzer auf die restlichen Server. Bei dem auftraggebenden Unternehmen arbeiten alle Mitarbeiter auf Windows Terminal Servern (WTS). Auf einem WTS arbeiten ca. 20-30 Benutzer. Wenn ein Server während des Betriebes ausfällt, stürzt die aktuelle Session des Nutzers auf dem Server ab und er muss sich über den Load-Balancer wieder neu auf einem Server anmelden. Alle nicht gespeicherten Daten gehen verloren. Der WAS unterstützt zwar den Mechanismus der Sessionübernahme, damit die Änderungen nicht verloren gehen. Die Anwendungen bei dem auftraggebenden Unternehmen können jedoch nicht mit diesem Mechanismus umgehen und die nicht gespeicherten Daten gehen verloren. [63]

3.2.4 Clustering

Bei dem auftraggebenden Unternehmen wird die Methodik des Clusterings nicht angewendet. Für den späteren Vergleich ist dieses Themengebiet somit eher optional. Dennoch sollen hier einige Funktionen des WAS dargestellt werden, die den Funktionsumfang rund um das Thema Clustering darstellt:

Funktionsweise

Der WAS mit dem Paket „Network Deployment“ bietet dem Nutzer die Möglichkeit, mehrere WAS in einer zentralen Verwaltungseinheit zu administrieren. Die Einteilung der WAS-Einheiten erfolgt über Knoten. Ein Knoten bildet dabei eine Gruppe von JEE-Applikationsservern auf einer Betriebssystem-Instanz. Pro Knoten existiert ein Knoten-Agent, der mit dem Deployment Manager, der zentralen Verwaltungseinheit, interagiert. Knoten bzw. eine Gruppe von Knoten kann mithilfe einer Zelle verknüpft und somit zentral verwaltet werden. [64] In Abbildung 8 ist die Architektur dieses Knotenkonzepts mit einem zentralen Deployment Manager abgebildet.

[63] Vgl. Kohler, F. (2013)
[64] Vgl. Agopyan, A. (2009), S.7 ff.

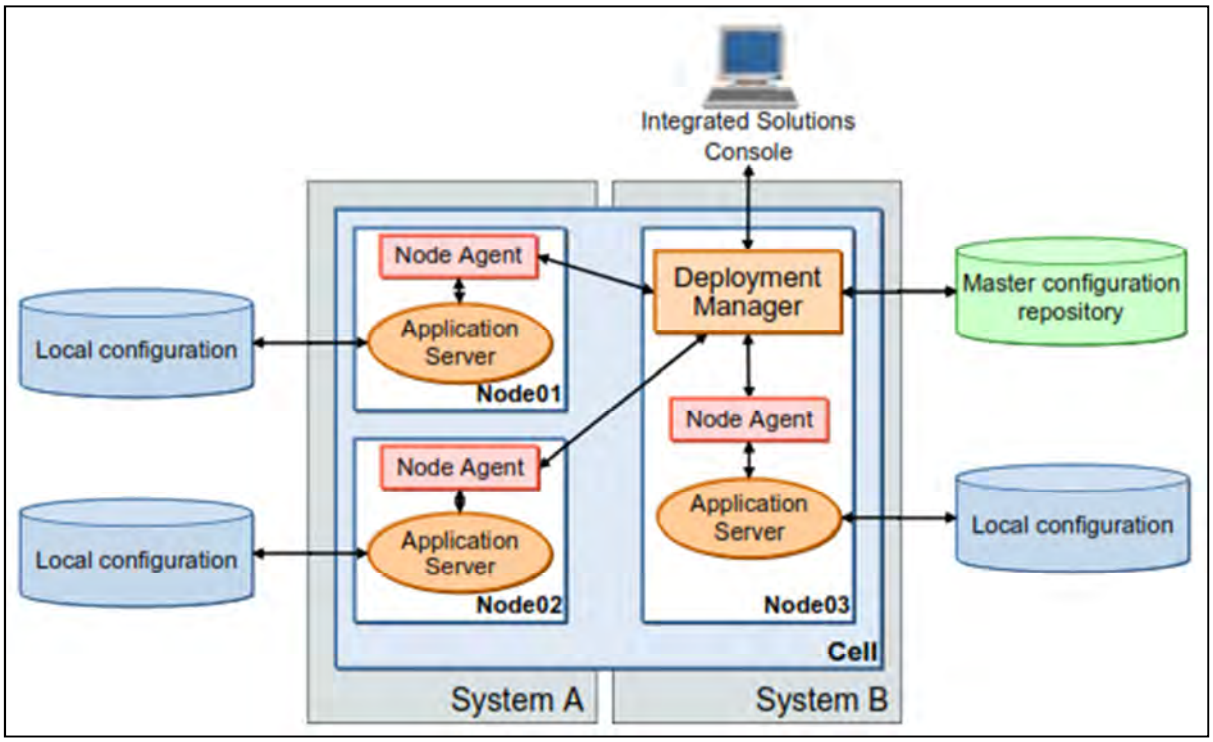


Abb. 8: Verteilte Serverlandschaft mit dem IBM WebSphere AS [65]

In der „Master configuration repository“ werden die Konfigurations- und Anwendungsdateien zentral für die einzelnen Knoten einer Zelle abgelegt.

3.3 Administration

Der Begriff Administration umfasst in der vorliegenden Studienarbeit die Verwaltung und Konfiguration von allen beteiligten Komponenten eines Systems. Mit der Hilfe einer entsprechenden Einrichtung wird dem Anwender ein Überblick über den aktuellen Zustand der Systemkomponenten ermöglicht. [66]

3.3.1 Verwaltbarkeit

Der WAS 7 von IBM besitzt neben der reinen administrativen Möglichkeiten verschiedener JEE-Anwendungsserver weitere Werkzeuge, die die Konfiguration und Administration der Laufzeitumgebung ermöglichen:

[65] Vgl. Agopyan, A. (2009), S.7
[66] Vgl. DATACOM Buchverlag GmbH (2012a)

- **Integrated Solutions Console**
  Browserbasierte Konsole, die innerhalb einer Webanwendung ausgeführt wird, um den WAS verwalten zu können.

- **WSadmin**
  Eine Kommandozeile über die der WAS gesteuert werden kann.

Die Integrated Solutions Console auch Admin Konsole genannt, besitzt folgende typische Verwaltungsmöglichkeiten eines JEE-Appliktationsservers im laufenden Betrieb:

- Hinzufügen, Löschen, Starten und Stoppen von JEE-Anwendungsservern
- Neue Applikationen auf den Server deployen
- Starten und Stoppen der Applikationen
- Veränderung der Konfiguration einer Applikation
- Verwaltung der Sicherheitseinstellungen
- Daten für Performanceauswertung sammeln

Die Abbildung 9 zeigt einen Screenshot der Integrated Solutions Console:

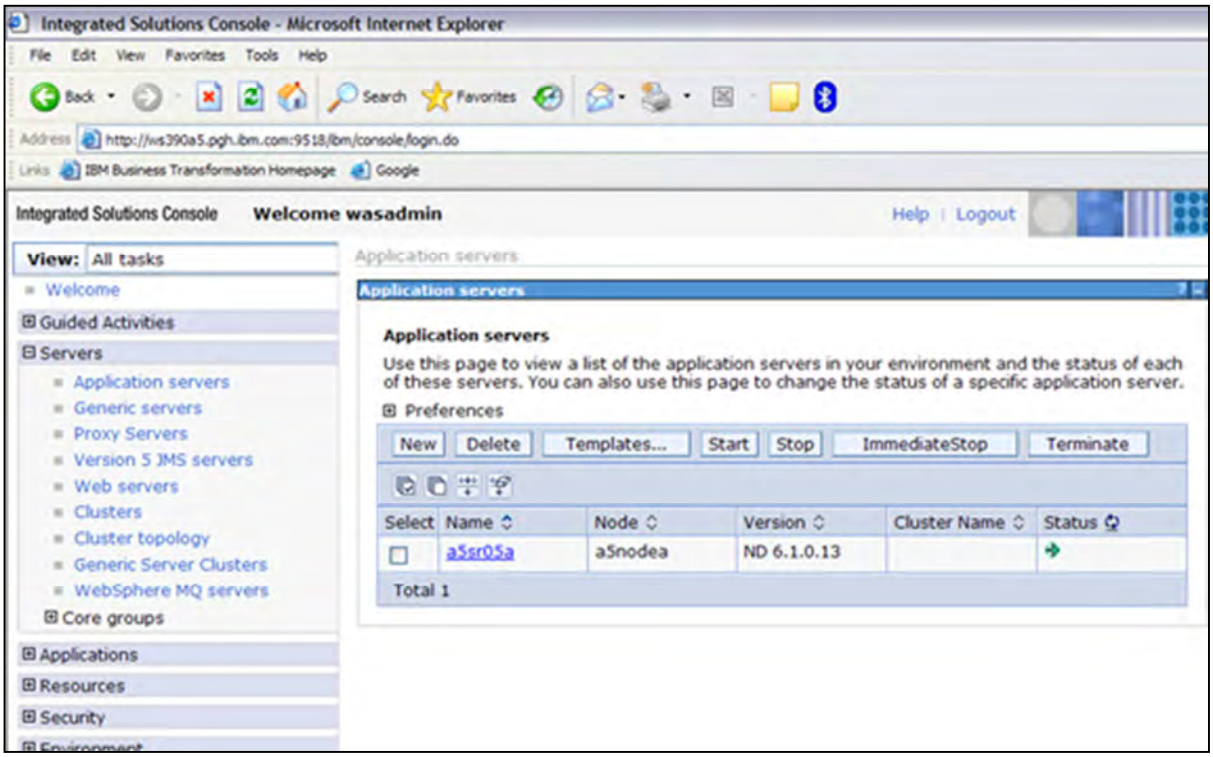


Abb. 9: Integrated Solutions Console

Darüber hinaus gibt es noch einige weitere Verwaltungsmöglichkeiten des WAS die im Folgenden vorgestellt werden. Seit der Version 7 des WAS ist es möglich, im stand-alone Umfeld eine Komponente namens Adminstrative Agents einzusetzen. Damit wird die Verwaltbarkeit der einzelnen WAS-Instanzen abstrahiert, damit sie in dieser Komponente verwaltet werden können. Der Verwaltungsagent muss sich auf derselben Maschine wie der JEE-Applikationsserver befinden. Bis Version 6 musste jeder JEE-Applikationsserver einzeln administriert werden, wie es auch in der folgenden Abbildung 10 ersichtlich ist: [67]

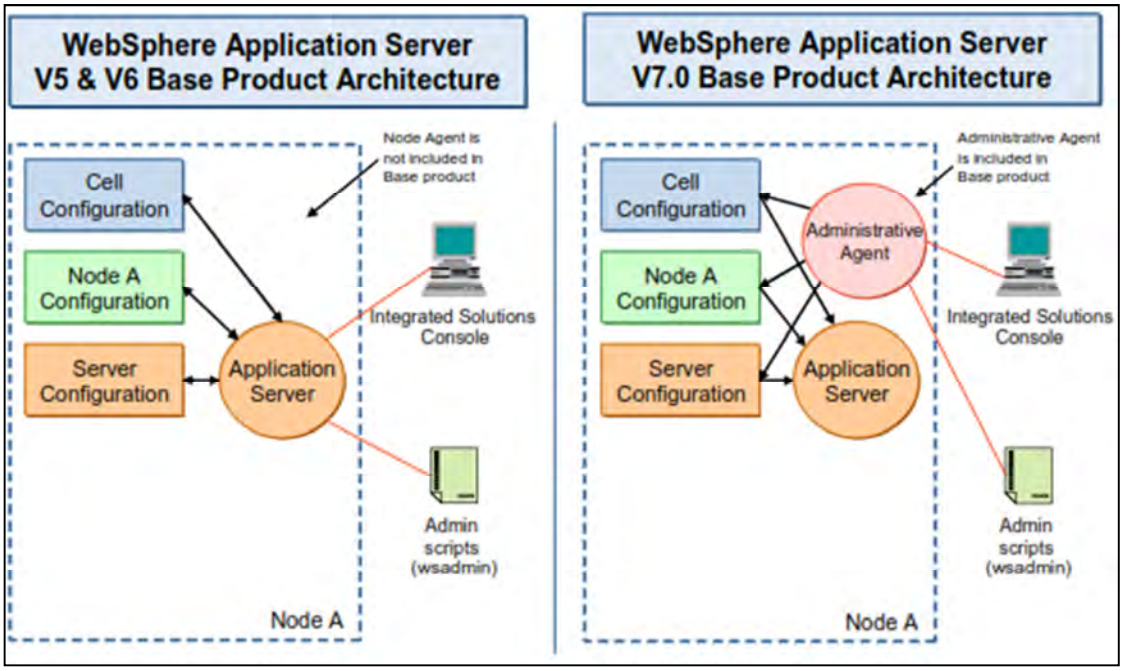


Abb. 10: Einsatz des Adminstrative Agent im stand-alone Umfeld [68]

Neu seit der Version 7 ist ebenfalls die Komponente namens Job Manager. Damit können eigenständige Applikationsserver, die bei einem Verwaltungsagenten oder Deployment Manager registriert sind, registriert werden. Damit können nun Verwaltungsaufgaben direkt über den Job Manager getätigt werden, wie in Abbildung 11 dargestellt ist.

[67] Vgl. Agopyan, A. (2009), S.13
[68] Vgl. ebenda

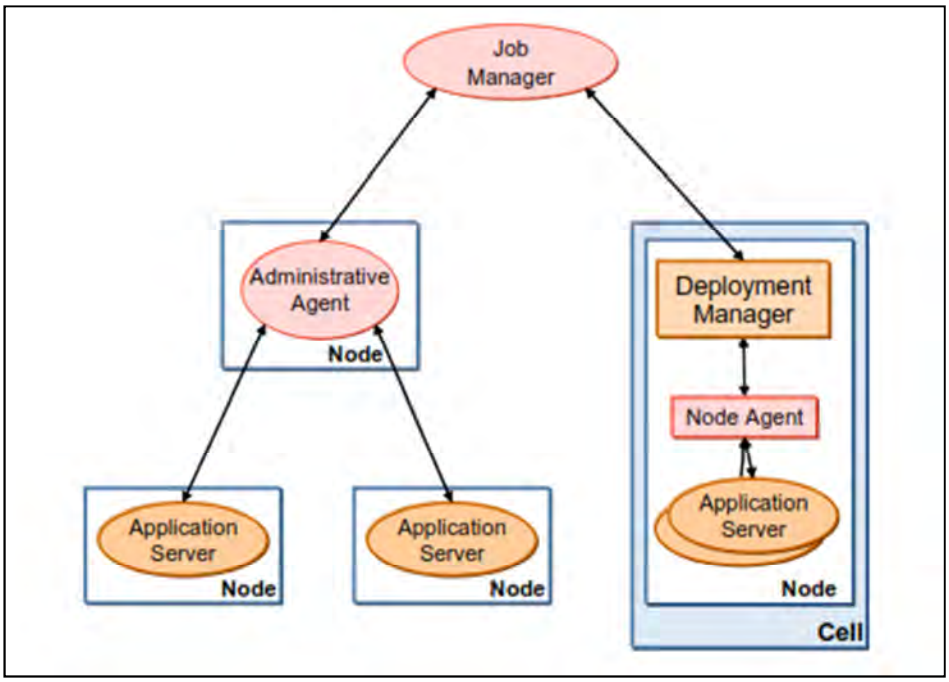


Abb. 11: IBM WebSphere - Einsatz des Job Managers [69]

Sinnvoll ist dieser Job Manager beispielsweise bei einer Umgebung, die aus mehreren eigenständigen Applikationsserver besteht. Jeder dieser stand-alone JEE-Applikationsserver kann mithilfe des Verwaltungsagenten beim Job Manager registriert werden. Damit kann der Administrator über den Job Manager die Verwaltungsbefehle für alle JEE-Applikationsserver zusammenfassen, um beispielsweise eine Anwendung zu installieren oder zu aktualisieren.[70]

3.3.2 Logging

Für das Logging wird bei dem auftraggebenden Unternehmen ebenfalls das Framework Log4J eingesetzt. Darüber hinaus werden auch die Standard Logs der WAS-Produkte genutzt, die WAS-Server-Log-Dateien. Mithilfe der WAS-Server-Log-Dateien werden Logs aus dem WAS geschrieben. Bei dem auftraggebenden Unternehmen existieren 3 verschieden Log-Dateien, die der WAS mit Hilfe von Log4J ausgibt:

Name	Inhalt
Sytemout.log	Alle Logs während des Betriebes (bis auf Fehler)
Startserver.log	Logs des Startvorgangs
Systemerr.log	Fehlermeldungen

Tabelle 2: Auftraggebende Unternehmen - Log-Dateien

[69] Vgl. Agopyan, A. (2009), S.14
[70] Vgl. IBM Deutschland GmbH (2012j)

In der Systemout.log befinden sich alle Logs zu dem WAS. Hier werden Konsolenausgaben niedergeschrieben, die der Server während des Betriebes ausgibt. Fehlermeldungen werden in der Systemerr.log-Datei abgelegt. Dennoch können in der Systemout.log-Datei ebenfalls Fehler abgelegt sein. Dazu kommt es, wenn der Server den Fehler nicht als Fehler identifiziert. Aus diesem Grund sollte man nach dem Startvorgang beide Serverlogs lesen, um sicher zu gehen, dass alle Anwendungen auch sauber gestartet wurden. In der Startserver.log-Datei befinden sich alle Logs, die der Server beim Startvorgang schreibt. [71, 72]

3.3.3 Monitoring

Für das Monitoring und Management verwendet das auftraggebende Unternehmen das Produkt IBM Tivoli Monitoring. Tivoli ist eine Firma die seit 1996 eine hundertprozentige Tochter von IBM ist. [73] Neben der Monitoring-Software, die im nächsten Abschnitt untersucht wird, bietet IBM Tivoli eine große Produktpalette [74] rund um die Themen System- und Servicemanagement, [75] beispielsweiße den Tivoli Performance Viewer (TPV). Damit lässt sich der Gesamtzustand des WAS direkt in der Administrationskonsole überwachen. Man muss die Konsole dazu nicht verlassen. [76]

IBM Tivoli Monitoring

IBM Tivoli Monitoring bietet für die Systemüberwachung bewährte Verfahren, die die Identifikation und Lösung von Infrastrukturproblemen gewährleisten. Die Software verwaltet und steuert Datenbanken und Server in dezentralen Umgebungen und Hostumgebungen. [77] Insgesamt bietet das Produkt folgenden Funktionsumfang an: [78]

- Systemressourcenüberwachung
- Dynamische Grenzwertüberwachung
- Visualisierung von Vorfällen und Langzeitansichten
- Überwachung auf zukünftige Kapazitätsengpässe
- Systemüberwachung anhand einer Browserschnittstelle

[71] Vgl. IBM Deutschland GmbH (2012k)
[72] Vgl. IBM Corporation (2012)
[73] Vgl. IBM Deutschland GmbH (2012l)
[74] Siehe dazu: IBM Deutschland GmbH (2012m)
[75] Vgl. IBM Deutschland GmbH (2012n)
[76] Vgl. IBM Deutschland GmbH (2012b)
[77] Vgl. IBM Deutschland GmbH (2012o)
[78] Vgl. ebenda

Dieser Funktionsumfang wird durch das Tivoli Monitoring Services Framework bereitgestellt. Dieses Framework bietet alle wichtigen Komponenten rund um das Thema Monitoring an. Nachfolgend wird die Funktionsweise dieser Komponenten beschrieben. Die Architektur dieser IBM Tivoli Monitoring Komponenten ist in Abbildung 12 dargestellt.

Funktionsweise

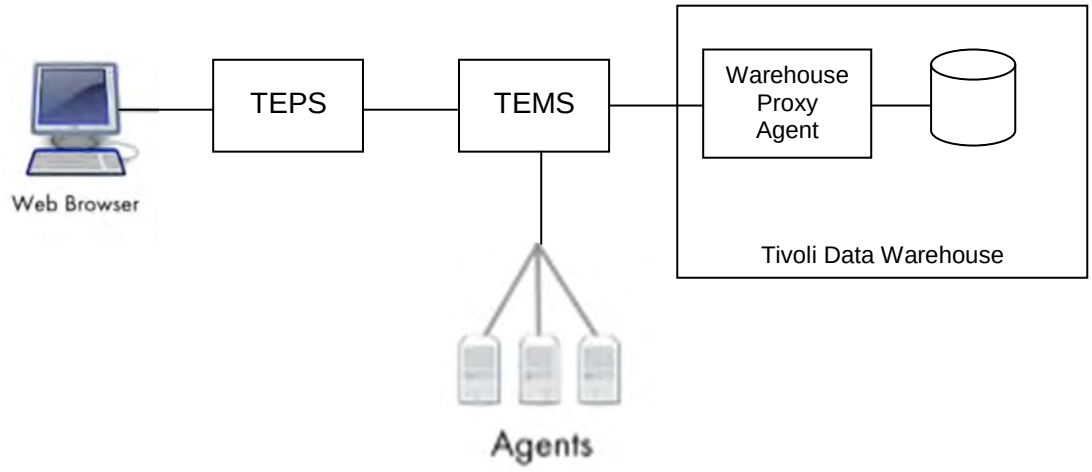


Abb. 12: Architektur des IBM Tivoli Monitoring

Die Kernkomponente für das Monitoring mit dem IBM Tivoli Monitoring ist der Tivoli Enterprise Monitoring Server (TEMS). Alle Tivoli-Komponenten kommunizieren mit diesem Server. Für die Datenbeschaffung und Überwachung werden sogenannte Agents verwendet. Diese werden in der zu überwachenden Umgebung eingesetzt um Daten bzw. Informationen zu senden. Agents können sowohl auf Ebene des Betriebssystems eingesetzt werden, als auch granular pro einzelner Anwendung.

Um die Daten dem Benutzer anzuzeigen verwendet die Software den Tivoli Enterprise Portal Server (TEPS). Durch den TEPS werden dem Benutzer alle Informationen des Systems im Web Browser zentral angezeigt. Der Warehouse Proxy Agent (WPA) hat die Aufgabe alle historischen Daten der verschiedenen Agents zu sammeln und im Tivoli Data Warehouse abzuspeichern. Durch den WPA ist es dem Nutzer möglich aktuellen sowie historische Informationen angezeigt zu lassen. [79]

[79] Vgl. Gucer, V./Godoy, A. (2005), S.1 ff.

3.4 Erweiterbarkeit

3.4.1 Modul-Schnittstellen

Da es sich bei dem WAS um einen zertifizierten JEE-Applikationsserver handelt, besitzt dieser Schnittstellen zu allen Modulen, die für die Spezifikation eines JEE-Applikationsservers wichtig sind. Die Version 7 des WAS ist zertifiziert für die JEE Version 5 und bietet demnach Schnittstellen zu den JEE-Spezifikationen der Version 5. [80] Darüber hinaus besitzt der WAS Schnittstellen zu weiteren Spezifikationen und Produkten von IBM, die auch bei dem auftraggebenden Unternehmen im Einsatz sind zum Beispiel IBM Tivoli Monitoring. [81]

3.4.2 Skalierbarkeit

Um eine höhere Skalierbarkeit bei JEE-Applikationsserver zu erreichen gibt es verschiedene Möglichkeiten, beispielsweise durch zusätzliche Rechenleistung, die Erhöhung der Verfügbarkeit oder die Erhöhung der Effizienz der Aufgabenverteilung. Die Skalierung kann vertikal oder horizontal durchgeführt werden.

Bei der vertikalen Skalierung werden WAS-Instanzen zu derselben Maschine hinzugefügt. Bei der horizontalen Skalierung hingegen werden physische Maschinen hinzugefügt. Durch das Hinzufügen einer Maschine erhöht sich beispielsweise die Leistung eines Cluster-Pools, jedoch nicht eins zu eins, da die Skalierung mit höheren Verwaltungsaufwänden verbunden ist. Eine gute Skalierbarkeit zeichnet hierbei nun die Linearität des Performancezuwachses aus. Je linearer sie ist, umso besser skaliert das System.

Auch nach intensiver wissenschaftlicher Recherche konnte kein Beispiel von der Effizienz der Skalierung des WAS gefunden werden. Aus diesem Grund wird im Folgenden ein Praxisbeispiel aus dem auftraggebenden Unternehmen dargestellt.

Zu Beginn der Aufnahme des KOS-Projektes arbeitete das auftraggebende Unternehmen mit sechs verschiedenen WAS im produktiven Umfeld. Mitte November wurde die Entscheidung gefällt, aufgrund der hohen Auslastung zum Jahreswechsel, zwei weitere Server im produktiven Umfeld einzusetzen. Vor dem Jahreswechsel konnte planmäßig einer der beiden Server erfolgreich installiert und genutzt werden. Zu Beginn des neuen Jahres soll der zweite Server hinzukommen.

[80] Siehe dazu: Anhang 5
[81] Siehe dazu: IBM Deutschland GmbH (2012p)

3.5 Schulungsaufwand

Der Begriff Schulungsaufwand umfasst im Allgemeinen Aufwendungen, die zur Erlernung oder Erneuerung von Wissen bezüglich eines bestimmten Systems, Produkts oder Verfahrens dient. Als Beispiel kann der Produktivitätsverlust durch fehlende Arbeitszeit der Mitarbeiter oder die anfallenden Schulungskosten dienen. Im Nachfolgenden wird kurz der Schulungsaufwand des WAS erläutert.

Neue Mitarbeiter die zum auftraggebenden Unternehmen gelangen werden in der Regel von internen Mitarbeitern geschult. Dennoch werden kurz einige Schulungsarten der IBM zum Thema WAS vorgestellt:

IBM bietet folgende Arten von Schulungen an:

- Firmeninterne Schulungen
- Schulungen in IBM Bildungszentren (u.a. in Stuttgart)
- Verschiedene E-Learning Varianten

Firmeninterne Schulungen können speziell auf die Bedürfnisse des Kunden ausgerichtet und individuell angepasst werden. In IBM Bildungszentren finden Standardschulungen von IBM statt. Diese sind in verschiedensten Städten in Deutschland verfügbar. [82] Des Weiteren bietet die IBM verschiedene virtuelle Schulungen an, bei dem der Schulungsteilnehmer online Zugriff auf alle Schulungsunterlagen erhält und gemäß seiner Auffassungsgabe und Lerntempo selbständig lernen kann. Dabei kann er wählen, ob er online an einer „Klassenraum-Schulung“ teilnimmt, an einer virtuellen und trainergesteuerten Schulung oder selbständig Lernvideos online abruft und sich einlernt. Tests und Übungen helfen bei der Lernerfolgskontrolle. Eine Auflistung aller Standardschulungen findet sich im Trainingskatalog der IBM. [83]

Für das auftraggebende Unternehmen von Interesse sind Schulungen rund um den WAS der Version 7. Auf der Homepage von IBM bekommt man zu diesem Themenkomplex eine vorgeschlagene „Lernroute“ welche Kurse ein Neuling zum Thema WAS 7 belegen sollte:

[82] Siehe dazu: IBM Deutschland GmbH (2013q)
[83] Siehe dazu: IBM Deutschland GmbH (2012r)

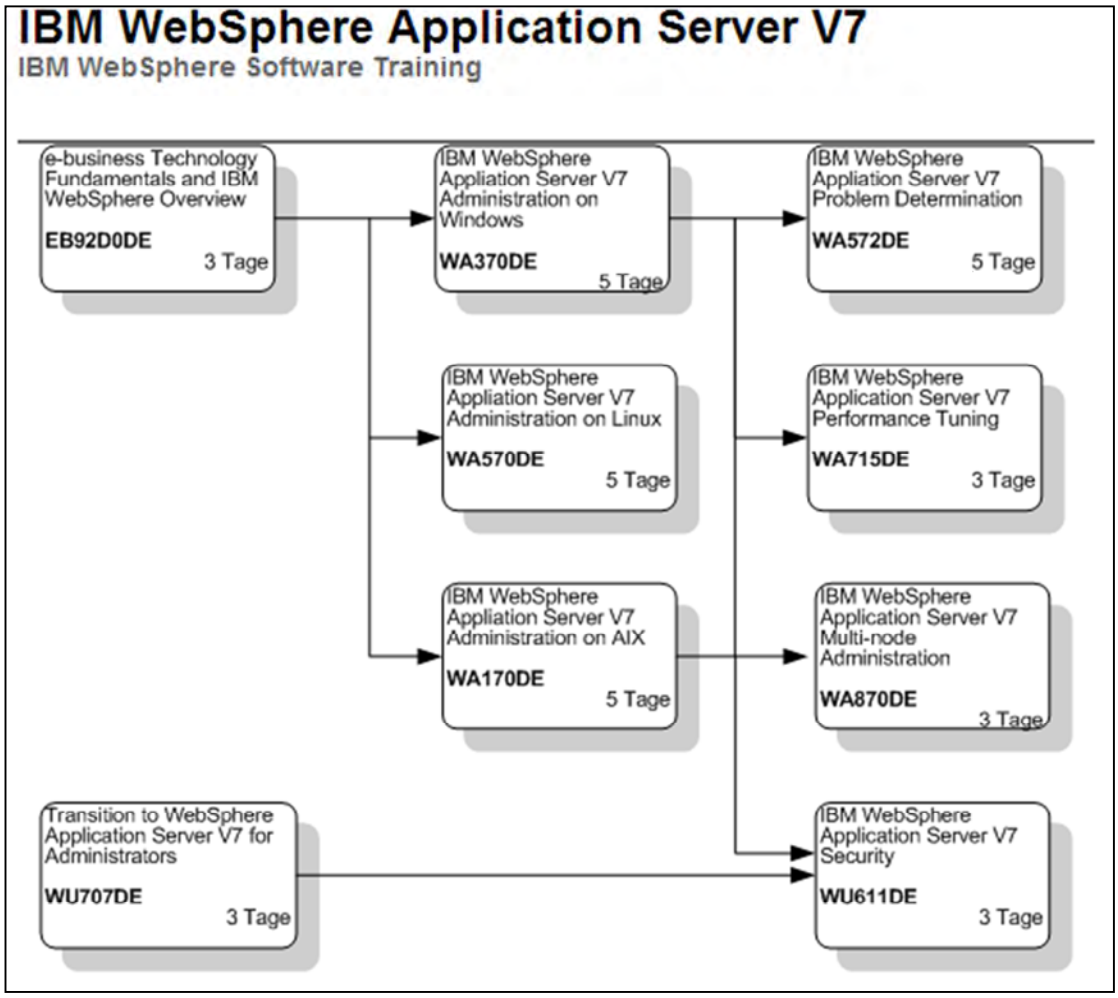


Abb. 13: Lernroute der WAS Version 7

Angefangen mit Grundlagenschulungen werden den Teilnehmern bei dieser Lernroute Wissen zu den Themenkomplexen der Administration, der Problemerkennung, dem Performancetuning und der Sicherheit beigebracht. Zudem kann die Schulung zu dem Themenbereich Administration speziell dem zugrundeliegenden Betriebssystem gewählt werden. Für das auftraggebende Unternehmen wäre somit der Kurs WA370DE sinnvoll.

Die Kosten für diese Schulungen variieren zwischen 1.790 € und 3.700 € [84] und können online abgerufen werden. [85] Es gibt zudem spezielle Rabattprogramme die im Folgenden aber nicht untersucht werden.

[84] Beträge ohne UMST
[85] Beispielhaft: IBM Deutschland GmbH (2012s)

4 JBoss AS

Der Open-Source-JEE-Anwendungsserver JBoss AS ist eine freizugängliche und konforme Alternative zu den kommerziellen JEE-Anwendungsservern, wie dem WAS, und wird weitgehend für die Anwendungsentwicklung und zu Testzwecken eingesetzt.

Die Entwicklung des JBoss AS begann im Jahr 1999 mit einer Vielzahl an aktiven Entwicklern. Kurze Zeit später wurde der JBoss AS unter der Lesser General Public License (LGPL), die von der gemeinnützigen Organisation „Free Software Foundation“ ausgegeben wird, veröffentlicht. Die LGPL ist eine Lizenz, die die Nutzung von einer Open-Source-Software in einer nicht-freien Software, die Weitergabe sowie die Modifizierung einer Open-Source-Software regelt und den kommerziellen Einsatz absichert. [86, 87, 88] Im Anhang 8 befindet sich eine Erläuterung zu den Lizenzen, General Public License (GPL) und LGPL.

Man unterscheidet zwischen der frei verfügbaren JBoss Community Edition (CE) und der JBoss Enterprise Edition (EE), die seit dem Jahr 2006 durch den US-amerikanischer Softwarehersteller Red Hat weiterentwickelt, gewartet und vertrieben wird. Die JBoss CE wird durch den Einsatz der JBoss-Community-Mitglieder, die mit dem Softwarehersteller Red Hat im engen Kontakt stehen, stetig weiterentwickelt. Diese Erweiterungen werden den Kriterien entsprechend genauestens überprüft, beurteilt und in die JBoss EE zusammen mit weiteren Eigenentwicklungen des Softwareherstellers Red Hat eingepflegt. Red Hat versucht die Gegenläufigkeit von Innovation aus der JBoss CE und den Werten für eine Geschäftsfähigkeit, wie zum Beispiel Stabilität, Sicherheit und Kompatibilität, abzuwägen und zu vereinen. [89] Tabelle 3 zeigt die wichtigsten Unterschiede im Detail: [90, 91]

[86] Vgl. Barbara, A (o.J.)
[87] Vgl. Termin, T./Kröger, R. (2006), S.22 f.
[88] Vgl. Pientka, F. (2009), S.23 f.
[89] Vgl. Red Hat, Inc. (2012d)
[90] Vgl. Red Hat, Inc. (2012e)
[91] Vgl. Red Hat, Inc. (2013d)

	JBoss EE	JBoss CE
Langzeitstabilität	• Bietet bis zu 5 Jahre Wartung und Support inklusive Abwärtskompatibilität	• API's können sich unter Releases unterscheiden • Abwärtskompatibilität nicht gesichert
Out-of-Box Erfahrung	• Verschiedene Community Versionen sind integriert und werden in einer Distribution ausgeliefert • Integration von ergänzenden JBoss Produkten ist gesichert • Wartbarkeit wird duch automatische Updates, Warnsystem und Mangement verbessert (z.B. JON)	• Als individuelles Projekt, liegt die Verantwortung zur Zusammenarbeit beim Entwickler • Integration mit anderen Systeme ist alleine die Verantwortung des Realisierers
Patches und Updates	• Abonnement beinhaltet einzelne Patches und Update-Streams für alle Komponenten • Patches, Sicherheitsupdates und Hot-Fixes sind zertifiziert	• Community-Projekte werden nicht ausgebessert, sondern werden insgesamt mit neuen Versionen ersetzt • Einige Fixes werden nie aufgenommen
Erweiterungen	• Releases sind strukturiert und ausgewiesen mit Release Notes, sowie Migrationsdokumentationen	• Code wird der Gemeinschaft für Tests und Feedback freigegeben (Fertigstellung nach mehrern Versionen) • Erweiterungen sind vermischt mit Patches
Sicherheit	• Standardisierter Sicherheitsprozess samt Fallidentifizierung, erforscht und gelöst durch das Red Hat Security Response Team	• Kein formaler Sicherheitsprzoess garantiert, der Sicherheitsprobleme löst
Support	• Industrieller Kunden und Entwickler-Support verfügbar • SLAs, Problemeskalation und End-of-Life-Garantie	• Support über öffentliche Foren

	• Zugriff auf Kunden-Support-Portal • Support durch Code-Entwickler	
Qualitätssicherung	• Qualitätssicherung durch Leistungs und Skalierbarkeittests zum Abschluss für jeden Release	• Kein formaler Quliätssicherungsprozess • Tests sind durch die Anzahl spezifischer Konfigurationen beschränkt
Kompatibilität	• Prüfung und Zertifizierung auf einer Vielzahl von JVM / Betriebssystemen und Datenbanken	• Keine Kompatibilitäszertifizierung
Beratung	• Beratung durch das Red Hat Professional Services Team	• Beratung durch Drittanbieter, jedoch mit Gewährleistungsausschluss
Rechtssicherheit	• Red Hat bietet im Rahmen der „Software Assurance“ Entschädigung und sichert vor bestimmten rechtlichen Risiken	• keine rechtliche Zusicherung wird angeboten

Tabelle 3: Gegenüberstellung der JBoss EE und JBoss CE

4.1 Verfügbarkeit

4.1.1 Deployment

An dieser Stelle wird eine kurze Beschreibung gegeben, welche Arbeitsschritte notwendig sind, um eine webbasierte Anwendung auf dem JBoss AS zum Einsatz zu bringen. Zunächst muss eine Kopie der Anwendungsdatei, zum Beispiel in dem Dateiformat .ear, .war, .sar oder .rar, in das deploy-Verzeichnis abgelegt werden. Anschließend muss der JBoss AS gestartet werden, indem die Batchdatei bin\run.bat ausgeführt wird. Die Bezeichnung der Batchdatei ist vom verwendeten Betriebssystem abhängig. Während diesem Startprozess erkennt der JBoss AS, aufgrund von voreingestellten Konfigurationseinstellungen, dass eine neue Datei für das Deployment vorhanden ist und führt die notwendigen Arbeitsschritte aus, um die Anwendung auf dem JEE-Anwendungsserver zu installieren. [92, 93] Detaillierte Informationen zu diesem Thema finden Sie in der Literatur „JBoss – Das Entwicklerheft“ von

[92] Vgl. Richards, N./Griffith, S. (2006), S.13
[93] Vgl. Langer, T./Reiberg, D. (2006), S. 122 f.

Norman Richards und Sam Griffith. [94] Der umgekehrte Prozess, sprich die Entfernung einer webbasierte Anwendung, die bereits auf dem JBoss AS läuft, ist mit der Löschung der entsprechende Anwendungsdatei aus dem deploy-Verzeichnis umzusetzen. [95]

Der zuvor beschriebene Vorgang kann manuell oder durch ein Build-Skript, welches den Kopiervorgang automatisch durchführt, stattfinden. Laut Expertenaussage besitzt der JBoss AS jedoch eine nicht allzu gute Administrationskonsole, sodass die meisten Vorgänge über die JMX-Konsole geregelt werden. [96] Die JMX-Konsole ist eine Verwaltungskonsole zur Übersicht der Komponenten, wie zum Beispiel JMX-MBeans, des laufenden Servers und zur Konfiguration, zum Starten sowie zum Beenden einzelner Server-Komponenten. [97] Zur Veranschaulichung ist in der folgenden Abbildung 14 die JMX-Konsole dargestellt.

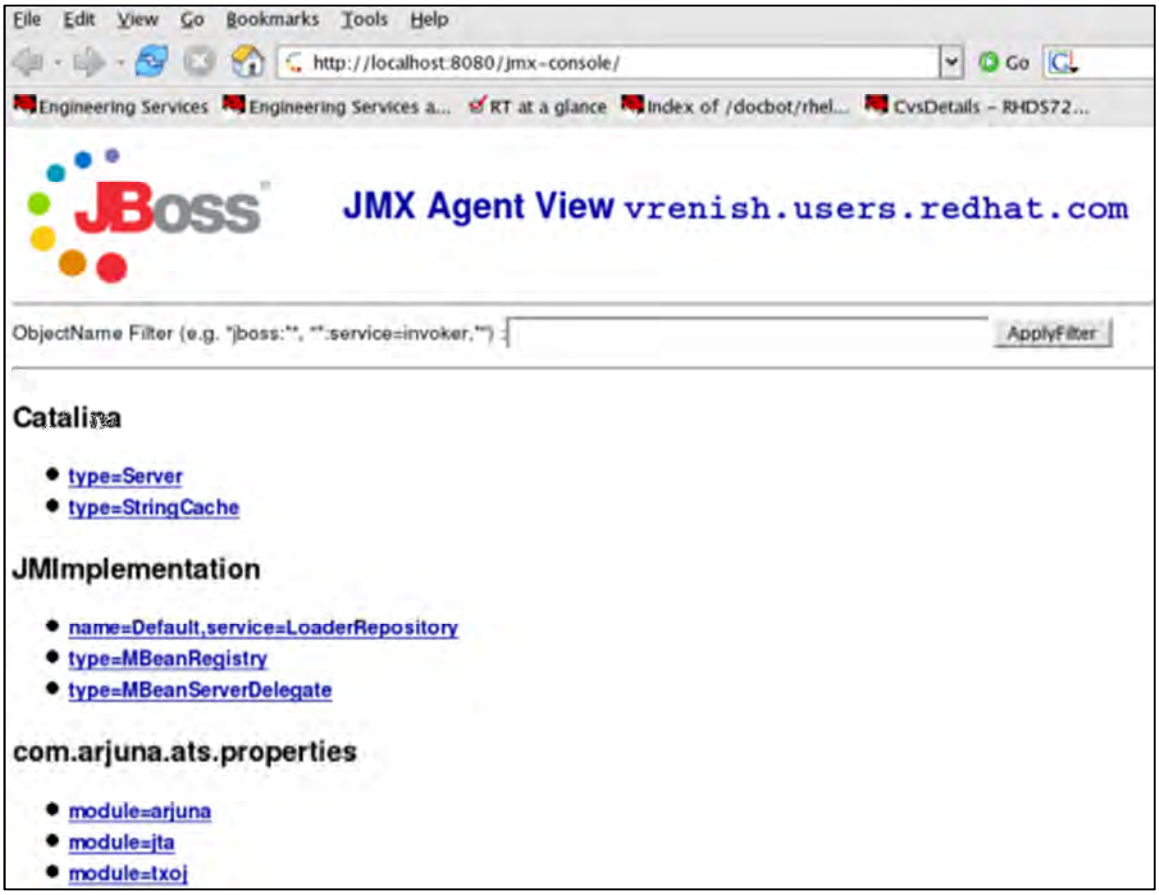


Abb. 14: JBoss AS - JMX-Konsole [98]

[94] Vgl. Richards, N./Griffith, S. (2006)
[95] Vgl. Jamae, J./Johnson, P. (2009), S.26
[96] Vgl. ebenda, S. 26
[97] Vgl. Red Hat, Inc. (2012f)
[98] Enthalten in: Red Hat, Inc. (o.J. c)

An dieser Stelle wird das Werkzeug JBoss Web Manager vorgestellt, der Funktionalitäten, die im Folgenden aufgelistet werden, zur Softwareverteilung bereitstellt. Der JBoss Web Manager ist Bestandteil der JBoss AS-Komponente JBoss Web Server, welche die Webanwendungen bereitstellt [99] und das statische Deployment, die Softwareverteilung bei nicht gestarteten JBoss AS, sowie das dynamische Deployment, die Softwaremodifikation während laufendem JBoss AS, ermöglicht. Es sind keine zusätzlichen Installationen für den JBoss Web Manager notwendig. [100] Wie zuvor erwähnt werden an dieser Stelle einige beispielshafte Funktionalitäten des JBoss Web Manager aufgelistet:

- Deployment einer neuen Webanwendung durch Laden einer Anwendungsdatei
- Deployment einer neuen Webanwendung durch Angabe des Dateipfades
- Auflistung der deployten Webanwendung
- Stoppen einer Webanwendungen, ohne diese vom Server zu entfernen
- Starten von gestoppten Webanwendungen
- Entfernen von Webanwendungen

Hot-Deployment

Der JBoss AS besitzt ebenfalls die Funktionalität des Hot-Deployments. Dies bedeutet, dass ein JEE-Anwendungsserver für eine Neuinstallation, Aktualisierung oder Deinstallation einer Anwendung im laufenden Betrieb nicht angehalten oder neu gestartet werden muss. [101] Dies bietet eine erhebliche Ressourceneinsparung, zum Beispiel die Arbeitszeit aller betroffenen Mitarbeiter.

4.1.2 Support

Die meisten Unternehmen, die sich mit dem möglichen Einsatz eines Open-Source-Produktes auseinandersetzen, haben große Bedenken hinsichtlich der Wartbarkeit und der schnellen Behebung von auftretenden Problemen, da selten ein unterstützender Support angeboten wird. [102]

Bei einer Nutzung des JBoss EE durch ein Unternehmen bietet der Softwarehersteller Red Hat einen kostenpflichten Wartungsvertrag an. Man unterscheidet zwischen dem Supportlevel Premium und Standard. Der Supportlevel Premium umfasst einen Support rund um die Uhr jeden Tag, wohingegen der Supportlevel Standard einen Support von Montag bis Freitag

[99] Vgl. Jamae, J./Johnson, P. (2009), S.125
[100] Vgl. Red Hat, Inc. (2011)
[101] Vgl. Red Hat, Inc. (2012g)
[102] Vgl. CENSIS (2012)

zu den gewöhnlichen Betriebszeiten beinhaltet. Entsprechend dem ausgewählten Supportlevel wird der Kunde bei der Wartung und Weiterentwicklung seiner Systeme unterstützt. Ebenso steht der Softwarehersteller Red Hat bei Systemausfällen dem Kunden mit kurzen Reaktionszeiten zur Seite. Die JBoss CE bietet den Vorteil, dass nicht nur die Nutzung, sondern auch der Support kostenfrei ist. Jedoch wird der Support durch die JBoss Community betrieben. Dieser ist in der Regel von hervorragender Qualität sowie ausreichend für Privatpersonen oder Kleinunternehmen, bei denen ein Systemausfall von beispielsweise einem Tag zu verkraften ist. Bei größeren Unternehmen, wie dem auftraggebenden Unternehmen, laufen geschäftskritische Dienste auf einem JEE-Anwendungsserver. Ein Systemausfall dieses JEE-Anwendungsserver über einen längeren Zeitraum ist mit hohen Kosten verbunden. Weitere Informationen zur Wartung der Produkte von Red Hat sind im Online-Kundenportal vorzufinden. [103]

Ein weiterer Faktor für den Erfolg dieses Open-Source-Produktes, der im Zusammenhang mit dem Support steht, ist die enge Zusammenarbeit zwischen dem Softwarehersteller Red Hat und der JBoss Community bei der Weiterentwicklung des JBoss AS. Im Zuge der Nutzung der JBoss EE erhält der Kunde die Möglichkeit seine gewünschten Anforderungen an den JBoss AS direkt dem Softwarehersteller Red Hat, sowie der JBoss Community, mitzuteilen.

Der Softwarehersteller Red Hat arbeitet mit einer Vielzahl von autorisierten Partnern eng zusammen. Durch das „JBoss Authorized Service Partner“ (JASP)-Programm werden Dienstleistungsunternehmen zertifiziert und erhalten die Erlaubnis einen professionellen JBoss-Service rund um den Weiterverkauf, den Support, die Schulungen und die Beratungen, anzubieten. Das JASP-Programm dient der Zunahme des Angebots professioneller Dienstleistungen und der Verbreitung des Open-Source-Produktes JBoss AS auf dem Markt. Dies bietet dem Unternehmen zusätzlich die Gelegenheit einen Vergleich der vorhandenen Dienstleister anhand unternehmensspezifischer Anforderungskriterien vorzunehmen und infolgedessen den qualifiziertesten und auch kostengünstigsten Dienstleister auszuwählen.[104] Im Anhang 9 ist eine Auflistung einiger zertifizierter Partner, die in Deutschland Ihren Geschäftssitz haben, aufgeführt.

Für ein Unternehmen steht eine komplette Dokumentation über den JBoss AS unter dem folgenden Link http://www.jboss.org/jbossas/docs zur Verfügung. Diese Dokumentation wird durch die JBoss Community erstellt und gepflegt.

[103] Vgl. Red Hat, Inc. (2012h)
[104] Vgl. United News Network GmbH (2012)

4.1.3 Ausfallsicherheit

Der JBoss AS ist aufgrund der Tatsache, dass der Softwarehersteller Red Hat eine qualitative und quantitative Qualitätssicherung vor jeder Auslieferung einer neuen Version des JBoss AS durchführt sehr stabil und ausfallsicher. Implementierte Mechanismen ermöglichen zum Beispiel dem JBoss AS eine selbständige Eigenüberwachung. Diesbezüglich erzwingt der JBoss AS bei unerwarteten Abstürzen von Komponenten einen automatischen Neustart der entsprechenden Komponente, um den laufenden Systembetrieb sicherzustellen.

Das JBoss Operations Network (JON), welches im späteren Kapitel 4.2.1 ausführlicher beschrieben wird, besitzt eine Funktionalität zur Überwachung von JBoss-Server-Instanzen. Bei Störungen im System können diese schnell ausfindig gemacht und gelöst werden. Zusätzlich kann ein Servicetechniker zur weiteren Unterstützung kontaktiert werden. Mit Hilfe dieser zentralen Kontrolle können die Betriebs- sowie Wartungskosten gesenkt und die Betriebseffizienz gesteigert werden. [105] Der JBoss AS ermöglicht ebenfalls eine problemlose Anbindung an eigenentwickelten Überwachungssystemen des Unternehmens oder an weitere Open-Source-Produkte, wenn zum Beispiel nicht auf Werkzeuge des Softwareherstellers Red Hat zurückgegriffen werden möchte. [106]

An dieser Stelle wird kurz auf die Thematik der Quellcodeoffenheit des JBoss AS eingegangen, da diese bei einigen potenziellen Anwendern Bedenken, hinsichtlich des Schutzes von unbefugten äußeren Zugriffen, auslöst. Die große Menge an Entwicklern der JBoss Community, die an der Weiterentwicklung beteiligt sind, überprüft ständig die Qualität des Quellcodes auf Schwachstellen. Dies ist zwar keine Garantie, dass unbefugte äußere Zugriffe möglich sind. Jedoch ist die Wahrscheinlichkeit wesentlich geringer als bei anderen Open-Source-Produkten, die nicht auf diese Vielzahl von Entwicklern, zurzeit 117.200 registrierte Mitgliedern [107], zurückgreifen können.

4.1.4 Clustering

Zurzeit ist das Thema Clustering für das auftraggebende Unternehmen nur ein Randthema, da sich die Bedeutung in der Zukunft ändern kann, wird in diesem Kapitel eine Darstellung des Clustering-Konzeptes im JBoss AS beschrieben.

[105] Vgl. Red Hat GmbH (2012)
[106] Vgl. Red Hat GmbH (2005)
[107] Vgl. Red Hat, Inc. (2013)

Der Cluster eines JBoss AS besteht aus mehreren gruppierten JBoss-Server-Instanzen. Eine JBoss-Server-Instanz kann zu jeder Zeit zu einem Cluster hinzugefügt bzw. entfernt werden. Eine einfache Möglichkeit einen Cluster zu starten, ist der Start von JBoss-Server-Instanzen in einem lokalen Netzwerk mit dem Run-Befehl „run.bat –c all“. Der JBoss AS wird mit drei verschiedenen voreingestellten Server-Konfigurationen ausgeliefert. Man unterscheidet zwischen minimal, default und all. Die Funktionalität des Clusterings ist nur in der all-Konfiguration betriebsbereit. [108]

An dieser Stelle werden nur ein paar ausgewählte Eigenschaften des JBoss AS, in Bezug auf Clustering, aufgelistet und erläutert: [109, 110]

- **Automatische Erkennung von beteiligten Einheiten**
  Die gestarteten JBoss-Server-Instanzen finden sich untereinander und formen automatisch einen Cluster. In der Datei „daploy/cluster-service.xml“ wird festgelegt, welche JBoss-Server-Instanz zu welchem Cluster gehört. Es ist keine zusätzliche Installation notwendig.

- **Dynamische Cluster-Erzeugung**
  Wenn ein entsprechendes Cluster nicht existiert, zurzeit der Hinzufügung einer neuen JBoss-Server-Instanz, wird ein Cluster automatisch gebildet. Dies geschieht ebenfalls in umgekehrter Weise. Ein Cluster wird entfernt, wenn alle JBoss-Server-Instanzen entfernt wurden.

- **Farming**
  Eine Installation bzw. Aktualisierung auf einer JBoss-Server-Instanz wird auf die übrigen JBoss-Server-Instanzen innerhalb eines Clusters automatisch nachgezogen. Diese Funktionalität ist kein Standard und muss selbstständig in einer entsprechenden XML-Datei eingefügt werden. [111] Das Konfigurieren dieser Funktionalität ist zu Beginn mit Arbeitsaufwand verbunden, jedoch bietet diese Funktionalität den Nutzen, dass der zukünftige Administrationsaufwands und die Fehleranfälligkeit bei der Softwareverteilung reduzierte werden kann.

[108] Vgl. VinayTech (2009)
[109] Vgl. Red Hat, Inc (o.J. a)
[110] Vgl. Red Hat, Inc. (2005b)
[111] Vgl. VinayTech (2009)

- **Fail-Over und Load-Balancing für JNDI, RMI und allen EJB-Typen**
  Aufgrund der automatischen Erkennung von beteiligten Einheiten in einem Cluster, ohne zusätzliche Installation, ist es einfach EJB im Cluster betriebsbereit einzustellen. Eine kleine Einstellung in der Datei „jboss.xml“ ist ausreichend für das Ermöglichen von Load-Balance, Fail-Over sowie statische Reproduktion für EJB. [112]

4.2 Administration

4.2.1 JBoss Operations Network

Das JON ist ein Werkzeug zur Abbildung der gesamten Infrastruktur des JEE-Anwendungsservers, welche sich vom Betriebssystem, über die einzelnen Anwendungsserver und Ressourcen der Anwendungen bis hin zu der Schicht des Enterprise Service Bus (ESB). [113] Der ESB ist eine Komponente, die eine lose Kopplung von verteilten Services ermöglicht, indem es ein Message-System aufsetzt und eine Infrastruktur bietet. [114] An dieser Stelle werden ausgewählte Funktionalitäten des JON aufgelistet, die der Anwender einsetzen kann: [115]

- Steuerung der Anwendungsinfrastruktur
- Deployen einer Anwendung
- Starten einer Anwendungsressourcen
- Stoppen einer Anwendungsressourcen
- Konfigurationseinstellungen des JBoss AS
- Überwachung von JBoss-Server-Instanzen

Des Weiteren kann die komplette Middleware-Umgebung und der JBoss AS selbst zentral über eine Stelle gesteuert werden. Dies bedeutet, dass keine weiteren Werkzeuge gebraucht und zusätzliche Kosten vermieden werden. Die Bedienbarkeit des JBoss AS und der damit verbundene Arbeitsaufwand für den Benutzer kann wesentlich verbessert werden, da dieser entsprechende Arbeitsaufgaben über die grafische Oberfläche ausführen kann. Zum Beispiel ist die Konfiguration des JBoss AS nicht mehr über das Anlegen von XML-Dateien notwendig. [116] Abbildung 15 zeigt die Einstiegsseite JONs.

[112] Vgl. Orientation in Objects GmbH (2012b)
[113] Vgl. Red Hat, Inc. (2005), S.2
[114] Vgl. Krechner, S. (2008)
[115] Vgl. Red Hat, Inc. (2005), S.2
[116] Vgl. Red Hat, Inc. (2005), S.2

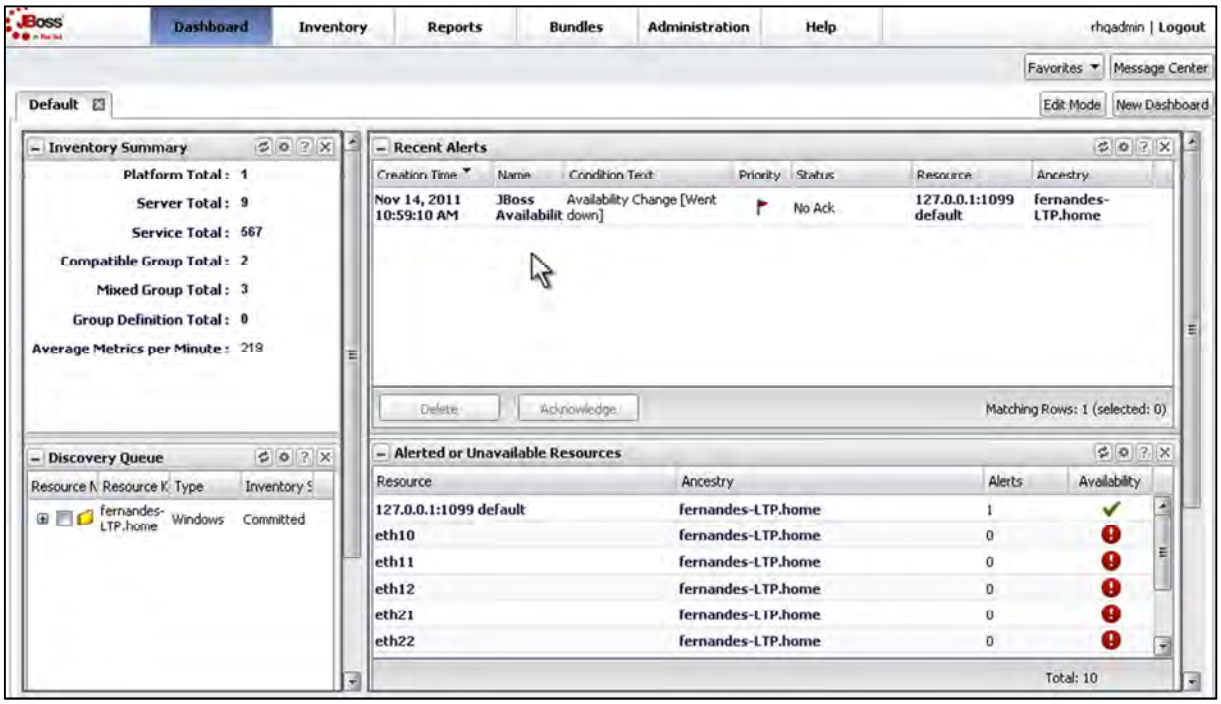


Abb. 15: JBoss Operations Network – Dashboard [117]

Das Monitoring des JONs basiert auf Software-Agenten, die eine integrierte Ansicht der gesamten Middleware-Infrastruktur, inklusive weiterer Abhängigkeiten wie zum Beispiel Betriebssystem und der Webanwendungen, ermöglichen. Software-Agenten sind Programme die mit anderen Software-Agenten in einem System kommunizieren und selbstständig auf entsprechende Situationen reagieren. [118] Diese Software-Agenten des JON erfassen kontinuierlich Betriebs- und Performance-Metriken, die in Statistiken für jede Ressource aufbereitet werden. Mit diesen Statistiken erhält man als Benutzer die Möglichkeit sich über den aktuellen Status der einzelnen Ressourcen zu informieren. [119]

Darauf aufbauend ermöglicht der JON die Definiton von Warnungen, die für das Verhalten der Middleware-Infrastruktur und deren einzelnen Komponenten ausschlaggebend sind. Infolgedessen können zum Beispiel einzelnen Ressourcen, die eine manuelle Aufmerksamkeit benötigen, ermittelt werden. Jedoch können auch vordefinierte Automatismen, beispielsweise der Neustart einer Server-Komponente, bei alarmierenden Messungen greifen. [120]

Eine weitere Funktionalität des JON ist die Historienverwaltung. Der Benutzer kann auf aktuelle, jedoch auch auf historische Daten zugreifen, um zum Beispiel ein früheres Fehlverhal-

[117] Vgl. Red Hat, Inc. (2013b)
[118] Vgl. Springer Fachmedien Wiesbaden GmbH (o.J. a)
[119] Vgl. Red Hat, Inc. (2005), S.2
[120] Vgl. Red Hat, Inc. (2005), S.2

ten zu analysieren oder um präventive Problemlösung zu erstellen. [121] Die Abbildung 16 zeigt eine grafisch aufbereitete Statistik einer ermittelten Ressource – der CPU Belastung.

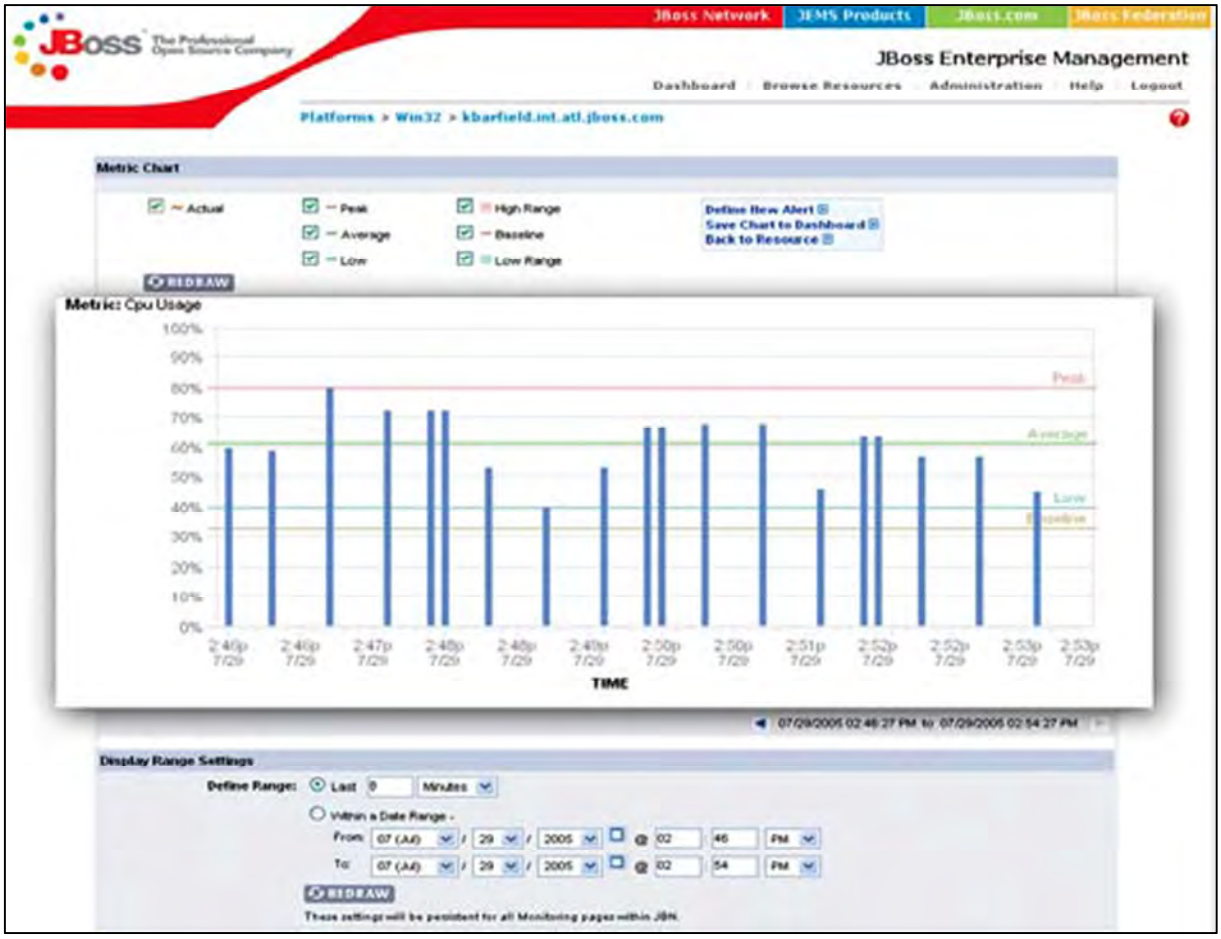


Abb. 16: JBoss Operations Network - Statistik der CPU-Belastung [122]

Der JON besitzt eine Schnittstelle zum Monitoring-Werkzeug Nagios, welches in Kapitel 4.2.4 erläutert wird, da es entsprechendes Erweiterungsmodul implementiert. [123] Dies hat zur Folge, dass das Monitoring-Werkzeug Nagios auch im JON eingesetzt werden kann. Diese Schlussfolgerung ist vor allem für die spätere Gegenüberstellung der beiden Anwendungsservern im Bereich des Monitorings von Bedeutung.

4.2.2 Verwaltbarkeit

Konfiguration

Die Konfiguration des JBoss AS ist benutzerfreundlich und in einer zentralen Konfigurationsdatei hinterlegt. Jegliche Konfigurationsänderungen sind an dieser Stelle vorzunehmen. Es

[121] Vgl. ebenda, S.3
[122] Vgl. ebenda
[123] Vgl. Red Hat, Inc. (2013c)

sind keine weiteren Änderungen außerhalb dieser Konfigurationsdatei notwendig. [124] Eine weitere Besonderheit der aktuellsten Version des JBoss AS ist, dass diese Konfigurationsdatei für mehrere JBoss-Server-Instanzen einsetzbar ist. Bei vorherigen Versionen musste jede JBoss-Server-Instanz einzeln verwaltet werden. Dies beruht unter anderem auf der Farming-Eigenschaft, die in Kapitel 4.1.4 erläutert wurde. Aufgrund der API, die in Kapitel 2.3 dargelegt wurde, sind Änderungen ebenfalls über die Benutzeroberfläche ohne weitere Schwierigkeiten möglich. Konfigurationsänderungen müssen nicht zwingend direkt in der Konfigurationsdatei vorgenommen werden. [125]

Administrator-Konsole

Seit der Version 5 des JBoss AS ist die Administrator-Konsole, die für die Verwaltung und die Beobachtung von JBoss-Server-Instanzen zur Laufzeit zuständig ist, implementiert. Im Laufe der Zeit wurde in den nachfolgenden Versionen diese Administrator-Konsole mit der Technologie des Java Management Extensions (JMX) ersetzt. [126] JMX ist eine Spezifikation um Anwendungen in der Programmiersprache Java verwaltbar zu machen. [127] Diese Technologie offeriert Werkzeuge zur Kontrolle von Anwendungen, technischen Komponenten und prozessangetriebenen Netzwerken, sowie Werkzeuge zur Erstellung von Statistiken und zur Aufführung von Prozessen an Komponenten. Der Bereich der Administration beinhaltet ebenso die Registrierung und die Verteilung von Zugriffsrechten der Benutzer einer Anwendung. Java Authentication and Authorization Service (JAAS) unterstützt diesen Vorgang und ermöglicht eine flexible und skalierbare Authentifizierung. [128] Die Abbildung 17 stellt die Administrator-Konsole des Anwendungsservers der aktuellen JBoss CE dar. Auf der linken Seite ist die Hauptnavigationsleiste vorzufinden. In dem Inhaltsfenster wird die Konfiguration der Datenablegung dargestellt.

[124] Vgl. Red Hat, Inc. (o.J. b)
[125] Vgl. Vertical Media GmbH (o.J.)
[126] Vgl. Oracle (o.J.)
[127] Vgl. Köhler, K. (2003)
[128] Vgl. Dpunkt Verlag (2002)

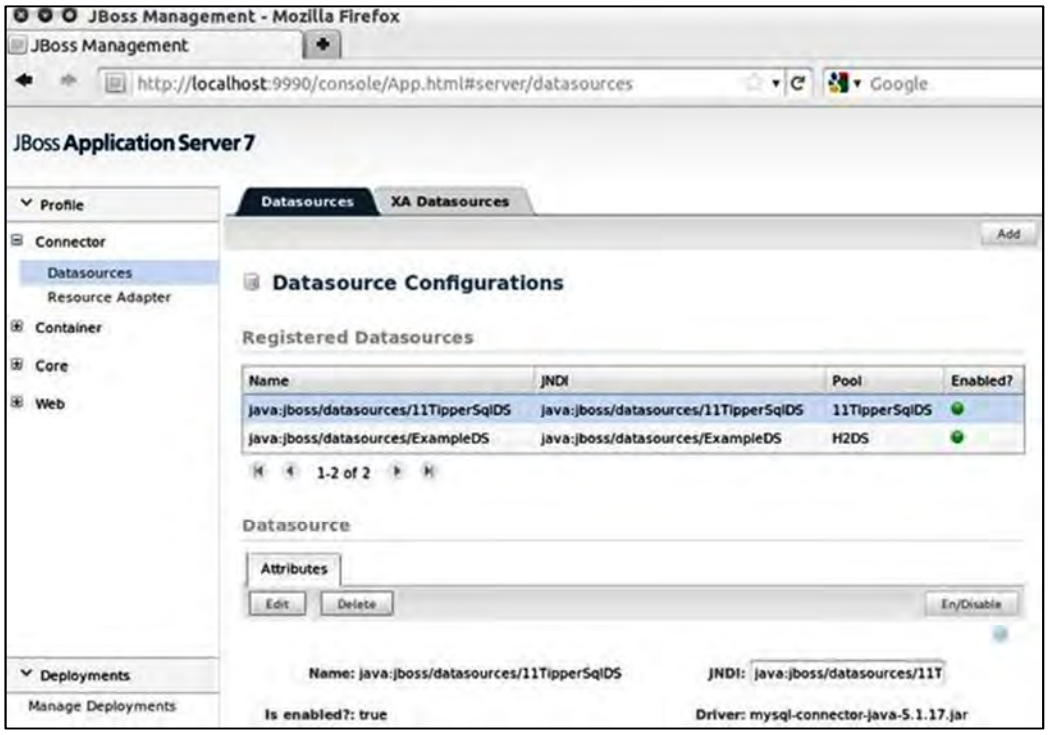


Abb. 17: Administrator-Konsole der JBoss CE (JBoss AS 7) [129]

Abbildung 18 zeigt die Administrator-Konsole des Anwendungsservers der aktuellen JBoss EE. Auf der linken Seite ist die Hauptnavigationsleiste vorzufinden. In dem Inhaltsfenster wird die Datenbankanbindung eingegeben.

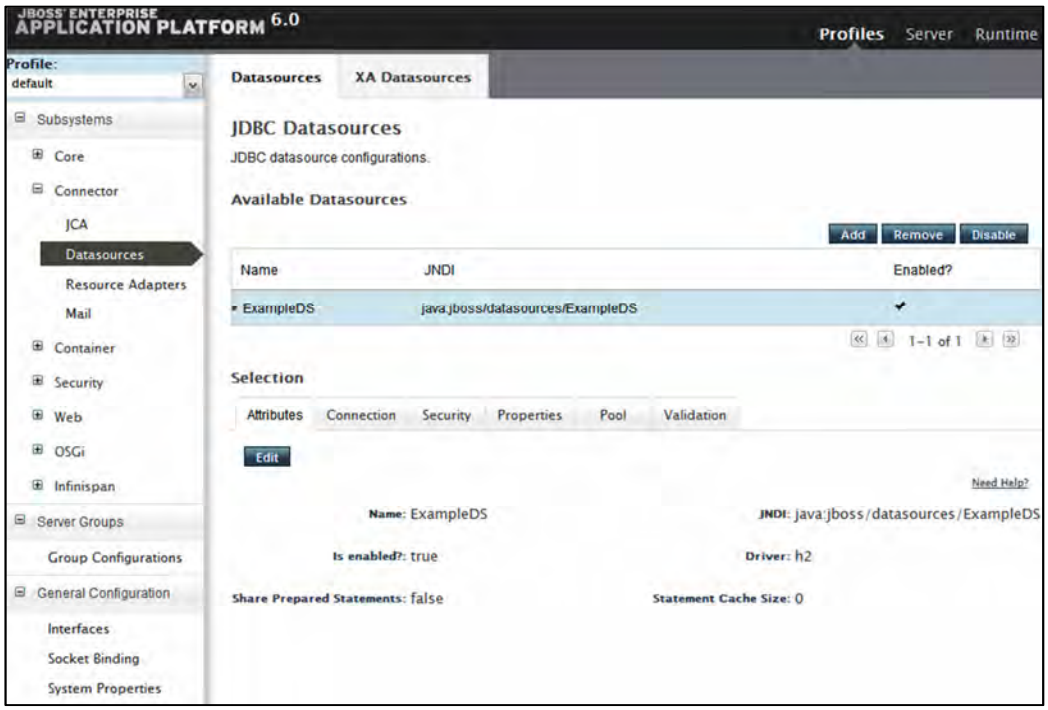


Abb. 18: Administrator-Konsole der JBoss EE (JBoss EAP 5) [130]

[129] Vgl. o.V. (o. J.a)
[130] Vgl. o.V. (o. J.b)

4.2.3 Logging

Der JBoss AS verwendet für das Logging das Open-Source-Logging-Framework Log4J. Die Kontrolle des Loggings ist in der Regel in der Datei „conf/jboss-log4j.xml“ konfiguriert. [131] Diese Datei definiert eine Anzahl von Appenders, die zur Präzisierung der Log-Dateien dienen, sowie das Meldungsformat, die Meldungskategorie und den Filterlevel. Der JBoss AS gibt beispielsweise eine Meldung über die Konsole aus und schreibt diese, bei Auftreten eines Anwendungsfehlers, in die Log-Datei. [132, 133]

Aufgrund der Standardeinstellungen werden während der Ausführung einer Anwendung Meldungen der Kategorie INFO, WARN und ERROR auf der Konsole ausgegeben. Meldungen der Kategorie Debug werden in die Log-Datei geschrieben. Ein Blick in diese Log-Datei hilft somit bei der Fehlererkennung und -behebung.

Zu den Standardeinstellungen lassen sich weitere Logging-Einstellungen konfigurieren. An dieser Stelle werden folgende zwei Punkte kurz erläutert, ohne auf den genauen technischen Ablauf einzugehen:

- **Erstellung einer automatisch generierten Log-Datei**
  Eine Log-Datei wird ab dem Zeitpunkt, an dem der Server gestartet wird, erstellt und wächst bis zu dem Zeitpunkt, an dem der Server beendet wird.
  Diese Vorgehensweise ist für die Produktionsumgebung nicht optimal, da große und unübersichtliche Log-Dateien entstehen. Aus diesem Grund ist die Verwendung einer automatisch generierten Log-Datei empfehlenswert. Beim Erreichen einer bestimmten Größe erzeugt die bisherige Log-Datei eine neue Log-Datei, in die zukünftige Informationen hineingeschrieben werden. Im Anhang 10 ist eine Veränderung des Appenders dargestellt, um eine maximale Anzahl der Server-Log-Dateien von 20 mit einer jeweiligen Maximalgröße von 10 Megabytes zu erzeugen, während der Ausführung einer Anwendung. [134]

- **Definition neuer Log-Datei**
  Die Standardeinstellungen beinhalten, dass alle Meldungen, unabhängig von der Kategorie, in eine Log-Datei geschrieben werden. Dies bietet den Vorteil, dass nur eine

[131] Vgl. Subbaro, M. (2008)
[132] Vgl. Red Hat, Inc. (2005c)
[133] Vgl. Subbaro, M. (2008)
[134] Vgl. Subbaro, M. (2008)

Log-Datei besteht und die Meldungen nicht zerstreut sind. Der Nachteil ist jedoch, dass ein langes und aufwendiges Suchen nach bestimmten Informationen entstehen kann. Die Lösung für dieses Problem ist die Erstellung von mehreren Log-Dateien, die jeweils Meldungen einer bestimmten Kategorie und Priorität enthalten. Im Anhang 11 ist ein Appender dargestellt, der nur Meldungen der Priorität DEBUG in einer Log-Datei speichert.

4.2.4 Monitoring

Zuvor wurde im Kapitel 4.2.1 das JON, inklusive seiner Monitoring-Eigenschaft, dargelegt und es wurde bereits auf das Monitoring-Werkzeug Nagios und die mögliche Einbindung in das JON verwiesen.

In diesem Kapitel wird das Monitoring-Werkzeug Nagios erläutert, da dies von keinem der beiden Unternehmen der zu untersuchenden Anwendungsserver, IBM und Red Hat, entwickelt wurde. Des Weiteren wird Nagios bereits von einigen Unternehmen, wie zum Beispiel Salinen Austria AG, Opsera, DekaBank, Linde und EDEKA Handelsgesellschaft Südbayern mbH, in modifizierter Form eingesetzt. [135, 136, 137] Diese Darstellung wird im späteren Vergleich des WAS und JBoss AS miteinbezogen.

Nagios

Nagios ist ein Monitoring-Werkzeug zur Darstellung und Überwachung von Komponenten eines Netzwerkes, wie beispielsweise Servern, Routern und ablaufenden Systemprozessen. [138] Dieses Monitoring-Werkzeug unterstützt die vollständige Überwachung des JBoss AS und dient daher zur frühzeitigen Erkennung und Beseitigung von Systemfehlern, um die Verfügbarkeit des Systems zu steigern. [139] Auf einzelne Funktionalitäten des Nagios wird an dieser Stelle nicht weiter eingegangen. Die Abbildung 19 zeigt das Monitoring-Werkzeug Nagios bei der Darstellung des Servicestatus der jeweiligen Systemkomponenten. Auf der linken Seite befindet sich die Hauptnavogationsleiste. Ein Service, der einem Host zugeordnet ist, wird im Inhaltsfenster einem Status zugeordnet. Ein Status kann die Ausprägung OK, WARNING, CRITICAL und UNKNOWN annehmen. Des Weitern werden Infomationen über den letzten Eintrag, die Dauer und die Anzahl der Anläufe des Tests angegeben.

[135] Vgl. Pressetext (2013)
[136] Vgl. Heise Zeitschriften Verlag (2013a)
[137] Vgl. NETWAYS GmbH (2013a)
[138] Vgl. NETWAYS GmbH (2012)
[139] Vgl. Nagios Enterprises, LLC. (2012)

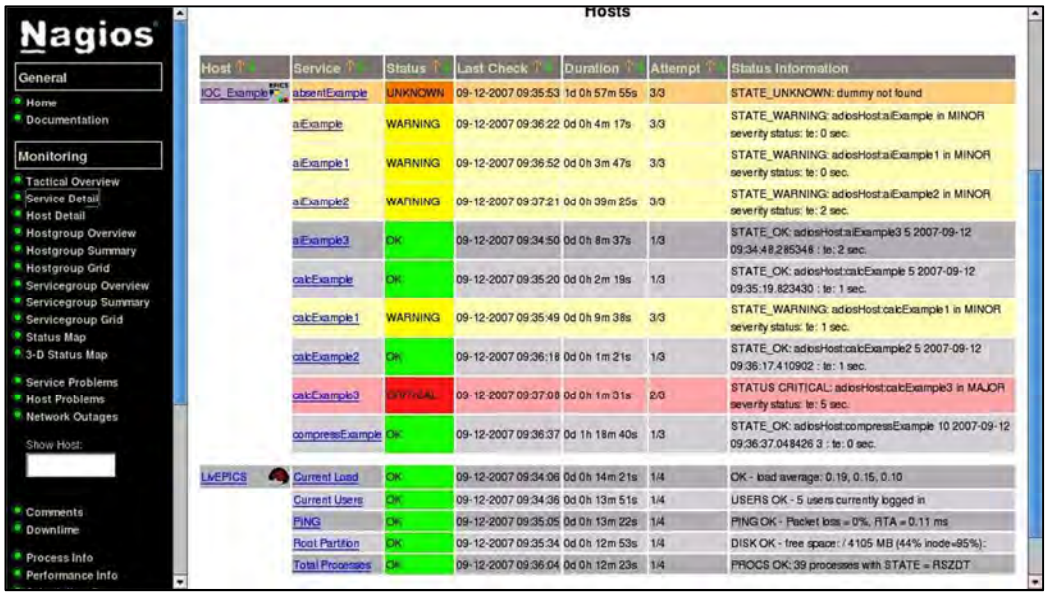


Abb. 19: Nagios - Service Status [140]

Abbildung 20 zeigt ebenfalls das Monitoring-Werkzeug Nagios. Auf der linken Seite befindet sich die Hauptnavogationsleiste. Im Inhaltsfenster werden die Verbindungen der einzelnen Host und Services im Netzwerk dargestellt.

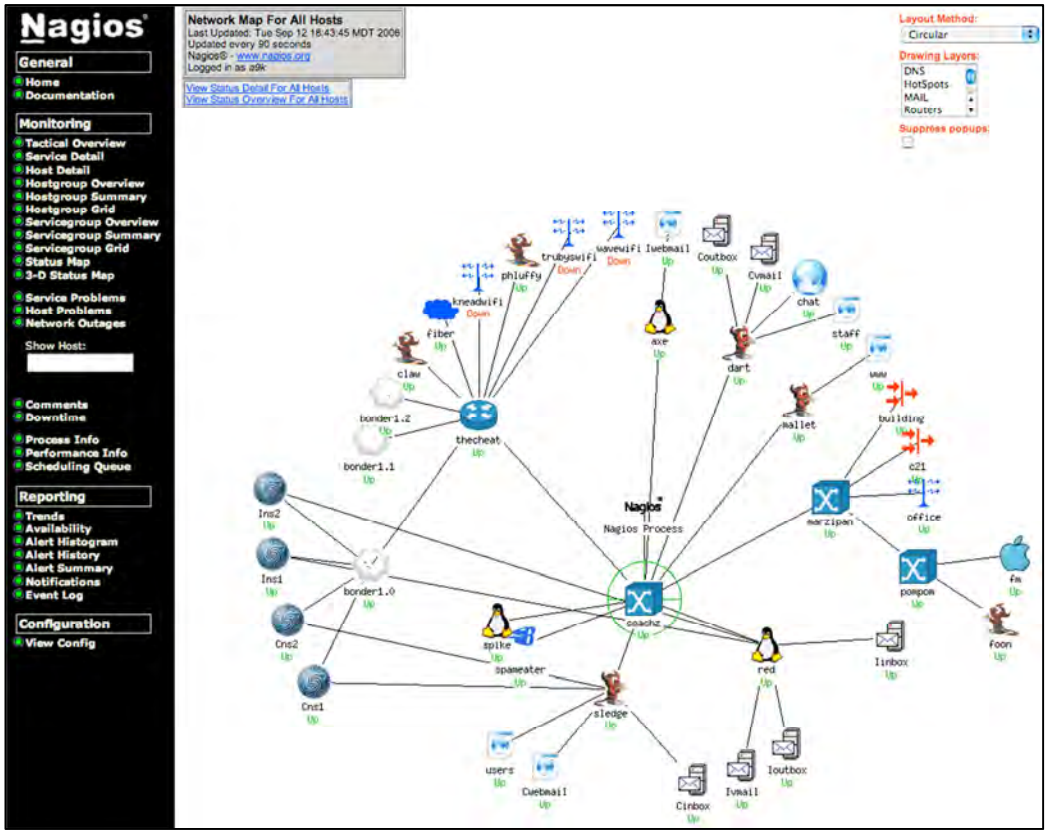


Abb. 20: Nagios - Netzwerkverbindungen [141]

[140] Vgl. o.V. (o. J.c)
[141] Vgl. o.V. (o. J.d)

4.3 Erweiterbarkeit

4.3.1 Modul-Schnittstellen

Der JBoss AS stellt, wie der WAS, dem Anwender Modul-Schnittstellen zur Verfügung, die für die Spezifikation eines JEE-Applikationsservers wichtig sind. In dem Anhang 7 befindet sich eine Tabelle von einigen ausgewählten Programmierschnittstellen der JEE Version 5. Weitere Informationen zum Thema API sind im Kapitel 2.3, Informationen zum Thema JMX sind im Kapitel 4.2.2 vorzufinden. Mit Hilfe dieser Programmierschnittstellen können somit zum Beispiel eigenentwickelte Überwachungssysteme des Unternehmens angebunden werden. Des Weiteren stellt die Integration von zusätzlichen Werkzeugen, wie zum Beispiel dem Programmierwerkzeug Eclipse, keine Schwierigkeit dar. Weitere Informationen zu dem Programmierwerkzeug Eclipse befinden sich in der Online-Eclipse-Dokumentation [142] sowie in der Literatur „Eclipse in der Java-Entwicklung“ von Patrick Kiwitter. [143]

An dieser Stelle wird das frei verfügbare Produkt JBoss-Tools und das kommerzielle Produkt JBoss Developer Studio vorgestellt. Beide Produkte sind von dem Softwarehersteller Red Hat und dienen zur Entwicklung von Webanwendungen, die auf der JSF-Technologie basieren.

Das Produkt JBoss-Tools beinhaltet eine Sammlung von Eclipse-Plugins. Weitere Informationen zu dem Produkt JBoss-Tools sind in der Online-Dokumentation von Red Hat zu finden.[144] Das JBoss Developer Studio basiert auf dem JBoss-Tools und offeriert eine komplette Eclipse-Umgebung mit Installationsprogramm. Darüber hinaus werden weitere Funktionalitäten angeboten, [145, 146] wie zum Beispiel die Service Provider Class Wizard, die eine Erstellung von Java-Klassen mit bestimmten Service-Typen ermöglicht, und die dazugehörigen Service-Dateien in das Metainformationsverzeichnis für Services schreibt. [147] Weitere Informationen zu dem Produkt JBoss Developer Studio sind in dem Datenblatt „JBoss Developer Studio“ von Red Hat enthalten. [148]

[142] Vgl. The Eclipse Foundation (2013)
[143] Vgl. Kiwitter, P. (2008)
[144] Vgl. Red Hat, Inc. (2012b)
[145] Vgl. IRIAN Solutions Softwareentwicklungs- und Beratungsgesellschaft mbH (o. J.b)
[146] Vgl. Red Hat, Inc. (2012c)
[147] Vgl. Heise Zeitschriften Verlag (2013b)
[148] Vgl. Red Hat, Inc. (2012i)

4.3.2 Skalierbarkeit

Die Skalierbarkeit und die Performance eines JEE-Anwendungsservers sind für den Einsatz in einem Unternehmen ein wichtiger Faktor, um die Hochverfügbarkeit der Anwendung zu gewährleisten. An dieser Stelle ist darauf hinzuweisen, dass auf keine wissenschaftlichen Quellen hinsichtlich der Skalierbarkeit und der damit verbundenen Performance zurückgegriffen werden kann. Ebenfalls bestand für die Projektgruppe keine Möglichkeit zur Durchführung einer eigenen Untersuchung, da die benötigten Ressourcen nicht zur Verfügung standen und die Ausarbeitung zeitlich begrenzt war.

Es liegt eine Untersuchung aus dem Jahr 2002 mit dem Titel „Performance and Scalability of EJB Applications“ von Emmanuel Cecchet, Julie Marguerite und Willy Zwaenepoel vor, in der die Performance und Skalierbarkeit einer webbasierten Anwendung auf dem JBoss AS mit verschiedenen Implementierungstechniken geschildert wird. Des Weiteren wurde die Performance des JBoss AS mit den JDK Versionen 1.3 und 1.4 untersucht. [149] Heutzutage steht jedoch bereits die JDK-Version 7 zur Verfügung. Anhand dieser Tatsache zeigt sich, dass diese Untersuchung nicht mehr zeitrelevant ist.

Aus einem Expertengespräch mit einem JBoss-Experte konnten die folgenden Informationen gewonnen werden. Der JBoss AS ermöglicht, dass mehrere Anwendungen auf einer Instanz oder auf mehreren Instanzen auf einer physikalischen Maschine betrieben werden können. Spezielle Anforderungen von Anwendern lassen sich durch diese Gruppierung von JBoss-Server-Instanzen realisieren. Dies bietet den Vorteil, dass technische Werkzeuge allgemeingültig für mehrere JBoss-Server-Instanzen gestaltet werden können. Die Skalierbarkeit ist vor allem von der verwendeten Anwendung eines Unternehmens abhängig. [150]

4.4 Schulungsaufwand

Es werden Schulungen vom Softwarehersteller Red Hat sowie von weiteren Unternehmen, unter anderem von zertifizierten Partnern, auf dem Markt angeboten. Eine Auflistung der zertifizierten Partner in Deutschland befindet sich im Anhang 9. Die Schulungen befassen sich unter anderem mit der Architektur, der Installation und der Konfiguration des JBoss AS. In den Anhängen 12, 13 und 14 befinden sich Zusammenfassungen von drei Schulungen, um eine Vorstellung darüber zu geben, wie eine solche Schulung aufgebaut sein könnte.

[149] Vgl. Cecchet, E./Marguerite, J./Zwaenepoel, W. (2002), S.15
[150] Siehe dazu: Anhang 15

Aus dem Gespräch mit einem JBoss-Experte ging hervor, dass bei der Einführung eines neuen Produktes ein entsprechender Schulungsaufwand notwendig ist. Es empfiehlt sich, die Schulungen individuell an die spezifischen Anforderungen anzupassen, egal ob diese extern oder intern betrieben werden. Die Anwendungen auf dem WAS sind unabhängig von einem JEE-Anwendungsserver implementiert. Infolgedessen sollte bei einer möglichen Migration vom WAS auf den JBoss AS der Aufwand für Schulungen gering ausfallen. Der genaue Schulungsaufwand kann dennoch nicht prognostiziert werden. Ergänzend ist jedoch zu erwähnen, dass die Möglichkeit besteht eine durch den Softwarehersteller Red Hat zertifizierte Prüfung abzulegen. Nach Bestehen dieser Prüfung werden Kenntnisse sowie Erfahrungen mit der Technologie JBoss bestätigt. [151]

Abschließend zu dem Kapitel JBoss AS wird auf den Anhang 22 hingewiesen. Dieser beinhaltet eine erfolgreiche Einführung des JBoss AS anhand des Autoversicherers GEICO, mit Sitz in den Vereinigten Staaten von Amerika.

[151] Vgl. NetMediaInteractive (2012)

5 Gegenüberstellung des JBoss AS und des IBM WebSphere AS

In diesem Kapitel wird zunächst auf die Problematik der Findung von neutralen wissenschaftlichen Quellen zur Ausarbeitung der vorliegenden Studienarbeit eingegangen, bevor eine Gegenüberstellung der beiden JEE-Anwendungsserver in den Bereichen Verfügbarkeit, Administration, Erweiterbarkeit und Schulungsaufwand auf Grundlage der vorherigen Darstellungen vorgenommen wird.

5.1 Neutraler Standpunkt

Bevor eine Auseinandersetzung mit dem zu bearbeitenden Thema stattfinden konnte, hat sich die Projektgruppe mit der Thematik des neutralen Standpunktes beschäftigt. Es zeigte sich, dass eine unvoreingenommene Herangehensweise Voraussetzung für eine wertneutrale Beschreibung ist. Die Bearbeitung muss sachlich, ohne persönlichen Meinungseinfluss, vorgenommen werden. [152]

Die Aufgabenstellung des Projektes sieht die Untersuchung der beiden JEE-Anwendungsserver in Form einer theoretischen Betrachtung vor. Infolgedessen war es der Projektgruppe nicht möglich, aufgrund fehlender Ressourcen und eines beschränkten Zeitrahmens dieser Ausarbeitung, umfassende Installationen bzw. Testszenarien beider JEE-Anwendungsserver vorzunehmen, um eigene und wertfreie Erfahrungen in die Untersuchung mit einfließen zu lassen.

Im Laufe der Auseinandersetzung mit den JEE-Anwendungsservern, WAS und JBoss AS, stellte sich heraus, dass die Findung nach wertneutralen und wissenschaftlichen Quellen recht problematisch ist. Eine große Anzahl an Quellen basieren zum Beispiel auf Forschungen und Darstellungen des Unternehmens IBM oder des Softwareherstellers Red Hat. Diese Quellen sind kritisch zu betrachten, da sie eventuell von bestimmten Bedingungen abhängig und somit nicht vergleichbar sind. Für eine objektive Gegenüberstellung, vor allem im Bereich der Kosten, ist es notwendig neutrale Quellen zu verwenden. Dies ist jedoch in unserer Ausarbeitung, wie bereits erwähnt, nicht der Fall gewesen. Je nach Ausgangsstellung und Gewichtung einzelner Aspekte der jeweiligen Quellen unterschieden sich die Ergebnisse. Dies ermöglichte einen Vergleich in Bezug auf einzelne Teilaspekte, aber nicht auf Grundlage von neutralen Zahlen.

[152] Vgl. gutefrage.net GmbH (2010)

Um diese Problematik zu verdeutlichen, werden beispielhaft zwei Studien des amerikanischen Markforschungsunternehmens Forrester Research genannt. Das Unternehmen untersuchte im Jahr 2009 den wirtschaftlichen Vorteil des JBoss AS und im Jahr 2012 den wirtschaftlichen Vorteil des WAS. Als Grundlage dienten Informationen aus Kundengesprächen der jeweiligen Hersteller, sowie teilweise die US-Bundesbehörde mit ca. 200 Entwicklern, die einen Support der Anwendungen für weitere Behörden gewährt. [153, 154] Bei den Studien wurden die Kosten, die Risiken, die Flexibilität und der Nutzen des jeweiligen JEE-Anwendungsservers, anhand einzelner Teilaspekte, untersucht, um den ROI zu ermitteln. Die Studie aus dem Jahr 2009 ergab einen ROI in Höhe von 63 Prozent für den Einsatz des JBoss AS. Diese Studie wurde von dem Softwarehersteller Red Hat in Auftrag gegeben. [155] Die zweite Studie aus dem Jahr 2012 wies ein Ergebnis eines ROI in Höhe von 44 Prozent aus bei einer Umstellung auf das kommerzielle Produkt WAS. Diese Studie wurde von dem Hersteller IBM in Auftrag gegeben. [156]

5.2 Vergleichsfaktoren

In diesem Kapitel werden die zu untersuchenden Aspekte der JEE-Anwendungsserver gegenübergestellt und für die anschließende Nutzwertanalyse bewertet. Die Bewertung der einzelnen Aspekte erfolgt auf einer Bewertungsskala von 1 bis 5. In dieser Bewertungsskala steht die Bewertung 1 für eine geringe Zielerfüllung, wohingegen die Bewertung 5 eine hohe Zielerfüllung deklariert. Die Bewertung basiert auf intensiver Recherche der Projektgruppe. Zur Veranschaulichung werden in einigen nachfolgenden Unterkapiteln Tabellen eingesetzt. An dieser Stelle werden die eingesetzten Symbole erläutert:

Eingesetzte Symbole	Bedeutung
	Erfüllt die Anforderungen komplett, laut Meinung der Projektgruppe.
	Erfüllt die Anforderungen nur teilweise, laut Meinung der Projektgruppe.
	Erfüllt die Anforderungen nicht, laut Meinung der Projektgruppe.
	Es liegen keine wissenschaftlichen Quellen vor.

Tabelle 4: JBoss AS vs. WAS - Eingesetzte Symbole

[153] Vgl. Forrester Research, Inc. (2012a)
[154] Vgl. Forrester Research, Inc. (2009)
[155] Siehe dazu: Forrester Research, Inc. (2012)
[156] Siehe dazu: Forrester Research, Inc. (2009)

5.2.1 Verfügbarkeit

Deployment

In diesem Abschnitt wird der Deployment-Vorgang beider JEE-Applikationsserver gegenübergestellt und anhand der Tabelle 5 veranschaulicht.

	JBoss AS	WAS	Informationen
Hot-Deplyoment [157]	⇧	⇧	Die Funktionalität des Hot-Deployments bieten beide JEE-Applikationsserver an. [158]
Deploy-Vorgang über eine graphische Benutzeroberfläche	⇧	⇧	Beide JEE-Applikationsserver offerieren an Anwendungen über eine graphische Benutzeroberfläche zu deployen. Bei IBM ist dafür die Integrated Solutions Console zuständig. Bei Red Hat funktioniert dies über die JMX-Konsole oder den JBoss Web Manager. [159]
Deploy-Vorgang über Kommandozeile	⇧	⇧	Beide JEE-Applikationsserver können den Deploy-Vorgang auch klassisch über eine Kommandozeile durchführen.

Tabelle 5: JBoss AS vs. WAS - Deployment

Wie aus der Tabelle 5 ersichtlich ist, bieten beide Produkte die grundlegenden Funktionen für das Deployment an. Im Hinblick auf den Deploy-Vorgang bei dem auftraggebenden Unternehmen, der größtenteils über die graphische Benutzeroberfläche vom WAS abläuft, reicht der hier dargestellte Funktionsumfang beim Deployment vollkommen aus. Beide Produkte erhalten auf Grund der Gegenüberstellung 3 Punkte für die Nutzwertanalyse.

Support

Der Support eines JEE-Anwendungsservers, unabhängig ob Open-Source-Produkt oder kommerzielles Produkt, ist für jedes Unternehmen von hoher Bedeutung. Die nachfolgende Tabelle 6 stellt auf Grundlage der Darstellungen der JEE-Anwendungsserver in Kapitel 3 und 4, einen Vergleich auf.

[157] Siehe dazu: Kapitel 4.1.1
[158] Vgl. Joseph, J. (o.J.)
[159] Siehe dazu: Kapitel 4.1.1

	JBoss AS	WAS	Informationen
Möglichkeit des Standard-Supports	↑	↑	IBM und Red Hat bieten einen Standard-Support für den jeweiligen JEE-Anwendungsserver an.[160] Bei dem JBoss AS ist zwischen JBoss CE und JBoss EE zu unterscheiden. Die JBoss CE ist kostenfrei, wird jedoch von der JBoss Community unterstützt. Die JBoss EE ist kostenpflichtig und wird von Red unterstützt. [161]
Möglichkeit des Premium-Supports	↑	↑	IBM und Red Hat bieten einen Premium-Support rund um die Uhr jeden Tag, für den jeweiligen JEE-Anwendungsserver an. [162]
Vereinbarung individueller Supporteigenschaften	↑	↑	IBM und Red Hat bieten dem Kunden die Möglichkeit zur Vereinbarung individueller Wartungsverträge, angepasst an die Unternehmensanforderungen, an.
Möglichkeit zur Anforderungsstellung	↑	↓	Für Unternehmen besteht die Möglichkeit zur direkten Zusammenarbeit mit Red Hat, um gewünschte Anforderungen an den JBoss AS einpflegen zu lassen. [163] Bei IBM besteht die Möglichkeit zur Anfrage über SRs, die jedoch nach Prioritätshöhe umgesetzt werden oder nicht.
Supportkosten	–	–	Es liegen keine Supportkostenangaben der beiden Hersteller vor.
Supportumfang	↑	↑	IBM und Red Hat bieten umfangreiche Supports an. IBM besitzt z.B. den Support-Lifecycle. Dieser regelt die Wartungsdauer einer bestimmten Version eines Produktes. Zurzeit wird standardmäßig die Option „5+3“ angeboten.

[160] Siehe dazu: Kapitel 3.2.2 und Kapitel 4.1.2
[161] Siehe dazu: Kapitel 4
[162] Siehe dazu: Kapitel 3.2.2 und Kapitel 4.1.2
[163] Siehe dazu: Kapitel 4.1.2

			[164]Red Hat arbeitet z.B. mit einer Vielzahl von autorisierten Partnern zusammen, die einen professionellen JBoss-Support anbieten.[165]
Online-Dokumentation			Beide JEE-Anwendungsserver verfügen über eine ausführliche, sowie qualitative hochwertige Online-Dokumentation. Red Hat: http://www.jboss.org/jbossas/docs IBM: http://pic.dhe.ibm.com/infocenter/wasinfo/v7r0/index.jsp

Tabelle 6: JBoss AS vs. WAS - Support

Auf Grundlage der Gegenüberstellung der beiden JEE-Anwendungsservern zeigt sich, dass beide Produkte einen hervorragenden Support von dem jeweiligen Hersteller erfahren. Ein wesentlicher Nutzen und damit wesentlicher Vorteil des JBoss AS gegenüber dem Produkt WAS ist die Möglichkeit der engen Zusammenarbeit mit Red Hat und die Mitentwicklung des JBoss AS. Individuelle Anforderungen des auftraggebenden Unternehmens könnten somit umgesetzt werden. Infolgedessen erhält der WAS 4 Punkte für die anschließende Nutzwertanalyse und der JBoss AS 5 Punkte.

Ausfallsicherheit

In diesem Kapitel wird das Themengebiet rund um die Ausfallsicherheit der beiden JEE-Applikationsserver gegenübergestellt und anhand der nachfolgenden Tabellen 7 und 8 veranschaulicht. Strenggenommen kann dieses Themengebiet nicht unabhängig untersucht werden, da Überschneidungen mit anderen Themen, wie Clustering, Monitoring und Logging existieren. Monitoring und Logging dienen als Indikator und Melder von Fehlern, die zum Ausfall führen. Clustering ist oft die Grundlage von Diensten, die die Ausfallsicherheit unterstützen. Des Weiteren hängt die Ausfallsicherheit auch immer von der zugrundeliegenden Architektur und von der eingesetzten Anwendung ab, die auf dem JEE-Anwendungsserver läuft. [166]

[164] Siehe dazu: Kapitel 3.2.2
[165] Siehe dazu: Anhang 9
[166] Siehe dazu: Anhang 16

	JBoss AS	WAS	Informationen
Singleton-Service			Sowohl JBoss AS als auch der WAS stellen Services zur Verfügung, die den Single-Point-Of-Failure Character von Singleton-Services entfernen. [167]
Session-Failover			JBoss AS und WAS bieten die Möglichkeit der Serverreplikation, um Sessions wiederherzustellen.
Performance			Genannte Dienste zur Steigerung der Ausfallsicherheit gehen sowohl beim JBoss AS als auch beim WAS auf Kosten der Performance.

Tabelle 7: JBoss AS vs. WAS - Ausfallsicherheit

Unabhängig von den beiden JEE-Applikationsservern			
Anpassen der Anwendungen			Die GUI der NiLS-Anwendung kann bei einem Session-Failover keine Sessionübernahme durchführen, da die Anwendung nicht mit dem komplexen Session Object umgehen kann. Dies muss erst implementiert werden und ist mit großem Aufwand verbunden. [168]

Tabelle 8: Ausfallsicherheit - Anwendung NiLS

Um Singleton-Services zu umgehen, bieten beide Produkte die Möglichkeit, die besonderen Dienste hoch verfügbar zu machen. Beim IBM Webshpere geschieht dies im Rahmen des HAM. Auch JBoss AS bietet beim Deployment an, dass Singleton-Services auf alle Nodes des Clustersystems verteilt werden und nur auf einem sogenannten Master laufen. [169]

Bei einem Ausfall sind beide JEE-Aplikationsserver in der Lage eine Replikation der Session zu erstellen, um diese umgehend wiederherstellen zu können. WAS nennt diesen Dienst Datenreplikationsservice (DRS) und stellt ihn als Bestandteil des HAM zur Verfügung. JBoss AS nennt einen vergleichbaren Dienst „Server Replication“ und bietet diesen Dienst im Rah-

[167] Siehe dazu: Kapitel 3.2.3
[168] Siehe dazu: Kohler, F. (2013)
[169] Vgl. Red Hat, Inc (o.J. d)

men des Projektes JBoss Messaging an. [170] Die Performance wird aber, bei beiden Produkten, durch diese Hochverfügbarkeitsdienste stark negativ beeinflusst. [171, 172] Wie aus den zuvor beschriebenen Aspekten hervor geht, beziehen sowohl der JBoss AS als auch der WAS für das Thema Ausfallsicherheit jeweils 4 Punkte.

Clustering

In diesem Kapitel wird die Methodik des Clusterings der beiden JEE-Anwendungsserver gegenübergestellt und anhand der nachfolgenden Tabellen 9 und 10 veranschaulicht. Wie bereits zuvor erwähnt, nutzt das auftraggebende Unternehmen aktuell die Methodik des Clusterings nicht.

	JBoss AS	WAS	Informationen
Farming			Beim Clustering bietet sowohl JBoss AS als auch der WAS die Möglichkeit an, Anwendungen zentral für das Cluster installieren zu können. Bei dem WAS übernimmt der Job Manager diese Funktionalität. [173]
Dynamisches Clustering			Das dynamische Clustering ist bei JBoss im Funktionsumfang enthalten. Bei dem WAS ist das dynamische Clustering erst ab Version 8.5 in der Standardlizenz enthalten. Das auftraggebende Unternehmen müsste um den Funktionsumfang zu erhalten, WAS Virtual Enterprise lizenzieren und einsetzen. [174]

Tabelle 9: JBoss AS vs. WAS - Clustering

[170] Vgl. Red Hat, Inc (o.J. e)
[171] Vgl. ebenda
[172] Vgl. IBM Deutschland GmbH (2012u)
[173] Siehe dazu: Kapitel 3.3.1
[174] Vgl. IBM Deutschland GmbH (2012t)

Unabhängig von den beiden JEE-Applikationsservern			
Anpassen der Anwendungen			Die GUI der NiLS-Anwendung ist nicht Clusterfähig und muss angepasst werden, um die Eigenschaft Clustering eines JEE-Anwendungsserver zu nutzen. Dies ist mit großem Aufwand verbunden. [175]

Tabelle 10: Clustering - NiLS

Wie aus der Tabelle 10 und der Vorstellung der einzelnen JEE-Applikationsserver aus Kapitel 3 und 4 ersichtlich ist, bieten beide Produkte ausgereifte Unterstützungen für die Thematik des Clusterings an. Das dynamische Clustering ist bei IBM ab der WAS Version 8.5 in der Standard Lizenz enthalten, was für das auftraggebende Unternehmen ein Nachteil darstellt, da aktuell die Version 7 eingesetzt wird. Aufgrund der erlangten Erkenntnisse erhalten beide Produkte jeweils 3 Punkte für den Themenbereich Clustering. Weder der WAS von IBM noch der JBoss AS von Red Hat kann sich entscheidend absetzen.

5.2.2 Administration

Verwaltbarkeit

Die Verwaltbarkeit als ein Unterpunkt der Administration beschreibt die Möglichkeit die JEE-Applikationsserver zu konfigurieren, die Konfigurationen zu ändern oder anzupassen. Hierzu bieten sowohl der WAS, als auch JBoss AS mehrere Werkzeuge an. Grundlage sind bei beiden JEE-Anwendungsserver Konfigurationsdateien. Diese können entweder manuell mittels Texteditor oder durch Skripte erstellt bzw verändert werden.

Darüber hinaus können beide JEE-Applikationsserver über eine grafische Oberfläche konfiguriert werden. Für den WAS ist hierfür die Integrated Solutions Console vorgesehen. JBoss AS bietet dafür mittels einer grafischen Administartor-Konsole diesen Funktionsumfang an. Bei professionellen Unternehmen sollte, wenn möglich, auf die Konfiguration über eine grafische Oberfläche verzichtet werden, da diese Option, genau wie die manuelle Anpassung der Konfigurationsdatei, zu fehleranfällig ist. Die Gefahr liegt vorallem darin, dass Mitarbeiter des Unternehmens einen Fehler eventuell aufgrund von Zeitdruck machen. Daher sollten nur getestete Skripte die Konfiguration anpassen. [176, 177]

[175] Siehe dazu: Kohler, F. (2013)
[176] Vgl. Red Hat, Inc. (2012e)
[177] Vgl. IBM Corporation (2010), S.440 ff.

	JBoss AS	WAS	Informationen
Manuelle Verwaltung über Edetierung			Sowohl JBoss AS als auch der WAS lassen sich über einen Texteditor anpassen.
Manuelle Verwaltung über grafische Oberfläche			JBoss AS und WAS bieten die Möglichkeit, die Konfiguration über eine grafische Oberfläche anzupassen. [178]
Verwaltung über Skript			Beide Produkte können über Schnittstellen Skripte zur Verwaltung durchführen. [179]

Tabelle 11: JBoss AS vs. WAS - Verwaltbarkeit

Da sowohl der WAS als auch der JBoss AS über alle möglichen Werkzeuge zur Konfiguration verfügen, werden für dieses Kriterium der Nutzwertanalyse an beide JEE-Anwendungsserver 4 Punkte vergeben.

Logging

In diesem Kapitel wird die Methodik des Loggings der beiden JEE-Anwendungsservern gegenübergestellt und anhand der Tabelle 12 veranschaulicht.

	JBoss AS	WAS	Informationen
Einsatz von Log4J			Beide JEE-Applikationsserver können das Framework Log4J einsetzen. [180]

Tabelle 12: JBoss AS vs. WAS - Logging

Log4J hat sich im Java-Umfeld zum Standard für das Loggen von Anwendungsmeldungen etabliert. Dieses Framework wird von dem auftraggebenden Unternehmen aktuell eingesetzt. Der JBoss AS verwendet ebenfalls das Log4J. Aus diesem Grund werden beide Produkte im Hinblick auf das Logging-Verfahren mit 3 Punkten bewertet.

Monitoring

In diesem Kapitel werden die Monitoring-Eigenschaften der beiden zu untersuchenden JEE-Anwendungsserver gegenübergestellt. Als Grundlage für den WAS dient das Kapitel 3.3.3

[178] Siehe dazu: Kaiptel 3.3.1 und 4.2.1
[179] Siehe dazu: Kapitel 3.3.1 und IBM Deutschland GmbH (2009)
[180] Siehe dazu: Kapitel 3.3.2 und 4.23

und für den JBoss AS das Kapitel 4.2.4 und 4.2.1, indem der JON vorgestellt wird. Dieser Vergleich wird in folgender Tabelle 13 dargestellt.

	JBoss AS	WAS	Informationen
Modulschnittstellen für den Einsatz von Monitoring-Werkzeugen			Das auftraggebende Unternehmen setzt das Monitoring-Werkzeug IBM-Tivoli-Monitoring [181] ein. Der JBoss AS bietet z.B. die Möglichkeit zur Einbindung von JON [182] oder Nagios. [183]
Überwachung von mehreren Server-Instanzen			Sowohl Nagios als auch IBM-Tivoli-Monitoring können einen oder mehrere Server-Instanzen überwachen.
Proaktive Überwachung			Der IBM-Tivoli-Monitoring sowie der JON können vordefinierte Mechanismen ausführen, wenn z.B. ein Verhalten einer Ressource von der üblichen Norm abweicht.
Einsatz von Agenten			Der IBM-Tivoli-Monitoring und JON verwenden zur Datenbeschaffung und Überwachung des Systems die Software-Agenten.
Anpassbarkeit an unternehmens-spezifische Anforderungen			Das Monitoring-Werkzeug Nagios kann durch vorhandene oder eigenentwickelte Plug-Ins erweitert werden. [184] Tivoli stellt keine Möglichkeit zur Erweiterung von eigenentwickelten Plug-Ins.
Lizenzkosten			Der IBM Tivoli Monitoring ist mit Lizenzkosten verbunden. [185] Der JON hat keine eigenen Lizenzkosten, ist jedoch nur in der JBoss EE verfügbar. Das Monitoring-Werkzeug Nagios ist ebenfalls kostenfrei. [186]

[181] Siehe dazu: Kapitel 3.3.3
[182] Siehe dazu: Kapitel 4.2.1
[183] Siehe dazu: Kapitel 4.2.4
[184] Siehe dazu: ebenda
[185] Vgl. IBM Deutschland GmbH (2012v)
[186] Siehe dazu: Kapitel 4.2.4

Historienverwaltung			Eine Historienverwaltung wird von IBM Tivoli Monitoring und JON zur Verfügung gestellt.
Weitere Funktionalitäten			Es können weitere Funktionalitäten, neben dem Monitoring, vom JON und vom IBM Tivoli Monitoring übernommen werden. JON: [187] • Steuerung der Anwendungsinfrastruktur • Deployen einer Anwendung • Starten einer Anwendungsressource • Stoppen einer Anwendungsressource • Konfigurationseinstellungen des JBoss AS IBM Tivoli Monitoring: [188] • Systemressourcenüberwachung • Dynamische Grenzwertüberwachung • Visualisierung von Vorfällen und Langzeitansichten • Überwachung auf zukünftige Kapazitätsengpässe • Systemüberwachung anhand einer Browserschnittstelle

Tabelle 13: JBoss AS vs. WAS - Monitoring

Die in dieser vorliegenden Studienarbeit behandelten Monitoring-Werkzeuge, Nagios, IBM Tivoli Monitoring und JON, bieten jeweils einen weitreichenden Funktionsumfang, wie Tabelle 13 zeigt, zur Darstellung und Überwachung von Systemkomponenten eines Netzwerks. Zum Beispiel offerieren diese die proaktive Überwachung des Systems.

Eine Differenz zwischen JBoss AS und WAS zeigt sich bei den Lizenzkosten. Der IBM Tivoli Monitoring ist bei einer Anschaffung mit Lizenzkosten verbunden. [189] Der JON sowie das Monitoring-Werkzeug Nagios haben keinerlei Lizenzkosten. Dies ist ein Kostenvorteil für den

[187] Vgl. Red Hat, Inc. (2005), S.2
[188] Vgl. IBM Deutschland GmbH (2012o)
[189] Vgl. Infinite Corporation (2013)

Einsatz des JBoss AS mit einem dieser beiden Werkzeuge. An dieser Stelle ist darauf hinzuweisen, dass für den JON sowie für das Monitoring-Werkzeug Nagios bei einem eventuellen Einsatz die Möglichkeit besteht, Supportverträge abzuschließen. Weitere Informationen zu den Wartungserträgen für das Monitoring-Werkzeug Nagios sind auf der Homepage von NETWAYS GmbH zu finden. [190] Informationen zu Supportmöglichkeiten der Red Hat-Produkten können im Kundenportal von Red Hat nachgelesen werden. [191]

Für das auftraggebende Unternehmen ist bei einer möglichen Umstellung vom WAS auf den JBoss AS wichtig, dass alle Monitoring-Funktionalitäten des IBM Tivoli Monitoring weiterhin abgebildet werden können. Mit Hilfe des Monitoring-Werkzeug Nagios oder dem JON ist dies gegeben. Basierend auf dieser Grundlage erhalten beide JEE-Anwendungsserver jeweils eine Punktzahl von 4 für die anschließende Nutzwertanalyse.

5.2.3 Erweiterbarkeit

Modul-Schnittstellen

In diesem Kapitel wird die Modul-Schnittstellen-Fähigkeit der beiden JEE-Anwendungsserver gegenübergestellt und anhand der nachfolgenden Tabelle 14 veranschaulicht.

	JBoss AS	WAS	Informationen
Basierend auf JEE-Technologie	↑	↑	Der JBoss AS und der WAS basieren auf der gleichen Technologie und den gleichen Spezifikationen. Es liegt lediglich eine Unterscheidung in der Implementierung durch das jeweilige Unternehmen vor.
Erweiterungen für individuelle Werkzeuge	↑	↑	Beide JEE-Anwendungsserver bieten die Möglichkeit zur Einbindung von weiteren Werkzeugen. In dem auftraggebenden Unternehmen ist zurzeit das Monitoring-Werkzeug Tivoli im Einsatz. In den JBoss AS kann z.B. das Produkt JBoss Developer Studio eingebunden werden.
Modularität vs. Out-Off-The-Box	↑	↓	Der JBoss AS bietet die Möglichkeit zur Anpassung des Leistungsumfangs anhand der Auswahl von einzelnen Modulen. Somit

[190] Siehe dazu: NETWAYS GmbH (2013b)
[191] Vgl. Red Hat, Inc. (2013e)

			kann der JEE-Anwendungsserver an die Bedürfnisse des auftraggebenden Unternehmens zielgenau angepasst werden. [192] WAS ist ein Komplettpaket, das sich modular erweitern lässt, jedoch keine Möglichkeiten zur Verringerung der Funktionaliät bietet. [193] Ein Großteil der Funktionalitäten wird vom auftraggebenden Unternehmen nicht verwendet und kann als Ressourcenverschwendung betrachtet werden. Des Weiteren steigt der Wartungsaufwand für dieses komplexe System an.
Schnittstellen zu IBM-Produkten			Aufgrund der Tatsache, dass der WAS ein IBM-Produkt ist, bietet dieser ohne Probleme die Möglichkeit zur Einbindung von weiteren IBM-Produkten, die bei dem auftraggebenden Unternehmen zurzeit im Einsatz sind. Der JBoss AS bietet zwar eine große Anzahl an Modul-Schnittstellen, jedoch kann nicht sichergestellt werden, dass das bei dem auftraggebenden Unternehmen eingesetzte Monitoring-Werkzeug Tivoli eingebunden werden kann. Diesbezüglich liegen keine wissenschaftlichen Quellen vor. Eine mögliche Lösung wäre, dass man über eine enge Zusammenabreit mit Red Hat die gewünschte Anforderung in den JBoss AS implementiert. [194] Des Weiteren wäre eine mögliche Einbindung von Nagios in JBoss AS vorstellbar.

Tabelle 14: JBoss AS vs. WAS - Modul-Schnittstellen

Aufgrund der Tatsache, dass beide JEE-Anwendungsserver auf derselben Technologie basieren und ausschließlich Differenzen in der implementierten Umsetzung vorliegen, bieten

[192] Siehe dazu: Kapitel 4.3.1 und IBM Corporation (o. J.)
[193] Siehe dazu: IBM Deutschland GmbH (2011)
[194] Siehe dazu: Kapitel 4.1.2

beide JEE-Anwendungsserver eine Vielzahl an Modul-Schnittstellen, zum Beispiel zur Erweiterung durch entsprechende Monitoring-Werkzeuge wie dem Nagios, an.

Für das auftraggebende Unternehmen ist bei einer möglichen Migration von WAS auf den JBoss AS der weitere Einsatz des Monitoring-Werkzeugs Tivoli zu hinterfragen. An dieser Stelle ist aufgrund von nicht vorhandenen wissenschaftlichen Quellen keine Aussage darüber zu treffen, ob der Tivoli auf dem JBoss AS eingebunden werden kann. Bei der Tatsache, dass der Tivoli nicht weiter verwendet werden kann, können andere Monitoring-Werkzeuge, wie zum Beispiel JON eingesetzt werden.

Ein wesentlicher Vorteil des JBoss AS gegenüber dem WAS ist die Anpassung des Leistungsumfangs anhand der Konfiguration und Installation von einzelnen Modulen. Ein Modul einer JBoss-Server-Instanz ist leicht austauschbar. [195] Zum Beispiel kann die JDK-Version durch eine aktuellere Version ersetzt werden, was bei dem WAS nicht der Fall ist. Des Weiteren können nichtbenötigte Komponenten des JBoss AS einfach entfernt werden. Dies verbessert die Performance sowie die Wartbarkeit des Anwendungsservers. Der WAS bietet diese Möglichkeit zum modularen Aufbau seiner Komponenten nicht. Infolgedessen können keine Anpassungen, bedarfsgerecht an die Anwendungen, vorgenommen werden und Funktionalitäten bleiben unbenutzt. Es birgt jedoch auch nicht die Gefahr, dass bei übereifriger Optimierung des Anwendungsservers Anwendungsprobleme auftreten.

Anhand der vorherigen Betrachtung erhält der WAS 4 Punkte für die anschließende Nutzwertanalyse und der JBoss AS 5 Punkte. Dies ist vor allem auf die Modularität des JBoss AS zurückzuführen.

Skalierbarkeit

In den vorherigen Darstellungen der beiden JEE-Anwendungsservern wurde bereits erwähnt, dass nach intensiver wissenschaftlicher Recherche keine Informationen hinsichtlich der Skalierbarkeit und der damit verbundenen Performance eines JEE-Anwendungsserver gefunden werden konnten. Des Weiteren bestand für die Projektgruppe keine Möglichkeit zur Durchführung einer eigenen Untersuchung, um ein Ergebnis vorzuzeigen.

Der Vergleich zwischen dem JBoss AS und dem WAS in Bezug auf Skalierbarkeit ist aufgrund der gerade genannten Problematik daher nicht möglich. Ergänzend ist noch darauf hinzuweisen, dass die Skalierbarkeit von einigen Faktoren der Anwendungsumgebung, wie zum Beispiel vom Volumen der Eingangsdaten, von der Anzahl der Knoten in einem Server-

[195] Vgl. Software & Support Media GmbH (2011)

Cluster, der Anzahl der konkurrierenden Benutzer, sowie von der Rate der ankommenden Systemaufträge, abhängig ist. [196] Aus diesem Grund ist eine Aussage in Bezug auf das auftraggebende Unternehmen nicht ohne weitere Analysen, die an die Anwendungsumgebung angepasst sind, möglich.
Basierend auf der Grundlage, dass beide Hersteller, IBM und Red Hat, jeweils die Thematik Skalierbarkeit definieren, aber nicht detailliert den Nutzen des jeweiligen JEE-Anwendungsserver vorstellen, erhalten beide Anwendungsserver die Punktzahl 3 für die folgende Nutzwertanalyse.

5.2.4 Schulungsaufwand

In der vorliegenden Studienarbeit wurde in den Darstellungen der beiden JEE-Anwendungsserver, JBoss AS und WAS, auf die jeweiligen Schulungen Bezug genommen. Es zeigte sich, dass eine große Anzahl unterschiedlichster Schulungen von beiden Herstellern, sowie von weiteren Dienstleistungs- und zertifizierten Partnerunternehmen auf dem Markt angeboten werden. Die Schulungen lassen sich jeweils individuell an die spezifischen Anforderungen des Unternehmens anpassen. Die Höhe der Kosten ist nicht klar zu benennen, da diese je nach Schulungsanforderung, Schulungsdauer, Schulungsziel und dem entsprechenden Dienstleister variieren. Aus diesem Grund erhält der JBoss AS für den Bereich Schulungsaufwand in der anschließenden Nutzwertanalyse die Punktzahl 4 und der WAS die Punktzahl 4.

Im Anhang 17 findet sich eine Gegenüberstellung von zwei Schulungen der beiden JEE-Anwendungsservern, die vom Unternehmen Ordix AG, einer Aktiengesellschaft für Softwareentwicklung, Schulung, Beratung und Systemintegration, durchgeführt werden. Beide Schulungen verfolgen das Ziel der Vermittlung von Wissen über den jeweiligen JEE-Anwendungsserver und die Schulungsdauer beträgt jeweils 3 Tage. Die Schulungen unterscheiden sich jedoch in Voraussetzungen, Schulungsdauer, Seminarinhalt sowie in den Kosten. [197]

[196] Vgl. Schindler, H. (2011), S.9
[197] Vgl. Ordix AG (2012)

6 Nutzwertanalyse

In diesem Kapitel wird eine Nutzwertanalyse, die ein nicht monetäres Bewertungsverfahren darstellt, auf Basis der vorherigen Darstellungen und der anschließenden Gegenüberstellung der beiden JEE-Anwendungsservern vorgenommen. Das Verfahren schafft einen Bewertungsrahmen, der zur Entscheidungshilfe einer möglichen Migration vom WAS auf den JBoss AS beiträgt, indem entscheidungsrelevante, nicht monetäre Aspekte subjektiv bewertet und transparent vermittelt werden. [198]

Die Nutzwertanalyse ist eine weitverbreitete Variante, die eine additive Nutzensynthese zu Grunde legt, und auf die sich in dieser Studienarbeit bezogen wird. [199] Weitere Informationen zu dieser Vorgehensweise sind in der Literatur „Nutzwertanalyse in der Systemtechnik“ von Christof Zangenmeister zu finden. [200] Der Nutzwert einer Entscheidungsalternative wird aus der Summe der definierten Teilnutzen, die von der bewertenden Person definiert und nach ihrer Relevanz gewichtet werden, gebildet. [201]

6.1 Aufbau

Der strukturierte Aufbau der Nutzwertanalyse erfolgt auf Grundlage der Untersuchung der beiden JEE-Anwendungsserver und orientiert sich an der Gliederung der vorliegenden Studienarbeit. Die Abbildung 21 stellt die Nutzwertanalyse in Form einer Tabelle dar.

Hauptkriterium	Gewichtung	Unterkriterium	Gewichtung	JBoss		Websphere	
				Teil-nutzen	gew. Nutze	Teil-nutzen	gew. Nutzen
Verfügbarkeit (A)		Deployment (A1)					
		Support (A2)					
		Ausfallsicherheit (A3)					
		Clustering (A4)					
Administration (B)		Verwaltbarkeit (B1)					
		Logging (B2)					
		Monitoring (B3)					
Erweiterbarkeit (C)		Modulschnittstellen (C1)					
		Skalierbarkeit (C2)					
Schulungsaufwand (D)		Schulungsaufwand (D)					

Abb. 21: Nutzwertanalyse

[198] Vgl. Springer Fachmedien Wiesbaden GmbH (o.J. b)
[199] Vgl. Zangenmeister, C. (1976), S.55 ff.
[200] Siehe dazu: Zangenmeister, C. (1976)
[201] Vgl. Zangenmeister, C. (2000), S.122

Die Gewichtung der Haupt- und Unterkriterien erfolgt anhand definierter Prozentpunkte, die durch das auftraggebende Unternehmen benannt wurden, um ein unterschiedslose Bewertungsskala schaffen zu können. Dies bietet den Vorteil, dass die Nutzwertanalyse zu einem späteren Zeitpunkt für weitere JEE-Anwendungsserver, wie zum Beispiel Apache Geronimo, angewendet werden kann und die Anpassung der Gewichtung, zum Beispiel aufgrund geänderter unternehmensbezogener Anforderungen, leicht anpassbar ist.

6.2 Gewichtung

Die Verteilung der Prozentpunkte der einzelnen Bewertungskriterien erfolgt auf Grundlage einer Priorisierung der einzelnen Aspekte durch das auftraggebende Unternehmen Die zu untersuchenden Aspekte wurden von Stufe 1 bis 4 eingeteilt. In diesem Fall stellt die Stufe 4 den wichtigsten zu untersuchenden Aspekt und die Stufe 1. den am wenigsten wichtigsten zu untersuchenden Aspekt dar. Aufgrund dieses Schemas wurden die Haupt- und Unterkriterien wie folgt priorisiert:

Hauptkriterien	**Priorisierung**
Verfügbarkeit	4
Administration	3
Erweiterbarkeit	2
Schulungsaufwand	1

Tabelle 15: Hauptkriterien - Priorisierung

Unterkriterien	**Priorisierung**
Deployment	2
Support	4
Ausfallsicherheit	4
Clustering	2
Verwaltbarkeit	2
Logging	2
Monitoring	4
Modulschnittstellen	1
Skalierbarkeit	4
Schulungsaufwand	1

Tabelle 16: Unterkriterien - Priorisierung

Bei 100 zu vergebenen Prozentpunkten ergeben sich für die Haupt- und Unterkriterien die folgenden Prozentpunkte:

Hauptkriterien	**Prozentpunkte**
Verfügbarkeit	8 + 15 + 15 + 8 ≈ 46
Administration	8 + 8 + 15 ≈ 31
Erweiterbarkeit	4 + 15 ≈ 19
Schulungsaufwand	≈ 4

Tabelle 17: Hauptkriterien - Prozentpunkte

Unterkriterien	**Prozentpunkte**
Deployment	≈ 8
Support	≈ 15
Ausfallsicherheit	≈ 15
Clustering	≈ 8
Verwaltbarkeit	≈ 8
Logging	≈ 8
Monitoring	≈ 15
Modulschnittstellen	≈ 4
Skalierbarkeit	≈ 15
Schulungsaufwand	≈ 4

Tabelle 18: Unterkriterien - Prozentpunkte

An dieser Stelle wird ein Rechenbeispiel für die Vergabe der Prozentpunkte der Unterkriterien anhand der Priorisierung zur Verdeutlichung dargestellt:

1. Schritt: Addition der Priorisierung

2 + 4 + 4 + 2 + 2 + 2 + 4 + 1 + 4 + 1 = 26

2. Schritt: Errechnung des Prozentpunktes

100/26 = 3,84 ≈ 4

2. Schritt: Verteilung der Prozentpunkte

Deployment besitzt Priorisierungsstufe 2.

2 * 3,84 = 7,68 ≈ 8

6.3 Ergebnis

Der daraus generierte Nutzwert drückt aus, wie gut ein Nutzen die subjektiven Zielvorgaben erfüllt. Je größer, desto optimaler die Zielerfüllung.

Hauptkriterium	Gewichtung	Unterkriterium	Gewichtung	JBoss		Websphere	
				Teil-nutzen	gew. Nutzen	Teil-nutzen	gew. Nutzen
Verfügbarkeit (A)	46%	Deployment (A1)	8%	3	0,24	3	0,24
		Support (A2)	15%	5	0,75	4	0,60
		Ausfallsicherheit (A3)	15%	4	0,60	4	0,60
		Clustering (A4)	8%	3	0,24	3	0,24
Administration (B)	31%	Verwaltbarkeit (B1)	8%	4	0,32	4	0,32
		Logging (B2)	8%	4	0,32	4	0,32
		Monitoring (B3)	15%	4	0,60	4	0,60
Erweiterbarkeit (C)	19%	Modulschnittstellen (C1)	4%	5	0,20	4	0,16
		Skalierbarkeit (C2)	15%	3	0,45	3	0,45
Schulungsaufwand (D)	4%	Schulungsaufwand (D1)	4%	4	0,16	4	0,16
				Summe :	3,88	Summe :	3,69

Abb. 22: Nutzwertanalyse - Auswertung

Das Ergebnis der Nutzwertanalyse zeigt, dass JBoss AS mit einem Nutzwert von 3,88 und der WAS mit einem Nutzwert von 3,69 bewertet wurden. JBoss AS besitzt somit eine optimalere Zielerfüllung als der JEE-Aplikationsserver von IBM. Diese höhere Zielerfüllung fällt mit einer Differenz von 0,19 Punkten aber gering aus.

Die Unterschiede liegen in den beiden Kriterien Support und Modulschnittstellen. Wie bereits in der Gegenüberstellung der beiden JEE-Applikationsserver beschrieben, sind die Gründe hierfür die Verfügbarkeit von Drittanbietern beim Support, sowie die Modularität des JBoss AS.

Das Ergebnis steht zudem exemplarisch für den Aufwind der Open-Source-Produkte in dem Umfeld der IT. Das Ergebnis zeigt, dass das kommerzielle Produkt im JEE-Applikationsserver-Umfeld keine funktionale Exklusivität besitzt, die nicht durch ein Open-Source-Produkt abgebildet werden kann. Beide Konzerne versuchen durch offensive Marketingstrategien, die Exklusivität und Funktionsvielfalt ihrer Produkte hervorzuheben. Bei näherer und kritischer Betrachtung unterscheiden sich diese Produkte aber nur in unwesentlichen Nuancen.

Die Unterschiede sind grafisch in den folgenden Abbildungen 23 und 24 dargestellt. Die Abbildung 23 zeigt das Ergebnis anhand der Priorisierung des auftraggebenden Unternehmens.

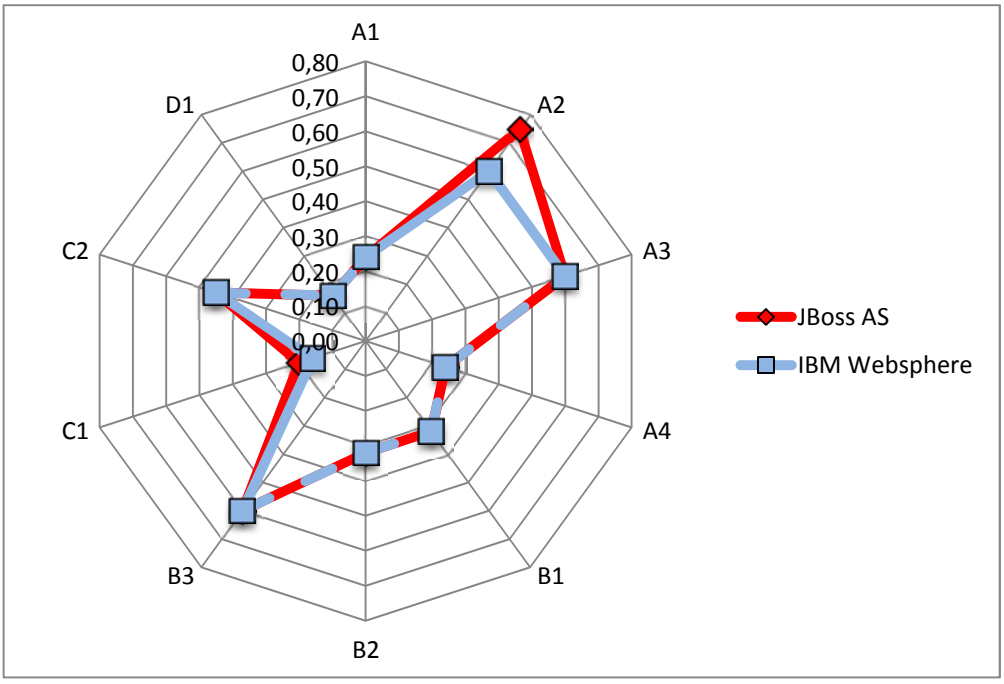


Abb. 23: Netzdiagramm - Mit Gewichtung

Die Abbildung 24 stellt die vom Projektteam definierten Teilnutzen für die einzelnen Kriterien, unberücksichtigt der Priorisierung des auftraggebenden Unternehmens, dar.

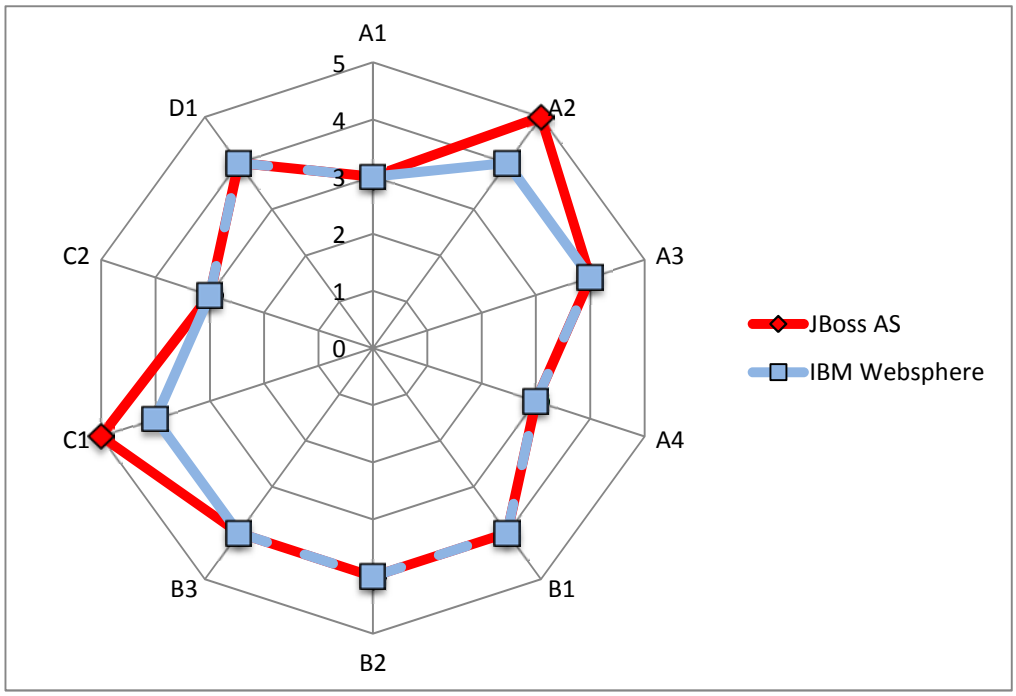


Abb. 24: Netzdiagramm - Ohne Gewichtung

7 Kostenanalyse

In diesem Kapitel sollte eine unvoreingenommene Kostenanalyse durchgeführt werden, die zur Entscheidungshilfe eines Einsatzes der beiden JEE-Anwendungsserver dienen soll. Zu diesem Zweck wurde, neben wissenschaftlichen Quellen, der direkte Kontakt zu den Herstellern gesucht, um eine Auflistung der Kosten, wie zum Beispiel Supportkosten, Lizenzkosten und Schulungskosten, zu erhalten. Man erhielt leider nur die Mitteilung vom Hersteller IBM, dass diese Informationen nur für IBM-Studenten oder potenzielle Kunden zur Verfügung stehen. Der Softwarehersteller Red Hat hat zu dieser Kontaktanfrage bisher keinen Kommentar abgegeben. [202] Des Weiteren war das auftraggebende Unternehmen nicht bereit genaue Kostenzahlen für die Lizenzen, sowie für den Support zu nennen. Infolgedessen bestand keine Möglichkeit eine unabhängige Kostenanalyse durchzuführen.

Aufgrund dieser Tatsache findet an dieser Stelle eine jeweilige kurze Vorstellung von zwei Studienarbeiten statt, in der sich unter anderem mit der Thematik der anfallenden Kosten beschäftigt wurde. Es ist klar darauf hinzuweisen, dass die beiden Studienarbeiten kritisch hinsichtlich des neutralen Standpunktes, zu betrachten sind. Es besteht die Vermutung, dass das jeweilige Produkt vom Hersteller gegenüber dem Konkurrenzprodukt besser dargestellt werde.

7.1 Studienarbeit von Summa Technologies [203]

Das amerikanische Unternehmen Summa Technologies, Inc. führte im Jahr 2009 eine Gesamtbetriebskosten-Analyse durch und veröffentlichte diese unter dem Titel „IBM WebSphere Application Server V7 vs. JBoss Application Server V5 TCO Analysis“. Diese Studienarbeit wurde von Rick Kotermanski, Jason Armstrong, Matt Holloway und Roman Kharkovski (IBM-Mitarbeiter) verfasst.

In dieser Studienarbeit werden die Gesamtbetriebskosten in Hardware, Software-Lizenzen, Applikation Management, Infrastruktur Management, Software Support und Training eingeteilt, die in der folgenden Tabelle 19 erläutert werden. Es wird nicht auf die Kosten des Deployments eingegangen, da es auf Grund der Variabilität der Unternehmen, zum Beispiel in Bezug auf Mitarbeitergröße, Fähigkeiten, sowie Erfahrungen, schwierig ist, eine einheitliche Kostendarstellung diesbezüglich vorzunehmen.

[202] Siehe dazu: Anhang 18-21
[203] Vgl. Kotermanski, R. (2009)

Kostenbereiche	Beinhaltete Elemente
Hardware	• Beschaffung der Hardware • Support der Hardware
Infrastruktur Management	• Support der Infrastruktur
Software-Lizenzen	• Beschaffungslizenz
Applikation Management	• Installation, Konfiguration und Aufrüstung der Software • Deployment und Monitoring von Anwendungen • Unerwartete Ausfallzeiten und anschließende Wiederherstellung
Software Support	• Wartungsvertrag
Training	• Ausbildung und Weiterbildung für Administratoren und Entwickler

Tabelle 19: Summa Technologies - Kostenbereiche

In der folgenden Grafik wird das Verhältnis der Kostenbereiche aufgezeigt, das auf Aussagen des Herstellers beruht.

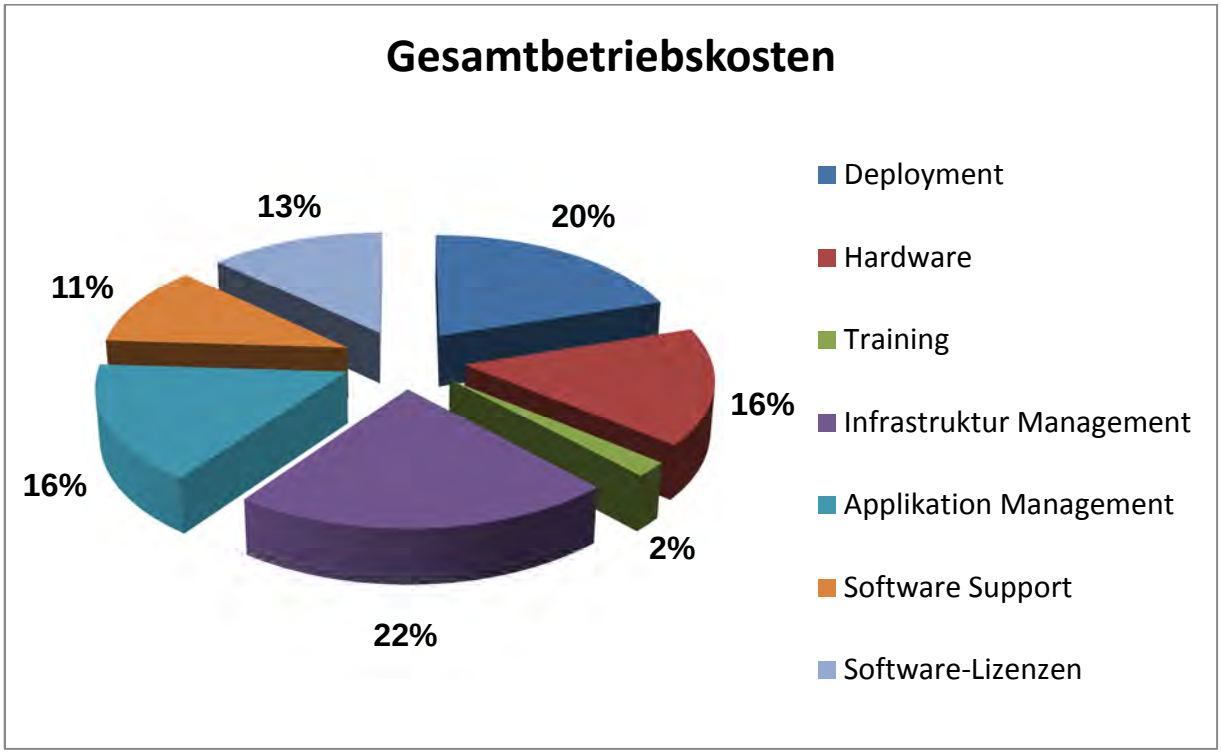


Abb. 25: Summa Technologies - Gesamtbetriebskosten [204]

[204] Vgl. Hohberger, R. (2010), S. 15

Die Studienarbeit vermittelt, dass die Kosten eines laufenden Open-Source-JEE-Anwendungsservers gegenüber einem kommerziellen JEE-Anwendungsserver deutlich höher liegen. In Abbildung 26 wird dies in unterschiedlichsten Konfigurationsstufen dargestellt. Anschließend ist in Tabelle 20 eine kurze Erläuterung zu den unterschiedlichsten Konfigurationsstufen zu sehen.

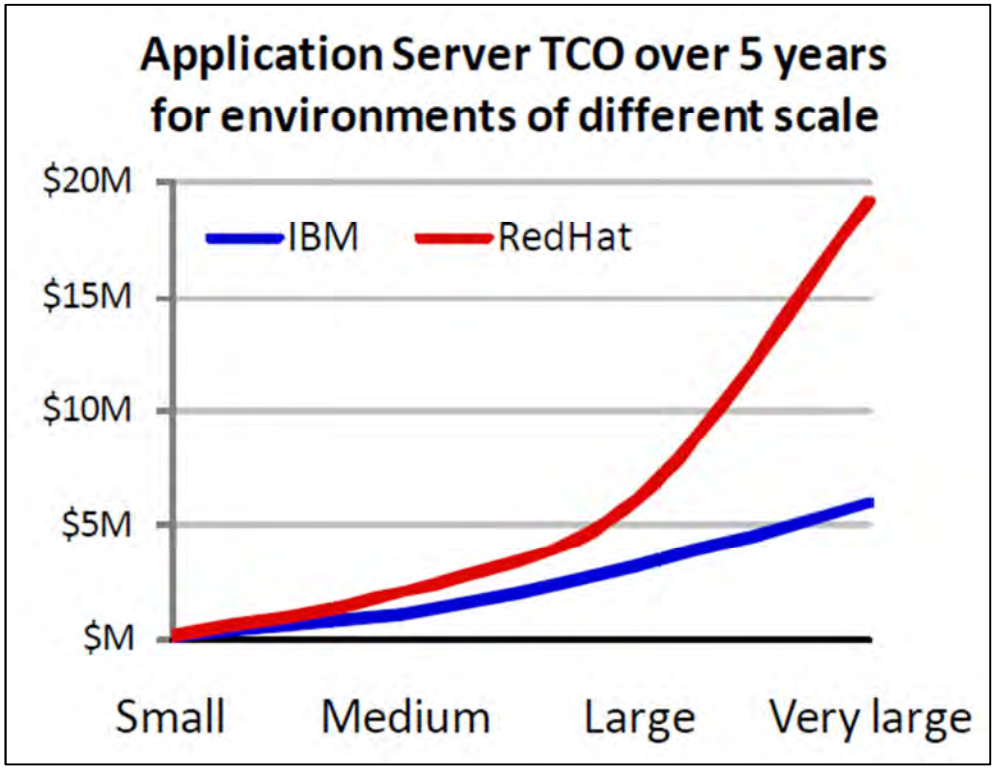


Abb. 26: Summa Technologies – Application Server TCO over 5 years

Konfigurationsstufen	Beschreibung	Fazit (in Bezug auf Abb. 26)
Small (Gering)	Konfiguration ohne besondere Anforderungen in Bezug auf Servicequalität mit einzelnen Servern, auf denen wenige Anwendungen laufen.	Das IBM-Produkt besitzt einen leichten Vorteil gegenüber dem Red Hat-Produkt. Der Grund dafür ist der Support des doppelt so teuren JBoss AS im Gegensatz zu dem WAS.
Medium (Mittel)	Konfiguration mit einem oder mehreren Clustern von JEE-Anwendungsservern.	Der JBoss AS ist beinahe 70 Prozent kostenintensiver im Gegensatz zu WAS. Ausschlaggebender Grund sind die Hardware-Kosten des JBoss AS, die mit zuneh-

		mender Serveranzahl exponentiell ansteigen.
Large (Groß)	Konfigurationen mit ansteigendem Umfang und Anforderungen im Verhältnis zu der „Medium“-Konfiguration.	Das Kostenverhältnis ähnelt dem der „Medium“-Konfiguration.
Very Large (Extrem groß)	Konfiguration bei der eine große Anzahl von Anwendungen auf einer geteilten Hardware mit strengen Anforderungen im Bezug auf Skalierbarkeit, Monitoring und Verwaltbarkeit, nebeneinander bestehen.	Die Kosten des JBoss AS steigen aufgrund der exponentiellen Zunahme der Hardwarekosten im Verhältnis zum WAS exponentiell an, wie zuvor erwähnt.

Tabelle 20: Summa Technologies - Konfigurationsstufen

Der JBoss AS besitzt keine Software-Lizenzkosten. Dies ist ein wesentlicher Vorteil. Jedoch zeigt die Studienarbeit, dass der WAS in den Gesamtbetriebskosten in einem Zeitraum von 5 Jahren kostengünstiger ist. Das Ergebnis stellt eine Einsparung von ca. 70 Prozent dar. Die Abbildung 27 zeigt das Verhältnis des WAS und dem JBoss AS in den bereits erwähnten Kostenbereichen ab.

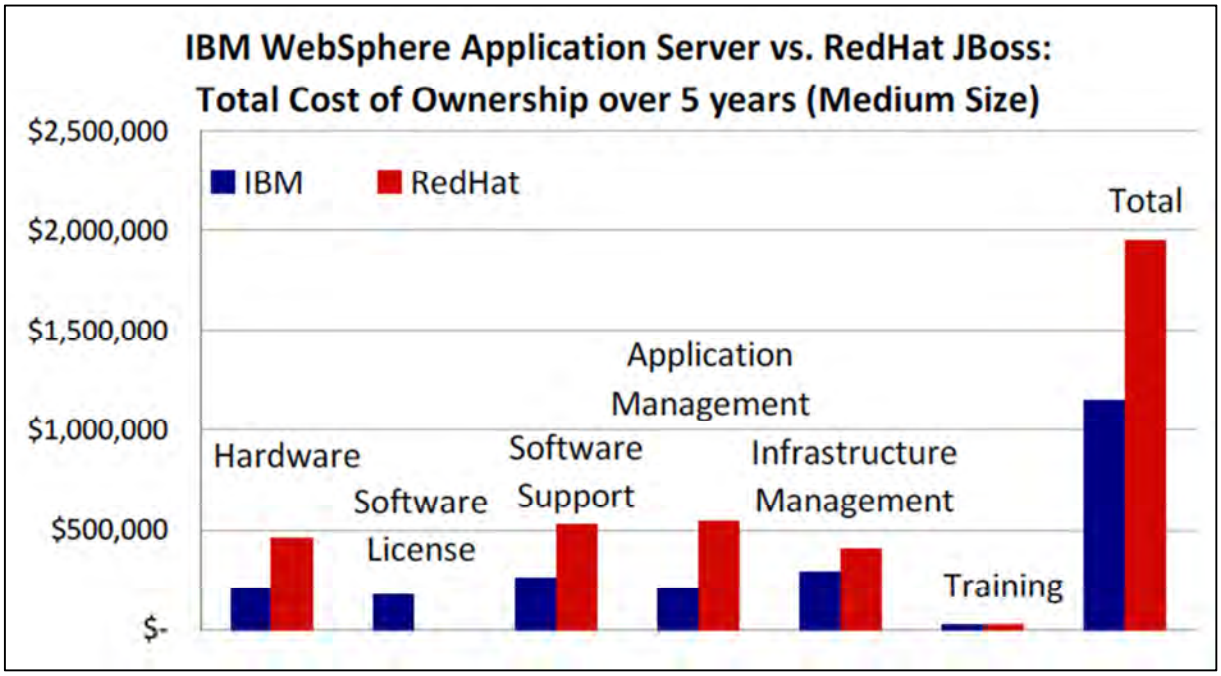


Abb. 27: Summa Technologies – IBM WebSphere AS vs JBoss AS

Im Nachfolgenden findet eine kurze Erläuterung der Ergebnisse der einzelnen Kostenbereiche statt, sowie eine Begründung zu den Gesamtbetriebskosten, die in der vorherigen Abbildung dargestellt werden.

Hardware

Die Kosten für den Bereich Hardware liegen bei einem Einsatz des JBoss AS um etwa 224 Prozent höher als bei einem Einsatz des WAS. Der Hauptgrund, laut der Studienarbeit von Summa Technologies, liegt in der Performance-Differenz der beiden JEE-Anwendungsserver. Der JBoss AS benötigt eine höhere Anzahl an JBoss-Server-Instanzen was zur Folge hat, dass eine kompliziertren Umgebung hinsichtlich der Wartbarkeit entsteht.

Software License (Softwarelizenz)

Der JBoss AS besitzt keine Softwarelizenzkosten im Gegensatz zum WAS. Dies ist ein erheblicher Vorteil hinsichtlich der Beschaffung eines Open-Source-Produktes.

Software Support

Die Supportkosten für den JBoss AS liegen in einem Zeitraum von 5 Jahren beinahe 206 Prozent höher als die Supportkosten für den WAS, laut der Studienarbeit von Summa Technologies. Ein Grund dafür sind die benötigten Hardware-Komponenten und Lizenzen, um dieselbe Auslastung wie beim WAS zu erreichen. Ein weiterer Vorteil ist, dass IBM ein komplettes Paket anbietet, was keine zusätzlichen Kosten verursacht. Bei dem JBoss AS müssen weitere Softwarebestandteile separat ergänzt werden, was mit Kosten verbunden ist.

An dieser Stelle ist wiederholend darauf hinzuweisen, dass diese Aussagen auf der Studienarbeit von Summa Technologies beruhen. Die Projektgruppe hatte die Ressourcen für eine eigene Untersuchung, um diese Aussagen zu überprüfen, nicht. Aus diesem Grund werden diese Aussagen weder unterstützt noch widerlegt.

Application Management (Anwendungsmanagement)

Nach einem Zeitraum von 5 Jahren liegen die Kosten für die Verwaltung der laufenden Anwendungen auf dem JBoss AS um 269 Prozent höher als die Kosten für die Verwaltung der laufenden Anwendungen auf dem WAS. Gründe für diese Tatsache sind die benötigte Differenz in den Verwaltungseigenschaften der beiden JEE-Anwendungsserver, die benötigte Zeit zur Durchführung von administrative Aufgaben sowie der größere Verwaltungsaufwand der benötigten Hardware, was zuvor bereits erwähnt wurde.

Infrastructure Management (Infrastrukturmanagement)

Die Installation und die anschließende Verwaltung des JEE-Anwendungsserver sind bei dem JBoss AS aufwendiger. Dies führt zu 140 Prozent höheren Kosten gegenüber dem WAS.

Training

Der Schulungsaufwand ist bei beiden JEE-Anwendungsservern nahezu identisch. Bei jedem neuen Produkt muss ein gewisser Aufwand betrieben werden, um die Mitarbeiter unter anderem mit der Handhabung vertraut zu machen.

Total

Die Betriebskosten für den Einsatz des WAS liegen deutlich unter den Betriebskosten für den Einsatz des JBoss AS. Dies ist nicht nur auf die Mehrkosten für Hardware oder weitere Lizenzkosten zu beschränken, sondern vor allem auf die Personalkosten, die zum Beispiel bei einem erhöhten Verwaltungsaufwand entstehen.

7.2 Studienarbeit von Forrester Research, Inc. [205]

Das amerikanische Unternehmen Forrester Research, Inc. führte ebenfalls im Jahr 2009 eine wirtschaftliche Auswirkungsanalyse für den Softwarehersteller Red Hat durch und veröffentlichte diese unter dem Titel „Total Economic Impact Of Red Hat JBoss Enterprise Application Platform“. [206] In der Studienarbeit werden die potenziellen Vorteile der JBoss EE gegenüber einer lizenzierten Java-Applikations-Plattform veranschaulicht. Diese basiert auf Informationen aus Kundengesprächen, die für eine Drei-Jahres-Kalkulation verwendet wurden. In der Tabelle 21 sind die wesentlichen Ergebnisse zusammengefasst dargestellt. [207]

	Ursprüngliche Schätzung	Risikobereinigte Schätzung (Zeitwert)
Return of Investment	76 Prozent	63 Prozent
Amortisationsdauer	18 Monate	19 Monate
Kosten	1.385.919 USD	1.434.509 USD.
Leistungen	2.443.538 USD	2.340.926 USD
Insgesamt	**1.057.619 USD**	**906.417 USD**

Tabelle 21: Forrester Research - Drei-Jahres-Kalkulation [208]

[205] Siehe dazu: Forrester Research, Inc. (2009)
[206] Vgl. ebenda, S.1
[207] Vgl. ebenda, S.21
[208] Vgl. ebenda, S.4

Ergänzend zu der vorherigen Tabelle 21, werden Return of Investment (ROI), Amortisationsdauer, Kosten, Leistungen sowie die risikobereinigte Schätzung erläutert.

Return on Investment (ROI)

Das ROI ist eine Messgröße für eine erwartete Rendite in Prozent für ein Projekt. Es wird berechnet, indem der Nettonutzen, sprich die Nutzen abzüglich der Kosten, durch die Kosten dividiert werden. Die ursprüngliche Schätzung liegt bei 76 Prozent, die risikobereinigte Schätzung bei 63 Prozent. Was unter dem Begriff risikobereinigte Schätzung zu verstehen ist, wird an späterer Stelle beschrieben.

Amortisationsdauer

Die Amortisationsdauer beschreibt den Zeitpunkt ab dem der Nettonutzen, sprich der Nutzen abzüglich der Kosten, die getätigten Investitionskosten ausgleicht. Die Amortisationsdauer beträgt bei der ursprünglichen Schätzung 18 Monate. Nach der Risikobereinigung 19 Monate.

Kosten

Die Kosten beinhalten alle Investitionen und notwendigen Ausgaben, um ein Projekt zu betreiben. In der Studienarbeit sind in den Kosten für Netzwerk-Monitoring, Schulungen, Premium Support, Unterhaltskosten, Planungs- sowie Implementierungsarbeiten mitinbegriffen. Die Kosten der ursprünglichen Schätzung betragen 1.385.919 USD. Die Kosten, laut risikobereinigter Schätzung, liegen bei 1.434.509 USD.

Leistungen

Unter den Leistungen versteht man die gewinnbringenden Ergebnisse eines Projektes für ein Unternehmen. In dieser Studienarbeit steht vor allem die Einsparung von Lizenzkosten und Supportkosten, die Steigerung des Umsatzes sowie die Verbesserung der Produkteinführungszeit im Vordergrund. Die ursprünglichen Leistungen liegen bei 2.443.538 USD. Die risikobereinigten Leistungen bei 2.340.926 USD.

Risikobereinigte Schätzung

Die ursprüngliche Schätzung basiert auf dem aktuellen Wert der Kosten und Leistungen bei einem bestimmten Zinssatz. Der Zeitwert beschreibt den Barwert der Kosten und Leistungen. Die risikobereinigte Schätzung basiert auf dem Zeitwert und dient zur vorsichtigen Einschätzung unter Berücksichtigung von geschäftsbeeinflussenden Risikofaktoren, die die originalen Kosten und Leistungen beeinflussen. Risikofaktoren beschreiben Unsicherheiten, die in Verbindung mit den Leistungen und Kosten einer Investition stehen. So gibt es zum Bei-

spiel Implementierungsrisiken und Folgerisiken. Implementierungsrisiken sind potenzielle Unsicherheiten, die zum Beispiel bei einer Integration eines Produktes auftreten können und infolgedessen zu höheren Kosten führen. Folgerisiken sind Gefahren bei einer Investition, die die Anforderungen des Unternehmens nicht vollkommen abdecken. Dies führt zu geringeren gewinnbringenden Leistungen. [209]

[209] Vgl. Forrester Research, Inc. (2009), S.22 f.

8 Fazit

Das Ziel dieser theoretischen Untersuchung war es, quelloffene Produkte, so genannte Open-Source-Produkte, im Bereich der JEE-Applikationsservern gegenüber den kommerziellen Varianten der JEE-Applikationsserver zu stellen. Die Untersuchung sollte anhand der Produkte WAS und JBoss AS erfolgen. Insbesondere sollte untersucht werden, welche Produkte in den Bereichen Verfügbarkeit, Ausfallsicherheit, Skalierbarkeit und Verwaltbarkeit besser abschneiden, bzw. ob sich die kommerzielle Variante des JEE-Applikationsservers in einem der Bereiche von einem Open-Source-Produkt abgrenzt.

Um diese Vorgaben zu erfüllen, ist das Projektteam dazu übergegangen, im ersten Schritt die Spezifikation eines JEE-Applikationsservers aufzuzeigen, sowie allgemein gültige Charakteristiken zu den zu untersuchenden Bereichen zu finden und zu beschreiben. Nachdem die Spezifikationen eines JEE-Anwendungsservers niedergeschrieben wurden, konnten beide spezifischen Produkte vorgestellt werden. Bereits in diesem Kapitel wurden die zu untersuchenden Bereiche aufgezeigt, vorgestellt und erörtert. Dabei wurde insbesondere auf Besonderheiten der einzelnen unterschiedlichen Produkte geachtet.

Auf Grundlage dieser Ausarbeitung konnte eine direkte Gegenüberstellung der beiden Produkte erfolgen. Bei der Analyse wurde sich dabei auf die Erkenntnisse der Produktvorstellungen gestützt und gegebenfalls weitere Unterschiede aufgezeigt. Die Gegenüberstellung hatte dabei nicht nur den Zweck eines direkten Vergleichs, sondern bildete auch die Grundlage für die spätere Nutzwertanalyse. In der Nutzwertanalyse wurden alle Aspekte der JEE-Applikationsserver aufgelistet und zusammengetragen, sodass die Ergebnisse, insbesondere für den Kunden, schnell ersichtlich und tabellarisch nachvollziehbar sind. Darüber hinaus ist mit dieser Form der Analyse gewährleistet, dass der Kunde ein weiteres Produkt aus dem Bereich JEE-Applikationsservers effektiv selbst beurteilen kann, da die Bewertung der Nutzwertanalyse universell für alle JEE-Applikationsserver angepasst wurde. Das Ergebnis der Nutzwertanalyse zeigt, dass das kommerzielle Produkt von IBM sich hinsichtlich des Funktionsumfangs kaum von der Open-Source Variante unterscheidet.

Die Einarbeitung in den Sachverhalt und die Erstellung der Nutzwert- und Kostenanalyse waren für die Projektgruppe herausfordernd. Zum einen lag dies an der Komplexität beider JEE-Applikationsserver, an den für die Projektgruppe neuartigen Technologien und der Vielzahl an Untersuchungskriterien. Zum anderen lag es an der Notwendigkeit der Arbeitskoordination, der Planung der einzelnen Arbeitsschritte und die Absprache über die gefundenen Teilergebnisse, die bei einer Gruppenarbeit anfallen. Aber auch an den zu Beginn unter-

schiedlichen Vorgaben des „KOS-Projektteams“ und dem auftraggebenden Unternehmen Der Kunde war darauf bedacht eine Ausarbeitung zu erhalten, bei der untersucht werden soll, ob die bestehenden Aufgaben des WAS relativ identisch durch die Funktionalität des Open-Source-Produkts JBoss AS ersetzt werden können. Die Aufgabenstellung des „KOS-Projektteams“ ging eher in die Richtung des Funktionalitätsumfangs der einzelnen JEE-Anwendungsserver, als auch in die der Kostenanalyse, die zu Beginn für das auftraggebende Unternehmen nicht von Relevanz war. Dies verlangte von dem Projektteam eine ständige Kommunikation mit den Auftragsgebern und eine Abnahme der erarbeiteten Teilergebnisse.

Darüber hinaus gab es Probleme mit der Findung von neutralen Quellen, was aufgrund seines gravierenden Charakters und der Wichtigkeit bereits in der Ausarbeitung der Studienarbeit thematisiert wurde. Da es sich bei der Ausarbeitung um eine theoretische Untersuchung handelt, ist man infolgedessen auf fremde Quellen angewiesen. Wenn diese Quellen jedoch voreingenommen und zum Teil widersprüchlich sind, ist eine uneinvorgenomme Ausarbeitung nicht durchführbar. Das Projektteam ist aufgrund dieser Erkenntnis noch kritischer bei der Nutzung und Verwendung von Quellen vorgegangen. Bei der Gegenüberstellung versuchte das Team möglichst wertfreie Quellen zu verwenden. Desweiteren verfolgte das Team das Ziel, Quellen direkt vom Hersteller zu beziehen, da hier die Herkunft und die Absicht ersichtlich sind. Das Problem der wertfreien Quellen versuchte das Team zudem mit der Hilfe von persönlichen Gesprächen mit Experten zu umgehen. Leider konnten sich die beiden Firmen IBM und Red Hat, auch auf wiederholte Anfragen, nicht dazu entschließen, Quellen bzw. Expertenmeinungen zur Verfügung zu stellen. Jegliche Anfragen wurden abgeblockt bzw. es wurde erst gar nicht darauf reagiert.

Handlungsempfehlung

Aufgrund der erlangten Erkenntnisse aus der Ausarbeitung empfiehlt das Projektteam dem auftraggebenden Unternehmen die Prüfung einer Umstellung des JEE-Anwendungsservers von IBM auf die Open-Source Variante von Red Hat. Das Projektteam empfiehlt dabei die Durchführung eines Pilotprojektes mit dem JEE-Applikationsserver JBoss. Anhand dieses Projektes können detaillierte Untersuchungen von JBoss AS erfolgen und der Server unter den Bedingungen des auftraggebenden Unternehmens getestet werden. Zudem kann dabei die Verträglichkeit von JBoss AS und den anderen Produkten von IBM untersucht werden.

Anhang

Anhangsverzeichnis

Anhang 1: Protokoll zum Kick-Off-Meeting des KOS-Projekt

Beteiligte Personen:	Ansprechpartner des auftraggebenden Unternehmen
	Timothy Dörr
	Sascha Jörg
	Julian Löbbers
Ort:	Auftraggebendes Unternehmen
	Straße
	Stadt
Datum:	28.11.2012
Beginn:	17 Uhr
Dauer:	1 Std.
Inhalte:	1. Besprechung der Kriterien für die Untersuchung
	2. Weitere Vorgehensweise
	3. Zukünftige Meetings

1. Besprechung der Kriterien für die Untersuchung

Die Kriterien orientieren sich in erster Linie an Betriebsaspekte. Unterschiedliche Funktionsmöglichkeiten von JBoss und WAS sollen nicht bzw. nur am Rande untersucht werden. Durch das Projekt soll untersucht werden, ob die Möglichkeit besteht, aktuelle auf dem WAS befindliche Anwendungen des auftraggebenden Unternehmens auch in JBoss einbetten zu können. Das auftraggebende Unternehmen als Kunde will deshalb, dass die nachfolgen denen Kriterien untersucht werden:

- Performance
- Ausfallsicherheit
- Deployment
- Administration
- Logging
- Monitoring

Zusätzlich können folgende Themen bearbeitet werden:

- Load Balancing
- Clustering
- Session-Fail-Over (Serverseitig, Clientseitg, bzw. nicht bei Stateless)

2. Weitere Vorgehensweise

Erstellung einer Grobgliederung anhand der festgelegten Kriterien. Zur Überprüfung der Gliederung an Ansprechpartner senden.

2. Zukünftige Meetings

Sobald sich Bedarf an internen Informationen ergibt -

Mitte Dezember 2012 (Abgabe des Management Resümee).

Anhang 2: Logging – Meldungskategorien [210, 211]

Meldungskategorie	Beschreibung
Info	Allgemeine und nützliche Informationen werden in dieser Meldungskategorie geloggt. Beispielsweise Informationen bezüglich des Startes und der Beendigung des Systems.
Warn	Diese Meldungskategorie loggt Informationen, die eine potenzielle Gefahr des Systems aufzeigen. Zum Beispiel bei der wiederholten Ausführung eines Prozesses nach einem fehlgeschlagenen Erstversuch.
Error	Fehlende Daten oder Probleme beim Öffnen einer Datei werden als schwerwiegende Fehler bezeichnet und in die Meldungskategorie „Error“ geloggt.
Debug	In dieser Meldungskategorie werden Informationen geloggt, die zur Analyse und zur anschließenden Behebung von Systemproblemen dienen.
Trace	Die Informationen dienen, wie bei der Meldungskategorie „Debug“, zur Analyse und zur anschließenden Behebung von Systemproblemen. Jedoch liegen umfangreichere und detailliertere Informationen vor.
Fatal	Diese Meldungskategorie loggt Informationen, die bei der Abschaltung eines Systems, aus bis dahin unerklärlichen Gründen, entstehen, um vorhandene Daten nicht zu verlieren.

[210] Vgl. Bruusgaard, T./Twist, J. (2012)
[211] Vgl. Chuvakin, A./Schmidt, J./Phillips, C. (2013), S.342

Anhang 3: E-Mail von einem Mitarbeiter des auftraggebenden Unternehmens

Hallo,

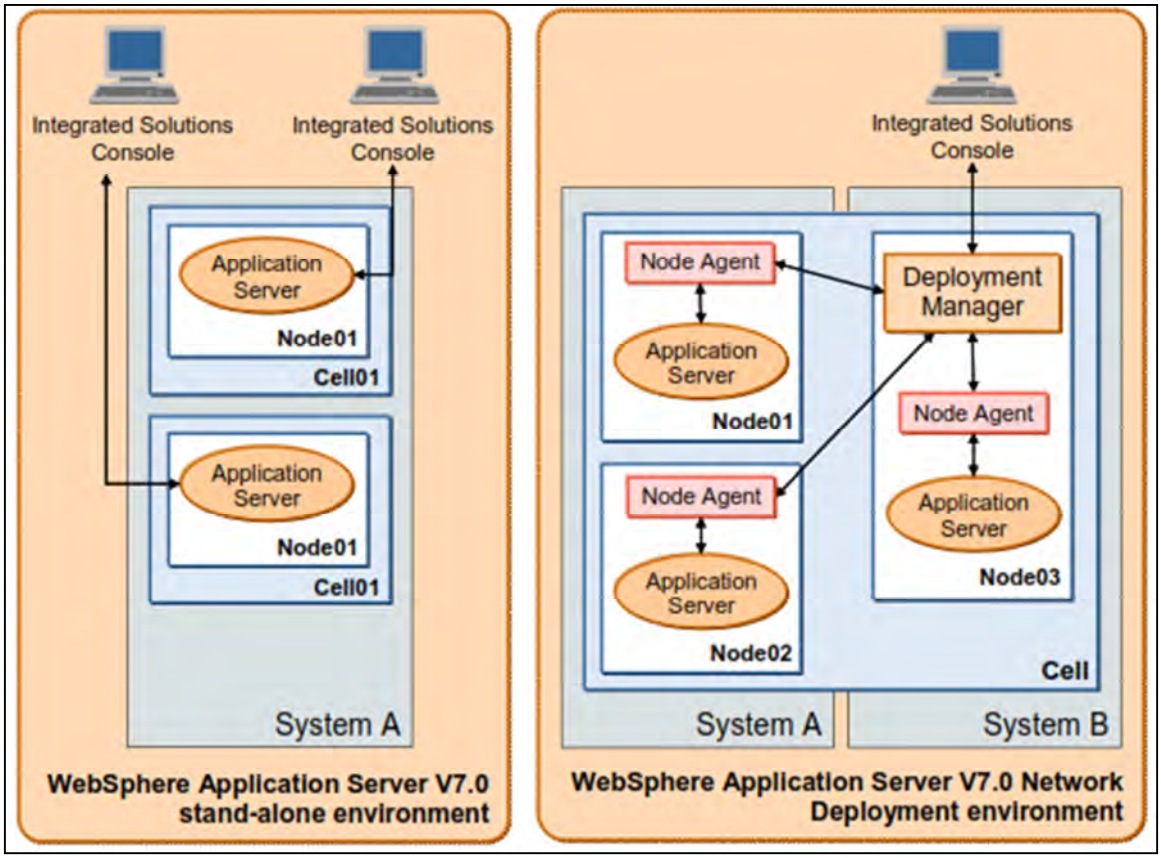
genutzt wir aktuell Websphere Application Server Network Deployment V7 - Lizenzpflichtig auf insgesamt 27 virtuellen Servern mit insgesamt 45 virtuellen CPU's --> 3150 PVU
Darüberhinaus noch auf weiteren 7 virtuellen Servern für Cognos bzw. ITIM

Lizenzkosten fallen einmalig bei Beschaffung an - laufende Kosten entstehen durch das Paket "Subscription & Support" - k.A.

[redacted]

14.12.2012 09:25 An [redacted]

Anhang 4: Unterschied zwischen WAS und WAS Network Deployment [212]



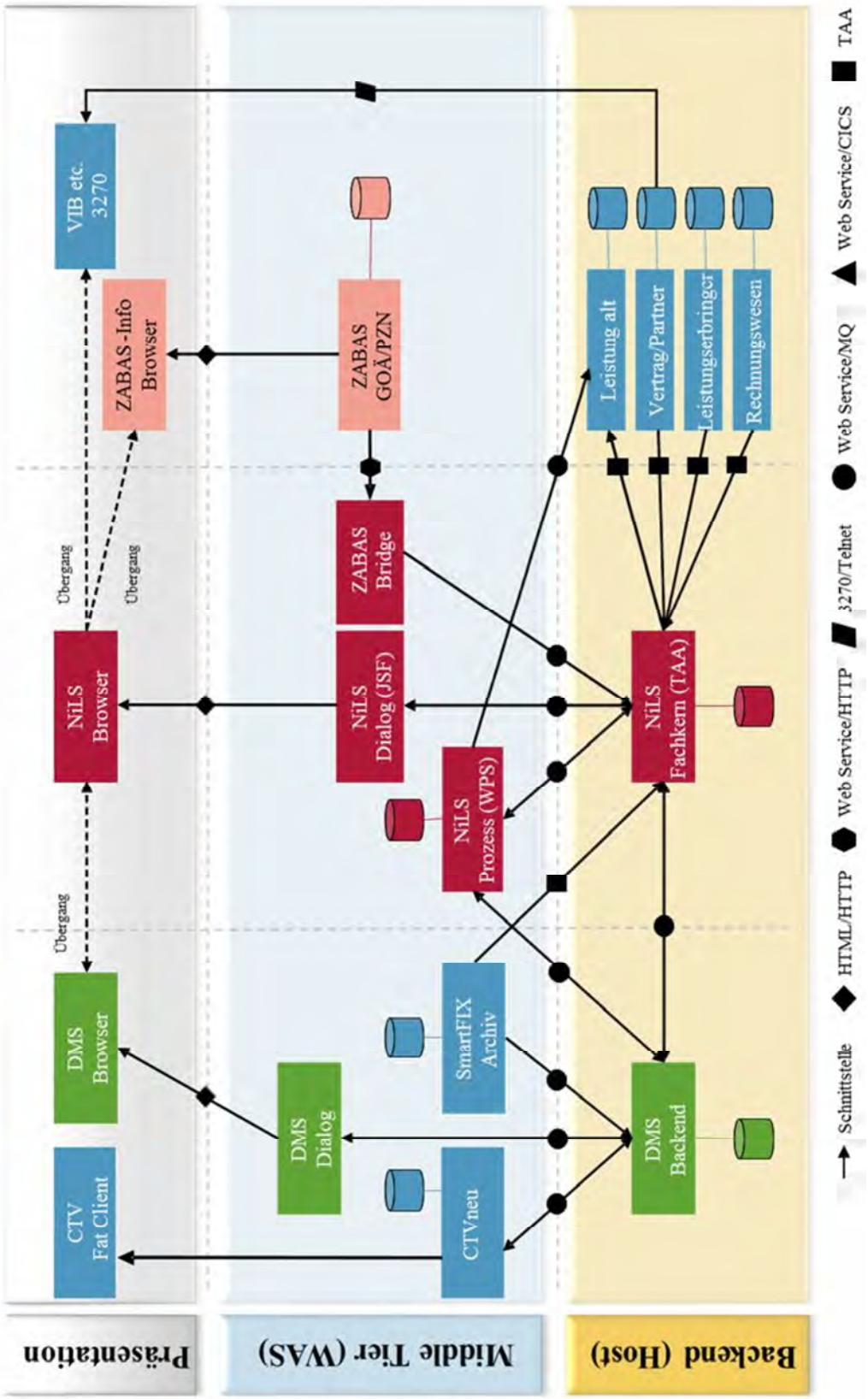
[212] Siehe dazu: Agopyan, A. (2009)

Anhang 5: Abgrenzung der Sicherheitslevel [213]

Severity Level	Severity Definitions	Examples
Severity 1	- Critical situation/System Down - Business critical software component is inoperable - usually applies to production environment - Critical interface has failed	- All users of Tivoli Problem Management are unable to register a call - The Lotus Notes mail server is down and affecting all users.
Severity 2	Severe Impact: A software component is severely restricted in its use, causing significant business impact	- All users of Tivoli Problem Management receive a database manager error while attempting to view open problems
Severity 3	Moderate impact: A non-critical software component is malfunctioning, causing moderate business impact	- A client cannot connect to a server
Severity 4	Minimal impact: A non-critical software component is malfunctioning, causing minimal impact, or a non-technical request is made.	- Documentation is incorrect. - Additional documentation requested

[213] Entnommen aus: IBM Corporation (2009), S.15

Anhang 6: Gesamtarchitektur von NiLS und DMS



Anhang 7: JEE-5-Spezifikationen [214]

Java EE 5 Platform Packages	
javax.activation	The JavaBeans(TM) Activation Framework is used by the JavaMail(TM) API to manage MIME data.
javax.annotation	-
javax.annotation.security	-
javax.ejb	The javax.ejb package contains the Enterprise JavaBeans classes and interfaces that define the contracts between the enterprise bean and its clients and between the enterprise bean and the EJB container.
javax.ejb.spi	The javax.ejb.spi package defines interfaces that are implemented by the EJB container.
javax.el	Provides the API for the Unified Expression Language shared by the JSP 2.1 and JSF 1.2 technologies.
javax.enterprise.deploy.model	Provides Tool Vendor implementation classes.
javax.enterprise.deploy.model.exceptions	Provides Tool Vendor exception implementation classes.
javax.enterprise.deploy.shared	Provides shared objects for Tool Vendor and Product Vendor implementation classes.
javax.enterprise.deploy.shared.factories	Provides shared factory manager object for Tool Vendor and Product Vendor implementation classes.
javax.enterprise.deploy.spi	Provides J2EE Product Vendor implementation classes.
javax.enterprise.deploy.spi.exceptions	Provides J2EE Product Vendor deployment

[214] Vgl. Sun Microsystems, Inc. (2007)

	exception implementation classes.
javax.enterprise.deploy.spi.factories	Provides J2EE Product Vendor deployment factory implementation classes.
javax.enterprise.deploy.spi.status	Provides J2EE Product Vendor deployment status implementation classes.
javax.faces	Top level classes for the JavaServer(tm) Faces API.
javax.faces.application	APIs that are used to link an application's business logic objects to JavaServer Faces, as well as convenient pluggable mechanisms to manage the execution of an application that is based on JavaServer Faces.
javax.faces.component	Fundamental APIs for user interface components.
javax.faces.component.html	Specialized user interface component classes for HTML.
javax.faces.context	Classes and interfaces defining per-request state information.
javax.faces.convert	Contains classes and interfaces defining converters.
javax.faces.el	DEPRECATED Classes and interfaces for evaluating and processing reference expressions.
javax.faces.event	Interfaces describing events and event listeners, and concrete event implementation classes.
javax.faces.lifecycle	Classes and interfaces defining lifecycle management for the JavaServer Faces implementation.
javax.faces.model	Standard model data beans for JavaServer Faces.
javax.faces.render	Classes and interfaces defining the ren-

	dering model.
javax.faces.validator	Interface defining the validator model, and concrete validator implementation classes.
javax.faces.webapp	Classes required for integration of JavaServer Faces into web applications, including a standard servlet, base classes for JSP custom component tags, and concrete tag implementations for core tags.
javax.interceptor	The javax.interceptor package contains classes and interfaces for use with EJB interceptors.
javax.jms	The Java Message Service (JMS) API provides a common way for Java programs to create, send, receive and read an enterprise messaging system's messages.
javax.jws	
javax.jws.soap	
javax.mail	The JavaMail™ API provides classes that model a mail system.
javax.mail.event	Listeners and events for the JavaMail API.
javax.mail.internet	Classes specific to Internet mail systems.
javax.mail.search	Message search terms for the JavaMail API.
javax.mail.util	Utility classes.
javax.management.j2ee	Provides the J2EE Management Enterprise Bean component (MEJB) interfaces.
javax.management.j2ee.statistics	Provides the standard interfaces for accessing performance data from J2EE managed objects Package Specification JSR 77, J2EE Management Related Documentation For overviews, tutorials, examples, guides, and tool documentation, please see:J2EE Tools

javax.persistence	The javax.persistence package contains the classes and interfaces that define the contracts between a persistence provider and the managed classes and the clients of the Java Persistence API.
javax.persistence.spi	The javax.persistence.spi package defines the classes and interfaces that are implemented by the persistence provider and the Java EE container for use by the container, provider, and/or Persistence bootstrap class in deployment and bootstrapping.
javax.resource	The javax.resource package is the top-level package for the J2EE Connector API specification.
javax.resource.cci	The javax.resource.cci package contains API specification for the Common Client Interface (CCI).
javax.resource.spi	The javax.resource.spi package contains APIs for the system contracts defined in the J2EE Connector Architecture specification.
javax.resource.spi.endpoint	This package contains system contracts for service endpoint interactions.
javax.resource.spi.security	The javax.resource.spi.security package contains APIs for the security management contract.
javax.resource.spi.work	This package contains APIs for the work management contract.
javax.security.jacc	This package contains the Java Authorization Contract for Containers API
javax.servlet	The javax.servlet package contains a number of classes and interfaces that describe and define the contracts between a servlet class and the runtime environment provided for an instance of such a class by a

	conforming servlet container.
javax.servlet.http	The javax.servlet.http package contains a number of classes and interfaces that describe and define the contracts between a servlet class running under the HTTP protocol and the runtime environment provided for an instance of such a class by a conforming servlet container.
javax.servlet.jsp	Classes and interfaces for the Core JSP 2.1 API.
javax.servlet.jsp.el	Provides the ELResolver classes that define the object resolution rules that must be supported by a JSP container with the new unified Expression Language.
javax.servlet.jsp.tagext	Classes and interfaces for the definition of JavaServer Pages Tag Libraries.
javax.transaction	Provides the API that defines the contract between the transaction manager and the various parties involved in a distributed transaction namely : resource manager, application, and application server.
javax.transaction.xa	Provides the API that defines the contract between the transaction manager and the resource manager, which allows the transaction manager to enlist and delist resource objects (supplied by the resource manager driver) in JTA transactions.
javax.xml.bind	Provides a runtime binding framework for client applications including unmarshalling, marshalling, and validation capabilities.
javax.xml.bind.annotation	Defines annotations for customizing Java program elements to XML Schema mapping.
javax.xml.bind.annotation.adapters	XmlAdapter and its spec-defined sub-

	classes to allow arbitrary Java classes to be used with JAXB.
javax.xml.bind.attachment	This package is implemented by a MIME-based package processor that enables the interpretation and creation of optimized binary data within an MIME-based package format.
javax.xml.bind.helpers	JAXB Provider Use Only: Provides partial default implementations for some of the javax.xml.bind interfaces.
javax.xml.bind.util	Useful client utility classes.
javax.xml.registry	This package and its sub-packages describe the API classes and interfaces for the JAXR API.
javax.xml.registry.infomodel	This package describes the information model for the JAXR API.
javax.xml.rpc	This package contains the core JAX-RPC APIs for the client programming model.
javax.xml.rpc.encoding	This package defines APIs for the extensible type mapping framework.
javax.xml.rpc.handler	This package defines APIs for SOAP Message Handlers
javax.xml.rpc.handler.soap	This package defines APIs for SOAP Message Handlers
javax.xml.rpc.holders	This package contains the standard Java Holder classes.
javax.xml.rpc.server	This package defines APIs for the servlet based JAX-RPC endpoint model.
javax.xml.rpc.soap	This package defines APIs specific to the SOAP binding.
javax.xml.soap	Provides the API for creating and building SOAP messages.

javax.xml.stream	-
javax.xml.stream.events	-
javax.xml.stream.util	-
javax.xml.ws	This package contains the core JAX-WS APIs.
javax.xml.ws.handler	This package defines APIs for message handlers.
javax.xml.ws.handler.soap	This package defines APIs for SOAP message handlers.
javax.xml.ws.http	This package defines APIs specific to the HTTP binding.
javax.xml.ws.soap	This package defines APIs specific to the SOAP binding.
javax.xml.ws.spi	This package defines SPIs for JAX-WS 2.0.

Anhang 8: GPL und LGPL

Die General Public License (GPL) und Lesser General Public License (LGPL) sind zwei verwandte Typen von Softwarelizenzen, die zum Schutz einer Open-Source-Software entstanden sind. Die erste Version der GPL entstand im Jahr 1989 und die erste Version der LGPL im Jahr 1999 mit Hilfe der gemeinnütziger Organisation Free Software, die zur Förderung von freier Software gegründet wurde.

Im Allgemeinen regeln beide Lizenzen das Vervielfältigungs-, Bearbeitungs- und Verbreitungsrecht von freiverfügbaren Softwaren. Die GPL unterliegt im kommerziellen Softwarebereich jedoch einigen Restriktionen. Um diesen Beschränkungen entgegenzuwirken wurde LGPL entwickelt. Der Hauptunterschied dieser Lizenzen ist folgender: „Die LGPL ermöglicht es, LGPL-Software in sein Programm einzubinden, ohne jedoch das eigene Programm ebenfalls unter die LGPL stellen zu müssen, da die Software nun die GPL-Software nutzt, aber nicht davon abgeleitet ist."

Um diesen Umstand in der Praxis realisieren zu können, wird die LGPL-Software mit Hilfe von dynamischen Links erst zur Laufzeit mit der Open-Source-Software verknüpft. Infolgedessen kann die Unabhängigkeit zwischen den Softwareprogrammen bewerkstelligt werden.

Anhang 9: Red Hat Partner [215]

Partner	Partnertyp	Hauptstandort	Spezialisierungen
Logica Deutsch-land GmbH & Co. KG	Vertriebspartner	Leinfelder Straße 60 Leinfelden-Echterdingen 70771	Middleware - JBoss
Millenux Gmbh	Vertriebspartner	Lilienthalstr 2 Korntal-Münchingen BW 70825	Infrastructure - RHEL Virtualization
adorsys GmbH & Co. KG	Vertriebspartner	Bartholomaeusstrasse 26c Nuremberg 90489	---
akquinet AG	Vertriebspartner	Bulowstr 66 Berlin Berlin 10783	Middleware - JBoss
Gonicus Gmbh	Vertriebspartner	Möhnestraße 11-17 Arnsberg 59755	Infrastructure - RHEL Virtualization
Steria Mummert Consulting AG	Vertriebspartner	Hans-Henny-Jahnn-Weg 85 Hamburg 22085	Middleware - JBoss
Computacenter AG & Co oHG	Vertriebspartner	Europaring 34-40 Kerpen Berlin 50170	Infrastructure - RHEL
SAP AG	ISV	Dietmar-Hopp-Allee 16 Walldorf 69190	---
agentbase AG	Vertriebspartner	Eggertstrasse 7 Paderborn 33100	---
CCP Software GmbH	Vertriebspartner	Rudolf-Breitscheid-Str. 1-5 Marburg 35037	---

[215] Vgl. Red Hat, Inc (2010)

Bechtle Logistik & Service GmbH	Vertriebspartner	Bechtle Logistik & Service GmbH Bechtle-Platz 1, 74172 Neckarsulm Neckarsulm 74172	Infrastructure - RHEL Virtualization
inoX-tech GmbH	Vertriebspartner	inoX-tech GmbH & Co. KG, Stahlgruberring 54, München 81829	Infrastructure - RHEL Virtualization
VIADA GmbH & Co. KG	Vertriebspartner	Rheinlanddamm 201 Dortmund Nordrhein-Westfalen 44139	Middleware - JBoss

Anhang 10: Quellcode - Erstellung einer automatisierten Log-Datei [216]

```
Zeile 01. <log4j>
Zeile 02.       <appender name="FILE"
Zeile 03        class="org.jboss.logging.appender.RollingFileAppender">                1
Zeile 04.       <param name="File" value="${jboss.server.log.dir}/server.log"/>         2
Zeile 05.       <param name="Append" value="true"/>                                     3
Zeile 06.       <param name="MaxFileSize" value="10MB"/>                                4
Zeile 07.       <param name="MaxBackupIndex" value="20"/>                               5
Zeile 09.       </appender>
Zeile 10. </log4j>
```

Erläuterung:

1 = Appender
2 = Ablageort der Log-Datei
3 = Hinzufügen zur existierenden Log-Datei (true oder false)
4 = Größe der Log-Datei
5 = Maximale Anzahl der zu speichernden Log-Datein

[216] Mit Änderungen entnommen aus: Vgl. Subbaro, M. (2008)

Anhang 11: Quellcode - Definierung einer neuen Log-Datei [217]

```
Zeile 01. <log4j>
Zeile 02.       <appender name="JBIA">                                          1
Zeile 03.       <param name="File" value="${jboss.server.log.dir}/jbia.log"/>    2
Zeile 04.       </appender>
Zeile 05.               <category name="org.jbia">                              3
Zeile 06.               <priority value="DEBUG"/>                               4
Zeile 07.               <appender-ref ref="JBIA" />                             5
Zeile 08.               </category>
Zeile 09. </log4j>
```

Erläuterung:

1 = Identifizierung des Appenders

2 = Name der Log-Datei

3 = Zu speichernde Klasse

4 = Logging-Level (z.B. Info, Warn, Error, Debug)

5 = Speicherung in Appenders “JBIA”

[217] Mit Änderungen entnommen aus: Vgl. Subbaro, M. (2008)

Anhang 12: JBoss Schulung - Orientation in Objects GmbH [218]

Unternehmen	Orientation in Objects GmbH Weinheimer Straße 68 68309 Mannheim
Thema	JBoss AS 7 – Konfiguration und Administration
Beschreibung	Dieses Seminar gibt einen umfassenden Überblick über die Architektur und die verfügbaren Dienste des Applikationsservers JBoss in der Version 7.x. Die Teilnehmer/-innen erlernen die Installation und Konfiguration des JBoss und die Bereitstellung von Anwendungen. Die eingesetzten Beispiele reichen von einer einfachen Umgebung mit nur einem Server bis hin zu redundant aufgebauten Clustern mit vorgelagerten Web-Servern.
Dauer	2 Tage
Kurs-Inhalt	• JavaEE im Überblick • JBoss AS im Überblick • Download und Installation • Architektur des JBoss AS • Domainkonzept • Modulkonzept • Deployment • Security • Messaging • Administrationswerkzeuge • Clustering & Hochverfügbarkeit • Exkurs: Migration
Preis	1.310 Euro (Preis pro Person zzgl. MwSt.)
Zielgruppe	Sie sind Administrator oder Java Entwickler und möchten den JBoss Application Server im Detail kennenlernen.
Voraussetzung für Teilnahme	Die Begriffswelt der Java EE sollte Ihnen vertraut sein, z. B. durch Besuch unserer Seminars Enterprise Java für Architekten. Erfahrung in Administration und Netzwerken. Java Kenntnisse sind hilfreich aber nicht notwendig.

[218] Mit Änderungen entnommen aus: Orientation in Objects GmbH (2012a)

Anhang 13: JBoss Schulung – Red Hat, Inc. [219]

Unternehmen	Red Hat, Inc.
Thema	JBoss Application Administration I – Installation, Konfiguration und Verwalten der JBoss AS Plattform
Beschreibung	JBoss Application Administration I lehrt die Methoden zur Insatallation und Konfiguration des JBoss AS. Mit Hilfe von praktischen Übungen werden die wesentlichen Aufgaben, die ein Systemadministrator benötigt, um den JBoss AS zu implementieren und zu verwalten.
Dauer	5 Tage
Kurs-Inhalt	• Installation der JBoss AS Plattform 6 • Konfiguration der Domain • Konfiguration der JMS • Konfiguration des Logging-Systems • Implementierung der Anwendungssicherung • Migration von dem JBoss AS Plattform 5 zu JBoss AS Plattform 6 • Vorstellung des Thema Clustering • Verteilung einer Anwendung auf den JBoss AS
Preis	2900 USD (ca. 2200 Euro)
Zielgruppe	Systemadministratoren, die die JBoss AS Plattform 6 kennenlernen möchten und eventuell schon Erfahrungen mit der JBoss AS Plattform 5 haben.
Voraussetzung für Teilnahme	Erfahrung mit Systemadministration auf den Betriebssystem Microsoft Windows, Unix oder Linux. Kein Wissen bezüglich Java und JBoss Developer Studio benötigt.

[219] Mit Änderungen entnommen aus: Red Hat, Inc. (2012a)

Anhang 14: JBoss Schulung – New Elements GmbH [220]

Unternehmen	New Elements GmbH (it-schulungen.com) Thurn-und-Taxis-Str. 10 90411 Nürnberg
Thema	JBoss AS 7 - Administration Grundlagen
Beschreibung	In dieser Schulung lernen Sie den JBoss 7 Server zu Administrieren und zu verwalten. Die einzelnen Lernziele sind: • JBoss Applikation Server Überblick und Architektur • JBoss Server installieren und konfigurieren • Standalone-Installation und Domain-based Installationen • Classloading-Eigenschaften, Einsatz und die Unterschiede zu JBoss Vorgängerversion • Einsatz der Web Management-Konsole • Management-Modell (Scripting über CLI (Command Line Interface)) • Konfiguration und Nutzung der wichtigsten Subsysteme • Web-, Messaging-und Logging • Web-Container, einschließlich des HTTP, HTTPS und AJP • Erstellung und Nutzung von Java EE-Anwendung Komponenten wie EJB, Servlets und JMS • Optimierung JBoss • Migration Szenarien Anwendungen auf vogänger Versionen
Dauer	5 Tage (jeweils von 09:00 Uhr bis 17:00 Uhr)
Kurs-Inhalt	<u>JBoss 7 Server Installation</u> • Java-Umgebung installieren • Herunterladen und installieren des Servers • Server starten und stoppen • Erster Überblick über die Dateistruktur

[220] Mit Änderungen entnommen aus: New Elements GmbH (2012)

	• Anpassung der wichtigsten Dienste <u>JBoss 7 Grundlegende Konfigurationsaufgaben</u> • JBoss Server Architektur • Konfigurieren von Erweiterungen (Extensions) • Pfade, Management Interfaces, Profile, Interfaces, Socket binding groups und Deployments konfigurieren • Core Subsysteme konfigurieren (Thread Pool u.a.) • Logging konfigurieren (JBoss logging, slf4j, log4j) <u>JBoss 7 Anwendungsdienste konfigurieren</u> • JDBC-Treiber konfigurieren • Data Sources einrichten • Enterprise Java Beans konfigurieren • Transaktions-Subsysteme einstellen <u>JBoss 7 Servlet Container konfigurieren</u> • Konfigurieren von Webserver Konnektoren • Abfragen von Container Configuration mit Hilfe des CLI • Konfiguration der HTTP, HTTPS, and AJP Connectors (in standalone.xml, und mit Hilfe von CLI) • Socket Binding Groups und Portkonfiguration • Konfigurieren von statischen und dynamischen Ressourcen • Webanwendung deployen <u>JBoss 7 Domain Konfiguration</u> • Überblick über die Domänenarchtitektur in JBoss 7 • Starten und stoppen einer Domäne • Domain.xml und host.xml verstehen • Management Interface, Network Interfaces und Domaincontroller einstellen • Konfiguration der JVM für JBoss 7 • Aufbau einer eigenen Domain an einem Beispiel

	Anwendung in JBoss 7 deployen • JAR-, WAR-, und EAR-Archive verstehen • Deployment über CLI vs. FileCopy • Deployment-Scanner konfigurieren • Standalone Deployment • Deployment in der Domäne • JBoss 7 Classloading und Deployment Verwaltung des Application Servers • Management über das CLI (Command Line Interface) • Management über die Administrationskonsole • Konfiguration von Web Services und Datasources mit Hilfe von JNDI Messaging • Konfiguration eines JMS Servers (Java Messaging Server) • Konfiguration der HornetQ Implementierung • Einrichtung von Queues und Topics Performance und Tuning des JBoss AS 7 • JBoss AS und die Java Virtual Machine (JVM) • Tuning des Heap (Memory Usage) • Garbage Collection (GC) mit Hilfe des G1GC • Tuning GC • Tuning the Web Tier (Tomcat) • Connector Tuning
Preis	Auf Nachfrage
Zielgruppe	Systemadministratoren Softwareentwickler
Voraussetzung für Teilnahme	Kenntnisse mit JEE-Anwendungsservern Vorteilhaft Kenntnisse in JBoss 5/6 JEE Basis-Kenntnisse

Anhang 15: Zusammenfassung des Gesprächs mit einem JBoss-Experte

Beteiligte Personen:	JBoss-Experte Julian Löbbers Philipp Brüßler Timothy Dörr
Datum:	19.12.2012
Beginn:	13:30 Uhr
Dauer:	1,5 Std.
Inhalte:	1. Deployment 2. Support 3. Ausfallsicherheit 4. Administration 5. Logging 6. Skalierbarkeit 7. Modul-Schnittstellen 8. Schulungsaufwand 9. Performance 10. Differenzen zwischen IBM WebSphere AS und JBoss AS 11. Quellcodeoffenheit 12. Mainframe

1. Deployment:

Frage:	Beim Deployment einer Anwendung muss, kurz gesagt, die Built-Target in ein entsprechendes Verzeichnis kopiert und im eine Batchdatei ausgeführt werden. Muss beim JBoss im Gegensatz zum IBM WebSphere AS weitere Schritte beachtet werden? Ist der Aufwand etwa vergleich?
Antwort:	Das Vorgehen ist bei beiden Applikationsservern gleich. Die Hauptfrage, die sich ein Unternehmen beim Deployment einer Anwendung jedoch fragen muss, ist die Revisionssicherheit. Es muss zu jedem Zeitpunkt nachvollziehbar sein, wer eine Änderung, zum Beispiel eine Neuinstallation einer Anwendung, vorgenommen hat. Ebenfalls muss sichergestellt werden das Änderungen wiederrückgängig gemacht werden können. Um diese Anforderungen zu erfüllen können eigenentwickelte Tool oder auch Open-Source-Produkte eingesetzt werden.
Frage:	Der JBoss besitzt die Eigenschaft des sogennaten „Hot-Deployment“. Ist dies eine Eigenschaft, die genutzt wird und auch einen Vorteil bringt?

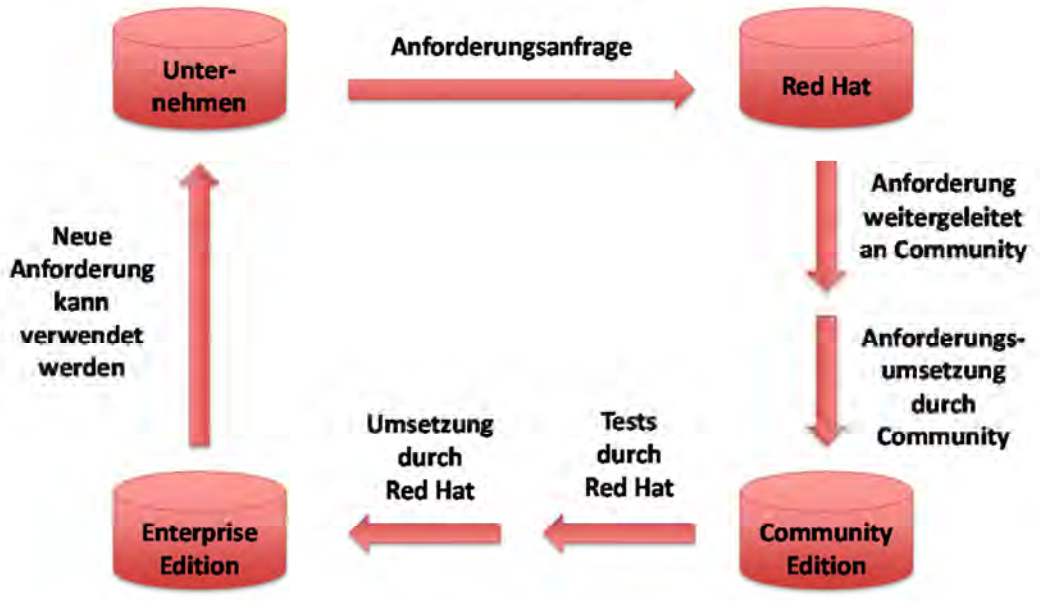
Antwort: Dies ist eine nette Eigenschaft des JBoss, jedoch wird dies in Produktion nicht eingesetzt. Der Grund ist einfach, dass die Gefahr zu hoch ist, dass ein unerwartetes Problem auftritt und infolgedessen die komplette Anwendung abstürzt. Vor allem Infrastrukturänderungen können zu Schwierigkeiten führen. Bei der Entwicklung auf dem lokalen Rechner ist die Verwendung dieser Eigenschaft jedoch hilfreich.

2. Support

Frage: Bei einem Einsatz eines Open-Source-Produktes stellt sich immer die Frage nach dem Support. Wie sieht der Support bei dem Produkt JBoss aus?

Antwort: Man unterscheidet zwischen der Community Edition und der Enterprise Edition. Die Community Edition wird von freien Entwicklern betreut. Die Enterprise Edition vom Softwarehersteller Red Hat. Bei der Verwendung der Enterprise Edition wird ein individueller Wartungsvertrag abgeschlossen, der an die Bedürfnisse des Unternehmens und an die Projektanforderungen angepasst sind. Dies bietet für das Unternehmen den Vorteil, dass dieser Fehler melden und neue Eigenschaften anfordern kann. Dies ist beim dem IBM WebSphere AS nicht so einfach machbar, da diese sich nicht individuell auf Unternehmen eingehen, sondern ein komplettes Paket anbieten.

Ablauf des Anforderungsprozesses bei der Enterprise Edition von Red Hat:



Frage: Sie sind die Kosten des Wartungsvertrages gegliedert?

Antwort: Zu dem Thema Kosten des Wartungsvertrages ist zu sagen, dass es individuell, den Anforderungen entsprechend, ausgehandelt wird. Ein wesentlicher

Vorteil ist, dass die Softwarelizenzkosten wegfallen. Diese Kosten machen in der Regel den größten Anteil der Kosten einer Software aus.

Frage: Wie ist Ihre Erfahrung mit dem Support des Softwareherstellers Red Hat zum Beispiel in Bezug auf Reaktionszeit?

Antwort: Aus meiner Erfahrung heraus muss ich sagen, dass der Support von Red Hat sehr gut ist. Sie bieten eine schnelle Fehlerbeseitigung an. Ebenfalls ist die Anforderung an neue Eigenschaften ein weiterer Pluspunkt und deren Umsetzung wird häufig, wenn möglich, erfüllt. Dies ist bei IBM eher schwierig.

Frage: Bei der Überlegung, ob JBoss eingeführt werden soll, wurde sich sicherlich mit der Thematik des Supports stark auseinandergesetzt. Wurde überlegt, ob man eine internen Support einführt oder mit einen externen Provider anstatt Red Hat zusammen arbeitet?

Antwort: Bei der Frage, ob man eine Abteilung mit dem Support von Red Hat beschäftigt, wurde relativ schnell festgelegt, dass man diese Vorgehensweise nicht weiterbrachtet, da man kein reines Softwarehaus ist und es eine Vielzahl an externen Dienstleistern auf dem Markt gibt. Es blieb dann nur noch die Frage offen, mit welchen externen Dienstleister man zusammenarbeiten möchte. Nach einer Marktanalyse fiel die Wahl dann auf den Softwarehersteller Red Hat.

3. Ausfallsicherheit

Frage: Welche Maßnahmen bzw. Werkzeuge werden eingesetzt, um die Ausfallsicherheit zu erhöhen?

Antwort: Die Software Jopr, was eine quellcodeoffene Variante des JBoss ON ist, bietet Funktionen zur Verwaltung, Konfiguration, Monitoring und Alarmierung. So lassen sich zum Beispiel die Anzahl der verwendeten CPU´s zählen, was für die Kostenfrage von Bedeutung ist. Ebenfalls lassen sich die Anwendungen damit komplett überwachen.

Frage: Der JBoss besitzt die Eigenschaft der Eigenüberwachung. Unerwartete Abstürze von Komponenten führen zu einem automatischen Neustart dieser Komponente, um laufenden Betrieb sicherzustellen. Wie wird diese Eigenüberwachung gesteuert bzw. erweitert?

Antwort: Mit Hilfe von Jopr lassen sich spezielle Bedingungen mit Folgeaktionen definieren. Wenn während der Laufzeit der Anwendung, gegen diese Bedingun-

gen verstoßen wird, erfolgt eine Fehlermeldung. Zum Beispiel kann die Heap-Size definiert werden mit einem anschließenden Heap-Dump.

Frage: Wie ist der Aufwand einzuordnen im Verhältnis zu dem IBM Produkt?

Antwort: Die Höhe des Aufwandes richtet sich an die Anzahl der gewünschten Bedingungen, die man umsetzten möchte. Zu Beginn ist der Aufwand sicherlich sehr hoch, jedoch ist der Arbeitsaufwand an einem späteren Zeitpunkt um einiges vereinfacht und weniger fehleranfällig.

4. Administration

Frage: Die Administration des Applikationsserver ist natürlich auch ein wichtiges Thema. Wie ist die Administration organisiert? Werden spezielle Werkzeuge verwendet?

Antwort: Es gibt die Möglichkeit des Einsatzes von Administrations-Werkzeugen, um eine grafische Oberfläche zur erleichterten Handhabung zu erreichen. Für den JBoss lässt sich der JON (JBoss Operations Network) einsetzen. Der Einsatz von solchen Werkzeugen bei der Administration ist jedoch genaustens zu hinterfragen, da Eingaben über eine Benutzeroberfläche schnell zu Fehleingaben führen können. Vor allem im Bezug auf die Revisionssicherheit muss jeder Schritt nachvollziehbar sein. Daher ist der Einsatz von Skripten emphelenswert.

Frage: Die Registrierung sowie die Verteilung von Zugriffsrechten der Benutzern ist zum Beispiel mit Hilfe des JAAS (Java Authentication and Authorization Service) möglich. In welche Kategorien werden die Zugriffsrechte unterteilt?

Antwort: 1. Entwicklungsumgebung (Local)

→ Entwickler erhält Admin-Recht

2. Testumgebung (JBoss AS)

→ Entwickler erhält Admin-Recht

3. Systemintegrationsumgebung (JBoss AS)

→ Einzelne Personen erhalten spezielle Rechte

4. Produktionsumgebung

→ Einzelne Personen erhalten spezielle Rechte

5. Logging

Frage: Das Framework Log4J dient zum Loggen von Anwendungsmeldungen. Wird dieses verwendet oder wird ein anderes Framework eingesetzt?

Antwort: Das Log4J wird eingesetzt und ist im Vergleich der beiden JEE-Anwendungsservern, IBM WebSphere und JBoss, gleich, da dies auf Java basiert.

6. Skalierbarkeit

Frage: Bietet der Einsatz des JBoss AS irgendwelche Vorteile gegenüber dem IBM WebSphere AS in Bezug auf die Skalierbarkeit?

Antwort: Aufgrund der Tatsache, dass beide Applikationsservern auf der Grundlage der Java-Technologie basieren, bietet keiner der beiden Applikationsservern einen erheblichen Vorteil in diesem Bereich.

Frage: Wo liegt in etwa die Grenze der Skalierbarkeit einer JBoss-Server-Instanz?

Antwort: Zur Zeit ist es so, dass bei dem JBoss AS in einem Cluster 16 Knoten (JBoss-Server-Instanz) vorhanden sind.

7. Modul-Schnittstellen

Frage: Lässt sich JBoss leicht erweitern? Wenn ja, nur mit bestimmten Tools? Gibt es Schwierigkeiten?

Antworten: Der JBoss lässt sich ohne weitere Schwierigkeiten leicht erweitern. Es ist dabei in der Regel egal, ob es Open-Source-Produkte, kommerzielle Produkte oder selbstentwickelte Produkte sind.

8. Schulungsaufwand

Frage: Wie wurde die Problematik bzgl. der Schulungen gelöst? Finden Schulungen intern oder extern statt?

Antwort: Bei jeder Einführung eines neuen Produktes muss man sich mit diesem Thema auseinandersetzen. Es gibt eine große Anzahl an Dienstleitern, die Schulungen für die Konfiguration, Installation, etc. anbieten. Es empfiehlt sich zunächst eine Ausschreibung aufzugeben und die Themen für den Einsatz im Unternehmen individuell anzupassen.

9. Performance

Frage: Welcher JEE-Anwendungsserver ist im Vergleich schneller? Gibt es überhaupt gravierende Unterschiede?

Antwort: Wie schon zuvor erwähnt, gibt es die Möglichkeit den JBoss schlanker zu gestalten, indem man nicht benötigte Module entfernt. Dies ist sicherlich beim Starten des JBoss ein erheblicher Vorteil. Sonst ist kein Performance-Unterschied vorhanden. Das Hauptproblem der Performance ist in der Regel die Anwendung selbst.

10. Differenzen der JEE-Anwendungsservern IBM WebSphere und JBoss

Frage: Der IBM WebSphere AS ist im Gegenteil zu dem JBoss ein kompaktes Paket. Der JBoss ist sehr modular. Welche Vorteile und Nachteile ergeben sich diesbezüglich?

Antwort: Aufgrund der Modularität ist JBoss flexibler und einfacher zu verwalten. Es können gewünschte Module ausgewählt werden und nicht benötigte müssen nicht mit installiert werden. Dies führt zur Einsparung von Ressourcen. Ein weiterer Vorteil ist, dass Dateien des JBoss im Nachhinein einfach entfernt werden können. Dies ist beim IBM WebSphere AS nicht möglich. Darin besteht auch die Gefahr, dass zu viele Dateien oder die falsche Datei gelöscht wird und infolgedessen das System nicht mehr läuft.

Frage: Muss man bei dem JBoss auf die neuste Version des SDK warten?

Antwort: Nein, da man flexibel in der Gestaltung des JBoss ist.

11. Quellcodeoffenheit

Frage: Wodurch zeichnet dich die sogenannte Quellcodeoffenheit gegenüber dem IBM WebSphere AS aus?

Antwort: Das Unternehme IBM stellt nur die Binaries zur Verfügung und nicht den Quellcode. Dadurch besteht keine Möglichkeit für einen Entwickler bei technischen Problemen zu debuggen nach dem Fehler im Quellcode.

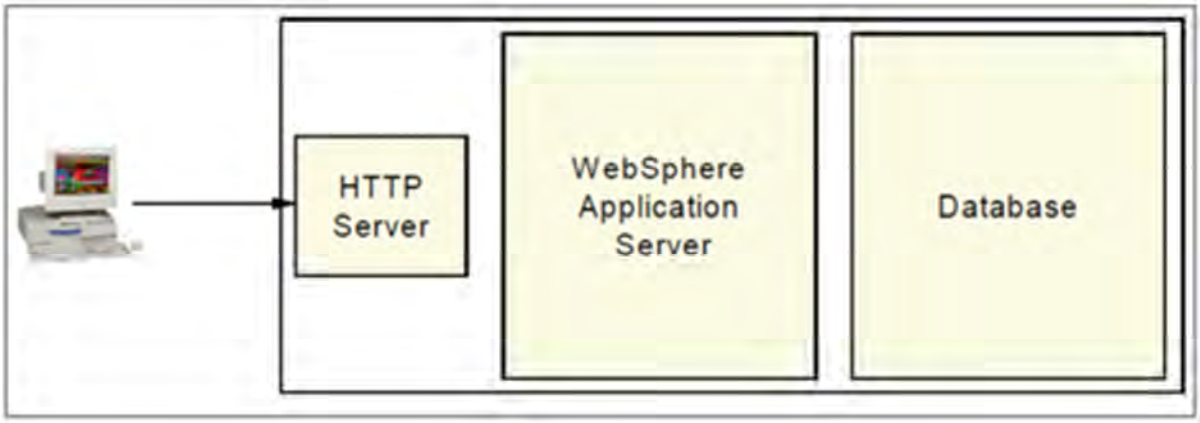
Dies ist ein Nachteil für den Entwicklungsprozess. Der JBoss liefert nicht nur die Binaries, sondern auch den Quellcode. Aufgrund dieser Tatsache kann auch debugged werden ohne auf den Hersteller angewiesen zu sein.

12. Mainframe

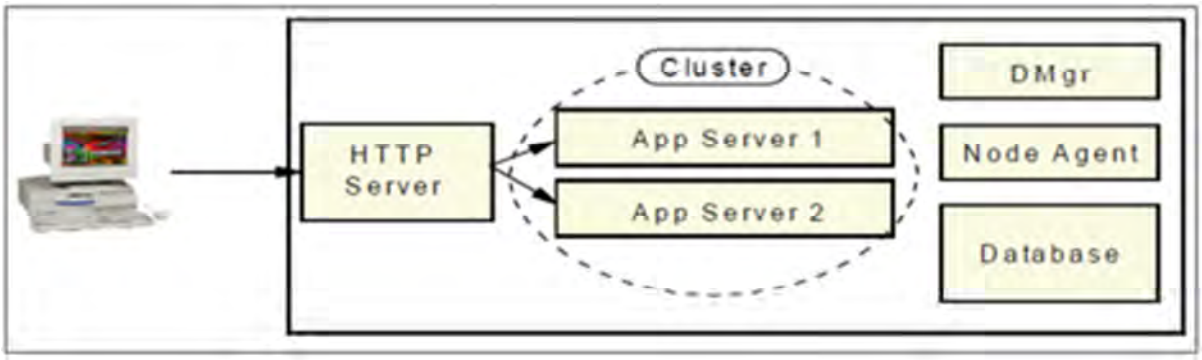
Frage:	Ist der JBoss auf dem Mainframe einsetzbar?
Antwort:	Die verwendete Technologie von IBM ermöglicht keinen Einsatz des JBoss auf dem Mainframe.

Anhang 16: Ebenen der Hochverfügbarkeit [221, 222, 223]

WebSphere system HA level 1: Es sind die drei Komponenten HTTP Server, WebSphere Application Server und Datenbank zu sehen. Sobald eine dieser Komponenten ausfällt, ist die Anwendung nicht mehr erreichbar.

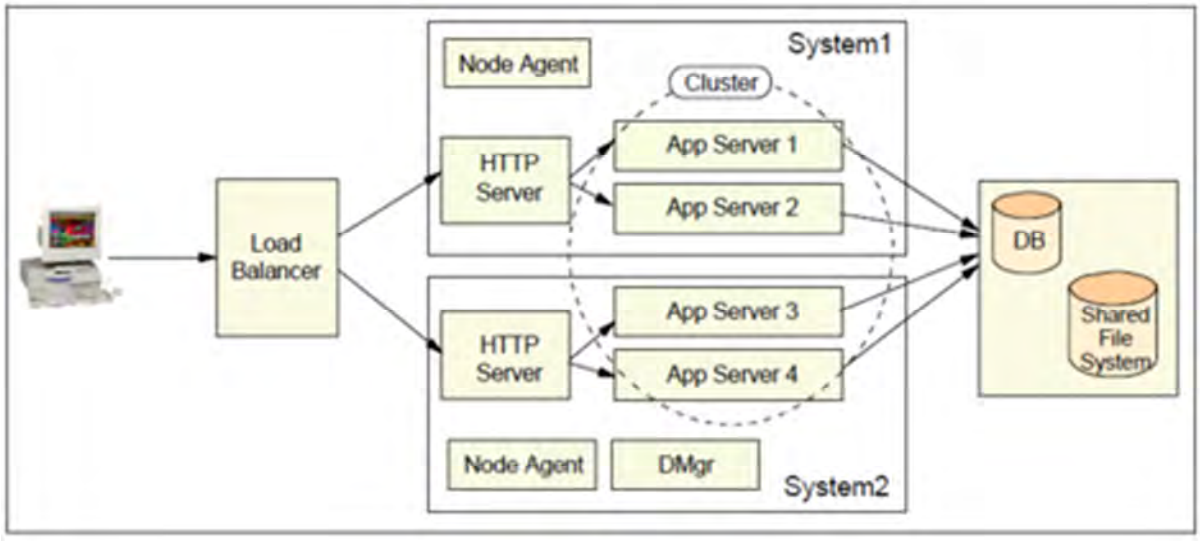


WebSphere system HA level 2: Im Unterschied zur vorherigen Ebene wird hier ein Cluster genutzt, der auf einem Server läuft. Sollte hier ein Application Server ausfallen ist die Anwendung noch immer erreichbar. Bei den anderen Komponenten verhält es sich wie in der vorherigen Ebene erläutert.

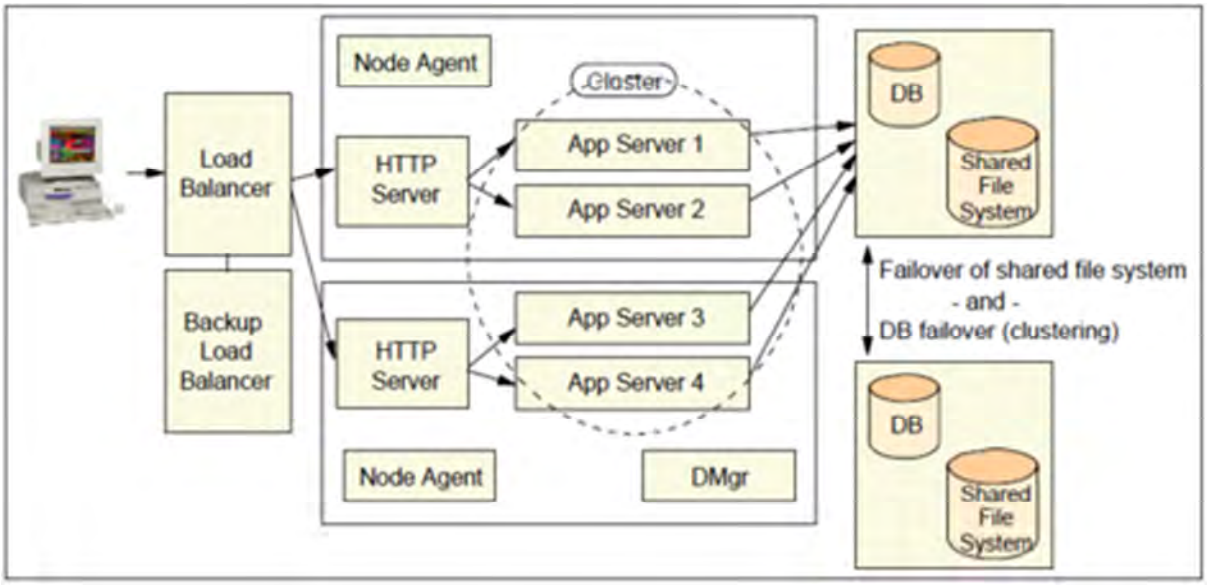


WebSphere system HA level 3: Hier wird das System aus der zweiten Ebene dupliziert und die Zugriffe über einen Boadbalancer verteilt. Zusätzlich werden die Daten der Datenbank auf einem separaten Storageserver gelagert, wo beide Systeme drauf zugreifen können. Sollte auf dieser Ebene der Loadbalancer oder die Datenbank ausfallen, ist die Anwendung nicht mehr erreichbar.

[221] Vgl. Schmitt, Michael (2005), S. 3ff.
[222] Vgl. Roehm, B. u. a. (2005), 35 ff.
[223] Vgl. WinfWiki (2010)



WebSphere system HA level 4: Im Gegensatz zur vorherigen Ebene gibt es hier auch für den Loadbalancer und die Datenbank eine Redundanz. Auf dieser Ebene gibt es keine Komponente mehr, die bei einem Ausfall die Anwendung unerreichbar macht.



WebSphere system HA level 5: ist die letzte Ebene und hier wird das komplette System vor dem Katastrophenfall geschützt. Das ganze System muss nach diesem Ansatz in einem anderen Rechenzentrum identisch aufgebaut werden. Sollte das eigentliche Rechenzentrum einer Katastrophe, wie z.B. einem Feuer, zum Opfer fallen, kann das Backup-System die ankommenden Anfragen bearbeiten.

Anhang 17: Gegenüberstellung des Schulungsaufwand - Ordix AG

Titel	Administration und Konfiguration für JBoss 7	WebSphere Application Server Installation und Administration
Beschreibung	In diesem Seminar werden Sie mit der Installation und Konfiguration der neuen JBoss Application Server-Version 7 vertraut gemacht. Sie lernen den internen Aufbau und die verschiedenen Betriebsmodi kennen. Des Weiteren erlernen Sie das Deployment von Java EE-Anwendungen und die Anbindung von Ressourcen wie Datenbanken und Nachrichten-Queues.	In diesem Seminar werden Sie mit der Installation und Administration des WebSphere Application Servers vertraut gemacht. Außerdem werden die Themen Hochverfügbarkeit, Lastverteilung und automatische Software-Verteilung intensiv in Theorie und Praxis behandelt.
Seminarpreis	1.190,00 € (zzgl. ges. gült. MwSt.)	1.390,00 € (zzgl. ges. gült. MwSt.)
Seminardauer	3 Tage	3 Tage
Voraussetzungen	Verständnis von Client/Server-Architekturen, Grundverständnis von Web-Applikationen. Kenntnisse der Java Enterprise Architektur (P-JEE-02) sind von Vorteil.	Grundlegende IT-Kenntnisse (Windows oder Unix). Kenntnisse der Java Enterprise Architektur (P-JEE-02) sind von Vorteil.
Zielgruppe	Administratoren, Planer, Entscheider für JBoss-Plattformen.	Systemadministratoren, Entwickler, Programmierer.
Seminarinhalte	• Kurze Einführung in Java EE und JBoss: Historie, Lizenzierung, Produkte • Architekturübersicht: Interner Aufbau, JBoss-Modules • Administration: Konfiguration des Kernsystems, erweiterte Konfigurationsmöglichkeiten • Standalone vs. Domain Mode • Die JBoss Admin CLI: Das Command-Line-Interface, Verwendung von Skripten • Installation einer Beispielapplikati-	• Überblick über die Java EE-Architektur • Architektur des WebSphere Application Servers • Installation des WebSphere Application Servers • Grundlagen der Administration mit der Administrations-Konsole und CLI-Werkzeugen • Installation einer Anwendung • Verwaltung und Installa-

	on: Deployment, Startparameter, Erstellen einer eigenen Konfiguration • Konfiguration des Loggings der Anwendungen und des Application Servers • Einbindungen von Datenbanken, XA und Non-XA-Datasources • Einbindung von Message Queues • JBoss Security: Überblick über JAAS, Authentifizierung, Autorisierung, Login-Module • JBoss Custering: Aufbau eines JBoss Clusters, Verteilen von Anwendungen im Cluster-Verbund • Vertiefung der Theorie durch praktische Übungen und Beispiele	tion von Data Sources • Fehlersuche • Absicherung der Administrations-Konsole • Erstellung eines Clusters zur Lastverteilung und Hochverfügbarkeit • Konfiguration von Java Messaging Service (JMS) • Konfiguration und Deployment von Message Driven Beans • Vertiefung der Theorie durch praktische Übungen und Beispiele

Anonymisiert (6 Seiten)

Anhang 22: GEICO migriert JBoss Middleware [224]

Es wird eine erfolgreiche Einführung des JBoss AS anhand des Autoversicherer GEICO, mit Sitz in den Vereinigten Staaten von Amerika, dargestellt. In den nachfolgenden Kapiteln werden die Hintergründe, die betrieblichen sowie technischen Herausforderungen, die Umsetzung, der Auswahlprozess sowie das resultierende Ergebnis beschrieben.

Hintergründe

Die Covernment Employees Insurance Company (GEICO) ist der drittgrößte Anbieter von Autoversicherungen in den USA basierend auf den Prämieneinnahmen der letzten 12 Monate (Stand 2009). Das Unternehmen ist für beinahe 9 Millionen versicherte Person über 14,4 Millionen Kraftfahrzeuge verantwortlich. Zusätzlich bietet das Unternehmen GEICO Versicherungspolicen für Motorräder, Gebäudeschutz und Hausrat an.

Der Hintergrund der Überlegung bezüglich einer Umstellung der Hauseigenen Middleware auf die Enterprise Edition des JBoss war in erster Linie die komplexe Verwaltbarkeit des aktuellen Systems. Ebenfalls waren die anfallenden Kosten im Verhältnis zu Performance und Skalierbarkeit nicht befriedigend. Diese Gründe waren ausschlaggebend für die Suche nach einem Alternativsystem, welches die Anforderungen des Unternehmens GEICO besser abdeckt.

Betriebliche und technische Herausforderungen

Im Vorfeld der bevorstehenden Umstellung identifizierte die IT-Abteilung des Unternehmens GEICO die Hauptprobleme, die durch den Einsatz der damals aktuellen Middleware entstanden:

- **Kosten**

Das Lizenzmodell war eine Art zeitgebundener Lizenzvertrag, welcher sich auf die Anzahl der eingesetzten Applikationsservern innerhalb eines definierten Zeitraums bezog. Zu dieser Zeit erfuhr das Unternehmen GEICO einen prägnanten Größenzuwachs. Dies führte zu einer exponentiellen Steigerung der Lizenzkosten und infolgedessen zu einem erheblichen Anstieg der Gesamtkosten.

- **Performance**

Die Umstellung des Java Development Kit (JDK) von Version 1.4 auf Version 1.5 sollte einen Anstieg der gesamten Systemleistungen erbringen. Vor allem eine Verbesserung der Antwortzeiten des Applikationsservers sollte durch dieses Upgrade erreicht werden. Diese

[224] Siehe dazu: Red Hat, Inc. (2005a)

erhofften Anforderungen konnten jedoch nicht erfüllt werden. Nach 8 Wochen intensivstem Testen und aufwendiger Neuabstimmungen des Systems konnte der Zustand vor dem Upgrade des JDK wiederhergestellt werden. Die Einführung war nicht nur sehr zeitaufwendig, sondern auch mit sehr hohen Kosten verbunden, da zusätzlich externe Berater rekrutiert werden mussten, um die ursprüngliche Performanceleistung wiederherzustellen.

- **Speicherprobleme**

Unter dem Einsatz der damals aktuellen Middleware wurden vermehrt unerwartete und ungeklärte Speicherprobleme festgestellt. Speichertests, die von Entwicklern des der IT-Abteilung durchgeführt wurden, lieferten häufig fehlerhafte Resultate zurück.

- **Dokumentation und Support**

Ein weiteres Problem war die mangelhafte Dokumentation der verwendeten Middleware. Infolgedessen war es für das Unternehmen GEICO sehr schwer das Java Application Programming Interface (API) zu verstehen und zu applizieren. Des Weiteren war der Erwerb von neuen Tools, die zum Beispiel Speicherprobleme beheben konnten, relativ schwierig. Für jedes Projekt, das dieses und andere Probleme zu lösen versuchte, mussten zwangsläufig wieder externe Berater eingebunden werden.

- **Inszenierung**

Auf Grund der oben genannten Schwierigkeiten setzten immer mehr Entwicklerteams JBoss Technologien ein und entwickelten erste Anwendungen unter dieser neuen Technologie. Im Laufe der Zeit wurde jedoch diese Parallelstrategie immer komplexer, da vorübergehend für zwei Technologien Konfigurationsänderungen vorgenommen werden mussten. Dieser zusätzliche Aufwand konnte nicht abfangen.

- **Auswahlprozess**

Das Unternehmen GEICO führte weitreichende Forschungsmaßnahmen durch, um die bestmöglichen Alternativen zu untersuchen und zu evaluieren. Im Zuge dieser Forschungs-maßnahmen kristallisierte sich die JBoss Middleware vom Softwarehersteller Red Hat als potenziell mögliche Systeme heraus. Dieses System erfüllt die geforderten Anforderungen der Anwendungen und der Infrastruktur am ehesten.

Das Unternehmen führte Performance- und Lasttests auf der alten Middleware und der neuen JBoss Middleware durch. Folgende zwei Resultate dieser Tests verdeutlichen stark den Unterschied beider Systeme:

1. CPU-Lasteinsparung:

Unter Verwendung derselben Umgebung und Hardware konnte die JBoss Middleware eine CPU-Lasteinsparung von 70% im Vergleich zur bestehenden Middleware verzeichnet werden.

2. Performanceverbesserungen:

Bei der bestehenden Middleware wurden 1440 Personenstunden zur Verbesserung der Performance benötigt. Beim Einsatz der JBoss Middleware konnte dasselbe Ziel innerhalb von 40 Personenstunden bewerkstelligt werden.

Neben den Tests wurden zusätzliche Referenzen von anderen Organisationen gleicher Größe zur Entscheidungshilfe herangezogen. Auf Grundlage dieser Informationen entschied sich das Unternehmen GEICO für die Umstellung auf die Enterprise Edition des JBoss.

Umsetzungslösung

Das GEICO implementierte im Rahmen der JBoss Enterprise Application Platform (EAP) in der Ersten Umgebung 540 Prozessoren. Weitere 350 Prozessoren sollten zu einem späteren Zeitpunkt folgen. Als Ziel wurde eine Immigration 2 Geschäftskritischer Applikationen innerhalb von 3 Monaten definiert. Weiterhin stand dem Unternehmen ein sogenannter Technical Account Manager (TAM) während der Einführung von JBoss für den Support zur Seite.

Ergebnis

Spätere Analysen zeigten, dass das Unternehmen GEIOCO durch die Umstellung der Middleware einige Vorteile erzielen konnten. An dieser Stelle werden fünf herausstechende Vorteile aufgelistet:

- Einsparung der Gesamtbetriebskosten von ca. 30%
- Verdreifachung der Datenverarbeitungsmenge
- Reduzierung der Ressourcennutzung um beinahe zweidrittel
- Reduzierung der Gesamtressourcenauslastung von über 50% auf unter 10%
- Steigerung der Skalierbarkeit ohne zusätzlichen Hardware-Einsatz

Quellenverzeichnisse

Literaturverzeichnis

Adler, S. (o.J.) — Java RMI, München: GRIN Verlag GmbH

Agopyan, A. (2009) — WebSphere Application Server V7.0: Technical Overview, New York: International Business Machines Corporation

Buss, M. (2005) — Aspektorientierte Programmierung, Konzepte und Werkzeuge für die Softwareentwicklung mit Java, o. O.: o. Verl.

Cecchet, E./Marguerite, J./ Zwaenepoel, W. (2002) — Performance and Scalability of EJB Applications, Seattle: Association for Computing Machinery

Chuvakin, A./Schmidt, J./ Phillips, C. (2013) — Logging and Log Management, The Authoritative Guide to Understanding the Concepts Surroounding Logging and Log Management, Waltham: Elsevier, Inc.

Faustmann, H. /u.a. (2009) — SAP NetWeaver AS Java – Systemadministration, Bonn: Galileo Press

Fischer, S./Koschel, A./ Wagner, G. (2006) — J2EE/Java EE kompakt, Enterprise Java: Konzepte und Umfeld, 2.Aufl., München: Elservier GmbH

Forrester Research, Inc. (2012) — The Total Economic Impact To IBM WebSphere Application Server Migrating From An Open Source Environment, Cambridge: Forrester Research, Inc.

Forrester Research, Inc. (2009) Total Economic Impact Of Red Hat JBoss Enterprise Application Platform, Cambridge: Forrester Research, Inc.

Fowler, M. (2003) Patterns für Enterprise Application-Architekturen, Heidelberg: mitp Verlag

Gucer, V./Godoy, A. (2005) Deployment Guide Series: IBM Tivoli Monitoring 6.1, New York: International Business Machines Corporation

Hodgson, J. u. a. (2001) IBM WebSphere Application Server Advanced Edition 3.5, WebSphere JMS/JTA support for MQSeries Overview, New York: International Business Machines Corporation

Hohberger, R. (2010) WebSphere Application Server, Die Plattform für unternehmenskritische Java Enterprise Anwendungen, New York: International Business Machines Corporation

IBM Corporation (2005) IBM WebSphere Application Server V&, Data Replication Service, New York: International Business Machines Corporation

IBM Corporation (2009) IBM Software Support Handbock, Worldwide World Class Software Support, New York: International Business Machines Corporation

IBM Corporation (2010) WebSphere Application Server V7 Administration and Configuration Guide, New York: International Business Machines Corporation

IBM Deutschland GmbH (2013q) IBM Training, IT-Training 2013, Ehningen: IBM Deutschland GmbH

IBM Passport Advantage and Passport Advantage Express (2012) — Subscription and Support FAQ, o. O.: o. Verl.

Jamae, J./Johnson, P. (2009) — JBoss im Einsatz: Den JBoss Application Server konfigurieren, München, Carl Hanser Verlag

Kiwitter, P. (2008) — Eclipse in der Java-Entwicklung, München: Addison-Wesley Verlag

Kotermanski, R. (2009) — IBM WebSphere Application Server V7 vs. JBoss Application Server V5 TCO Analysis, Pittsburgh: Summa Technologies, Inc.

Langer, T./Reiberg, D. (2006) — J2EE und JBoss: Grundlagen und Profiwissen, Verteilte Enterprise Applikationen auf Basis von J2EE, JBoss & Eclipse, München: Hanser Verlag

Kohler, F. (2013) — Evaluation der Clusterbetriebsfähigkeit einer JSF-Anwendung mit Fokus auf der transiente Datenspeicherung beim Maskenwechsel, Stuttgart: Kohler, F.

Leander, R. (2000) — Building Application Servers, Cambridge: Cambridge University Press

Lee, R. (2008) — Software Engineering Research, Management and Applications, Berlin: Springer Verlag

Pientka, F. (2009) — Apache Geronimo, Handbuch für den Java-Applikationsserver, Heidelberg: dpunkt.verlag GmbH

Red Hat, Inc. (2005) — JBoss Network Monitoring, Atlanta: Red Hat, Inc.

Red Hat, Inc. (2005a) — GEICO migrates to JBoss, JBoss Innovation Award Winner 2009: Superior Alternatives

Red Hat, Inc. (2012e) — Jboss enterprise brms: the power of jboss drools. The assurance of Red Hat, Red Hat

Richards, N./Griffith, S. (2006) — JBoss, Das Entwicklerheft, Köln: O'Reilly Verlag GmbH & Co. KG

Robinson, S. (2011) — IBM WebSphere Application Server 8.0 Administration Guide, Lear nto administer a reliable, secure, and scalable environment for running apllications with WebSphere Application Server 8.0, Birmingham: Packt Publishing Ltd.

Roehm, B. u. a. (2005) — WebSphere Application Server Network Deployment V6: High Availability Solutions, New York: International Business Machines Corporation

Schindler, H. (2011) — Nichtfunktionale Eigenschaften, 3. Architekturen und Algorithmen zur Herstellung nichtfunktionaler Eigenschaften, Ilmenau: Technische Universität Ilmenau

Siedersleben, J. (2003) — Softwaretechnik, Praxiswissen für Softwareingenieure, 2. Aufl., München: Hanser Verlag

Stark, T. (2010) — Java EE 5, Einstieg für Anspruchsvolle, München: Pearson Studium

Sun Microsystems, Inc. (2006) — Java™ Platform, Enterprise Edition 5 (Java™ EE 5) Specification ("Specification"), Santa Clara: Sun Microsystems, Inc.

Termin, T./Kröger, R. (2006) — Eine generische Performance-Instrumentierung des JBoss Application Servers, Wiesbaden: o. Verl.

Vogel, O. u.a. (2009) Software-Architektur, Grundlagen – Konzepte – Praxis, 2. Aufl., Heidelberg: Spektrum Akademischer Verlag

Zangemeister, C. (1976) Nutzwertanalyse in der Systemtechnik,. 4. Auflage, München : Wittemannsche Buchhandlung

Zangenmeister, C. (2000) Erweiterte Wirtschaftlichkeitsanalyse (EWA), Bremerhaven : Wirtschaftsverlag NW

Verzeichnis der Internet- und Intranet-Quellen

Barbara, A (o.J.)	Ausarbeitung im Fach Open Source zum Thema Lizenzen: LGPL, MS-PL, MS-RL, http://www.feyrer.de/OPS/vortraege/ws0708-ascher-paper.pdf, Abruf: 03.12.2012
Bögel, S. (2005)	RMI – Remite Method Invocation, http://www.sbgl.de/rmi/, Abruf: 11.12.2012
Bruusgaard, T./Twist, J. (2012)	Logging Levels and how to use them, http://www.Thejoyofcode.com/Logging_Level_and_how_to_use_them.aspx,
CENSIS (2012)	Open Source VS kommerzielle Produkte, http://www.censis.de/internet-ebusiness/content-management/open-source-vs-kommerzielle-produkte/, Abruf: 05.01.2013
DATACOM Buchverlag GmbH (2012a)	Administration, http://www.itwissen.info/definition/lexikon/Administration-administration.html, Abruf: 03.12.2012
DATACOM Buchverlag GmbH (2012b)	Application Server, http://www.itwissen.info/definition/lexikon/Anwendungs-Server-application-server.html, Abruf: 11.01.2013
DATACOM Buchverlag GmbH (2012c)	AOP (aspect oriented programming) http://www.itwissen.info/definition/lexikon/Aspektorientierte-Programmierung-aspect-oriented-programming-AOP.html, Abruf: 09.01.2013

DATACOM Buchverlag GmbH (2012d) Drei-Schiten-Architektur http://www.itwissen.info/definition/lexikon/Drei-Schichten-Architektur-three-tier-architecture.html Abruf: 03.01.2013

Dpunkt Verlag (2002) Der Java Authentication and Authorization Service (JAAS), http://www.dpunkt.de/java/Programmieren_mit_Java/Sicherheit/14.html, Abruf: 28.11.2012

Faure, M. (2012) Loadbalancer http://www.global-village.de/hosting/server-erweiterungen/loadbalancer/ Abruf: 14.01.2013

Galileo Press (2009) Logging mit Java, http://www.iks.hs-merseburg.De/~uschroet/Literatur/Java_Lit/JAVA_Insel/Javainsel_25_001.htm, Abruf: 16.12.2012

gutefrage.net GmbH (2010) Was bedeutet Neuralität? – Eine Begriffsdefinition, http://www.helpster.de/was-bedeutet-neutralitaet-eine-begriffsdefinition_106358#zur-anleitung, Abruf: 27.12.2012

Heise Zeitschriften Verlag (2013a) Monitoring-Software Opsview 3.0 setzt auf Nagios 3 auf, http://www.heise.de/open/meldung/Monitoring-Software-Opsview-3-0-setzt-auf-Nagios-3-auf-206818.html, Abruf: 07.01.2013

Heise Zeitschriften Verlag (2013b) JBoss-Werkzeuge mit mehr Assistenz, http://www.heise.de/developer/meldung/JBoss-Werkzeuge-mit-mehr-Assistenz-1742252.html, Abruf: 12.01.2013

IBM Corporation (o. J.) WebSphere Application Server Feature Pack for Web 2.0 and Mobile, http://www-01.ibm.com/software/webservers/appserv/was/featurepacks/web20-mobile/, Abruf: 14.01.2013

IBM Corporation (2012) IBM WebSphere Application Server system log files, http://publib.boulder.ibm.com/infocenter/iisinfsv/v8r5/index.jsp?topic=%2Fcom.ibm.swg.im.iis.found.admin.common.doc%2Ftopics%2Fwsadmin_was_log_files.html, Abruf: 25.12.2012

IBM Deutschland GmbH (2012a) WebSphere Application Server, http://www-142.ibm.com/software/products/de/de/appserv-was/, Abruf: 21.12.2012

IBM Deutschland GmbH (2012b) Information Center von WebSphere Application Server Version 7.0, http://pic.dhe.ibm.com/infocenter/wasinfo/v7r0/index.jsp, Abruf: 27.12.2012

IBM Deutschland GmbH (2012c) List of supported software for WebSphere Application Server V7.0, http://www-01.ibm.com/support/docview.wss?uid=swg27012369, Abruf: 27.12.2012

IBM Deutschland GmbH (2012d) Wsadmin tool, http://publib.boulder.ibm.com/infocenter/wsdoc400/v6r0/index.jsp?topic=/com.ibm.websphere.iseries.doc/info/ae/ae/rxml_commandline.html, Abruf: 28.12.2012

IBM Deutschland GmbH (2012e) Functions or features in Rational Application Developer that are and are not in WebSphere Integration Developer, http://www-01.ibm.com/support/docview.wss?uid=swg21614641, Abruf: 11.01.2013

IBM Deutschland GmbH (2012f) Software Support Handbook, http://www-304.ibm.com/webapp/set2/sas/f/handbook/offerings.html#3, Abruf: 13.01.2013

IBM Deutschland GmbH (2012g) Erzielen Sie maximalen Return on Investment in IBM Software, http://www-01.ibm.com/software/de/support/priority.html, Abruf: 27.12.2013

IBM Deutschland GmbH (2012h) Support Lifecycle, http://www-01.ibm.com/software/websphere/support/lifecycle/, Abruf: 28.12.2012

IBM Deutschland GmbH (2012i) High Availability Manager, http://pic.dhe.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=%2Fcom.ibm.websphere.nd.doc%2Finfo%2Fae%2Fae%2Fcrun_ha_hamanager.html, Abruf: 09.01.2013

IBM Deutschland GmbH (2012j) High Availability Manager, http://pic.dhe.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=%2Fcom.ibm.websphere.nd.doc%2Finfo%2Fae%2Fae%2Fcrun_ha_hamanager.html, Abruf: 11.01.2013

IBM Deutschland GmbH (2012k) Application server log files, http://pic.dhe.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=%2Fcom.ibm.commerce.admin.doc%2Fconcepts%2Fclswaslogs.htm, Abruf: 15.15.2012

IBM Deutschland GmbH (2012l) 1996, http://www-03.ibm.com/ibm/history/history/year_1996.html, Abruf: 12.01.2013

IBM Deutschland GmbH (2012m) Tivoli IBM Software Produktesuche, http://www-142.ibm.com/software/products/de/de/tivoli, Abruf: 12.01.2013

IBM Deutschland GmbH (2012n) IBM Tivoli software für integrierte Service Management, http://www-01.ibm.com/software/de/tivoli/, Abruf: 09.12.2012

IBM Deutschland GmbH (2012o) Bewährte Verfahren für die Erkennung und Lösung von Infrastrukturproblemen, http://www-142.ibm.com/software/products/de/de/tivomoni/, Abruf: 05.01.2013

IBM Deutschland GmbH (2012p) Spezifikationen und API-Dokumentation, http://pic.dhe.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=%2Fcom.ibm.websphere.web20fepja-xrs.doc%2Finfo%2Fae%2Fae%2Frovr_specs.html&resultof%3D%2522%2569%2573%256f%2522%2520%2522%2538%2538%2535%2539%2522%2520, Abruf: 12.01.2013

IBM Deutschland GmbH (2012r) Frisch veröffentlicht: Neuer IBM Trainingskatalog 2013, http://www-03.ibm.com/press/de/de/pressrelease/39208.ws, Abruf: 18.12.2012

IBM Deutschland GmbH (2012s) Kursbeschreibung: IBM WebSphere Application Server V7 Administration on Windows, http://www-304.ibm.com/jct03001c/services/learning/ites.wss/de/de?pageType=course_description&includeNotScheduled=y&courseCode=WA370DE, Abruf: 08.01.2013

IBM Deutschland GmbH (2012t) WebSphere Virtual Enterprise, http://www-01.ibm.com/software/webservers/appserv/extend/virtualenterprise/, Abruf: 14.01.2013

IBM Deutschland GmbH (2012u) Wann sollte ein High Availability Manager verwendet werden?, http://pic.dhe.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=%2Fcom.ibm.websphere.nd.doc%2Finfo%2Fae%2Fae%2Fcrun_ha_ham_required.html, Abruf: 12.01.2013

IBM Deutschland GmbH (2012v) Preise anzeigen und kaufen, https://www-112.ibm.com/software/howtobuy/buyingtools/paexpress/Express?part_number=D0IAMLL%2CD0IAQLL%2CD0IATLL&catalogLocale=de_DE&Locale=null&country=DEU&PT=html&TACTICS=%26S_TACT%3D%26S_CMP%3D%26brand%3D&ibm-submit=Preise+anzeigen+%2F+kaufen, Abruf: 12.01.2013

IBM Deutschland GmbH (2011) Implementierung während des Betriebs und dynamisches Neuladen, http://pic.dhe.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=%2Fcom.ibm.websphere.express.doc%2Finfo%2Fexp%2Fae%2Ftrun_app_hotupgrade.html, Abruf: 16.01.2013

IBM Deutschland GmbH (2009) Verwenden von Scripts zur Implementierung von WSRR in WebSphere Application Server in separaten Stufen, http://publib.boulder.ibm.com/infocenter/sr/v6r3/index.jsp?topic=%2Fcom.ibm.sr.doc%2Ftwsr_install_deploy_scripts_stages.html, Abruf: 16.01.2013

Infinite Corporation (2013) Red Hat, http://www.infinitecorporation.com/shop/category.php?id_category=39, Abruf: 12.01.20013

IRIAN Solutions Softwareentwicklungs- und Beratungsgesellschaft mbH (o. J.a) Einführung in JavaServer Faces, http://jsfatwork.irian.at/book_de/introduction.html#!idx:/introduction.html:1, Abruf: 12.01.2013

IRIAN Solutions Softwareentwicklungs- und Beratungsgesellschaft mbH (o. J.b) Eclipse und JBoss-Tools, http://jsfatwork.irian.at/book_de/appendix_eclipse.html#!idx:/appendix_eclipse.html:11, Abruf: 12.01.2013

IT-Administrator (o.J.a) Softwareverteilung, http://www.it-administrator.de/themen/server_client/grundlagen/110904.html, Abruf: 10.01.2013

IT-Administrator (o.J.b) — Der sichere Weg zu neuer Software, http://www.it-administrator.de/magazin/heftarchiv/artikel/24584.html, Abruf: 10.01.2013

java-server.net (2007) — Java Server Pages Grundlagen http://java-server.net/jsp_grundlagen.html Abruf: 20.12.2012

Joseph, J. (o.J.) — JBoss application server production settings and performance tuning, http://www.jaysonjc.com/programming/how-to-configure-jboss-as-production-settings-and-tuning-tips.html, Abruf: 02.12.2012

Kioskea Deutschland (2013) — Das HHTP Protokoll, http://de.kioskea.net/contents/internet/http.php3, Abruf: 12.01.2013

Köhler, K. (2003) — JMX - Management ohne Wasserkopf, http://www.oio.de/public/java/jmx/jmx.htm, Abruf: 29.11.2012

Krechner, S. (2008) — Mit dem ESB zur SOA, http://www.gi-hb-ol.de/esbsoa.pdf, Abruf: 07.01.2013

Microsoft Corporation (2012) — Skalierbarkeit der Server, http://technet.microsoft.com/de-de/library/cc757735(v=ws.10).aspx, Abruf: 13.12.2012

Nagios Enterprise, LLC. (2012) — JBOSS Monitoring, http://www.nagios.com/solutions/jboss-monitoring, Abruf: 14.12.2012

Nagios Enterprise, LLC. (2008) Nagios XI Screenshots, http://www.nagios.com/products/nagiosxi/screenshots, Abruf: 14.12.2012

NetMediaInteractive (2012) Red Hat zertifiziert JBoss-Entwickler, http://www.zdnet.de/41552508/red-hat-zertifiziert-jboss-entwickler/, Abruf: 23.12.2012

New Elements GmbH (2012) JBoss AS 7 – Administration Grundlagen- Seminare, http://www.it-schulungen.com/seminare/applikationsserver-middleware/jboss/jboss-7-administration.html, Abruf: 16.12.2012

NETWAYS GmbH (2012) Nagios Monitoring System, http://www.netways.de/de/produkte/nagios, Abruf: 29.11.2012

NETWAYS GmbH (2013a) Nagios Referenzen, http://www.netways.de/Referenzen.272.0.html?&no_cache=1&tx_t3references_pi1%5bem%5d=3, Abruf: 07.01.2013

NETWAYS GmbH (2013b) Nagios Supportverträge, http://www.netways.de/de/produkte/nagios/supportvertraege/, Abruf: 11.01.2013

o. V. (o. J.a) JBoss Application Server 7, http://javathreads.de/data/uploads/2011/09/110918_JBossAdminConsole.jpg, Abruf: 10.01.2013

o. V. (o. J.b) JBoss Enterprise Application Platform 6.0, https://access.redhat.com/knowledge/docs/resources/docs/en-US/JBoss_Operations_Network/3.1/html/How_to_Manage_JBoss_Servers/images/as7/profiles-console.png, Abruf: 05.12.12

o. V. (o. J.c) Nagios, http://www.lnl.infn.it/~epics/images/NagiosServicesDetail.jpg, Abruf: 11.01.2013

o. V. (o. J.d) Nagios, http://dcala.files.wordpress.com/2010/02/imagenreferencia1nagios.png, Abruf: 11.01.2013

Oletzky, T. (2012) Defintion Dunkelverarbeitung http://www.versicherungsmagazin.de/Definition/33903/dunkelverarbeitung.html, Abruf: 16.01.2013

Oracle (o. J.) Java Management Extensions (JMX) Technology, http://www.oracle.com/technetwork/java/javase/tech/javamanagement-140525.html, Abruf: 28.11.2012

Ordix AG (2012) Seminare zum Thema Web und Application Server, http://training.ordix.de/index_webserver.htm, Abruf: 14.01.2013

Orientation in Objects GmbH (2012a) Schulung: JBoss A7 - Konfiguration und Administration, http://www.oio.de/seminar/open-source/jboss-schulung.htm, Abruf: 16.12.2012

Orientation in Objects GmbH (2012b) EJB Clustering mit JBoss, http://www.oio.de/public/java/ejb/jboss-clustering.htm, Abruf: 16.12.2012

Pressetext (2013) Salin AG setzt auf Nagios a la LINBIT, http://www.pressetext.com/news/20090519041, Abruf: 08.01.2013

Red Hat GmbH (2012) JBoss Operations Network, http://de.redhat.com/products/jbossenterprisemiddleware/operations-network/, Abruf: 05.12.2012

Red Hat, Inc. (2005a) Getting Started with JBoss 4.0, http://docs.jboss.org/jbossas/getting_started/v4/html/tour.html, Abruf: 05.12.2012

Red Hat, Inc. (2005b) High Availability Enterprise Services via JBoss Cluster, http://docs.jboss.org/jbossas/jboss4guide/r4/html/cluster.chapt.html, Abruf: 09.12.2012

Red Hat, Inc. (2005c) The JBoss Server - A Quick Tour, http://docs.jboss.org/jbossas/getting_started/v4/html/tour.html, Abruf: 24.11.2012

Red Hat, Inc (2010) Partner-Finder, http://redhat.force.com/finder/Partner_Finder, Abruf: 23.12.2012

Red Hat, Inc. (2011) JBoss Web, http://www.jboss.org/jbossweb/, Abruf: 11.01.2013

Red Hat, Inc. (2012a) JBoss Application Administration I with Exam (JB249),http://www.redhat.com/training/courses/jb249/, Abruf: 16.12.2012

Red Hat, Inc. (2012b) JBoss Tools, http://www.jboss.org/tools, Abruf: 12.01.2013

Red Hat, Inc. (2012c) JBoss Tools, https://community.jboss.org/en/tools/blog/2012/12/10/jboss-tools-4-and-developer-studio-6-is-ready, Abruf: 12.01.2013

Red Hat, Inc. (2012i) Developer Studio Portfolio Edition, http://de.redhat.com/products/jbossenterprisemiddleware/developer-studio/, Abruf: 12.01.2013

Red Hat, Inc. (2012d) JBoss Community or JBoss enterprise, http://www.redhat.com/products/jbossenterprisemiddleware/community-enterprise/, Abruf: 11.01.2013

Red Hat, Inc. (2012e) JBoss AS rolling deployment using shell script, https://community.jboss.org/wiki/JBossASRollingDeploymentUsingShellScript, Abruf: 02.01.2013

Red Hat, Inc. (2012f) The JMX Console https://access.redhat.com/knowledge/docs/en-US/JBoss_Enterprise_Web_Platform/5/html/Getting_Started_Guide/The_JBoss_Server___A_Quick_Tour-The_JMX_Console.html Abruf: 14.01.2013

Red Hat, Inc. (2012g) Hot-deployment of services in JBoss https://access.redhat.com/knowledge/docs/en-US/JBoss_Enterprise_Web_Platform/5/html/Getting_Started_Guide/The_JBoss_Server___A_Quick_Tour-Hot-deployment_of_services_in_JBoss.html Abruf: 14.01.2013

Red Hat, Inc. (2012h) JBoss Enterprise Middleware and JBoss Operations Network Product Update and Support Policy https://access.redhat.com/support/policy/updates/jboss_notes/ Abruf: 28.12.2012

Red Hat, Inc. (2013) JBoss Community - Leute, https://community.jboss.org/people, Abruf: 09.01.2013

Red Hat, Inc. (2013b) Overview of JBoss Operations Network, http://www.redhat.com/v/swf/jboss_on/2122_RedHat_JONOverview.html, Abruf: 11.01.2013

Red Hat, Inc. (2013c) Nagios Plugin Service, https://access.redhat.com/knowledge/docs/en-US/JBoss_Operations_Network/1/html/Users_Guide/sect-JBoss_Operations_Network-Users_Guide-Script_Services-Nagios_Plugin_Service.html, Abruf: 10.01.2013

Red Hat, Inc. (2013d) JBoss Community and JBoss Enterprise, http://www.europe.redhat.com/products/jboss/community-enterprise/, Abruf: 10.01.2013

Red Hat, Inc. (2013e) Produktaktualisierungs- und Supportrichtlinie für JBoss Enterprise Middleware und JBoss Operations Network, https://access.redhat.com/support/policy/updates/jboss_notes/, Abruf: 13.01.2013

Red Hat, Inc (o.J. a) Scalability and High Availability fort he JBoss Apllication Server, http://www.jboss.org/jbossclustering, Abruf: 11.12.2012

Red Hat, Inc. (o.J. b) Elegant Administration, http://www.jboss.org/jbossas, Abruf: 13.12.2012

Red Hat, Inc. (o.J. c) Installation And Getting Started Guide, http://docs.jboss.org/jbossas/docs/Installation_Guide/beta500/html-single/index.html, Abruf: 28.11.2012

Red Hat, Inc (o.J. d) Chapter 4. Clustered Deployment Options, http://docs.jboss.org/jbossclustering/cluster_guide/5.1/html/deployment.chapt.html, Abruf: 11.01.2013

Red Hat, Inc (o.J. e) Setting the Standard for High Performance Messaging, http://docs.jboss.org/jbossmessaging/docs/usermanual-2.0.0.beta4/html_single/index.html#ha, Abruf: 10.01.2013

SEO-united.de (o.J.) Monitoring, http://www.seo-united.de/glossar/monitoring, Abruf: 11.12.2012

Springer Fachmedien Wiesbaden GmbH (o.J. a) Agent, http://wirtschaftslexikon.gabler.de/Definition/agent.html?referenceKeywordName=Software+Agent, Abruf: 11.01.2013

Springer Fachmedien Wiesbaden GmbH (o.J. b) — Nutzwertanalyse, http://wirtschaftslexikon.gabler.de/Definition/nutzwertanalyse.html, Abruf: 09.01.2013

Software & Support Media GmbH (2011) — JBoss AS7: Das Leichtgewicht der Enterprise-Java-Entwicklung, http://it-republik.de/jaxenter/artikel/JBoss-AS7-Das-Leichtgewicht-der-Enterprise-Java-Entwicklung-3986.html, Abruf: 0601.2013

Stansberry, Brian (o.J.) — Cluster Definition, http://docs.jboss.org/jbossas/docs/Clustering_Guide/4/html/clustering-intro-def.html, Abruf: 10.12.2012

Subbaro, M. (2008) — Configuring Logging in JBoss, http://java.dzone.com/articles/configuring-logging-jboss, Abruf: 10.11.2012

Sun Microsystems, Inc. (2007) — JavaTM 2 Platform Enterprise Edition, v 5.0 API Specifications, http://docs.oracle.com/javaee/5/api/, Abruf: 08.01.2013

Sys-Con Media (2013) — Birth of a Platform: Interview with Don Ferguson, http://websphere.sys-con.com/node/43095, Abruf: 07.01.2013

TechTerms (2012) — API, http://www.techterms.com/definition/api, Abruf: 11.01.2013

The Eclipse Foundation (2013) — Eclipse documentation, http://www.eclipse.org/documentation/, Abruf: 12.01.2013

United News Network GmbH (2012) SOFTCON wird JBoss Authorized Service Partner, http://www.pressebox.de/pressemitteilung/jboss-group-europe/SOFTCON-wird-JBoss-Authorized-Service-Partner/boxid/15136, Abruf: 23.12.2012

Vertical Media GmbH (o.J.) Application-Programming-Interface (API), http://www.gruenderszene.de/lexikon/begriffe/application-programming-interface-api, Abruf: 04.12.2012

VinayTech (2009) Clustering in JBoss Application Server, http://www.vinaytechs.com/2009/11/clustering-in-jboss-application-server.html, Abruf: 10.12.2012

WinfWiki (2010) Leistungsanalyse Websphere, http://winfwiki.wifom.de/index.php/Leistungsanalyse_Websphere#cite_note-ftn28-25, Abruf: 14.01.2013

DHBW
Duale Hochschule
Baden-Württemberg
Stuttgart

Marktanalyse Enterprise Service Busse

kommerziell vs. Open Source

Schriftliche Ausarbeitung
im Rahmen der Lehrveranstaltung „Projekt des 5. Semesters“
Kompetenzzentrum Open Source (KOS)

Vorgelegt von

Natascia Ferro, Robert Kolanda,
Erik Sattelmair, Alexander Sikora und
Moritz Thies

am 25.01.2013

Fakultät Wirtschaft
Studiengang Wirtschaftsinformatik
WWI2010V

Inhaltsverzeichnis

Abkürzungsverzeichnis

ACID	**A**tomicity, **C**onsistency, **I**solation and **D**urability
AE	**A**nwendungs**e**ntwicklung
ALE	**A**pplication **L**ink **E**nabling
API	**A**pplication **P**rogramming **I**nterface
AQ	**A**dvanced **Q**ueuing
B2B	**B**usiness **to B**usiness
BMW	**B**ayerische **M**otoren**w**erke
BPEL	**B**usiness **P**rocess **E**xecution **L**anguage
CBR	**C**ontent **B**ased **R**outing
CEI	**C**ommon **E**vent **I**nfrastructure
CLI	**C**ommand **L**ine **I**nterface
COBOL	**C**ommon **B**usiness **O**riented **L**anguage
CORBA	**C**ommon **O**bject **R**equest **B**roker **A**rchitecture
CSV	**C**omma **S**eparated **V**alues
CXF	**C**elti**XF**ire
DHBW	**D**uale **H**ochschule **B**aden-**W**ürttemberg
DPF	**D**istributed **P**rocessing **F**ramework
EDI	**E**lectronic **D**ata **I**nterchange
EDV	**e**lektronische **D**aten**v**erarbeitung
EIS	**E**nterprise **I**nformation **S**ystem
EJB	**E**nterprise **J**ava **B**ean
ESB	**E**nterprise **S**ervice **B**us
FAQ	**F**requently **A**sked **Q**uestions

FTP	**F**ile **T**ransfer **P**rotocol
GE	**G**eneral **E**lectric
GEICO	**G**ouvernement **E**mployees **I**nsurance **Co**mpany
GNU	**G**NU is **n**ot **U**nix
HTTP(**S**)	**H**yper**t**ext **T**ransfer **P**rotocol (**S**ecure)
IBM	**I**nternational **B**usiness **M**achines Corporation
IMAP	**I**nternet **M**essage **A**ccess **P**rotocol
IP	**I**nternet **P**rotocol
IT	**I**nformations**t**echnologie
J2EE	**J**ava **to E**nterprise **E**dition
JAAS	**J**ava **A**uthentication and **A**uthorization **S**ervice
JAXR	**J**ava **A**PI for **X**ML **R**egistries
JBI	**J**ava **B**usiness **I**ntegration
(**j**)**BPM**	(**j**ava) **B**usiness **P**rocess **M**odeling
JCA	**J**ava EE **C**onnector **A**rchitecture
JDBC	**J**ava **D**ata**b**ase **C**onnectivity
JMS	**J**ava **M**essage **S**ervice
JMX	**J**ava **M**anagement **E**xtensions
JNDI	**J**ava **N**aming and **D**irectory **I**nterface
JSR	**J**ava **S**pecification **R**equest
JTA	**J**ava **T**ransaction **A**PI
(**j**)**UDDI**	(**j**ava) **U**niversal **D**escription, **D**iscovery and **I**ntegration
KOS	**K**ernkompetenzzentrum **O**pen **S**ource
KPI	**K**ey **P**erformance **I**ndicator
LDAP	**L**ightweight **D**irectory **A**ccess **P**rotocol

LGPL	**L**esser **G**eneral **P**ublic **L**icense
MOM	**M**essage **O**riented **M**iddleware
MQ	**M**essage **Q**ueue
NCAR	**N**ational **C**enter for **A**tmospheric **R**esearch
NMR	**N**ormalized **M**essage **R**outer
NYSE	**N**ew **Y**ork **S**tock **E**xchange
ODE	**O**rchestration **D**irector **E**ngine
OSGi	**O**pen **S**ervices **G**ateway **i**nitiative
OSI	**O**pen **S**ource **I**nitiative
PDF	**P**ortable **D**ocument **F**ormat
PGP	**P**retty **G**ood **P**rivacy
PHP	**H**ypertext **P**rocessor
POJO	**P**lain **O**ld **J**ava **O**bject
POP	**P**ost **O**ffice **P**rotocol
QoS	**Q**uality **o**f **S**ervice
REST	**Re**presentational **S**tate **T**ransfer
RMI	**R**emote **M**ethod **I**nvocation
SAP	**S**oftware, **A**nwendungen und **P**rodukte in der Datenverarbeitung
SCA	**S**ervice **C**omponent **A**rchitecture
SLA	**S**ervice **L**evel **A**greement
SMTP	**S**imple **M**ail **T**ransfer **P**rotocol
SOA	**S**ervice **O**riented **A**rchitecture
SOAP	**S**imple **O**bject **A**ccess **P**rotocol
SS4TLS	**S**ecure **S**ocket **for T**ransport **L**ayer **S**ecurity

SSL	**S**ecure **S**ockets **L**ayer
TCP	**T**ransfer **C**ontrol **P**rotocol
TIBCO	**T**he **I**nformation **B**us **C**ompany
TOE	**T**ransaction **O**riented **E**ndpoint
UDP	**U**ser **D**atagram **P**rotocol
URI	**U**niform **R**esource **I**dentifier
VM	**V**irtual **M**achine
WAS	IBM **W**ebSphere **A**pplication **S**erver
WID	IBM **W**ebSphere **I**ntegration **D**eveloper
WS	**W**eb **S**ervice
WSDL	**W**eb**S**ervices **D**escription **L**anguage
WSRR	IBM **W**ebSphere **S**ervice **R**egistry and **R**epository
XA	E**x**tended **A**rchitecture
(**X**)**HTML**	E**x**tensible **H**ypertext **M**arkup **L**anguage
XML	E**x**tensible **M**arkup **L**anguage
XSLT	E**x**tensible **S**tylesheet **L**anguage **T**ransformation
z/OS	**z**ero downtime/**O**peration **S**ystem

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

In den letzten Jahren etablierte sich in Unternehmen für die Datenverarbeitung großer IT-Systeme die Anwendung von Enterprise Service Bussen (ESB). Dies ist damit zu begründen, dass serviceorientierte Architekturen (SOA), dessen zentraler Bestandteil ein ESB ist, Business-Services zentral verwalten und über eine einheitliche Schnittstelle zur Verfügung stellen.

Das Forschungsprojekt Kernkompetenzzentrum Open Source (KOS), welches an der DHBW Stuttgart stattfindet, führt unter anderem eine Marktanalyse von Enterprise Service Bus Systemen durch. Das Ergebnis wird für ein Stuttgarter Versicherungsunternehmen als Kooperationspartner angefertigt. Aus diesem Grund wird sich das Forschungsergebnis speziell auf die Bedürfnisse des Partnerunternehmens ausrichten. Das Ziel der Arbeit ist es, in einer abschließenden Empfehlung ein Produkt zu bestimmen, welches die funktionalen und nichtfunktionalen Anforderungen des Partners auf höchstem Qualitätsniveau erfüllt und gleichzeitig aus wirtschaftlichen Gesichtspunkten betrachtet, ein Kosten-Nutzen-Optimum darstellt. Das Forschungsteam hat sich darüber hinaus entschieden, im Rahmen der Forschungsarbeit einen Prototypen anzufertigen, um einen typischen Anwendungsfall in der praktischen Umsetzung zu veranschaulichen.

Der Aufbau der vorliegenden Arbeit untergliedert sich in zwei Themenbereiche. Zum einen findet die Untersuchung und Bewertung der zielrelevanten Informationen im **Theorieteil** statt. In diesem werden zu Beginn die Grundlagen vermittelt, die zum Verständnis der anschließenden Analyse ausgewählter ESB-Systeme, sowohl aus dem kommerziellen als auch dem Open Source Bereich, dienen. Auf Basis der daraus gewonnenen Erkenntnisse und der Untersuchung der Kundenanforderungen findet eine Gegenüberstellung statt, dessen Evaluierungsergebnis maßgeblich für die Empfehlung an den Kooperationspartner sein wird. Zusätzlich wird die Forschung die Anfertigung eines Prototyps einschließen, um ausgewählte Anforderungen zu veranschaulichen.

Da dieses Forschungsprojekt die erstmalige Zusammenarbeit des Kooperationspartners mit dem Team darstellt, wird neben der eigentlichen Zielerreichung des Theorieteils angestrebt, eine hohe Transparenz des Vorgehens für den Kunden zu liefern. Aufgrund der Veranschaulichung der Ressourcenplanung und des Projektverlaufs soll dem Kooperationspartner eine Einordnung der fachlichen Ergebnisse aus einer **Projektmanagement**-Sicht im gleichnamigen Kapitel ermöglicht werden.

2 Projektmanagement

Das Projektmanagement dient der Planung, Kontrolle und Steuerung des Forschungsprojektes, um die zur Verfügung stehenden Ressourcen optimal zu nutzen. Beim Auftreten konkurrierender Einflussfaktoren werden durch das Management Steuerungsmaßnahmen ermöglicht, die zur Risikovermeidung oder -minimierung dienen. Als Rahmenbedingungen werden personelle und zeitliche Ressourcen betrachtet, um anschließend einen Zeitplan zur Meilensteinplanung anzufertigen. Weiterhin werden Teamentscheidungen über die Auswahl von Werkzeugen dokumentiert und kurz erläutert.

2.1 Ressourcenplanung

Das Forschungsprojekt findet im Zeitraum vom 20.11.2012 bis 25.01.2013 statt. Es stehen insgesamt 15 Termine à 3 Stunden zur Verfügung, in welchen das Team, bestehend aus 5 Personen, die Marktanalyse betreibt und die Ergebnisse zusammenstellt. Das Projekt Marktanalyse Enterprise Service Busse umfasst somit insgesamt 225 Mannstunden[1].

2.1.1 Forschungsteam

Zur optimalen Nutzung individueller Stärken der Teammitglieder ist zu Beginn des Projektes folgende Rollenaufteilung zu den hauptsächlichen Themenschwerpunkten erfolgt:

- Projektmanagement: Natascia Ferro
- Fachliche Grundlagen: Erik Sattelmair, Alexander Sikora
- Analyse kommerzieller Tools: Robert Kolanda
- Analyse Open Source Tools: Robert Kolanda, Alexander Sikora, Moritz Thies
- Umsetzung des Prototypen: Erik Sattelmair

Die Analyse der Kundenanforderungen und die Beurteilung der gesammelten Erkenntnisse erfolgte im gesamten Team.

2.1.2 Zeitplan

Zu Beginn des Projektes wurden sämtliche im Projekt durchzuführenden Aktivitäten formuliert und in Einzelschritten festgehalten, sodass eine zeitliche Planung unter Berücksichti-

[1] Berechnung mit 1 Mannstunde = 60 Minuten

gung der Ressourcen und der inhaltlichen Abhängigkeiten durchgeführt werden konnte. Zu Beginn jedes Termins hat ein Statusabgleich stattgefunden, bei dem der aktuelle Fortschritt eines jeden Forschungsmitglieds mitgeteilt, offene Fragen geklärt und eine Übersicht über den weiteren Verlauf gegeben worden ist. Die anfängliche Planung ist Anhang 1 zu entnehmen. Dieser Zeitplan diente als Planungsorientierung während des Projektes, musste jedoch im Laufe des Fortschritts angepasst werden. Die geplante und letztlich benötigte Zeit in Mannstunden ist anschließend aus Tabelle 1 zu entnehmen.

	Thematik	Geplante Zeit	Benötigte Zeit	Erledigt
Projektmanagement	Projektplanung und Aktualisierung	16	12	✓
	Statusbericht	13	10	✓
	Qualitätsmanagement	30	28	✓
Forschung	Grundlagen	12	12	✓
	Kommerzielles Tool	12	11	✓
	Open Source Tools	12	15	✓
	Koop.-Partner Analyse	12	13	✓
	Bewertung	75	73	✓
	Prototyp	35	54	✓
Dokumentation	Dokumentaufbereitung	8	12	✓
	SUMME	225	240	

Tabelle 1: Geplanter und benötigter Aufwand in Mannstunden

Die Entscheidung, dass über die Termine hinaus Zeit in das Projekt investiert wird, ist innerhalb des Forschungsteams mit dem Vorhaben getroffen worden, dass die geplante Qualität umgesetzt wird. Der hauptsächliche Mehraufwand hat sich bei der Anfertigung des Prototyps ergeben, da der Aufwand für die Einarbeitung in das Tool bei der Planung unterschätzt worden ist.

2.1.3 Überblick über den Prüfablauf

Während des Projektes hat eine stetige Qualitätskontrolle stattgefunden. Der Prüfablauf aus Abbildung 1 zeigt den zeitlichen Verlauf der Aktivitäten und gleichzeitig den qualitativen Aspekt der Ergebnisse. Er spiegelt wider, dass die Fertigstellung der Ergebnisse in der geforderten Qualität frühzeitig erreicht worden ist. Lediglich der Aufwand für die Ausarbeitung des Prototyps wurde von der Planung unterschätzt, weshalb mehr Zeit investiert worden ist.

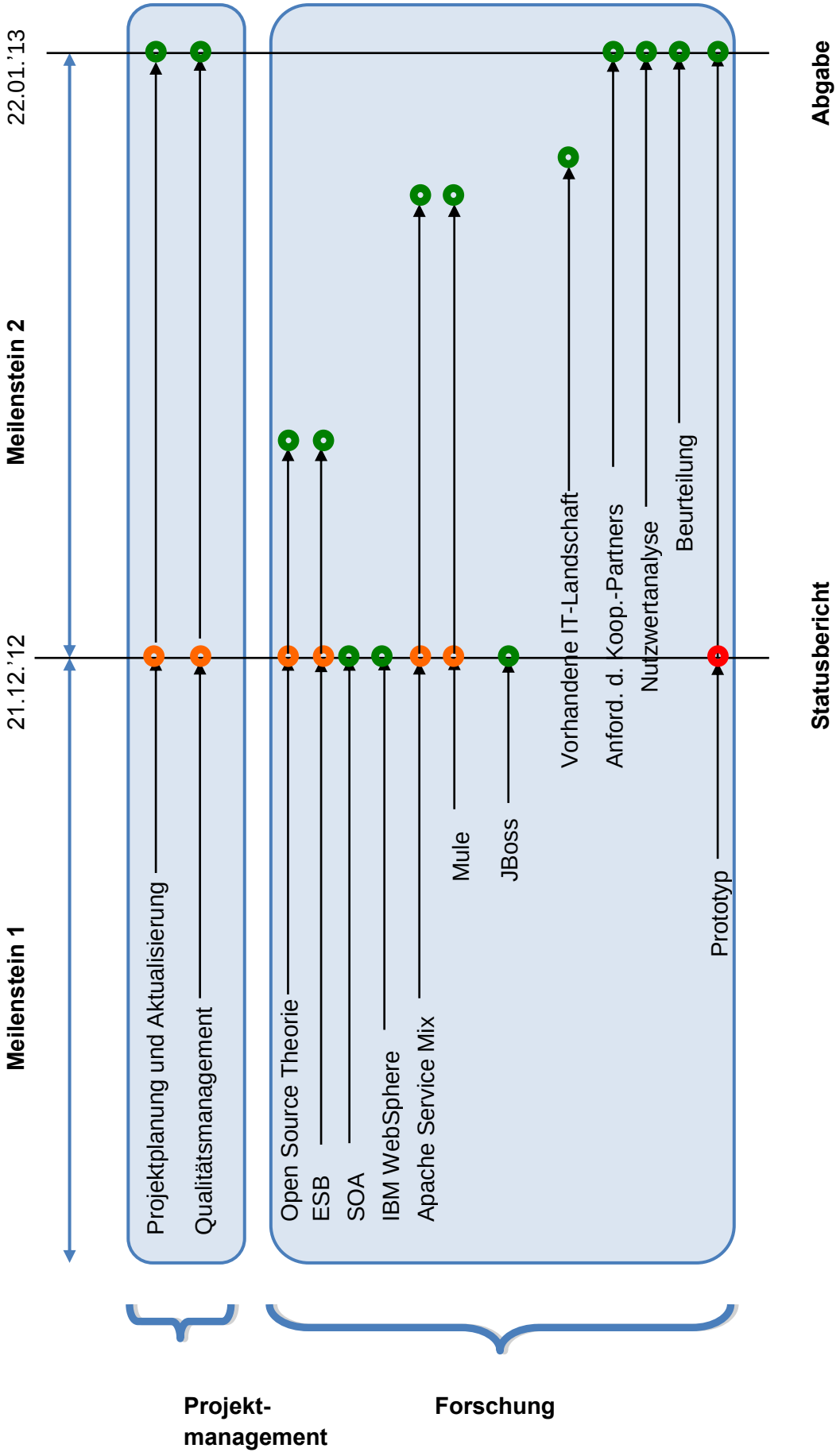


Abbildung 1: Zeitlicher Prüfablauf

2.2 Werkzeugauswahl

Zur Erstellung des Prototyps und zur Evaluierung der gegenübergestellten Produkte hinsichtlich der Kundenanforderungen werden Werkzeuge benötigt. Welche Kriterien zur Auswahl ausschlaggebend gewesen sind, werden im Folgenden benannt.

2.2.1 Tool zur Prototyperstellung

Das Tool, welches im Rahmen des Forschungsprojektes praktische Anwendung erfährt, ist aufgrund der Vorgaben des Kooperationspartners und der kostenlosen Verfügbarkeit aus dem Open Source Bereich auszuwählen. Zur Entscheidungsfindung trägt hierbei hauptsächlich eine große Popularität und eine hohe Nutzung bei, um von einer breiten und aktiven Entwicklercommunity zu profitieren. Ein weiteres Kriterium zur Auswahl des Tools ist eine schnelle Einarbeitungszeit und leichte Bedienbarkeit durch vorzugsweise einer grafischen Benutzeroberfläche zum Erstellen des Workflows, welcher als anschauliches Ergebnis erstellt wird. Diese Kriterien sind darüber hinaus vollständig in die Nutzwertanalyse eingeflossen.

2.2.2 Excel zur Nutzwertanalyse

Die Nutzwertanalyse untersucht qualitative und quantitative Aspekte der ESBs. Ziel dabei ist es, aufgrund von, in Abstimmung mit dem Partner, gewichteten Kriterien eine Entscheidungsgrundlage bereitzustellen, inwiefern die analysierte Software mit den Kundenanforderungen übereinstimmt.

In Kapitel 4.4 ist die Nutzwertanalyse mit Hilfe des Scoring-Modells dargestellt. Der maximal zu erreichende Score-Wert für ein Tool beträgt 100 Punkte. Die in funktional und nichtfunktional unterteilten Anforderungen weisen eine Gewichtung von 60% zu 40% auf. In Absprache mit dem Partner sind diese Anforderungen in Muss- und Wunschkriterien aufgeteilt worden. Dabei sind die Wunschkriterien selbst nochmals mit den Werten 1 (weniger wichtig) bis 10 (sehr wichtig) gewichtet. Die abschließende Score-Summe trifft sowohl als absolute als auch prozentuale Angabe eine Aussage darüber, welchen Grad an Übereinstimmung die jeweilige Software mit den Kundenanforderungen aufweist. Diese Umsetzungsbedingungen sind mit Excel einfach zu erstellen und übersichtlich darzustellen. Aus diesem Grund fiel die Entscheidung zur Durchführung der Nutzwertanalyse auf Microsoft Excel 2010.

Nach der Klärung organisatorischer Rahmenbedingungen wurden, erläutert der nächste Teil die fachlichen Grundlagen, welche für das weitere Verständnis essenziell sind.

3 Fachliche Grundlagen

In diesem Teil der Arbeit werden die Begriffe Open Source, serviceorientierte Architektur (SOA) und Enterprise Service Bus (ESB) detailliert betrachtet und deren Bedeutung für den weiteren Verlauf der Arbeit hervorgehoben.

3.1 Open Source

Der Begriff Open Source wurde, obwohl bereits zuvor quelloffene Software verwendet wurde, erst mit Gründung der Open Source Initiative (OSI) im Jahre 1998 etabliert. Als Open Source Software wird Software bezeichnet, die unter einer, von der OSI anerkannten Lizenz vertrieben wird und zehn, von der OSI definierte, Kriterien erfüllt (siehe Anhang 2).[2]

Wie sich Open Source Software von kommerzieller Software unterscheidet und in welchen Bereichen sich spezielle Vor- oder Nachteile ergeben, kann in der Studienarbeit Open Source – Der Weg in Das Unternehmen[3] recherchiert werden.

3.2 Serviceorientierte Architektur

In der Literatur finden sich viele Definitionen der Serviceorientierten Architektur (SOA). Stellvertretend soll die umfassende Definition nach Dostal u. a. angeführt werden:

> *„Service-orientierte Architekturen, kurz SOA, sind das abstrakte Konzept einer Software-Architektur, in deren Zentrum das Anbieten, Suchen und Nutzen von so genannten Diensten über ein Netzwerk steht. Dabei werden Dienste fast ausschließlich von Applikationen oder anderen Diensten genutzt. Hierfür ist es unerheblich, welche Hard- oder Software, Programmiersprache oder Betriebssystem die einzelnen Beteiligten verwenden. Unter anderem durch einheitliche Standards sollte es möglich sein, Dienste durch entsprechende Suchfunktionen zu finden, die von einem Anbieter genauso einfach publiziert werden können. [. . .] Im Idealfall ist sogar eine einfache Integration ganzer Anwendungen möglich.“*[4]

Wie obiger Definition zu entnehmen ist, besteht die Grundidee einer SOA darin, bestehende Funktionalität in feingranulare Dienste zu kapseln und in einem Netzwerk anzubieten.

[2] Harhoff, D. u. a. (2004), S. 20
[3] Zalewski, S. (2008), S. 8
[4] Dostal, W. u. a. (2005), S. 7

Wichtig zu erwähnen ist, dass eine SOA kein konkretes Produkt darstellt, sondern ein Architekturparadigma, das heißt ein abstrakter IT-Architekturansatz.[5]

3.2.1 Dienst und Dienstschnittstelle

Unter einem Dienst ist ein Programm oder auch eine Softwarekomponente zu verstehen, welche andere, wie z.B. Clients, Applikationen oder Services, lokal oder über ein Netzwerk nutzen kann. Damit diese Nutzung möglich ist, muss die Schnittstelle des Dienstes für potenzielle Nutzer in maschinenlesbarer Form (engl. servicedescription) öffentlich zugänglich sein. In diesem Zusammenhang hat sich WSDL als Schnittstellenbeschreibungssprache etabliert. Durch die Verwendung einer Schnittstelle wird das Konzept des Information-Hiding umgesetzt (Kapselung). Der Nutzer kennt dabei nur die Schnittstelle und interessiert sich nicht dafür, wie der Dienst realisiert ist.[6]

3.2.2 Ziele und Visionen einer SOA

Die Systemlandschaft vieler Unternehmen ist historisch gewachsen und damit zumeist heterogen. Das primäre Ziel einer SOA ist deshalb die Systemlandschaft effizient an Geschäftsprozessänderungen anpassen zu können. Ein weiteres Ziel einer SOA ist das Einsparen von Entwicklungs- und Wartungskosten. Durch Wiederverwendbarkeit von Businessfunktionen, interner sowie von Drittanbietern, werden Redundanzen verringert. Eine SOA verfolgt zudem das Ziel die Flexibilität der Systeme zu verbessern. Dies geschieht durch Modularisierung und damit einhergehend einer leichteren funktionalen Erweiterbarkeit. Dadurch wird der Servicegedanke realisiert. Bei einer Gesetzesänderung z. B. kann die Anwendung angepasst werden, indem betroffene Dienste ausgetauscht werden. Damit kann schnell auf externe Einflüsse reagiert werden.[7]

3.2.3 Aufbau einer SOA

Nachstehende Abbildung 2 veranschaulicht, welche Teilnehmer in einer SOA agieren:

[5] Vgl. Enzyklopädie der Wirtschaftsinformatik (2012)
[6] Vgl. Melzer, I. u. a. (2010), S. 14 f.
[7] Vgl. Commercial IT (o. J.)

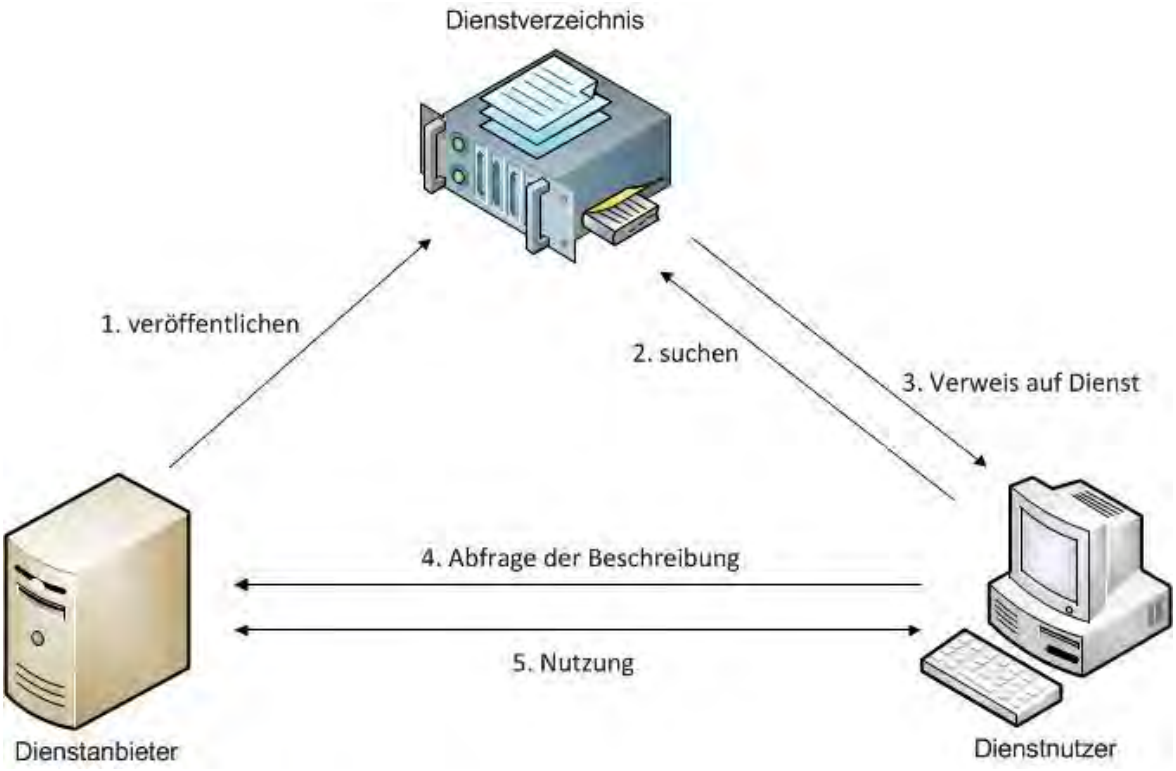


Abbildung 2: Das magische Dreieck in einer SOA[8]

Dienstanbieter

Der Dienstanbieter (engl. service provider) stellt seinen Dienst auf einer Plattform zur Verfügung. Auf dieser Plattform wird der Dienst betrieben. Damit der Dienst in einem Netzwerk gefunden werden kann, publiziert der Dienstanbieter seinen Dienst in einem Dienstverzeichnis.

Zu den Aufgaben eines Dienstanbieters zählen die Bereitstellung der Infrastruktur, Betrieb und Wartung, Datensicherung, Sicherheitsaspekte wie Authentifizierung und Autorisierung und Quality of Service (Verfügbarkeit).[9]

Der Dienst eines Dienstanbieters kann sich wiederum aus mehreren fremden Diensten zusammensetzen. Dies befreit den Dienstanbieter nicht von den im vorhergehenden Abschnitt genannten Aufgaben.

Dienstverzeichnis

Das Dienstverzeichnis (engl. registry) hat die Aufgabe potenziellen Nutzern das Auffinden von benötigten Diensten zu ermöglichen. In einem Dienstverzeichnis findet der Dienstnutzer

[8] Mit Änderungen entnommen aus: Melzer, I. u. a. (2010), S. 15
[9] Vgl. Melzer, I. u. a. (2010), S. 16

alle notwendigen Informationen, die für die Nutzung des Dienstes erforderlich sind. Ein Beispiel für solch eine Information stellt die physische Adresse des Dienstes dar.

Als abstraktes Modell hat sich diesbezüglich UDDI etabliert, welches verschiedenste Informationen über den Dienstanbieter (White Pages), die Kategorie des Dienstes (Yellow Pages) und die Schnittstellenbeschreibung des Dienstes (Green Pages) verwaltet.[10]

Dienstnutzer

Der Dienstnutzer (engl. service consumer) kann entweder ein Mensch sein, der mittels eines Browsers mit dem System interagiert oder eine Anwendung, die einen Dienst verwendet.

3.2.4 Serviceaufruf in einer SOA

Im Folgenden soll auf den Serviceaufruf und der Servicenutzung eingegangen werden[11]:

1. Der Dienstanbieter publiziert seinen Dienst in einem Dienstverzeichnis.
2. Der Dienstnutzer, welcher einen Dienst benötigt, durchsucht das Dienstverzeichnis nach einem geeigneten Dienst.
3. Wenn der Dienstnutzer den geeigneten Dienst gefunden hat, holt er sich die physische Adresse des Dienstes.
4. Dienstanbieter und Dienstnutzer einigen sich auf einen Dienstvertrag. Dabei wird die Schnittstellenbeschreibung abgefragt und Voraussetzungen für die Nutzung des Dienstes überprüft (z. B. Authentifizierung).
5. Bei Einigung kann der Dienst durch den Dienstnutzer genutzt werden.

Dieses Vorgehen ist jedoch noch mit einigen Defiziten behaftet. So existieren zum Beispiel weiterhin Punkt-zu-Punkt-Verbindungen zwischen den Diensten. Bei einer Vielzahl von Diensten ist dies nachteilig, da jede Punkt-zu-Punkt Verbindung individuell entworfen, entwickelt und administriert werden muss. Bei steigender Zahl der beteiligten Systeme kann die Anzahl der verwendeten Transportprotokolle und Datenformate ansteigen. Dies hat zur Folge, dass eine Kommunikation zwischen Dienstnutzer und Dienstanbieter nur mit hohem Aufwand stattfinden kann.

Bei einer Vielzahl an Kommunikationswegen ist die Verwendung eines zwischengeschalteten Vermittlers, einer Middleware in Form eines ESBs, sinnvoll. Zu den Aufgaben einer Midd-

[10] Vgl. Münz, S. (2008), S. 944
[11] Vgl. Melzer, I. u. a. (2010), S. 19

leware zählen unter anderem Protokollumwandlung und Routing von Nachrichten und damit eine sichere Zustellung. Kann diese Middleware beliebig erweitert werden und zeichnet sie sich durch einen protokollunabhängigen Aufbau aus, spricht man von einem Enterprise Service Bus.[12] Das Konzept eines ESBs wird im nächsten Kapitel vorgestellt.

3.3 Enterprise Service Bus

In diesem Kapitel wird erläutert, was unter einem ESB zu verstehen ist, wie er aufgebaut ist und welche Funktionen er bietet.

3.3.1 Definition

Analog zur serviceorientierten Architektur existiert auch für den ESB keine einheitliche Definition.[13] Aus diesem Grund sollen hier verschiedene Definitionen herangezogen werden, anhand derer der Begriff näherungsweise spezifiziert werden soll.

Zunächst soll eine erste Definition nach Roy Schulte, Analyst bei Gartner, betrachtet werden, welche als erste überhaupt gilt:[14]

> *„Ein Enterprise Service Bus (ESB) ist eine neue Architektur, die Web Services, nachrichtenorientierte Middleware, intelligentes Routing und Transformation nutzt. ESBs fungieren als ein leichtgewichtiges, allseits verfügbare Integrationsgerüst, durch welches Softwaredienste und Anwendungskomponenten fließen.“*[15]

Wie der Definition entnommen werden kann, versteht Schulte ein ESB als eine abstrakte Architektur, durch die eine Integration heterogener Technologien mittels verschiedener Funktionen möglich sein soll.

Der Autor Stefan Kischel liefert in seinem Artikel in der Fachzeitschrift OBJEKTspektrum aus dem Jahr 2003 eine etwas detaillierte Definition:

> *„Ein ESB ist ein messaging-basierter Softwarebus, der die Module eines Verbundes von Anwendungen zu einer verteilten Geschäftsanwendung verbindet. Seine Aufgaben sind:*
>
> - *Verbinden der Anwendungsteile, der Services, zu einer verteilten Geschäftsanwendung,*

[12] Vgl. Dangelmaier, W. u. a. (2002), S. 61
[13] Vgl. Gramm, A. (2007), S. 12
[14] Vgl. Hiekel, S. (2007), S. 23
[15] Vgl. ebenda, S. 23

- *Transformation und Zusammenstellen des Contents, den die Services austauschen, sowie*
- *Koordination des Verarbeitungsablaufes (BPM).*

Zentrale Elemente eines ESB sind:

- *ein Messaging-System, das die physikalische Ebene des Busses realisiert und die Vielzahl der notwendigen Fähigkeiten, wie z. B. sichere und garantierte Übermittlung, Persistenz bei asynchronen und lose gekoppelten Services bieten und*
- *ein Distributed Processing Framework (DPF), das als logische Ebene des Busses die Services verknüpft und die Prozesssteuerung übernimmt.*

Der ESB erfüllt diese Aufgabe durch eine Implementierung auf zwei Ebenen, die Transportschicht und die Prozessschicht.“[16]

Im Unterschied zur Schulte beschreibt Kischel den ESB konkret als einen Softwarebus und nicht als eine abstrakte Architektur. Als die Aufgaben eines ESB nennt er die Gewährleistung der Konnektivität der Services in einer SOA, Transformation der Daten und Koordination des Arbeitsablaufes. Neu im Vergleich zu Schulte ist hier der zuletzt genannte Punkt, wobei diese Definition das intelligente Routing vermissen lässt. Neu hingegen ist der exemplarische Aufbau eines ESBs, der hier gegeben wird. Auf den Aufbau wird an späterer Stelle tiefer eingegangen.

Als nächstes soll die Definition des Autors David Chappell zur Thematik ESB betrachtet werden:

„Ein ESB ist eine standardbasierte Integrationsplattform, die sowohl Nachrichten, Web Services, Datentransformation als auch intelligentes Routing miteinander verbindet, um eine erhebliche Zahl unterschiedlicher Anwendungen entlang erweiterter Unternehmen zuverlässig zu verbinden, ihre Interaktionen zu koordinieren und für transaktionale Integrität zu sorgen.“[17]

Chappell versteht den ESB als eine standardbasierte Integrationsplattform, welche unterschiedliche Anwendungen mittels Datentransformation und intelligentem Routing zuverlässig verbinden soll. Neu sind hier die Hervorhebung von Standards sowie die Aufgabe des ESB, für transaktionale Integrität zu sorgen.

[16] Vgl. Kischel, S. (2003), S. 37
[17] Vgl. Chappell, D. (2004), S. 1

Für diese Ausarbeitung soll abschließend noch Domenico Lorenzelli-Scholz zitiert werden:

> *„Moderne ESBs kann man sich wie einen Middleware-Switch vorstellen. Er funktioniert nach dem Prinzip eines Telekommunikations-Switches, der Aufrufe über ein weit verteiltes, heterogenes Netzwerk vermittelt. Diese neue Generation von ESBs vermittelt Nachrichten verschiedener Formate über eine verteilte Infrastruktur heterogener Middleware-Transportprotokolle, wie Tuxedo, MQ, CORBA, TIBCO oder HTTP. Dabei kann sich jeder beliebige Service in die SOA mittels ESB einklinken. Hierfür müssen alle relevanten Plattformen unterstützt werden (von mobilen Endgeräten bis hin zum Mainframe), die notwendige Servicequalität (Sicherheit, Transaktion, Hochverfügbarkeit usw.) muss bereitgestellt werden und nicht zuletzt müssen ESBs natürlich hoch performant sein.“*[18]

Aus obigen Definitionen wird für diese Ausarbeitung abgeleitet, dass ein ESB als Architektur gesehen werden kann, die mittels zuvor genannter Funktionalitäten heterogenen Systemen eine einheitliche Schnittstelle bietet und somit eine Integration dieser Systeme ermöglicht.

3.3.2 Architektur

Um das Verständnis über die Funktionsweise eines ESBs zu steigern, soll in diesem Abschnitt detailliert auf seinen technischen Aufbau eingegangen werden. Nach Kischel lässt sich der Aufbau eines ESBs in eine Transportschicht, eine Prozessschicht und Service-Container gliedern.

3.3.2.1 Transportschicht

Die Transportschicht wird durch ein Messaging-System realisiert, das den Bus physikalisch repräsentiert und über welches Nachrichten physisch übertragen werden. Das Messaging-System besteht aus einer auf Standards basierenden Message Oriented Middleware (MOM). Solche Standards sind nach Kischel JMS, JCA, SOAP sowie JDBC. Zusätzlich müssen von der Transportschicht Adapter unterstützt werden, durch welche proprietäre Protokolle und Anwendungen an den ESB angeschlossen werden können. Diese Adapter bilden die Schnittstelle zum proprietären System und sorgen dafür, dass ausgetauschte Daten in Nachrichten umgewandelt werden und mittels Queues oder Topics der Prozessschicht zur Verfügung stehen.[19]

[18] Vgl. Lorenzelli-Scholz, D. (2005), S. 18
[19] Vgl. Kischel, S. (2003), S. 48

3.3.2.2 Prozessschicht

Die Prozessschicht ist eine logische Schicht, die über der Transportschicht liegt und mittels eines Distributed Processing Framework (DPF) umgesetzt wird. Das DPF kann als Vermittler und Koordinator gesehen werden, der den Datenfluss und Verarbeitungsablauf zwischen einzelnen Services steuert. Dazu nimmt das DPF die Nachrichten aus den Warteschlangen entgegen und reicht sie in einer bestimmten Reihenfolge, die zuvor mittels spezieller Tools als Prozesskette definiert wurde, an die Services weiter. Bestandteile einer solchen Prozesskette sind die beteiligten Services, die Eintrittspunkte, über die die Daten an die Services weitergereicht werden, die Austrittspunkte, über welche die verarbeitete Nachricht weiterverwendet werden kann, sowie die Verarbeitungsreihenfolge und das Ziel, an dem das Resultat der Prozesskette gespeichert werden soll. Die Auswahl der Services geschieht dynamisch zur Laufzeit.[20]

Folgende Abbildung zeigt beispielhaft den Aufbau eines ESB:

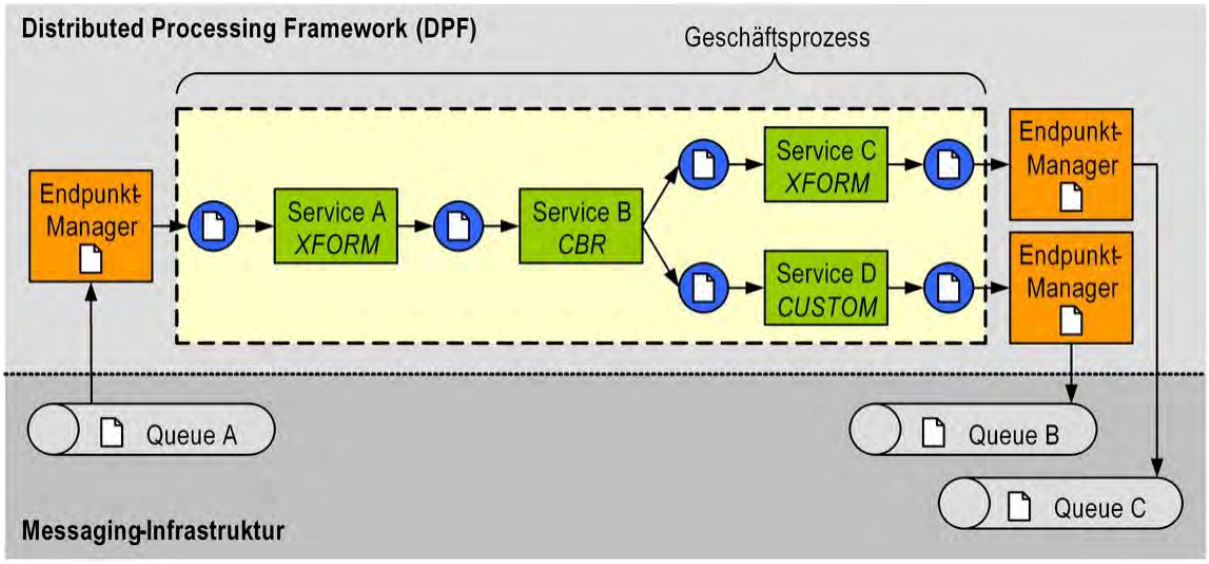


Abbildung 3: Exemplarischer Aufbau eines Bussystems im ESB[21]

Zunächst wird eine Nachricht über die Nachrichtenqueue A an einen Endpunkt-Manager geleitet, der sie dem DPF zur Verfügung stellt. Das DPF ruft nun einen Transformationsdienst auf (XFORM), welcher die Nachricht nun in ein ESB-internes Format transformiert. Anschließend werden mittels inhaltsbasierten Routings die nächsten Endpunkte bestimmt, an welche die Nachricht versendet werden sollen. Darauf folgend wird die Nachricht, in ein für den Empfänger verständliches Format, abermals transformiert. Nach diesem abschließenden Transformationsschritt wird die Nachricht an zwei Endpunktmanager gegeben, wel-

[20] Vgl. Kischel, S. (2003), S. 40
[21] Entnommen aus: Kischel. S. (2003), S. 38

che sie dann in zwei Nachrichtenqueues übergeben, von denen sie dann an den jeweiligen Endpunkt weitergereicht werden.

Da nun der Kern des Bussystems erläutert wurde, aber die Funktion der Endpunkte noch unklar ist, soll im folgenden Abschnitt auf eben diese und den Service-Container eingegangen werden.

3.3.2.3 Endpunkte und Service-Container

(Abstrakte-)Endpunkte dienen dazu, Anwendungen zu kapseln, um diese bei der Modellierung von Prozessketten nutzen zu können ohne die Implementierung der Services zu kennen. Diese Endpunkte werden in Service-Containern gehalten.[22]

Ein Service-Container wiederum wird als physische Repräsentation von Endpunkten beschrieben, welche die Implementierung der Serviceschnittstellen bereitstellt und ferngesteuert aufgerufen werden kann. Dabei ist er für die Versorgung der Services mit Daten verantwortlich und hat dafür Sorge zu tragen, dass bearbeitete Daten vom Bus wieder geholt werden können.[23]

Ein Service-Container kann nicht nur einen Service oder eine Art von Services anbieten, sondern mehrere unterschiedliche Services.[24] Dies ist in der nachstehenden Grafik zu erkennen:

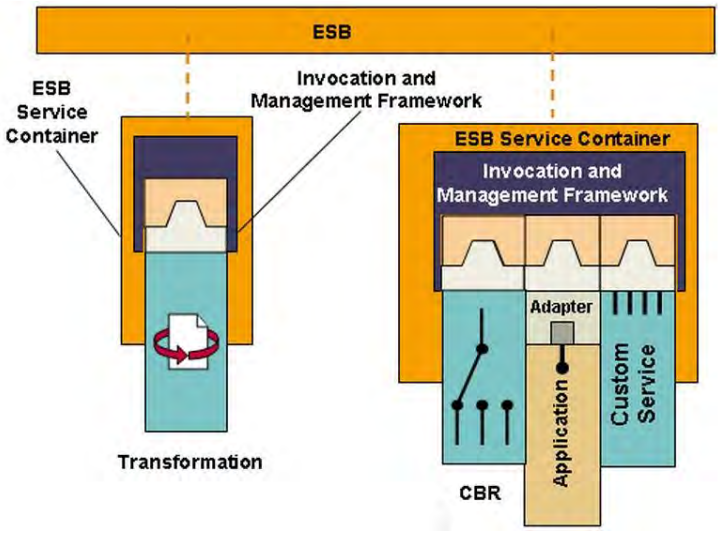


Abbildung 4: Exemplarischer Aufbau eines Service Containers[25]

[22] Vgl. Chappell, D. (2005)
[23] Vgl. ebenda
[24] Vgl. ebenda
[25] Entnommen aus: ebenda

Jeder Service besitzt mindestens einen Eingangspunkt (engl. entrypoint), über den ein Service mit Daten versorgt wird und mindestens einen Ausgangspunkt (engl. exitpoint), über welchen die verarbeiteten Daten wieder vom Bus geholt werden können. Dazwischen wird die Nachricht entsprechend vom Service manipuliert, sodass z. B. mehrere Nachrichten in den Ausgangspunkt gelegt werden können oder die Ausgangsnachricht inhaltlich der Eingangsnachricht entspricht, nur das die Zieladresse eine andere ist (CBR).[26]

Zusätzlich kann ein Service mehrere Eingangs- und Ausgangspunkte besitzen. So könnte es einen weiteren Eingangspunkt geben, um nachvollziehen zu können, welche Services die Nachricht bereits durchlaufen hat, und wie sie im Zuge dessen verarbeitet wurde.[27]

Weitere Ausgänge kann es ebenfalls für das Verfolgen einer Nachricht geben, aber auch für fehlerhafte Verarbeitungen oder zurückgewiesene Nachrichten, welche dann vom Invocationand Management Framework, ausführlich im nächsten Absatz beschrieben, weiterbehandelt werden.[28] Die untenstehende Grafik verdeutlicht dies:

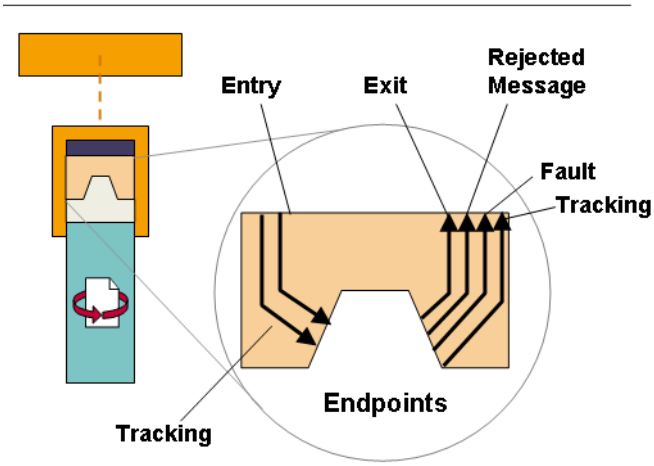


Abbildung 5: Entry- und Exitpoints im Service-Container[29]

Zusätzlich verfügt jeder Service-Container über ein Invocation and Management Framework. Dieses Framework dient dazu den Container mittels einer geeigneten Schnittstelle (hier exemplarisch Java Management Extensions (JMX)) mit Managementdaten zu versorgen sowie angeschlossenen Überwachungskonsolen die Managementdaten des Containers zur Verfügung zu stellen. Solche Daten können die Konfiguration (Starten, Stoppen oder Herun-

[26] Vgl. Chappell, D. (2005)
[27] Vgl. ebenda
[28] Vgl. ebenda
[29] Entnommen aus: ebenda

terfahren) sowie Fehlermeldungen und -behandlungen des Containers oder Benachrichtigung bei erfolgreichem Verlassen einer Nachricht aus dem Container betreffen.[30]

Des Weiteren ist das Invocation and Management Framework, ebenfalls mittels JMX, mit dem Dienstverzeichnis verbunden, aus dem es benötigte Daten bezieht und hält zusätzlich noch eine lokale Kopie des Dienstverzeichnisses, um im Fehlerfall ohne das eigentliche Dienstverzeichnis weiter arbeiten zu können.[31] Unten stehende Grafik veranschaulicht den Einsatz des Invocation and Management Frameworks:

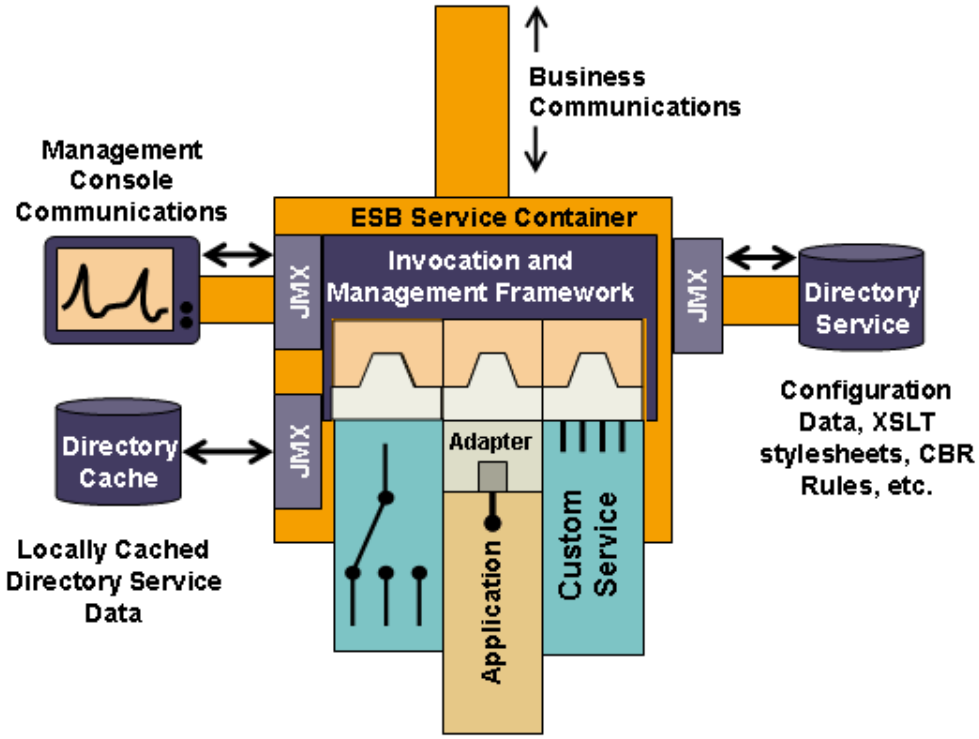


Abbildung 6: Veranschaulichung des Dienstverzeichnisses[32]

3.3.3 Funktionen

In diesem Abschnitt werden die elementaren Funktionen eines ESB beleuchtet. Dies ist insofern von Relevanz, da sie die Grundlage der Anforderung des Kooperationspartners bilden und im Verlauf der Arbeit wiederholt als Referenzgröße herangezogen werden.

Da die Funktionalitäten eines ESB sehr vielfältig sind, werden sie im Folgenden in Kernfunktionalitäten und weitere Funktionalitäten unterschieden. Als Kernfunktionen lassen sich nennen:[33]

[30] Vgl. Chappell, D. (2005)
[31] Vgl. ebenda
[32] Entnommen aus: ebenda
[33] Vgl. Younes, Y. (2010), S. 7

- Nachrichtenaustausch / Herstellung der Konnektivität
- Intelligentes Routing
- Nachrichtentransformation
- Protokolladaptierung
- Skalierbarkeit
- Monitoring
- Umsetzung von Web-Service Standards

Die Hauptaufgabe eines ESB besteht darin, die **Konnektivität** der SOA-Teilnehmer herzustellen und einen **Nachrichtenaustausch** bzw. eine Kommunikation zwischen Ihnen zu ermöglichen.[34] Ein wichtiger Aspekt diesbezüglich ist die sichere Nachrichtenübertragung. Diese wird durch Verwendung einer Message Oriented Middleware (MOM) (siehe 3.3.2.1) gewährleistet. Mit ihrer Hilfe ist es außerdem möglich, Nachrichten nicht nur synchron zu übertragen sondern auch mittels Persistierung asynchron. Als Kommunikationsmuster werden unter anderem Request-Reply und Publish-Subscribe unterstützt.[35]

Das interne Nachrichtenformat ist in der Regel XML oder ein Derivat davon. Dies bringt einige Vorteile mit sich, die im Rahmen der Integrationsaufgaben von hoher Bedeutung sind. Zum einen ist XML sehr verbreitet und plattformunabhängig, was einen Nachrichtenaustausch sehr erleichtert. Ein weiterer Grund liegt in der Erweiterbarkeit und Validierbarkeit von XML begründet. Dies sind wichtige Attribute, da damit Routing, Transformation, Nachrichtenaggregation sowie Nachrichtenanreicherung erleichtert werden.[36]

Die zweite Kernfunktionalität stellt das **intelligente Routing** dar. Durch das Routing wird mittels festdefinierter Kriterien und Ansätze bestimmt, an welchen Endpunkt eine Nachricht gesendet wird.[37]

Als wichtigste Ansätze nennt Chappell das inhaltsbasierte Routing (CBR).[38] Bei diesem Prinzip wird eine Nachricht anhand ihres Inhaltes an einen bestimmten Endpunkt gesendet. Als einen weiteren Ansatz nennt Chappell das Routing auf Reisewegen, bei dem eine Nachricht an eine festdefinierte Anzahl von Empfängern in einer fest vorgegebenen Reihenfolge geschickt wird.[39] Vollmer und Gilpin verweisen zusätzlich noch auf das typbasierte Routing, bei

[34] Vgl. Zeppenfeld, K. / Finger, P. (2009), S. 27
[35] Vgl. Davis, J. (2009), S. 57
[36] Vgl. ebenda, S. 258
[37] Vgl. Scheiner, D. (2010), S. 7
[38] Vgl. Chappell, D. (2004), S. 129 ff.
[39] Vgl. ebenda, S. 129 ff.

dem eine Nachricht anhand ihres Nachrichtentyps an einen bestimmten Endpunkt gesendet wird.[40]

Ebenfalls eine elementare Funktion stellt die **Nachrichtentransformation** dar. In diesem Zusammenhang ist zu verstehen, dass Nachrichten sowohl syntaktisch, sprich das Nachrichtenformat oder enthaltene Datentypen betreffend, und semantisch, den Nachrichteninhalt betreffend, transformiert werden können. So ist es möglich, eine CSV-Datei in eine XML-Datei zu transformieren (syntaktische Transformation) oder bei der Repräsentation des Wohnortes aus dem Kürzel *Str.* für Straße, dieses Kürzel auszuschreiben (semantisch).[41]

Unter dem Punkt der **Protokolladaptierung** ist zu verstehen, eine Kommunikation zwischen Anwendungen über Protokollgrenzen hinweg zu ermöglichen. Konkret ist es mit der Hilfe eines ESBs möglich, eine Applikation, die mittels FTP kommuniziert, mit einer anderen Applikation, deren Kommunikationsbasis HTTP bildet, interagieren zu lassen.[42]

Des Weiteren bietet ein ESB Mechanismen, die es ihm ermöglichen die Verarbeitungslast zu verteilen und somit trotz steigender Last, ohne größere Performanzverluste weiterzuarbeiten und **skalierbar** zu bleiben.[43]

Eine ebenfalls enorm wichtige Funktionalität wird durch das **Monitoring** bereitgestellt. Durch sie ist es dem Anwender möglich, den Nachrichtenfluss einzelner Services zu überwachen und zu protokollieren. So können gewisse Kenngrößen, wie die Zahl versendeter und erhaltener Nachrichten innerhalb einer gewissen Zeitspanne, ermittelt und Fehler schneller lokalisiert werden.[44]

Als weitere wichtige Funktionalitäten können noch genannt werden:

- **Sicherstellen der Dienstgüte (QoS):** Hierunter fallen Mechanismen, die gewährleisten, dass Nachrichten sicher übermittelt werden. So zum Beispiel das Persistieren von Nachrichten in einer MOM sowie das wiederholte Aufrufen eines Services im Fehlerfall oder das Ausweichen auf einen anderen Service, um eine Verletzung der SLAs zu vermeiden.[45]

[40] Vgl. Hiekel, S. (2007), S. 27
[41] Vgl. ebenda
[42] Vgl. Dostal, W. u. a. (2005), S. 20
[43] Vgl. Kischel, S. (2003), S. 37
[44] Vgl. Davis, J. (2009), S. 262 f.
[45] Vgl. ebenda, S. 28

- **Aggregieren von Services:** Darunter ist zu verstehen, dass ein ESB die Möglichkeit bietet, durch Kombination einzelner Services eine größere Gesamtanwendung zu aggregieren.[46]

- **Serviceorchestrierung:** Ein ESB ist in der Lage die Serviceaufrufe mittels Prozessketten zu orchestrieren, das heißt den Ablauf der Serviceaufrufe zu steuern.[47]

- **Sicherheit:** Bezüglich Sicherheit stellt ein ESB Funktionen zur Verfügung, die es durch Authentifizierung und Autorisierung erlauben, Zugriffe auf den ESB und einzelne Services zu beschränken oder ganz zu untersagen.[48] Ein weiteres Mittel ist die Verschlüsselung des Übertragungskanals (HTTPS, SSL).[49]

- **Serviceverzeichnis:** Ein ESB kann ebenfalls die Funktion eines Serviceverzeichnisses übernehmen.[50]

- **Timer:** Zusätzlich bietet ein ESB noch die Möglichkeit, Prozessketten oder einzelne Verarbeitungsschritte in einer solchen Kette zeitgesteuert auszuführen, um Arbeitsvorgänge zu automatisieren.[51]

[46] Vgl. Davis, J. (2009), S. 27
[47] Vgl. ebenda, S. 27
[48] Vgl. Chappell, D, (2004), S. 12
[49] Vgl. Hiekel, S. (2006), S. 39
[50] Vgl. ebenda, S. 28
[51] Vgl. Davis, J. (2009), S. 261

4 Durchführung der Marktanalyse

Die in der vorliegenden Arbeit dargestellte Forschung stellt den Hauptteil des Projektes KOS dar. Es findet eine Marktanalyse über einen Vertreter der kommerziellen ESB-Systeme und über drei Stellvertreter der Open Source Landschaft statt. Zur Zielerreichung des Projektes werden die technischen Voraussetzungen und Anforderungen an den ESB des Kooperationspartners als Kunden untersucht. Für die anschließende Bewertung, welches System als Empfehlung für den Kunden ausgesprochen werden kann, sind Grundlagen als Basisverständnis voraussetzend. Diese Basis wird mit Hilfe des Folgenden Kapitels geschaffen.

4.1 IBM WebSphere ESB

WebSphere ESB ist die SOA-basierte Integrationsplattform der IBM. Die neueste Version 7.5.1 der kommerziellen Enterprise Service Bus Lösung wurde 2011 publiziert. WebSphere ESB stellt eine dynamische Konnektivitätsinfrastruktur für die Integration von Anwendungen und Services zur Verfügung und bietet standardbasierte ESB-Funktionalitäten für die unternehmensweite IT-Umgebung.[52]

4.1.1 Architekturübersicht

WebSphere ESB kann abhängig von den Anforderungen in verschiedenen Bus-Topologien installiert und konfiguriert werden. Es ist sowohl möglich alle Komponenten auf einem Einzelserver (engl. stand-alone configuration) als auch auf mehreren verteilten Systemen (engl. network deployment configuration) einzurichten. Um eine möglichst hochverfügbare und ausfallsichere IT-Umgebung zu schaffen, kann WebSphere ESB ebenfalls in einer Cluster-Umgebung installiert werden. Der Cluster-Mechanismus wird von der grundlegenden Laufzeitumgebung des WebSphere ESB - dem IBM WebSphere Application Server (WAS) – zur Verfügung gestellt.

WAS ist ein Software-Produkt, welches die Rolle eines Webanwendungsservers in der IT-Architektur eines Unternehmens einnimmt und als Middleware für diverse andere IBM Software-Produkte fungiert. Aufgrund der technischen Integration im WAS, profitiert WebSphere ESB maßgeblich vom Transaktions- und Sicherheitskonzept, der Hochverfügbarkeit des Anwendungsservers, sowie einer reibungslosen Interoperabilität mit anderen Produkten aus dem WebSphere-Portfolio. Beispiele hierfür sind der IBM WebSphere MQ oder der IBM WebSphere Message Broker.

[52] Vgl. IBM WebSphere ESB (2011), S. 1

Die SOA-Funktionalität ist im WebSphere ESB durch den sogenannten service-oriented architecture core (SOA core) realisiert. Der SOA core ermöglicht die Entwicklung und den Einsatz von standardbasierten Prozess-Integrationslösungen in einer SOA durch die Bereitstellung notwendiger Serviceschnittstellen auf Geschäftsebene. Darüber hinaus implementiert der SOA core zwei weitere Servicekomponenten:

- die Service Component Architecture (SCA) und
- die Common Event Infrastructure (CEI).[53]

Die SCA stellt eine Architektur dar, in welcher alle Geschäftsprozesselemente, wie etwa Service-Assets des betrieblichen Informationssystems, Schnittstellen zu Webservices, Betriebsregeln, Arbeitsabläufe, Datenbankverwaltungssysteme, etc., service-orientiert dargestellt sind. Des Weiteren bietet die Service Component Architecture sowohl ein einheitliches Programmiermodell für den Aufruf als auch für die Darstellung von Daten, welche für die Ausführung von Anwendungen in einer Java EE-basierten Softwarearchitektur verwendet werden.[54]

Die Common Event Infrastructure ist eine in WebSphere ESB integrierte Technologie, die grundlegende Monitoring- und Verwaltungsfunktionalitäten für laufende Anwendungen beziehungsweise Services zur Verfügung stellt. Unter anderem ermöglicht die CEI das Logging und die Ablaufverfolgung, sowie die Konsolidierung, Persistenz und Verteilung einzelner Ereignisse und Prozesse.[55]

Eine weitere wichtige Komponente der WebSphere ESB Architektur bildet das IBM WebSphere Service Registry and Repository (WSRR). Das WSRR wird verwendet, um Informationen über Service-Endpunkte zu speichern und diese während der Mediation abfragen und nutzen zu können.[56]

Die folgende Abbildung zeigt die Architektur des IBM WebSphere ESB und seiner wichtigsten Komponenten:

[53] Vgl. IBM WebSphere ESB (2011), S. 3
[54] Vgl. ebenda, S. 3 f.
[55] Vgl. ebenda, S. 7
[56] Vgl. ebenda, S. 15

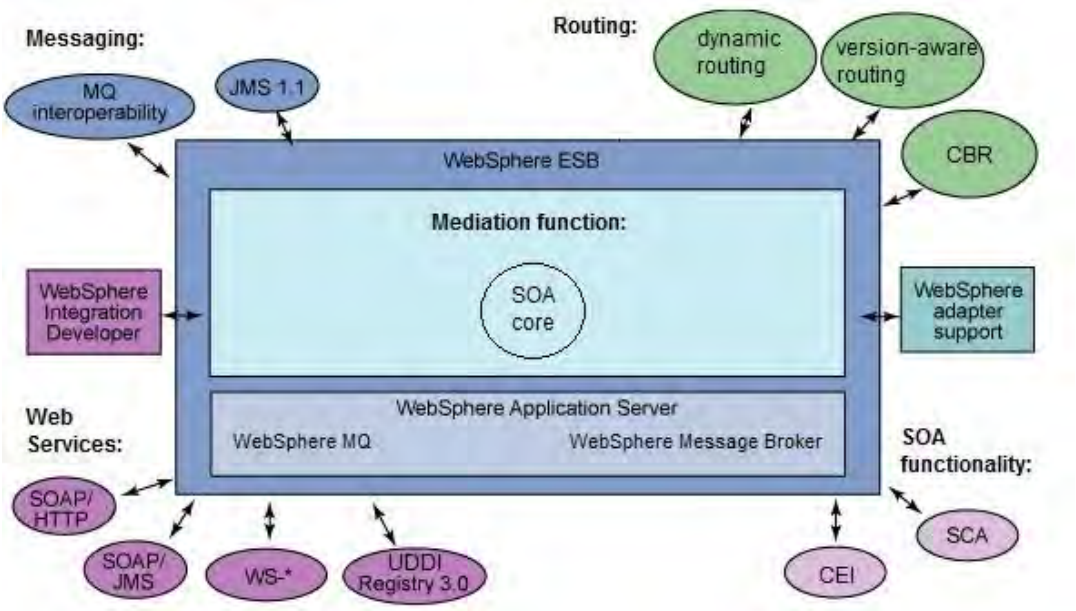


Abbildung 7: Architekturübersicht des IBM WebSphere ESBs[57]

4.1.2 Funktionale Features

WebSphere ESB ermöglicht eine fortgeschrittene Interaktion zwischen Service-Endpunkten über eine Vielzahl von Webservice-Standards, Protokollen, Programmiersprachen und Anwendungsprogrammierschnittstellen, wie zum Beispiel:[58]

- SOAP/HTTP, HTTP/JMS, WSDL 1.1
- Java Message Service (JMS) 1.1. wird vom WAS zur Verfügung gestellt, auf dem WebSphere ESB aufbaut. Anwendungen können eine Vielzahl an Transportprotokollen nutzen, einschließlich TCP/IP, SSL, HTTP und HTTPS.
- UDDI 3.0 Service Registry, Web Services Gateway
- WS-Security, WS-Atomic Transactions
- Business Process Execution Language (BPEL) zur Realisierung von Workflows

Das Tool unterstützt diverse Arten zum Nachrichtenaustausch zwischen Dienstnutzern und Dienstanbietern. Dazu zählt unter anderem das dynamische Routing von Nachrichten. Dabei erfolgt der Nachrichtenfluss zwischen zwei oder mehr Service-Endpunkten dynamisch, während die Auswahl der eigentlichen Service-Endpunkte sowohl dynamisch, als auch anhand vordefinierter Komponenten in der SCA erfolgen kann. Des Weiteren besteht die Möglichkeit des inhaltsbasierten- sowie versionsgerechten Routings.[59] Unabhängig vom gewählten Rou-

[57] Mit Änderungen entnommen aus: IBM developerWorks (2012)
[58] Vgl. IBM WebSphere ESB (2011), S. 1 f.
[59] Vgl. ebenda, S. 245 ff.

tingverfahren wird zur Auswahl der Service-Endpunkte und ihrer Informationen entweder ein vordefiniertes Registry, wie das WSRR oder eine Datenbank eingesetzt.[60]

Das Kommunikationsmuster im WebSphere ESB hängt maßgeblich vom verwendeten Service Integration Bus ab, welcher für die synchrone beziehungsweise asynchrone Kommunikation zwischen Integrationsdiensten einer Anwendungslandschaft zuständig ist. Durch den Einsatz sogenannter bindings ist es sowohl möglich ein Point-to-Point als auch ein Publish-Subscribe Nachrichtenaustauschverfahren in WebSphere ESB einzubinden. Während das MQ binding lediglich das Point-to-Point Messaging unterstützt, bietet das MQ JMS binding beide genannten Kommunikationsmuster an.[61]

Der Einsatz von bindings ist auch bei der Protokollumwandlung erforderlich. Um zwei Service-Endpunkte miteinander zu verbinden, die unterschiedliche Protokolle verwenden, muss zunächst ein Mediationsmodul (engl. mediation module) geschaffen werden.[62] Mediationsmodule sind Komponenten der SCA, die der Umwandlung des Formats, des Inhalts und des Ziels von Nachrichten zwischen dem Dienstanbieter und Dienstnutzer dienen.[63] Zur Schaffung eines solchen Moduls sind die entsprechenden bindings erforderlich. Soll zum Beispiel ein Web Service, der SOAP/HTTP verwendet, mit einem Java Message Service (JMS) verbunden werden, ist sowohl das SOAP/HTTP binding als auch das JMS binding für die Protokollumwandlung notwendig.

Beide bindings müssen im Mediationsmodul als Import beziehungsweise Export implementiert werden, wie es in der nachfolgenden Abbildung zu sehen ist:

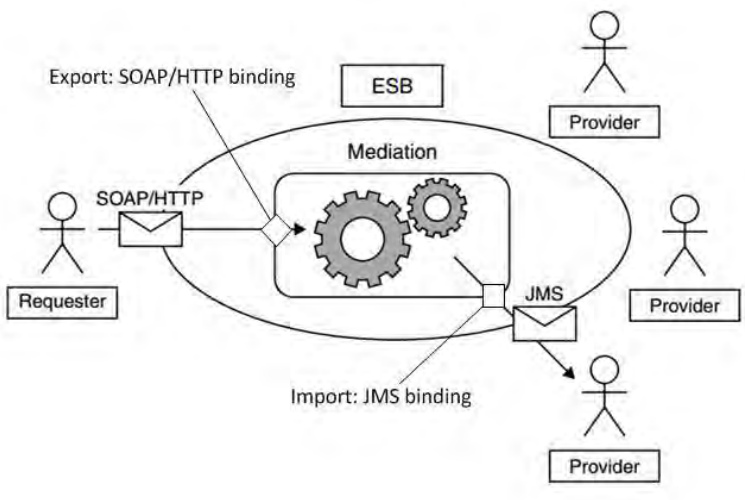


Abbildung 8: Funktionsweise des Mediationsmodul im WebSphere ESB[64]

[60] Vgl. IBM WebSphere ESB (2011), S. 176
[61] Vgl. ebenda, S. 197
[62] Vgl. ebenda, S. 259 ff.
[63] Vgl. IBM WebSphere ESB (2011), S. 11 ff.
[64] mit Änderungen entnommen aus: ebenda, S. 261

WebSphere ESB unterstützt die Verbindung zwischen den Service-Endpunkten durch eine Vielzahl von standardmäßig integrierten Protokollen und Schnittstellen zur Anwendungsprogrammierung, wie zum Beispiel:

- Webservices SOAP 1.1 und 1.2
- WebSphere MQ
- Enterprise Java Beans (EJB)
- Enterprise Information System (EIS) Integration durch die Verwendung von WebSphere- bzw. Business Integration-Adapter:
    - IBM WebSphere-Adapter für E-Mail, FTP, JDBC, SAP Software, u. a.
    - Business Integration-Adapter[65]

Die aufgeführten IBM WebSphere-Adapter sind, anders als Business Integration-Adapter, kompatibel mit der Java EE Connector Architecture (JCA) 1.5.[66] Es ist jedoch zu beachten, dass die IBM WebSphere-Adapter für Standard-Software, beispielsweise der IBM WebSphere Adapter für Oracle E-Business Suite oder der IBM WebSphere Adapter für SAP Software, nicht standardmäßig im WebSphere ESB integriert sind, sondern bei Bedarf explizit gekauft werden müssen. Der Preis für diese Adapter beträgt jeweils 170.458,- Euro.[67] Bei Business Integration-Adaptern handelt es sich um eine Sammlung von Tools und APIs, die den Datenaustausch zwischen Anwendungen über eine zusätzliche Transportschnittstelle wie JMS ermöglichen.[68] Die zusätzliche Komponente WebSphere-Adapter Toolkit bietet darüber hinaus die Möglichkeit eigene JCA-kompatible Adapter für den IBM WebSphere ESB zu entwickeln. Die Komponente ist im Standardfunktionsumfang nicht enthalten, wird jedoch auf der IBM Webseite zum Download angeboten.[69]

Wie bereits angesprochen profitiert WebSphere ESB von der technischen Konnektivität mit dem WebSphere Application Server, ist aber zugleich von diesem abhängig, da das Tool lediglich mit diesem Anwendungsserver kompatibel ist. Unter anderem stellt der WAS die Webservice Mechanismen WS-Security und WS-Atomic Transactions zur Verfügung. Der WS-Security Mechanismus kann verwendet werden, um eine Vielzahl von Sicherheitserweiterungsmodulen und Verschlüsselungstechniken im SOAP-Nachrichtenaustauschverfahren unterzubringen. Dazu zählen unter anderem die Nachrichtenintegrität, -vertraulichkeit und -authentifizierung sowie die Verschlüsselung des Transfers und Inhalts von SOAP-Nachrichten.[70] Darüber hinaus verfügt WebSphere ESB über ein fortgeschrittenes Sicherheitskonzept auf Anwendungsebene. Zu den wichtigsten Sicherheitsverfahren zählen die

[65] Vgl. IBM WebSphere ESB (2011), S. 23
[66] Vgl. ebenda, S. 547 f.
[67] Vgl. IBM Software Prices (2013)
[68] Vgl. IBM WebSphere ESB (2011), S. 547
[69] Vgl. IBM WebSphere Adapter Toolkit (2012)
[70] Vgl. IBM WebSphere ESB (2011), S. 76 f.

Authentifizierung, Autorisierung, Zugriffskontrolle, Datenintegrität sowie ein Rollenkonzept zur Vergabe administrativer Rechte (Lese- und Schreibrechte) an andere User im System. Der WS-Atomic Transactions Mechanismus dient im WAS und somit auch im WebSphere ESB, zur Koordinierung globaler Transaktionen.[71]

4.1.3 Nicht-Funktionale Features

Zur Entwicklung der – für die Funktionalität von WebSphere ESB fundamentalen - Mediationsmodule wird der WebSphere Integration Developer (WID) verwendet. Der von WebSphere ESB zur Verfügung gestellte visuelle Editor für Eclipse ermöglicht die Erstellung von Service-Endpunkten, Knoten und Mediationsabläufen mit Hilfe des Drag-And-Drop-Verfahrens und bietet dadurch eine für den User komfortable Möglichkeit zur Entwicklung von Workflows. Zahlreiche Knotenarten und Operationen, wie zum Beispiel verschiedene Datentransformatoren, Routingverfahren, Protokollumwandlungen, Datenbank Endpoint-Lookup, etc. sind bereits standardmäßig integriert. Der WID Editor verfügt außerdem über eine komplette Unit-Testumgebung und ermöglicht somit die einfache Durchführung von Modultests und das Debugging erstellter Mediationsmodule.[72]

Für das Produkt bietet IBM sehr umfangreiche Online-Dokumentationen, FAQ‘s, sowie Onlinevideos zur Installation, Konfiguration, Bedienung und Finanzierung des Tools. Alle Interessenten und Kunden können sich im Forum von IBM anmelden und auf die zahlreichen Einträge der durchaus aktiven Community zugreifen.

Um den Finanzierungsaspekt in die Betrachtung ebenfalls einzubeziehen, werden an dieser Stelle noch drei Preise aus dem IBM Preiskatalog zu WebSphere ESB genannt. Eine Lizenz für den IBM WebSphere Enterprise Service Bus für das zEnterprise-System ohne WSRR kostet 373,- Euro. Dieselbe Lizenz würde mit der IBM WebSphere Service Registry und Repository 623,- Euro kosten. Beim Erwerb einer Lizenz wird von IBM technischer und vertraglicher Support für die ersten 12 Monate garantiert.[73]

Mit WebSphere ESB erhält der Kunde eine funktionell sehr umfangreiche Enterprise Service Bus Lösung. Von Routing, Transformation und Protokollumwandlung, über Sicherheit und Skalierbarkeit, bis hin zu einer Vielzahl unterstützter Standards und dem direkten sowie professionellen Support des Herstellers - IBM überzeugt mit seinem qualitativ hochwertigen und allumfassenden Produkt.

[71] Vgl. IBM WebSphere ESB (2011), S. 188 f.
[72] Vgl. IBM WebSphere Integration Developer (2009), S. 7 ff.
[73] Vgl. IBM Software Prices (2013)

4.2 Open Source ESB-Lösungen

Nachdem das kommerzielle ESB-Produkt von IBM betrachtet wurde, folgt in diesem Kapitel die Vorstellung der Open Source ESB-Lösungen von JBoss, Mule und Apache.

4.2.1 JBoss ESB

Das Enterprise Service Bus System JBoss ESB wurde von der Firma JBoss entwickelt und 2006 von RedHat übernommen. RedHat's Ziel war es mit JBoss stärker in der SOA-Markt vorzudringen, weshalb bereits im Juni 2006 JBoss ESB (damals unter dem Namen Rosetta) von JBoss gekauft und in deren Produktpalette eingegliedert wurde. Rosetta war zu dem Zeitpunkt schon drei Jahre bei Kanadas zweitgrößter Versicherung erfolgreich im Einsatz.[74] Seit April 2007 bietet JBoss eine Community- und eine Enterprise-Version seiner ESB-Lösung an. Auf die wichtigsten Unterschiede dieser beiden Versionen wird in folgender Architekturübersicht und Featureanalyse eingegangen.

4.2.1.1 Architekturübersicht

Der JBoss Enterprise Service Bus (JBoss ESB) ist ein Teil der JBoss Enterprise SOA Platform, einer Open Source SOA-Software, welche auf Java EE basiert. Die SOA-Plattform ist ein Teilprodukt des JBoss Enterprise Middleware Portfolios zudem unter anderem auch der JBoss Web Server oder die JBoss Enterprise Application Platform gehören.[75]

Abbildung 9 zeigt die Kerntechnologien der JBoss Enterprise SOA Platform. Die Kernkomponente bildet der Enterprise Service Bus. Er bietet die fundamentalen Features zur Nachrichtenübertragung, Dienstverwaltung und -implementierung sowie Integration mit entfernten Subsystemen und externen Drittsystemen. Die inhaltsbasierte Weiterleitung, die das Routing einer Nachricht anhand ihres Inhalts ermöglicht, kann per Drools, einen implementierten Regelinterpreter oder per XPath erfolgen. Als leichtgewichtiger Prozessinterpreter für die Orchestrierung der Dienste wird jBPM verwendet. Für die Transformation von Nachrichten kommt Smooks zum Einsatz, ein fragmentbasiertes Dateitransformations- und Analysewerkzeug der Gruppe Codehaus, das sowohl XML als auch nicht-XML Nachrichten unterstützt und ebenfalls mit großen Datenmengen umgehen kann. Als Bus beziehungsweise Queue zum Transport von Nachrichten wird JBoss Messaging eingesetzt. JBoss Messaging stellt weitere Sicherheitsimplementierung wie den SSL Transport und die JAAS Security Integration bereit. Für die Integration SOAP-basierter Webdienste werden die beiden externen Pro-

[74] Vgl. JBoss ESB Background (2012)
[75] Vgl. JBoss Enterprise Middleware (2012)

dukte Wise von Java Linux Labs und SoapUI von eviware verwendet. Um die einzelnen Dienste im Sinne einer SOA-Governance verwalten zu können, werden alle Dienstinformationen in einem UDDI-Repository abgelegt. Diese Service Registry Komponente von JBoss ESB unterstützt standardmäßig eine JAXR-Implementierung, die mit einer jUDDI Registry verbunden ist. Es ist ebenfalls möglich eine eigene Java Naming and Directory Interface (JNDI) Registration in Verbindung mit dem Remote Methode Invocation (RMI) Service einzubinden.[76]

Zur Entwicklung und zum Test von Webanwendungen und SOA-Services kann das JBoss Developer Studio - eine Eclipse-Erweiterung - eingesetzt werden. Gerade in einer IT-Umgebung mit vielen kooperierenden Subsystemen ist es wichtig, den Status und die Antwortzeiten der installierten Anwendungen operativ überwachen zu können. Hierfür stellt JBoss das Monitoring- und Managementwerkzeug Operations Network bereit, welches es ermöglicht, Kennzahlen der Applikationsserver und des Betriebssystems, aber auch des ESBs, zu erheben. Bei erkannten Fehlern und Problemen kann ein Alarm ausgelöst werden.[77]

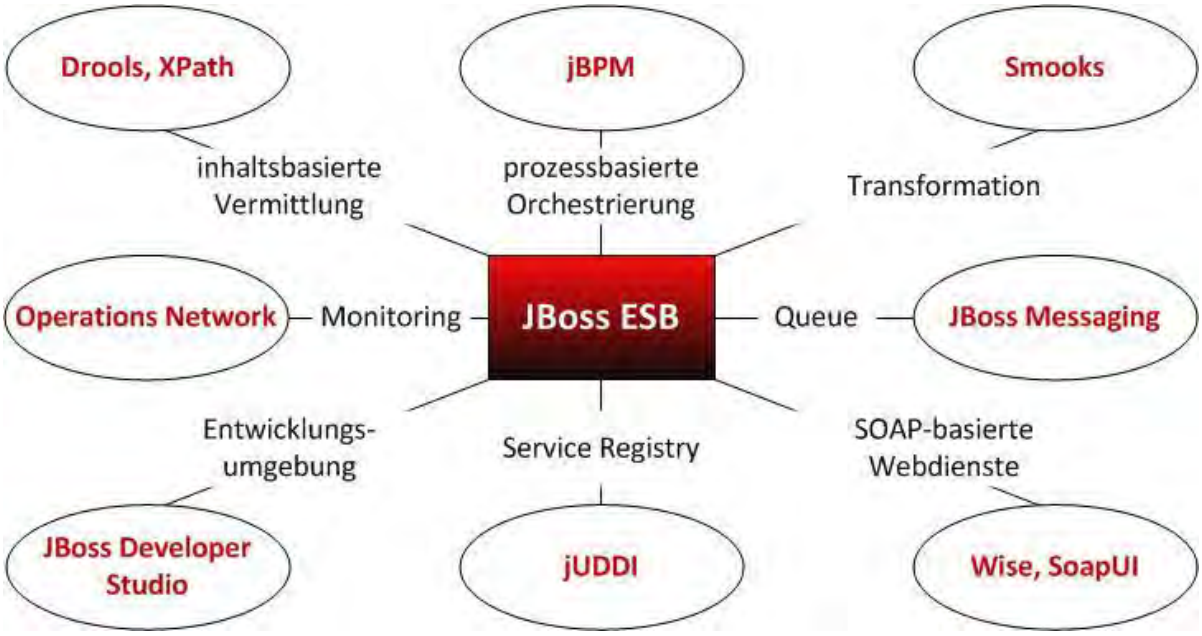


Abbildung 9: Architekturkomponenten der JBoss Enterprise SOA Plattform

Zur Installation bietet JBoss das ESB-Server-Paket zum Download an, welches den JBoss Applikationsserver (JBossAS) mit darin installiertem ESB beinhaltet. Es ist ebenso möglich einen bereits existierenden JBossAS um die ESB-Funktionalität zu erweitern. Dies hat den

[76] Vgl. JBoss Registry Guide (2010)
[77] Vgl. Fink, T. (2009), S. 10 ff.

Vorteil, dass bei bereits vorhandener JBoss Infrastruktur lediglich das Tool beschafft werden muss. Zu diesem Zweck kann der ESB als Quelltext- oder Binärversion separat heruntergeladen werden. Der JBossAS stellt die grundsätzliche Laufzeitumgebung für den JBoss ESB bereit. Die Installation von JBoss ESB erfolgt out-of-the-box, das bedeutet, dass JBoss ESB keiner Installationsroutine im engeren Sinne folgt, sondern durch einen Batchvorgang gestartet werden kann.[78] Die Vorgehensweise bei der Installation wird Schritt für Schritt in einem Getting Started Guide[79] erklärt.

Dank der Unterstützung einer Loose-Coupling-Architektur kann davon ausgegangen werden, dass die JBoss Enterprise Middleware stark skalierbar ist.[80] Dies nicht nur über mehrere Applikationen an einem Bus, sondern über mehrere Busse, ausgelegt für B2B-Cases. Damit ist die Architektur sehr gut geeignet für Dienste mit hohen Zuverlässigkeitsanforderungen. Ein weiterer Vorteil der JBoss Middleware besteht in dem einfachen Aufsetzen eines ESB-Clusters, um so eine höhere Ausfallsicherheit und eine optimierte Verteilung der Ressourcen zu erzielen.[81]

4.2.1.2 Funktionale Features

Wie aus der Architekturübersicht zu erkennen ist, weist der JBoss ESB alle wichtigen ESB-Funktionalitäten auf. Darüber hinaus unterstützt das Produkt folgende Standards, Protokolle und Programmierschnittstellen:

- Webservices, inkl. SOAP 1.1 und 1.2
- WSDL, BPEL
- JEE, inkl. JMS, JTA und EJB3
- HTTP(S), FTP
- TCP/IP
- UDDI
- u. a.[82]

Außerdem implementiert JBoss ESB eine Vielzahl weiterer Webservice-Spezifikationen wie WS-Atomic Transactions, WS-Business Activity, WS-Coordination, WS-Eventing, WS-Policy und WS-Security, wobei die ersten drei genannten Spezifikationen nicht standardgemäß implementiert sind, jedoch durch die Integration des JBoss Transaction Projects zur Verfü-

[78] Vgl. Diller, J., Viola, J. (2011), S. 48
[79] Vgl. JBoss Getting Started Guide (2012)
[80] Vgl. Thiel, T. (2007), S. 9 ff.
[81] Vgl. Fink, T. (2009), S. 12
[82] Vgl. JBoss Standards (2012)

gung gestellt werden können. Des Weiteren ist es grundsätzlich möglich weitere Adapter für Standard-Software selbst zu entwickeln.[83]

Wie bereits erwähnt, wird für die Orchestrierung von Diensten mittels Prozessen das eingebettete Prozessverwaltungswerkzeug JBoss jBPM verwendet. Mit Hilfe des Prozessinterpreters lassen sich Abläufe und die dazugehörigen Regeln im Unternehmen grafisch modellieren und deren Steuerung automatisieren. Die zweite Alternative zur Einbindung von Prozessen ist der Einsatz eines BPEL-Interpreters. Der JBoss ESB liefert allerdings keinen eigenen mit, weshalb zunächst Konfigurationsaufwand betrieben werden muss, um ein BPEL-System mit JBoss ESB verbinden zu können.[84]

Beim inhaltsbasierten Routing mit Hilfe des standardmäßig integriertem Regelinterpreters Drools (in der kommerziellen Enterprise-Version JBoss Rules genannt), wird eine eingehende Nachricht durch selbstdefinierte Regeln ausgewertet und dann an den entsprechenden Dienst weitergeleitet. Neben CBR ermöglicht JBoss ESB auch das statische Routing von Nachrichten zwischen den Diensten einer Anwendungslandschaft.[85] Die Kommunikation kann dabei sowohl synchron als auch asynchron erfolgen, wobei letztere die vom JBoss empfohlene Variante ist. Im Programmers Guide[86] weist der Hersteller jedoch darauf hin, dass die Verwendung des asynchronen Nachrichtenaustauschverfahrens die Fehlerfindung bei der Entwicklung von Anwendungen erschweren kann.

Die ebenfalls vorgestellte Komponente Smooks, welche von JBoss ESB zur Transformation von Nachrichten eingesetzt wird, unterstützt eine breite Reihe von Dateitypen, wie zum Beispiel XML, CSV, EDI, etc. Der momentan noch eingesetzte SmooksTransformer wird jedoch in naher Zukunft durch die leistungsstärkere SmooksAction-Klasse ersetzt und als *deprecated* gekennzeichnet. Neben den Funktionalitäten von SmooksTransformer, unterstützt SmooksAction vor allem eine Vielzahl interner Java-Transformationen, wie beispielsweise Strings, Byte Arrays, InputStreams, Readers, etc. und erleichtert somit die Typtransformation innerhalb von Java.[87]

Der Transaktionsschutz wird beim JBoss ESB durch den eingesetzt Bus, dem Medium zum Transport von Nachrichten, gewährleistet. Dieser ist im JBoss ESB durch eine Queue, gewöhnlich eine JMS-Queue, wie zum Beispiel JBoss Messaging, Oracle AQ oder MQ Series, realisiert.[88] Standardmäßig integriert ist JBoss Messaging, die hohen Wert auf Performance, Verlässlichkeit und Skalierbarkeit sowie hohen Durchsatz und schnelle Antwortzeiten legt. Es

[83] Vgl. JBoss Programmers Guide (2012)
[84] Vgl. ebenda
[85] Vgl. Fink, T. (2009), S. 11 ff.
[86] Vgl. JBoss Programmers Guide (2012)
[87] Vgl. ebenda
[88] Vgl. ebenda

handelt sich um eine JMS 1.1 kompatible und Sun zertifizierte Implementierung. Durch die Unterstützung von JMS wird sowohl das Point-to-Point als auch das Publish-Subscribe Kommunikationsmuster im JBoss ESB zur Verfügung gestellt. Dank der in JBoss Messaging implementierten QoS-Merkmale unterstützt das Tool auch die Verschlüsselung und benutzerspezifische Authentifizierung sowie Rollenverteilungen für Benutzer.[89]

4.2.1.3 Nicht-Funktionale Features

Als Dokumentation zu JBoss ESB stehen neben Online-Ressourcen auch zahlreiche PDF-Dokumente zur Verfügung[90], welche unterteilt nach Versionen und Themenschwerpunkten leserorientiert sehr ausführlich verschiedene Konfigurationen beschreiben. Neben der schriftlichen Dokumentation können sich Käufer der Enterprise-Version auf den 24/7 Hersteller-Support verlassen. Dieser umfasst Fehlerkorrekturen, Updates und Behebung von Sicherheitslücken über mehrere Jahre hinweg. Des Weiteren hat die kommerzielle Version den Vorteil, dass sich Kunden auf ein zertifiziertes und qualitätsgesichertes Produkt verlassen können, das weniger fehleranfällig, stabiler und für die Produktion zuverlässiger ist.

Eine Hochrechnung des Herstellers selbst, sieht die Kosten im ersten Jahr für einen Server mit 16 Cores bei $29.000,-.[91] User der Community-Version, deren Quellcode frei zugänglich unter der GNU Lesser General Public License (LGPL) steht, profitieren lediglich vom Support der Community in Form von eingerichteten Wikis, FAQ's und Foren.[92] Im Gegensatz zur Enterprise-, ist die Community-Version - aufgrund der öfter publizierten Releases - dynamischer[93] und implementiert häufig sehr schnell auch die neusten IT-Standards.

Zur komfortablen Bedienbarkeit des Tools trägt lediglich das JBoss Developer Studio bei. Dieses dient jedoch ausschließlich zur Entwicklung und dem Test der Anwendungen und Services. JBoss ESB selbst verfügt über keine Benutzeroberfläche im engen Sinne und lässt sich nur über die Kommandozeile starten. Nach dem Start des Application Servers kann der Benutzer über eine Weboberfläche einige Monitoring-Funktionen durchführen und weiter - falls es unabhängig von JBoss ESB eingerichtet wurde - auf die jBPM-Konsole zugreifen, die zur Prozessverwaltung dient.[94]

Mit der SOA-Plattform und insbesondere mit dem JBoss ESB steht ein Produkt zur Verfügung, das sich aufgrund der sehr umfangreichen Funktionalität und Unterstützung der Lose-

[89] Vgl. JBoss Messaging (2012)
[90] Vgl. JBoss Docs (2012)
[91] Vgl. JBoss Enterprise Calculator (2011)
[92] Vgl. JBoss Community (2012)
[93] Vgl. JBoss Downloads (2012)
[94] Vgl. JBoss Programmers Guide (2012)

Coupling-Architektur sehr gut für mittlere und große SOA-Anwendungslandschaften eignet. JBoss ESB zeigt deutliche Schwächen bei der reinen Betrachtung der Tool-Bedienbarkeit. Diese kann es aber mit seinem qualitativ hochwertigen, breit gefächerten Funktionsumfang in Sachen Verarbeitung und Umgebung wieder ausgleichen.

4.2.1.4 Unterschiede der JBoss Community- und JBoss Enterprise-Version

Eine ausführliche Übersicht der Unterschiede zwischen der JBoss Community- und JBoss Enterprise-Version kann in Anhang 3 eingesehen werden. Die folgende Aufzählung stellt die wichtigsten Vorteile der kommerziellen JBoss ESB Lösung dar:

- professioneller 24/7 Support durch den Hersteller
- getestete und zertifizierte Version des JBoss ESB und seiner Komponenten
- automatisierte Software- und Sicherheitsupdates zur Gewährleistung der Stabilität[95]

4.2.2 Mule ESB

Mule ist ein schlanker, auf Java basierender ESB und wird von der Firma MuleSoft Inc. entwickelt und vertrieben. Mit dieser Integrationsplattform können Entwickler verschiedene Applikationen, die unterschiedliche Technologien verwenden, miteinander verbinden. Für den Austausch von Daten wird ein gemeinsamer Message Bus verwendet.[96]

Nach eigenen Angaben ist Mule ESB der weltweit am häufigsten verwendete Open Source Enterprise Service Bus. Er wird von über 3.200 Unternehmen eingesetzt, darunter von fünf der zehn erfolgreichsten Banken der Welt. Anfang Januar 2013 wurde Mule ESB über 3.500.000 Mal heruntergeladen.[97/ 98] Dies unterstreicht die Relevanz des Mule ESB, weshalb er im Rahmen dieser Marktanalyse unabdingbar ist.

Neben dem kostenlosen Mule ESB existiert die kostenpflichtige Version Mule ESB Enterprise. Hierfür ist eine Anmeldung bei MuleSoft erforderlich. Diese besitzt gegenüber der Basisversion weitergehende Features in den Bereichen Sicherheit, Hochverfügbarkeit, Ausfallsicherheit, Performance Management und technischen Support. Diese Kriterien spielen insbesondere im Unternehmensumfeld eine wichtige Rolle. Auf die genauen Unterschiede der beiden Versionen wird im Laufe dieses Abschnitts genauer eingegangen.

[95] Vgl. JBoss Community and Enterprise Features (2010)
[96] Vgl. MuleSoft (2012a)
[97] Vgl. MuleSoft (2012b)
[98] Vgl. MuleSoft (2012c)

4.2.2.1 Architekturübersicht

Die Architektur des Mule ESB besteht aus mehreren Komponenten. Diese arbeiten zusammen, um die Features bereitzustellen, die einen ESB kennzeichnen. Die nachfolgende Abbildung zeigt die grundlegendsten Konzepte von Mule ESB.

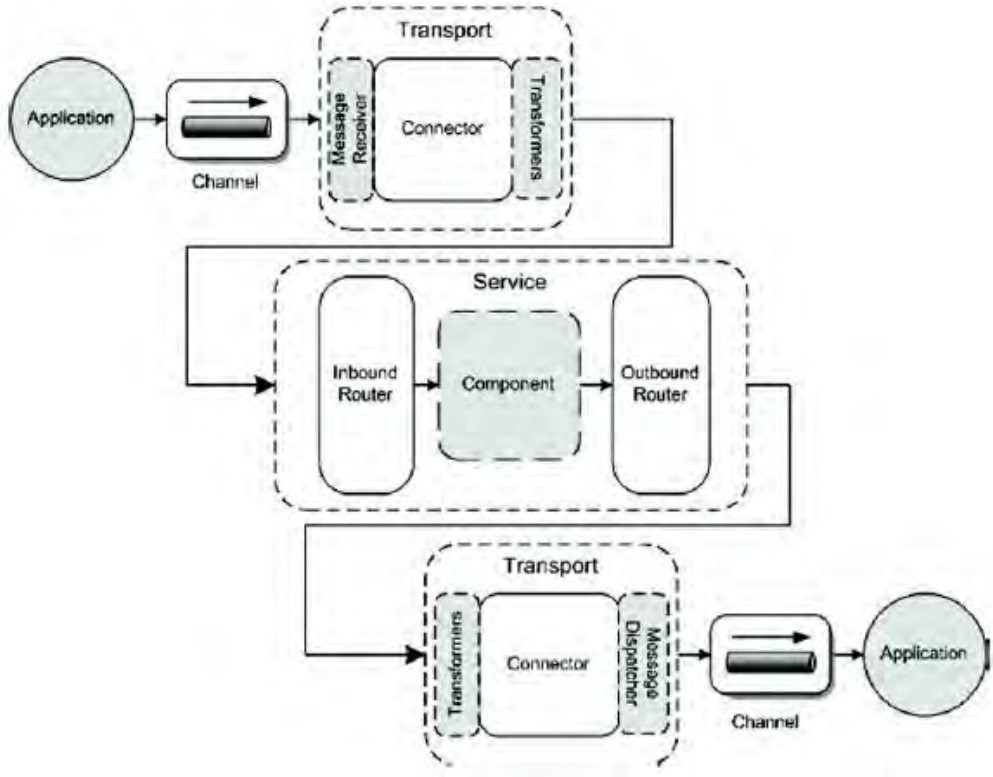


Abbildung 10: Grundlegende Konzepte von Mule ESB[99]

Nachfolgend werden zunächst die wichtigsten Komponenten der Mule-Architektur erläutert:[100]

Application

Eine Application kann alles sein – Cobol-System, .NET Anwendung, J2EE Applikation oder Web-Anwendung eines Unternehmens.

Channel

Durch einen Channel wird es einer externen Applikation ermöglicht mit Mule zu kommunizieren. Der Channel dient auch dazu, dass verschiedene Komponenten innerhalb des Mule ESB miteinander interagieren können.

[99] Entnommen aus: MuleSoft (o. J.)
[100] Vgl. ebenda

Message Receiver

Mule verwendet einen Message Receiver, um eingehende Nachrichten aus einem bestimmten Channel zu empfangen, damit diese weiterverarbeitet werden können. Um die Nachricht zu empfangen, benutzt der Message Receiver die spezifische Technologie, die für einen bestimmten Channel benötigt wird (z.B. HTTP). Mule stellt nach eigenen Angaben für viele gebräuchliche Standards und Technologien Message Receiver bereit.

Connector

Ein Connector versteht es, wie Daten aus bestimmten Channels zu empfangen und zu senden sind. Der Connector ist deshalb sowohl am empfangenden als auch am sendenden Ende vorhanden. Der Message Receiver und der Message Dispatcher sind Bestandteil des Connectors.

Transformer

Mit einem Transformer werden Daten von einem Format in ein anderes umgewandelt. Mule verwendet standardmäßig eine sinnvolle Transformation für ankommende und ausgehende Nachrichten. Wenn beispielsweise über JMS eine Nachricht empfangen wird, wandelt Mule eine Textnachricht in einen String und eine ObjectMessage in ein Java-Object um. Es besteht die Möglichkeit die Standard-Transformationen mit eigenen Transformationsregeln zu überschreiben.

Inbound Router

Wie in

Abbildung 10 zu sehen ist, kommt der Inbound Router zum Einsatz, nachdem die Nachricht transformiert wurde. Mit diesem Router kann festgelegt werden, was mit einer eingehenden Nachricht geschieht, d. h. an welches Ziel diese weitergeleitet werden soll.

Der Inbound Router wird auch Selective consumer genannt. Mit einem Filter kann bestimmt werden, welche Art von Nachrichten empfangen werden soll, z. B. nur Nachrichten, die einen String beinhalten.

Component

Innerhalb der Mule Architektur stellt eine Component den Ort dar, in dem Integrationslogik implementiert wird, die nicht von anderen Bestandteilen von Mule zur Verfügung gestellt

wird. Eine Component wird aufgerufen, wenn ein Inbound Router eine Nachricht empfangen und sie alle Filter erfolgreich durchlaufen hat.

Outbound Router

Ein Outbound Router ist im Wesentlichen dasselbe wie ein Inbound Router. Nach dem die Nachricht von der Component bearbeitet wurde, bestimmt der Outbound Router, wohin die Nachricht gesendet werden soll.

Message Dispatcher

Der Message Dispatcher stellt das Gegenstück zum Message Receiver dar. Diese Komponente weiß, wie Informationen über einen bestimmten Channel versendet werden und stellt die Nachricht in den Ziel-Channel ein.

Gesamtzusammenhang

Nachdem die einzelnen Komponenten der Architektur des Mule ESB vorgestellt wurden, folgt nun die Beschreibung eines beispielhaften Anwendungsfalles, um den Gesamtzusammenhang aufzuzeigen.

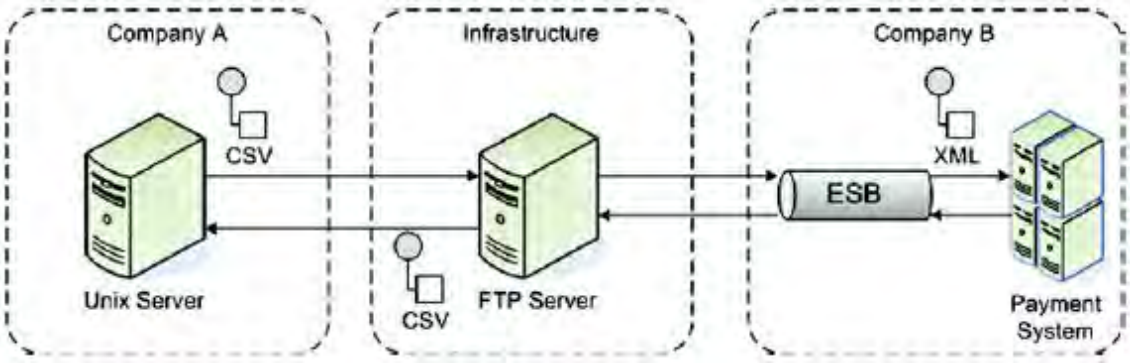


Abbildung 11: Anwendungsfall Mule ESB[101]

Wie die Abbildung 11 zeigt, wird im folgenden Beispiel eine CSV-Datei von einem FTP Server gelesen, in eine XML-Nachricht transformiert und anschließend zu einem Payment Service gesendet. Nachdem der Payment Service die Nachricht verarbeitet hat, wird sie erneut in das CSV-Format transformiert und im Dateiverzeichnis abgelegt.

[101] Entnommen aus: MuleSoft (o. J.), S. 45

Damit dieser Anwendungsfall mit Mule ESB abgebildet werden kann, ist der Einsatz der vorgestellten Komponenten erforderlich. Folgende Aktionen finden statt:[102]

1. Eine Applikation von Unternehmen A legt die CSV-Datei in ein vorgegebenes FTP Directory ab.

2. Dieses Directory fungiert als **Channel** in der laufenden Mule Instanz.

3. Mule empfängt diese Nachricht aus dem Datei-Channel mit dem **Message Receiver**, der ein Teil des eingehenden **Connectors** ist, damit sie weiterverarbeitet werden kann.

4. Der **Transformator** wandelt die eingehende CSV-Datei in das XML-Format um, welches der Payment Service seinerseits benötigt.

5. Nun überprüft der **Inbound Router**, wohin die Nachricht gesendet werden soll. Im beschriebenen Beispiel an den Payment Service.

6. Der Payment Service ist als **Component** deklariert. Implementiert ist der Service als POJO (Java-Objekt ohne externe Abhängigkeiten wie Schnittstellenimplementierungen, Annotationen, Namenskonventionen), das Nachrichten von Mule empfangen kann.

7. Nachdem die Nachricht vom Payment Service verarbeitet wurde, kommt der **Outbound Router** zum Einsatz, um zu bestimmen, wohin die Nachricht als nächstes gesendet wird.

8. Mit dem **Message Dispatcher**, der Teil des ausgehenden Connectors ist, wird die Nachricht zu einem Channel gesendet. Im Beispiel ist der Channel erneut ein Directory auf dem FTP Server. Von dort kann die Applikation von Unternehmen A die Datei wieder aufnehmen.

4.2.2.2 Funktionale Features

Nachrichtenverarbeitung

Für den Transport von Nachrichten stellt der Mule ESB zahlreiche Transportarten zur Verfügung, dabei werden alle gängigen unterstützt. Zu den wichtigsten Transportarten zählen un-

[102] Vgl. MuleSoft (o. J.), S. 45

ter anderem: File, FTP, HTTP/S, IMAP, JDBC, JMS, SMTP, SOAP, TCP und UDP. Der Nachrichtentransport kann synchron sowie asynchron erfolgen.[103] Mit JMS sind die beiden Kommunikationsmuster Point-to-Point und Publish-Subscribe möglich.

Ein wichtiges Feature bei einem ESB stellt das Routing dar. Im Mule ESB existieren mehrere Router, welche durch ihren Einsatz und Kombination miteinander die Nachrichten an den richtigen Zielendpunkt weiterleiten. Die Routing-Regeln müssen manuell in einer speziellen XML-Konfigurationsdatei eingetragen werden.[104] Das Routing von Nachrichten im Mule ESB kann auch durch einen Workflow erfolgen. Hierfür wird u. a. BPEL unterstützt.[105]

Durch die Vielzahl von Router kann der Entwickler je nach Anwendungsfall eine Nachricht auf dem, am geeignetsten erscheinenden Weg routen. Die wichtigsten Router im Mule ESB sind:

Router	Beschreibung
Pass-Through-Router	Routet eine Nachricht direkt zum Zielendpunkt
Filtering-Router	Prüft die Nachricht auf ihren Inhalt und verarbeitet sie davon abhängig weiter (Content-Based-Routing)
Exception-Based-Router	Bei fehlerhaften Transports findet eine alternative Nachrichtenübermittlung statt
Collection Aggregator	Ankommende Nachrichten werden gruppiert. Dies ermöglicht deren Verarbeitung im Batch-Betrieb
Wiretap-Router	Bestimmte Nachrichten können zusätzlich zu alternativen Zielendpunkten umgeleitet werden

Tabelle 2: Die wichtigsten Router im Mule ESB[106]

Für die Transformation von Daten zwischen verschiedenen Formaten bietet der Mule ESB eine Reihe von Datenformaten an. Folgend werden die wichtigsten genannt: CSV, HTML/XHTML, Java Objects, Byte Arrays.

[103] Vgl. Buchholz, A. / Hohloch, R. / Rathgeber, T. (2007), S. 27
[104] Vgl. ebenda, S. 33
[105] Vgl. ebenda, S. 33
[106] Vgl. NetWays (2012)

Sicherheit

Bezüglich der Sicherheit steht das Spring Framework, Acegi, JAAS, PGP und SS4TLS zur Verfügung. Mit diesen Methoden können Services authentifiziert werden.[107] Es wird auf WS-Security zurückgegriffen, um Nachrichten zu verschlüsseln.

Transaktionen müssen ACID-Eigenschaften aufweisen, damit verlässliche Ergebnisse erreicht werden können. Der Mule ESB besitzt einen Transaktionsmanager, welcher JDBC, XA und JMS Transaktionen ermöglicht.[108]

Skalierbarkeit

Der Mule ESB ist nach eigenen Angaben skalierbar. So unterstützt dieser z. B. mehrere Topologien.[109] Des Weiteren wird die Möglichkeit geboten, einen Cluster aufzusetzen. Dies wird durch Mules in-memory data grid erreicht. Das bedeutet, dass alle Instanzen dieselbe Applikation ausführen und die Arbeitslast untereinander verteilen können. Sollte eine Mule Instanz ausfallen, nehmen die anderen die Last ohne Betriebsunterbrechung unverzüglich auf.[110]

Weitere Funktionen

Zu Beginn dieses Kapitels wurde angesprochen, dass neben der kostenlosen Version eine kostenpflichtige Enterprise-Version zur Verfügung steht. Ein Nachteil der kostenlosen Version findet sich im Monitoring. So stehen nur beim Mule ESB Enterprise u. a. ein Monitoring Integration Framework, SLA Alerts (Warnungen vor potenziellen Systemproblemen, bevor sie passieren), ein Runtime Performance Manager sowie ein Operational Dashboard (einheitliche Sicht über alle Vorgänge in den Mule Instanzen) zur Verfügung. Ebenfalls nur in der kostenpflichtigen Version enthalten ist ESB Remote Control, mit dem sich Ressourcen des ESB starten, stoppen und neu starten lassen.[111]

Mit dem Mule Studio stellt Mule eine komfortable Art der Bedienung zur Verfügung. Dieses auf Eclipse basierende Tool ermöglicht es, grafisch Abläufe zu entwerfen, zu testen und auszuführen. Es kann zwischen der grafischen Aufbereitung und dem darunterliegenden XML-Code gewechselt werden.[112] Durch dieses Tool ist der Mule ESB gut bedienbar. Diese Einschätzung basiert auf eigener Erfahrung, da Mule ESB im Rahmen des Projektes getestet wurde.

[107] Vgl. Buchholz, A. / Hohloch, R. / Rathgeber, T. (2007), S. 34
[108] Vgl. ebenda, S. 33
[109] Vgl. ebenda, S. 35
[110] Vgl. MuleSoft (2012d)
[111] Vgl. MuleSoft (2012f)
[112] Vgl. MuleSoft (2012e)

Der Mule ESB unterstützt im Netz bereitgestellte Komponenten (Web Services), welche „eine Abstraktionsebene einer Anwendungslogik darstellt".[113] Mit Internetprotokollen kann auf diese Komponenten zugegriffen werden. Für die Nachrichtenkodierung wird XML verwendet.[114] Unterstützte Sprachen und Protokolle sind: .NET Web Services, REST, WS-Addressing, WS-Policy, WS-Security, WS-I BasicProfile, WS-I SecurityProfile und WSDL.

Für die Registrierung von Services verwendet Mule ESB eine eigene RMI-Registry, wobei die Services nicht automatisch registriert werden.[115]

Im Mule ESB können eigene Adapter entwickelt werden. Dem Entwickler stehen dafür genügend Informationen und eine Vielzahl an Beispielen zur Verfügung.[116]

4.2.2.3 Nicht-Funktionale Features

Eine Dokumentation des Mule ESB ist auf der Webseite www.mulesoft.org enthalten. Behandelt werden Themen wie Einstieg, Erstellen von Connectors, Management Console und Mule Studio (Registrierung in der Community erforderlich) sowie die Entwicklung von Mule Applications und FAQs.[117]

Die Mule Community besitzt zurzeit rund 105.000 Mitglieder.[118] 24/7 Premium Support sowie Service Packs & Hot Patches sind nur bei der kostenpflichten Version Mule ESB Enterprise enthalten.[119] Der Support erfolgt durch MuleSoft Consulting Services.

Der Mule ESB verfügt über eine Vielzahl an Referenzen. Unter anderem verwenden Facebook, Barclays, Forbes, Sky, BWM, MasterCard, Nestle und Olympus den Mule ESB.[120]

4.2.2.4 Unterschiede der Mule Community- und Mule Enterprise Version[121][122]

Die kostenpflichtige Version des Mule ESB Enterprise verfügt über eine Vielzahl von Features, welche in der kostenlosen Version des Mule ESB nicht enthalten sind (siehe Anhang 4). Nachfolgend werden die wichtigsten aufgelistet:

[113] Vgl. Golem (o. J.)
[114] Vgl. ebenda
[115] Vgl. Buchholz, A. / Hohloch, R. / Rathgeber, T. (2007), S. 34
[116] Vgl. ebenda, S. 36
[117] Vgl. MuleSoft (2012h)
[118] Vgl. MuleSoft (2012c)
[119] Vgl. MuleSoft (2012f)
[120] Vgl. MuleSoft (2012g)
[121] Vgl. MuleSoft (2012f)
[122] Vgl. MuleSoft (2012i)

- **Mule Studio DataMapper**
  Leistungsfähige, grafische Datenintegrationsmöglichkeiten.
- **Premium JDBC Connector**
  - Batched Statements, JDBC – XML / CSV Transformation und Ausführung von Stored Procedures.
- **Enterprise Gateway für SAP**
  - Bidirektionale Kommunikation via Application Link Enabling (ALE) und iDocs Technologie mit SAP-Lösungen.
- **Mule Enterprise Security**
  - Bereitstellung von sechs Tools, welche sicherstellen, dass autorisierte Enduser sicheren Zugang zu geschützten Daten haben.
- **High Availability**
  - Durch eine Cluster-Lösung können mehrere Mule Server gruppiert werden und diese als ein virtueller Server agieren lassen. Zwischen Knoten findet automatisch eine Lastverteilung sowie Ausfallsicherung statt und garantiert, dass keine Nachrichten verloren gehen.
- **Caching**
  Vermeidung von unnötigen Zugriffen auf Datenbanken oder Web Services durch Zwischenspeichern der Ergebnisse von Aufrufen.
- **Performance Management**
  - Bereitstellung der Tools Monitoring Framework Integration, Runtime Performance Manager und Performance KPI Tracking.
- **Operational Dashboard**
  - Monitoring des gesamten Systems auf einem einzigen Bildschirm. Es wird eine einheitliche Sicht über alle Vorgänge in den Mule Instanzen gewährleistet.
- **24/7 Premium Support**
- **Service Packs & Hot Patches**

4.2.3 Apache ServiceMix

Apache ServiceMix ist ein aus ausschließlich von OSGi zertifizierten Komponenten bestehender ESB. Das bedeutet, dass er aus modularen Bausteinen besteht, die ohne großen Aufwand ausgetauscht oder um weitere Bausteine erweitert werden können. Dabei ist der Benutzer nicht an einen bestimmten Anbieter gebunden.[123] Apache ServiceMix ist aktuell in

[123] Vgl. Apache Software Foundation (2013c)

der Version 4.4.2 verfügbar und läuft unter der Apache License 2.0. Seit dem 13.09.2007 ist er ein Apache Top Level Projekt.[124]

Da es ich bei Apache um eine Stiftung handelt, die sich nicht über ihre entwickelten Produkte finanziert,[125] existiert - im Gegensatz zu den beiden vorherigen Produkten - nur eine kostenlose Version des Apache ServiceMix. Daher entfällt ein Vergleich zwischen einer Community- und Enterprise-Version.

4.2.3.1 Architekturübersicht

Die Funktionalität des Apache ServiceMix basiert auf der Java-Spezifikation Java Business Integration (JBI), die eine komponentenbasierte Architektur beschreibt, d. h. die Anwendung ist ein Container, der durch verschiedene Plugins erweitert werden kann. Diese Plugins können über einen Normalized Message Router (NMR) Nachrichten austauschen ohne sich gegenseitig kennen zu müssen.[126]

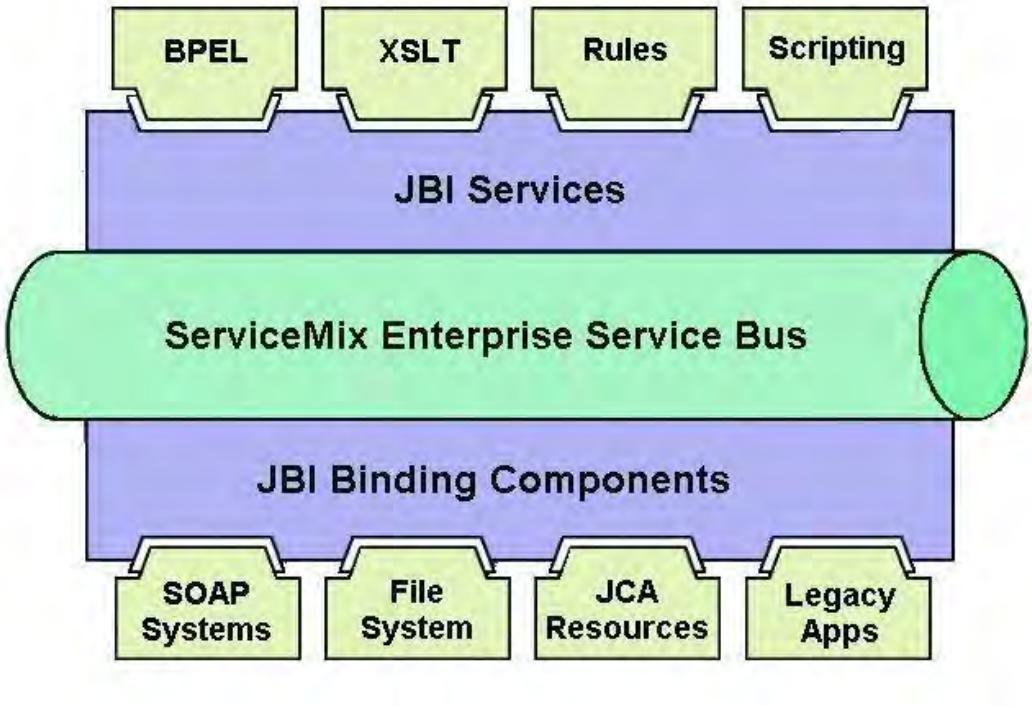


Abbildung 12: Architektur von Apache ServiceMix[127]

[124] Vgl. Apache Software Foundation (2011a)
[125] Vgl. Apache Software Foundation (2012)
[126] Vgl. De Angelis, R. (2009)
[127] Vgl. Hanson, J. J. (2005)

Die oben stehende Abbildung zeigt die typische Architektur des Apache ServiceMix. Dieser bildet die Kernkomponente, welche mittels Service Container mit den angeschlossenen Diensten verbunden ist. Die Service Container werden in zwei Kategorien unterteilt: JBI Services und JBI Binding Components.

JBI Services stellen lediglich Kontakt mit dem NMR her. Sie liefern die Logik, die der ESB zur Bewältigung seiner Aufgaben benötigt. Im Schaubild werden die Services BPEL, XSLT, Rules[128] und Scripting angegeben. Rules ist hier als eine Komponente anzusehen, die Regeln für die Implementierung eines Routers oder eines Services, auch als ServiceMix Drools bezeichnet, bereitstellt. Scripting stellt eine Funktion dar, die es erlaubt, alle mit Java Specification Request 223 (JSR)[129] kompatiblen Programme zu benutzen.

JBI Binding Components verbinden den ESB mit der Außenwelt. Sie wandeln eingehende bzw. ausgehende Nachrichten um, sodass diese nicht an spezielle Protokolle gebunden sind. Es werden die externen Systeme SOAP-Systeme, File Systeme, JCA Resources und Legacy Apps genannt.

Technisch besteht Apache aus mehreren, auch alleine lauffähigen Komponenten: die zusammen gesteckt als ein Programm funktionieren.

Der Apache ActiveMQ fungiert als Message Brooker, über Apache Camel wird Routing sowie Konvertierung umgesetzt. Apache CXF ergänzt den ServiceMix um ein Framework zur Erstellung von Web Services, Apache ODE um eine Workflow Engine und Apache Karaf fügt ein Run-Time-System hinzu. Ergänzt wird der ServiceMix durch einen NMR und um einige kleinere Komponenten.[130]

4.2.3.2 Funktionale Features

Basierend auf den Ausführungen des Kapitels 3.3.3, soll in diesem Abschnitt auf die Funktionalitäten der Komponenten des Apache ServiceMix näher eingegangen werden.

Über Apache Camel wird das direkte Messaging mit einer Vielzahl von Protokollen unterstützt. Diese sind unter anderem HTTP, FTP, SMTP, POP3, JMS, ActiveMQ und JDBC. Für nicht unterstützte Protokolle können eigene Adapter programmiert werden.

Apache Camel beherrscht Content-Based- sowie dynamisches Routing zur Unterstützung des Nachrichtenaustausches. Des Weiteren wird das Filtern, Aufteilen und Aggregieren von Nachrichten gewährleistet. Über eine weitere Funktionalität, den Resequenzer, können ver-

[128] Vgl. Apache Software Foundation (2011a)
[129] Vgl. o.V. (2010)
[130] Vgl. Apache Software Foundation (2011b)

teilt bearbeitete Nachrichten in die korrekte Reihenfolge gebracht werden. Ebenso besteht die Möglichkeit den Nachrichtenfluss zu drosseln bzw. zu beschleunigen, um z. B. die SLAs einzuhalten.

Eine weitere wichtige Funktion ist die Implementierung der JBI Komponente. Diese enthält die Registry der Endpunkte, der Camel TOEs. Diese Registry wird maschinell verwaltet.[131]

Abgerundet werden die Funktionen von Camel mit einem Transformator, der eine Transformation sowohl syntaktischer wie semantischer Natur durchführen kann, sowie einem Content Enricher und seinem Gegenstück, einem Content Filter. Ein Content Enricher versieht eine Nachricht mit weiterem Informationsgehalt wohingegen ein Content Filter den Nachrichteninhalt reduziert.

Apache ActiveMQ erweitert die Messaging Funktionen des ServiceMix: Es unterstützt JMS, wodurch eine plattformneutrale und programmiersprachenunabhängige Kommunikation möglich ist, sowohl nach dem Point-to-Point als auch dem Publish-Subscribe Prinzip. Eine weitere wichtige Aufgabe dieser Komponente ist die Verschlüsselung des Nachrichtenkanals. Dies geschieht per SSL oder HTTPS. Neben SSL werden auch andere Kommunikationsprotokolle, wie HTTP, Apache-VM, TCP und Multicast unterstützt. Weiterhin integriert ActiveMQ das Cached LDAP authorization module. Dieses erlaubt es, Autorisierung und Authentifizierung umzusetzen und zu verwalten.[132]

In Apache ActiveMQ ist es möglich, die Leistungsfähigkeit eines einzelnen Brookers zu steigern (vertikale Skalierung), oder mehrere vernetzte Brooker zu benutzen (horizontale Skalierung). Auch eine Hybrid-Lösung ist möglich.[133] Dadurch kann der Apache ServiceMix als gut skalierbares Produkt bezeichnet werden.

Ein ebenso wichtiger Punkt für einen ESB ist die Umsetzung von Web Service Standards. Diese Aufgabe wird von Apache CXF durchgeführt. Es unterstützt die Standards SOAP, das WS-I Basic Profile, WSDL, WS-Addressing, WS-Policy, WS-ReliableMessaging, WS-Security, WS-SecurityPolicy, WS-SecureConverstation, und zum Teil WS-Trust.[134]

Mit Apache ODE existiert zudem eine Workflow Engine, die dem WS-BPEL Standard folgt.[135]

Zuletzt soll noch auf die Komponente Apache Karaf eingegangen werden, womit der ServiceMix um eine CLI-basierte Konsole erweitert wird. Diese erlaubt Hot Deployment von

[131] Vgl. Apache Software Foundation (2011c)
[132] Vgl. Apache Software Foundation (2011b)
[133] Vgl. Bosanac, D. (2010), Folie 38 ff.
[134] Vgl. Apache Software Foundation (2013b)
[135] Vgl. Apache Software Foundation (2013a)

OSGi-Bundels, Logging, Monitoring, Administration und bietet ein auf JAAS basierendes Security-Framework.[136]

Eine Transaktionsverwaltung ist in der Dokumentation nicht explizit erwähnt.Basierend auf dieser, wird davon ausgegangen, dass sie teilweise, jedoch nicht vollständig umgesetzt wird.

4.2.3.3 Nicht-Funktionale Features

Der Apache ServiceMix ist ein Community Projekt. Daher findet ein direkter Support nur über die Community statt. Es existiert allerdings die Möglichkeit über ein Service-Unternehmen Support zu kaufen. Obwohl der Community-Support äußerst umfangreich ist (neben dem Forum gibt es Mail-Listen, ein Chat und ein Issue Tracker (eine Datenbank für Fehler)), so umfasst dieser weder telefonische noch persönliche Beratung.

Auf der Webseite des Produktherstellers wird für kommerziellen Support Savoir Technologies Inc empfohlen. Es existieren allerdings noch weitere Firmen, die Support anbieten. Als Beispiel soll hier die deutsche Firma X-Integrate genannt werden.[137]

Apache ServiceMix kann kostenlos installiert werden. Ein Preis für den professionellen Support bei der Installation und Bedienung, ist von den anbietenden Firmen nicht öffentlich genannt.

Beim Apache ServiceMix existiert keine graphische Oberfläche, stattdessen wird in der Konsole gearbeitet. Dadurch sind Konfiguration und Einarbeitung relativ schwierig. Die Guides, die zur Verfügung gestellt werden, decken lediglich einen Teil ab, bzw. sind zum Teil sogar unvollständig, wie der Architektur Guide[138], weshalb diese Probleme auch auf diesem Wege nicht behoben werden können.

Der Hersteller macht auf seiner Webseite keine Angaben, in welchen Unternehmen Apache ServiceMix eingesetzt wird. Lediglich die Savoir Technologie Inc. nennt Kunden, die sie im Umgang mit Apache beraten haben, wie Progress Software, das National Center for Atmospheric Research (NCAR) und GE–Healthcare.[139]

[136] Vgl. Apache Software Foundation (2011d)
[137] Vgl. Apache Software Foundation (2013c)
[138] Vgl. Apache Software Foundation (2011a)
[139] Vgl. Savoir Technologies Inc. (2012)

4.3 Betrachtung des Kooperationspartnerunternehmens

Bevor die in den vorigen Kapiteln betrachteten ESB-Systeme zielführend bewertet werden können, müssen die Voraussetzungen für den Einsatz beim Kooperationspartner als Kunden eruiert werden. Im Folgenden werden die Fragen geklärt, welche Rahmenbedingungen durch die derzeitig vorhandene IT-Landschaft beim Kunden gegeben sind und welche Anforderungen künftig an das ESB-System gestellt werden.

4.3.1 Vorhandene IT-Anwendungslandschaft

Der z/OS Großrechner ist ein zentraler Bestandteil der IT-Landschaft des Kooperationspartners und damit grundlegend für den Betriebsablauf. COBOL-Programmierungen stellen hierbei den Fachkern dar.

Abbildung 13 veranschaulicht dies in vereinfachter Weise und darüber hinaus das konkrete Ziel des ESB Einsatzes, für heterogene Systeme eine einheitliche Schnittstelle zur Verfügung zu stellen.

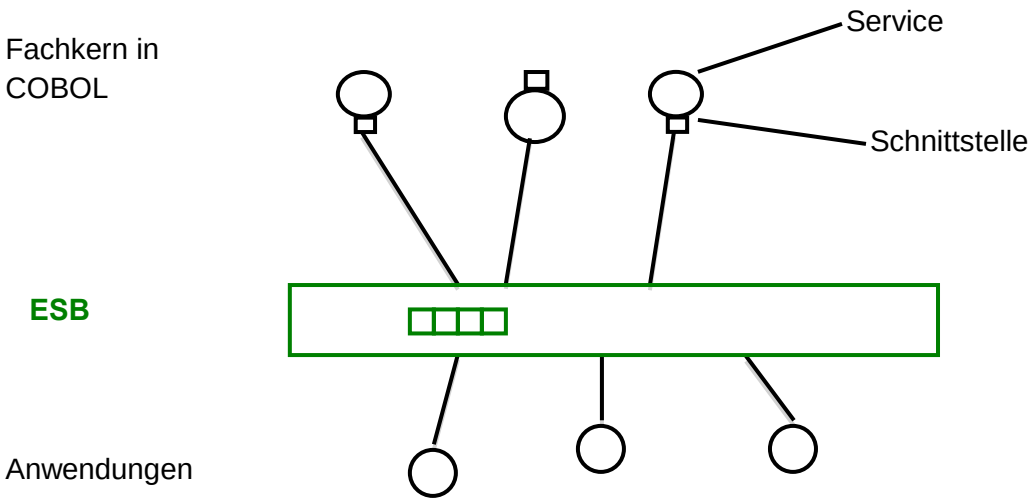


Abbildung 13: ESB in der IT-Landschaft des Kooperationspartners

Der Abbildung ist zu entnehmen, dass die in COBOL programmierten Services von mehreren Anwendungen verwendet werden. Bislang erfolgt deren Aufruf über unterschiedliche Schnittstellen. Zur Veranschaulichung sind hierbei Aufrufe durch unterschiedliche Anwendungen eines Sachbearbeiters oder Maklers denkbar, welche beispielsweise über den Browser erfolgen. Hierbei existiert von jeder Anwendung zu jedem Service eine Point-to-Point Verbindung, wodurch mehrere Schnittstellen an einem Service nötig sind. Anhand der einheitlichen, zentralen Schnittstelle des ESBs sind die heterogenen Schnittstellen der Services durch jeweils nur eine Schnittstelle abzulösen, welche allein mit dem ESB kommuniziert,

statt mit mehreren Anwendungen. Dies bringt neben den bereits in Kapitel 3.2.2 beschriebenen Vorteilen, wie hauptsächlich der erhöhten Unabhängigkeit und Flexibilität, eine Reduzierung des Integrationsaufwandes mit sich. Durch den Zugriff auf den zentralen Kommunikationsbus ist die Komplexität der Schnittstellenverwaltung deutlich reduziert und bringt damit Kosteneinsparungen im Bereich Entwicklung und Wartung mit sich.

Welche individuellen Anforderungen an das einzusetzende ESB-System vom Kooperationspartner gestellt werden, ist im anschließenden Abschnitt beschrieben.

4.3.2 Anforderungen an den ESB

In gemeinsamen Terminen mit dem Kooperationspartner sind Kriterien festgelegt und ihrer Wichtigkeit für das Unternehmen geklärt worden. Daraufhin hat das Team eine Bewertung in Zahlen vorgenommen, welche für die Nutzwertanalyse nötig waren. Somit ist ein messbares Verhältnis der Anforderungen und damit einhergehend eine Priorisierung umgesetzt worden. Der endgültige Anforderungskatalog, welchem die untersuchten ESB-Systeme gerecht werden müssen, wird im Folgenden vorgestellt:

Anforderung	Beschreibung	Gewichtung
Routingfähigkeit	Die Routingfähigkeit, die ein grundlegendes Element der Funktionsweise eines ESB-Systems ist, stellt folglich ein **Musskriterium** dar. Bei den zu bewertenden ESB-Systemen ist insbesondere auf inhaltsbasiertes und versionsgerechtes Routing zu achten.	Muss
Kommunikationsmuster	Dies stellt ebenfalls ein **Musskriterium** dar. Bei der Analyse der Produkte wird ein besonderes Augenmerk auf die beiden Kommunikationsmuster Queue (Point-to-Point) und Topic (Publish-Subscribe) gelegt.	Muss
Protokollumwandlung	Bei der Protokollumwandlung, die ebenfalls als **Musskriterium** gilt, ist nur auf wichtigsten Standards zurückzugreifen wie bei-	Muss

	spielsweise SOAP, HTTP, FTP, IMAP, CSV.	
Nachrichtentransformation	Die Nachrichtentransformation wurde ebenfalls als **Musskriterium** definiert. Dabei ist insbesondere auf syntaktische und semantische Transformation zu achten.	Muss
Unterstützung des Routings von Nachrichten durch Workflow-Engine	Diese Fähigkeit von ESBs ist von größerer, aber nicht essenzieller Bedeutung. Daher ist es als wichtiges **Wunschkriterium** zu behandeln.	7
Verschlüsselungsmechanismen	Für ein Unternehmen, deren Produkte an sich aus Informationen bestehen, ist es von **höchster Wichtigkeit**, dass eine hohe Sicherheit durch Verschlüsselung sowohl auf Nachrichten- als auch auf Protokollebene gegeben ist.	9
Transaktionskonzept (ACID)	Damit die Konsistenz der Daten garantiert wird, ist das ACID-Prinzip[140] ein **sehr wichtiges** Kriterium bei der Auswahl eines ESB-Systems und daher hoch zu bewerten.	8
Authentifizierung, Autorisierung	Dies stellt aktuell für den Kooperationspartner ein **weniger wichtiges** Kriterium dar, da diese Funktionalitäten von übergeordneten Komponenten übernommen werden.	4
Skalierbarkeit	In einem Unternehmen ist es eine **unabdingbare Anforderung**, dass der ESB auch bei Höchstlast skalierbar ist und Performanceverluste in geringstmöglichem Maß spürbar sind.	Muss
Monitoring (Logging)	**Unerlässlich** für die Aufrechterhaltung des	Muss

[140] **A**tomacity**C**onsistency**I**solation**D**urability – Datenoperationen müssen abgeschlossen, konsistent, isoliert und dauerhaft sein, um die Richtigkeit der Daten zu jedem Zeitpunkt sicherzustellen

	Betriebes des ESB ist die Funktionalität diesen zu überwachen. Aus diesem Grund ist diese Anforderung als **Muss-Kriterium** zu bewerten.	
Umsetzung von Web-Service Standards	Das Umsetzen von Web Service Standards wie beispielsweise SOAP, WSDL oder WS-Security ist ebenfalls ein **Muss-Kriterium**.	Muss
Realisierung von Workflows mit BPEL	BPEL, als XML-basierte Sprache zur Modellierung von Geschäftsprozessen, ist lediglich ein Zusatzkriterium für eine komfortable Nutzung und damit **weniger wichtig**.	3
Selbstständige Verwaltung einer Registry	Dieses Kriterium ist derzeit von untergeordneter Bedeutung. Aus diesem Grund wird das Kriterium als zusätzlicher Komfort angesehen und als **unwichtig** bewertet.	2
Möglichkeit der Eigenentwicklung von Adaptern	Diese Fähigkeit muss das ESB laut Aussage des Partners **nicht erfüllen**.	1
Support	Der Support zur Sicherstellung des Betriebs auch bei Problematiken in der täglichen Anwendung muss gegeben sein. Bei Open Source Anwendungen kann dies beispielsweise durch eine hohe und qualitativ hochwertige Aktivität in Foren gemessen werden. Daher weist dieses Kriterium eine **sehr hohe Wichtigkeit** auf.	10
Referenzen	Referenzen einer Software sind grundsätzlich hilfreich vor der Entscheidung einer größeren Investition beziehungsweise der Integration eines neuen IT-Systems, da die Folgen dieser meist mehrjährigen Ausrich-	8

	tung vorab bestmöglich durchdacht sein sollten. Referenzen sind hierbei ein einfaches und oft aussagekräftiges Mittel und damit im Kriterienkatalog als **sehr wichtig** zu bewerten.	
Bedienbarkeit und Einarbeitungszeit	Da die Einarbeitung einmalig stattfindet und die tägliche Bedienung daher hauptsächlich ausschlaggebend ist, ist diese dementsprechend **hoch** zu bewerten. Für die **eher unwichtige** Einarbeitung ist der Partner zur Investition in zusätzliche Schulungen bereit.	7

Nachdem die Anforderungen mit den jeweiligen Gewichtungen seitens des Kooperationspartners bestätigt worden sind, hat die Bewertung der ESB-Systeme hinsichtlich der individuellen Bedürfnisse stattfinden können. Diese ist im folgenden Kapitel aufgeführt.[141]

[141] Vgl. Bereichsleiter SE und AE-PP (2012)

4.4 Gegenüberstellung der ESB-Lösungen

Aufbauend auf den erarbeiteten Anforderungen im vorherigen Kapitel, wird im ersten Schritt die Umsetzung jener Anforderungen durch die Testkandidaten evaluiert. Im Anschluss wird auf Hindernisse und Schwierigkeiten in Bezug auf die Bewertung der ausgewählten ESB-Produkte. Abschließend wird, basierend auf der vorangegangen Nutzwertanalyse und den gesammelten Erfahrungen während der Forschung, eine Empfehlung über den adäquaten Einsatz eines Open Source ESBs als Alternative zum IBM WebSphere ESB gegeben.

4.4.1 Nutzwertanalyse

In diesem Kapitel wird die Nutzwertanalyse, wie bereits in Kapitel 2.2.2 erwähnt, durchgeführt. Im Folgenden wird der Erfüllungsgrad der einzelnen Kriterien bewertet. Hierbei handelt es sich um Muss- und Wunschkriterien, wobei bei letzteren eine Gewichtung angegeben ist. Das vollständige Scoring-Modell ist in Anhang 4 dokumentiert.

4.4.1.1 Funktionale Anforderungen

Routingfähigkeit

Art:	Musskriterium				
Gewichtung:	-				
IBM WebSphere	Mule		JBoss		Apache
	Community	Enterprise	Community	Enterprise	
✓	X	X	X	X	X

Tabelle 3: Nutzwertanalyse - Routingfähigkeit

Das Kriterium der Routingfähigkeit wird von keinem Open Source Produkt erfüllt, da die Anforderungen des Kooperationspartners neben dem inhaltsbasiertem Routing auch das versionsgerechte Routing verlangen. Letztere Anforderung wird lediglich durch den IBM WebSphere ESB umgesetzt.

Kommunikationsmuster

Art:	Musskriterium				
Gewichtung:	-				
IBM WebSphere	Mule		JBoss		Apache
	Community	Enterprise	Community	Enterprise	
✓	✓	✓	✓	✓	✓

Tabelle 4: Nutzwertanalyse - Kommunikationsmuster

In Bezug auf Kommunikationsmuster erfüllen alle ESBs die Anforderungen. Dies liegt insbesondere daran, dass die ESBs JMS als Programmierschnittstelle (API) für die Ansteuerung einer Message Oriented Middleware zur Verfügung stellen. Diese API bietet die Ansätze Queue (Point-to-Point) und Topic (Publish-Subscribe). Aus diesem Grund schneiden alle Produkte bezüglich dieser Anforderung gleich ab.

Protokollumwandlung

Art:	Musskriterium				
Gewichtung:	-				
IBM WebSphere	Mule		JBoss		Apache
	Community	Enterprise	Community	Enterprise	
✓	✓	✓	✓	✓	✓

Tabelle 5: Nutzwertanalyse - Protokollumwandlung

Auch bei der Protokollumwandlung kann sich keines der Produkte von der Konkurrenz abheben. Alle untersuchten ESBs unterstützen die gängigsten Protokollstandards wie beispielweise HTTP, FTP, SMTP, SOAP sowie JDBC. Darüber hinaus werden weitere Protokollstandards unterstützt, die jedoch für die Erfüllung der Anforderung unerheblich sind.

Nachrichtentransformation

Art:	Musskriterium				
Gewichtung:	-				
IBM WebSphere	Mule		JBoss		Apache
	Community	Enterprise	Community	Enterprise	
✓	✓	✓	✓	✓	✓

Tabelle 6: Nutzwertanalyse - Nachrichtentransformation

Das Kriterium der Nachrichtentransformation wird erneut von allen ESBs gleichermaßen erfüllt. Dabei unterstützen alle Produkte sowohl die syntaktische als auch semantische Transformation, welche in den Anforderungen explizit erwähnt sind.

Unterstützung des Routings von Nachrichten durch Workflow-Engine

Art:	Wunschkriterium				
Gewichtung:	7				
IBM WebSphere	Mule		JBoss		Apache
	Community	Enterprise	Community	Enterprise	
10/10	10/10	10/10	10/10	10/10	10/10

Tabelle 7: Nutzwertanalyse - Routing von Nachrichten durch Workflow-Engine

Alle untersuchten ESBs unterstützen das Routing von Nachrichten durch eine Workflow-Engine. Deshalb erhält jeder ESB 10 von 10 möglichen Punkten. Der IBM WebSphere ESB verwendet dafür den WebSphere Process Server, JBoss und Mule die externe Komponente jBPM sowie Apache die interne Komponente ODE. Bei den Open Source ESBs besteht im Vergleich zum kommerziellen WebSphere ESB keine Bindung an einen bestimmten Hersteller.

Verschlüsselungsmechanismen

Art:	Wunschkriterium				
Gewichtung:	9				
IBM WebSphere	Mule		JBoss		Apache
	Community	Enterprise	Community	Enterprise	
10/10	10/10	10/10	10/10	10/10	10/10

Tabelle 8: Nutzwertanalyse - Verschlüsselungsmechanismen

Alle untersuchten ESBs unterstützen durch Sicherheitsmechanismen wie SSL und HTTPS die Verschlüsselung des Nachrichtenkanals. Zusätzlich wird zur Verschlüsselung der Nachricht an sich, der WS-Security Standard umgesetzt. Damit erfüllen alle ESBs in vollem Umfang die Anforderungen des Kooperationspartners und erhalten deshalb die maximale Punktzahl im Scoring-Modell.

Transaktionskonzept (ACID)

Art:	Wunschkriterium				
Gewichtung:	8				
IBM WebSphere	Mule		JBoss		Apache
	Community	Enterprise	Community	Enterprise	
10/10	10/10	10/10	10/10	10/10	5/10

Tabelle 9: Nutzwertanalyse - Transaktionsprinzip (ACID)

Der IBM WebSphere ESB und der JBoss ESB verwenden WS-Atomic Transactions, wohingegen der Mule ESB einen eigenen Transaktionsmanager besitzt, um die ACID-Eigenschaften sicherzustellen. Der Apache ServiceMix gewährt laut der Dokumentation nicht immer alle Eigenschaften des Transaktionskonzepts. Deshalb hat sich das Forschungsteam dafür entschieden, in diesem Fall nur 5 von 10 Punkten zu vergeben.

Authentifizierung, Autorisierung

Art:	Wunschkriterium				
Gewichtung:	4				
IBM WebSphere	Mule		JBoss		Apache
	Community	Enterprise	Community	Enterprise	
7/10	9/10	10/10	8/10	9/10	8/10

Tabelle 10: Nutzwertanalyse - Authentifizierung, Autorisierung

Alle untersuchten ESBs können auf das LDAP-Protokoll zur Authentifizierung und Autorisierung zurückgreifen. Zur Authentifizierung der Clients unterstützen alle untersuchten ESBs zudem den Standard WS-Security. Der Mule ESB unterstützt zudem PGP Security und Acegi. Darüber hinaus ist in der Enterprise-Version ein zusätzlicher Sicherheitsaspekt durch die Bereitstellung zusätzlicher sechs Tools gegeben. Sowohl in diesem ESB als auch im JBoss ESB ist JAAS als zusätzlicher Sicherheitsmechanismus integriert. Als Besonderheit ist zu beachten, dass bei der JBoss Enterprise-Version, im Gegensatz zur Community Version, zertifizierte Sicherheitsupdates des Herstellers zur Verfügung gestellt werden, was zu einer unterschiedliche Bewertung der zwei JBoss Versionen führt. Im Endergebnis schneidet die Enterprise-Version des Mule ESB mit der maximalen Punktzahl am besten ab.

Skalierbarkeit

Art:	Musskriterium		
Gewichtung:	-		
IBM WebSphere	Mule	JBoss	Apache
✓	✓	✓	✓

Tabelle 11: Nutzwertanalyse - Skalierbarkeit

Das Kriterium der Skalierbarkeit wird von allen ESBs erfüllt. Der IBM WebSphere ESB sowie der JBoss ESB lassen sich auf einem Server, auf mehreren verteilten Systemen oder auch in einer Cluster-Architektur einrichten. Bei Mule ESB können für die Skalierung mehrere Netzwerktopologien verwendet und ein Cluster aufgesetzt werden. Der Apache ServiceMix bietet ebenfalls Clustering Möglichkeiten. Es handelt sich hierbei um Herstellerangaben, neutrale Quellen waren nicht vorhanden. Eigene Tests zur Skalierbarkeit konnten nicht durchgeführt werden. Auf Grundlage dieser Ergebnisse erfüllen alle Produkte die Anforderungen vollumfänglich.

Monitoring (Logging)

Art:	Musskriterium				
Gewichtung:	-				
IBM WebSphere	Mule		JBoss		Apache
	Community	Enterprise	Community	Enterprise	
✓	X	✓	✓	✓	✓

Tabelle 12: Nutzwertanalyse - Monitoring

Bis auf die Community-Version von Mule ESB wird das Kriterium des Monitoring von allen untersuchten ESBs erfüllt. In der Enterprise-Version des Mule ESB existieren unter anderem die Tools Monitoring Framework Integration, Runtime Performance Manager und Performance KPI Tracking und ein Operational Dashboard. Der IBM WebSphere ESB nutzt dazu die Servicekomponente Common Event Infrastructure (CEI). JBoss stellt das Open Source Monitoring- und Managementwerkzeug Operations Network zur Verfügung. Beim Apache-Service Mix wird das Monitoring über die Komponente Karaf gewährleistet. Daher bleibt festzuhalten, dass diese Anforderung, mit Ausnahme der Mule ESB Community Version, von allen Produkten erfüllt wird.

Umsetzung von Web Service Standards

Art:	Musskriterium				
Gewichtung:	-				
IBM WebSphere	Mule		JBoss		Apache
	Community	Enterprise	Community	Enterprise	
✓	✓	✓	✓	✓	✓

Tabelle 13: Nutzwertanalyse - Umsetzung von Web Service Standards

Alle untersuchten ESBs unterstützen Web-Service Standards wie SOAP, WSDL und WS-* und erfüllen damit das Musskriterium.

Realisierung von Workflows mit BPEL

Art:	Wunschkriterium				
Gewichtung:	3				
IBM WebSphere	Mule		JBoss		Apache
	Community	Enterprise	Community	Enterprise	
10/10	10/10	10/10	10/10	10/10	10/10

Tabelle 14: Nutzwertanalyse - Realisierung von Workflows mit BPEL

Da alle untersuchten Tools BPEL zur Realisierung von Workflows anbieten, erhalten alle ESBs die maximale Bewertung.

Selbstständige Verwaltung einer Registry

Art:	Wunschkriterium				
Gewichtung:	2				
IBM WebSphere	Mule		JBoss		Apache
	Community	Enterprise	Community	Enterprise	
10/10	10/10	10/10	10/10	10/10	0/10

Tabelle 15: Nutzwertanalyse - Selbstständige Verwaltung einer Registry

Abgesehen vom Apache ServiceMix bieten alle Produkte eine eigene Service Registry an. Diese wird beim Mule ESB mittels einer eigenen RMI-Registry realisiert, beim IBM WebSphere über eine eigene Komponente namens WebSphere Service Registry and Repository (WSRR) und beim JBoss ESB über eine jUDDI Implementierung. Daher erhalten alle ESBs, ausgenommen der Apache ServiceMix, die maximale Punktzahl.

Eigenentwicklung von Adaptern möglich

Art:	Wunschkriterium				
Gewichtung:	1				
IBM WebSphere	Mule		JBoss		Apache
	Community	Enterprise	Community	Enterprise	
0/10	10/10	10/10	10/10	10/10	10/10

Tabelle 16: Nutzwertanalyse - Eigenentwicklung von Adaptern

Durch die zusätzliche Komponente WebSphere Adapter Toolkit können im WebSphere ESB eigene Adapter entwickelt werden. Da dieser im Standardfunktionsumfang nicht enthalten ist, erhält der IBM WebSphere ESB bei dieser Anforderung keine Punkte. Die anderen Tools erfüllen diese Funktion und weisen daher die Höchstpunktzahl auf. Insbesondere beim Mule ESB ist zu nennen, dass dem Kunden zur Eigenentwicklung genügend Informationen und eine Vielzahl an Beispielen zur Verfügung stehen. Dies geht aus der eigenen Erfahrung des Forschungsteams bei der Erstellung des Prototypen hervor.

4.4.1.2 Nicht-Funktionale Anforderungen

Support (Entwicklercommunity, Forum, Dokumentation, FAQ)

Art:	Wunschkriterium				
Gewichtung:	10				
IBM WebSphere	Mule		JBoss		Apache
	Community	Enterprise	Community	Enterprise	
10/10	7/10	9/10	7/10	9/10	8/10

Tabelle 17: Nutzwertanalyse - Support

Bei den Community-Versionen der Open Source Produkte wird insbesondere durch die Entwicklercommunity darauf geachtet, einen Support durch Foren, Wikis und anderweitigen Arten der Kommunikation im Netz zu gewährleisten. Mule ESB überzeugt durch die größte Community und erhält daher in der Community-Version 7 Punkte. Der JBoss ESB und Apache ServiceMix bieten dagegen eine Mailing-Liste und gleichen somit die etwas geringere Größe der Community aus. Bei den Enterprise-Versionen des Mule und JBoss ESB ist ein direkter 24/7 Support des Herstellers im Leistungsumfang enthalten. Beim Apache Service-

Mix ist es möglich zusätzlich Support von einem externen Dienstleister einzukaufen. Das kommerzielle Produkt bietet in allen Bereichen professionellen Support, angefangen von einem ausführlichen Produkthandbuch, über Multimedia-Tutorials bis hin zum Direktsupport. Deshalb bekommt der IBM WebSphere ESB im Gegensatz zu den Open Source Produkten die volle Punktzahl.

Referenzen

Art:	Wunschkriterium				
Gewichtung:	8				
IBM WebSphere	Mule		JBoss		Apache
	Community	Enterprise	Community	Enterprise	
-/10	-/10	-/10	-/10	-/10	-/10

Tabelle 18: Nutzwertanalyse - Referenzen

Eine Bewertung der Referenzen ist aus zweierlei Gründen nicht möglich: Zum einen lassen sich beim IBM-WebSphere ESB aufgrund der mangelnden Informationslage keine Kundenreferenzen ausfindig machen. Zum anderen ist nicht klar differenzierbar auf welches Produkt - Community oder Enterprise-Version von Mule ESB und JBoss ESB - sich die Referenzen beziehen.

So lässt sich für das kommerzielle Produkt einzig die HypoVereinsbank als Referenz nennen. Bei Mule und JBoss hingegen kann eine Vielzahl an Referenzen herangezogen werden. Unter anderem gehören Facebook, Barclays, Forbes, Sky, BWM, MasterCard, Nestle und Olympus zu den Nutzern des Mule ESB. JBoss nennt auf seiner Webseite Priceline.com, GEICO, NYSE Euronext und McKesson als seine Kunden. Bei Apache ServiceMix verweist lediglich der externe Dienstleister Savior Technologie Inc. auf Referenzen. Darunter befinden sich Unternehmen wie Progress Software, das National Center for Atmospheric Research (NCAR) und GE – Healthcare.

Bedienbarkeit, Einarbeitungszeit

Art:	Wunschkriterium				
Gewichtung:	7				
IBM WebSphere	Mule		JBoss		Apache
	Community	Enterprise	Community	Enterprise	
10/10	10/10	10/10	8/10	8/10	5/10

Tabelle 19: Nutzwertanalyse - Bedienbarkeit, Einarbeitungszeit

Für den IBM WebSphere ESB wird zur Modellierung von Mediationsmodulen und -abläufen ein grafischer, eclipsebasierter Editor namens WebSphere Integration Developer eingesetzt. Analog dazu bietet Mule ESB eine grafische Oberfläche zur Modellierung. JBoss ESB stellt ein eigenes Prozessmodellierungstool namens JBoss jBPM. Zusätzlich kann zur Entwicklung und zu Testzwecken das JBoss Developer Studio verwendet werden. Apache ServiceMix verfügt hingegen über keine grafische Benutzeroberfläche. Es ist jedoch möglich ein zusätzliches Eclipse-Plugin zur Komponentenprogrammierung zu integrieren. Sowohl das JBoss Developer Studio als auch das angesprochene Eclipse-Plugin tragen zur komfortableren Bedienbarkeit bei, erfordern jedoch zunächst eine explizite Implementierung in die jeweilige ESB-Lösung. Die Steuerung dieser beiden Produkte erfolgt größtenteils über Konsoleneingaben, weshalb ein angemessener Punktabzug vorgenommen wurde.

Grundsätzlich sind diese Kriterien in einer Forschungsarbeit mit den Rahmenbedingungen des KOS-Projektes nur schwer zu beurteilen, da die Möglichkeiten der praktischen Anwendung stark begrenzt sind. Die Bewertung erfolgt vorwiegend über eine Einschätzung der Bedienbarkeit.

4.4.1.3 Gesamtergebnis der Nutzwertanalyse

IBM WebSphere	Mule		JBoss		Apache
	Community	Enterprise	Community	Enterprise	
83,3%	**81,7%**	**85,6%**	**78,7%**	**82,7%**	**66,4%**

Tabelle 20: Nutzwertanalyse - Gesamtergebnis

Die Gesamtbetrachtung des erstellten Scoring-Modells (siehe Anhang 5) zeigt, dass lediglich der IBM WebSphere ESB alle definierten Anforderungen erfüllen konnte. Das Musskriterium

Routingfähigkeit, bei dem insbesondere das inhaltsbasierte und versionsgerechte Routing im Vordergrund stehen, konnte von keinem der untersuchten Open Source Produkte erfüllt werden. Zu beachten ist jedoch, die Open Source Tools grundsätzlich die Routingfähigkeit unterstützen, denn alle Produkte bieten zumindest das inhaltsbasierte Routing an. Kann aus Sicht des Kooperationspartners von dieser Tatsache abgesehen und weiterhin vernachlässigt werden, dass die Monitoringfähigkeit - als ein weiteres Musskriterium - bei der Community-Version des Mule ESB nicht gegeben ist, geht aus dem Gesamtergebnis der durchgeführten Nutzwertanalyse die Enterprise-Version des Mule ESB mit einer Gesamtpunktzahl von 85,6 Punkten als Beste der untersuchten ESB-Lösungen hervor. Selbst das kommerzielle Tool von IBM, der WebSphere ESB, kann mit der kostenpflichtigen Open Source ESB-Version von Mule nicht mithalten und landet mit 83,3 Punkten auf dem zweiten Platz. Den dritten Platz belegt die Enterprise-Version des JBoss ESBs mit insgesamt 82,7 Punkten. Auf den weiteren Plätzen folgen die Community-Versionen des Mule ESBs (81,7 Punkte) sowie des JBoss ESBs (78,7 Punkte). Das Schlusslicht bildet mit deutlichem Abstand der Apache ServiceMix mit insgesamt 66,4 erreichten Punkten. Das Gesamtergebnis wird durch die folgende Grafik visualisiert:

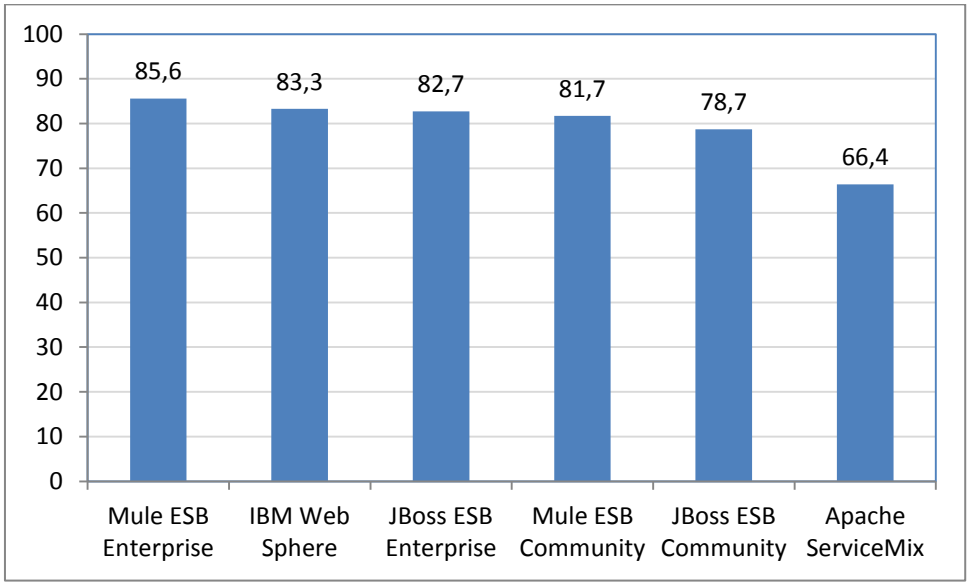


Abbildung 14: Gesamtergebnis der Marktanalyse

4.4.2 Hindernisse bei der Bewertung

Anhand der Nutzwertanalyse aus dem vorangegangenen Kapitel sind die funktionalen und nicht-funktionalen Kriterien gegenübergestellt und hauptsächlich auf Basis der Fakten aus den Datenblättern der Anbieter bewertet worden. Da es sich dabei um keine objektiven Quel-

len handelt und darüber hinaus auch weiterhin kaum neutrale Quellen ausfindig zu machen gewesen sind, ist allein die Beurteilung der genannten Fakten zur Schlussbewertung der Tools nicht ausreichend. Die hier dargestellten Performanceangaben oder beispielsweise Hinweise zur Skalierbarkeit sind aus Sicht der Anbieterangaben gefärbt und stellen damit eine einseitige Sichtweise dar, die aus dem Marketinggedanken heraus als zu positiv interpretiert werden können. Um jedoch konkretere Aussagen treffen zu können, ist über das KOS Projekt hinaus ein Lasttest durchzuführen. Ein denkbares Werkzeug hierfür ist JMeter, ein freies, in Java geschriebenes Werkzeug zur Ausführung von Lasttests.

Bei der Bewertung der nicht-funktionalen Kriterien ist die Quelle beim kommerziellen IBM WebSphere ebenfalls hauptsächlich der Hersteller und damit nicht unmittelbar überprüfbar. Bezüglich der Open Source ESBs hat die jeweilige Community vielseitigere Aussagen getroffen, welche zur Ergebnisbewertung geführt haben. Trotz diesem Aspekt bietet ein weitergehender Lasttest auch hier die Möglichkeit, insbesondere die Kriterien wie beispielsweise Support, Bedienbarkeit und Einarbeitungszeit im Umfeld des Unternehmens genauer zu bewerten.

Hinsichtlich der anfallenden Gesamtkosten ist dies für die Betrachtung nur schwer ein ausschlaggebendes Unterscheidungskriterium. Da der Kunde bereits IBM Produkte nutzt sind abweichende Preise zu den genannten denkbar. Aus diesem Grund ist der Verweis auf den Basispreis nicht alleinig ausschlaggebend, da eventuelle Rabatte berücksichtigt werden müssen. Darüber hinaus sind Kosten zu den Open Source Tools kaum öffentlich zugänglich. Allein über JBoss sind konkrete Preise bekannt, welche in Kapitel 4.2.1 aufgeführt werden. Bei Mule konnten selbst nach mehrfachem E-Mail Schriftwechsel mit einem zuständigen Ansprechpartner keine Preise in Erfahrung gebracht werden (Anhang 5). Obwohl sowohl die Rahmenbedingungen des KOS Projektes, als auch das Konzept der Dualen Hochschule erläutert worden sind, war Mule nicht bereit, Angaben über anfallende Kosten zu machen. Aus den genannten Schwierigkeiten ist keine detaillierte Kostenkalkulation möglich.

Insgesamt lässt sich anmerken, dass aufgrund der allgemein sehr undurchsichtigen und nicht neutralen Angaben zu den Tools die Kriterienbewertung erschwert worden ist.

Unabhängig der konkreten Bewertung anhand der Kriterien aus dem Scoring-Modell sollte die strategische Ausrichtung des Unternehmens bei der Wahl eines Tools bedacht werden. Fällt die Wahl auf den kommerziellen WebSphere von IBM, wäre hiermit eine Abhängigkeit zu dem Großunternehmen IBM gegeben. Der Kooperationspartner würde hierbei das Konzept des One-Stop-Shoppings befolgen, das darauf basiert, dass ein Unternehmen von nur einem Unternehmen Produkte für mehrere Funktionseinheiten einkauft. Vorteile sind darin zu sehen, dass der Kunde sich auf eine Anlaufstelle konzentrieren kann, was sich meist sowohl beim Support als auch bei Preisverhandlungen positiv auswirkt. Eine Alternative stellt die

Verwendung mehrerer Systeme von jeweils anderen Anbietern dar, was unter dem Begriff Best-Of-Breed zu verstehen ist. Dies wäre der Fall, wenn die Auswahl auf einen Open Source ESB fallen würde. Nachteilig kann sich hierbei der Support auswirken, da sich die Zuordnung und das Nachweisen des Systemfehlers, in Bezug auf das System eines bestimmten Anbieters, in der Praxis häufig als zeitaufwendig und schwierig herausstellt. Dies ist damit zu begründen, dass die Anbieter den Fehler nicht sofort ohne weiteres in ihrem System anerkennen. Ein großer Vorteil ist jedoch, dass bei diesem Ansatz auf die jeweiligen Spezialisten auf dem Markt zurückgegriffen werden kann. Diesbezüglich kann im Rahmen des KOS Projektes jedoch keine eindeutige Vorgehensweise bestimmt werden, da der Kooperationspartner dies in Bezug auf seine Unternehmensstrategie ausrichten muss. Im Folgenden Kapitel wird das Forschungsteam jedoch aus seiner Sicht eine Empfehlung geben, welche Vorgehensweise es für den Kunden als vorteilhaft ansieht.

4.4.3 Empfehlung

Bevor eine Empfehlung bezüglich des Einsatzes eines bestimmten ESB Produktes abgegeben werden kann, sind die folgenden weiterführenden Punkte zu berücksichtigen:

- Die Unabdingbarkeit des versionsgerechten Routings sollte erneut kritisch reflektiert werden. Sollte diese Reflektion zum Ergebnis führen, dass das die Bedeutung des versionsgerechten Routing relativiert werden kann, so ist der Mule ESB in der Enterprise Version zu empfehlen. Anderenfalls ist ein Verbleib beim IBM WebSphere ESB anzuraten.

- Die strategische Ausrichtung des Unternehmens im Bereich der IT-Integration sollte bei der Entscheidungsfindung bedacht werden. Wenn die strategische Ausrichtung es vorsieht auf einen Hersteller bei dem Einsatz der Integrationsplattform zurückzugreifen, ist der kommerzielle WebSphere ESB von IBM zu empfehlen. Hiermit ist eine Abhängigkeit mit allen Vor- und Nachteilen zu dem Großunternehmen IBM gegeben. Der Kooperationspartner profitiert in diesem Fall vom Konzept des One-Stop-Shoppings. Dieses basiert darauf, dass ein Unternehmen mehrere Hard- und Softwareprodukte von nur einem Hersteller bezieht. Vorteile sind darin zu sehen, dass der Kunde sich auf eine Anlaufstelle konzentrieren kann. Dies kann sich sowohl beim Support als auch bei Preisverhandlungen positiv auswirken.

  Eine Alternative stellt das Konzept Best-Of-Breed dar, unter welchem die Verwendung von Systemen unterschiedlicher Anbieter zu verstehen ist. Wird dieser Ansatz verfolgt, profitiert der Kooperationspartner von erhöhter Flexibilität und der Möglichkeit sich in einem bestimmten Einsatzgebiet für den jeweiligen Marktspezialisten zu

entscheiden. Nachteilig kann sich hierbei der Support auswirken, da sich die Zuordnung und das Nachweisen eines Fehlers zu einem bestimmten System - und damit einem Anbieter - in der Praxis häufig als zeitaufwendig und schwierig herausstellt. Ob die beschriebene Strategie verfolgt werden kann, ist von der Entscheidung abhängig, wie der Kooperationspartner die Bedeutung des versionsgerechten Routings einschätzt. Wird dieses Kriterium weiterhin als Musskriterium eingestuft, ist die Umsetzung Best-Of-Breed Konzeptes nicht möglich.

Aufgrund der genannten Punkte kann, ohne eine Entscheidung des Kunden, keine eindeutige Handlungsempfehlung ausgesprochen werden. Um dennoch eine Produktempfehlung abzugeben, wird im Rahmen des Forschungsprojektes KOS angenommen, dass das versionsgerechte Routing weiterhin als Musskriterium eingestuft wird und der Kooperationspartner das Konzept des One-Stop-Shoppings verfolgt. Hinsichtlich der getroffenen Annahmen wird vom Forschungsteam der IBM WebSphere ESB empfohlen.

4.5 Erstellung eines Prototyps

Ein Prototyp dient der vorab Erprobung gängiger Funktionen, welche im Betriebsablauf zum Einsatz kommen. Aufgrund der genannten Kriterien in Kapitel 2.2.1 und des erreichten Ergebnisses in der Nutzwertanalyse, ist die Entscheidung auf den Mule ESB gefallen. In Zusammenarbeit mit dem Kooperationspartner und unter der Priorisierung der Anforderungskriterien wurde ein möglicher Anwendungsfall besprochen, welcher exemplarisch alltagstypische Funktionen integriert. Auf dieser Basis ist der folgende Prototyp mit Mule ESB umgesetzt worden.

Der Anwendungsfall sieht vor, eine CSV-Datei mit Personentestdaten so zu verarbeiten, dass am Ende des Workflows pro Personendatensatz eine XML-Datei erstellt und in einen Zielordner abgelegt wird. Der Aufbau der CSV-Datei aus dem Testfall gestaltet sich folgendermaßen:

```
Adam,Mustermann,01.01.1960,Architekt
Eva,Musterfrau,31.01.1970,Zahnarzthelferin
```

Zunächst wird, wie in der Entwicklungsumgebung Eclipse, ein Mule-Projekt erstellt. Die Projektestruktur besteht zunächst grundsätzlich aus der Java-Standardbibliothek einer Mule-spezifischen Bibliothek, Quellordner für selbstentwickelte Komponenten oder Transformatoren sowie dem Flow, der umgesetzt werden soll.

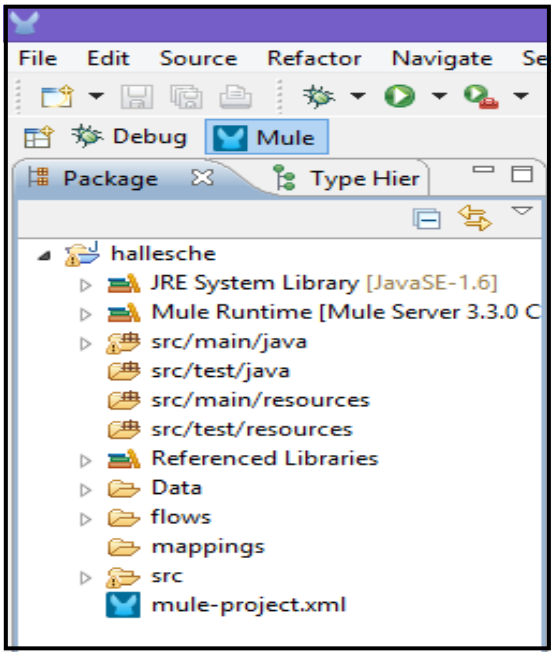


Abbildung 15: Projektordnerstruktur des Prototyps

Anschließend wird mit Hilfe des integrierten grafischen Editors der Workflow erstellt. Dazu werden die benötigten Endpunkte, Transformatoren, Filter oder andere Komponenten einfach per Drag-and-Drop in den Flow gezogen und mittels Assistenten konfiguriert. Die folgende Abbildung zeigt den Konfigurationsassistenten der im Workflow eingebauten Splitterkomponente:

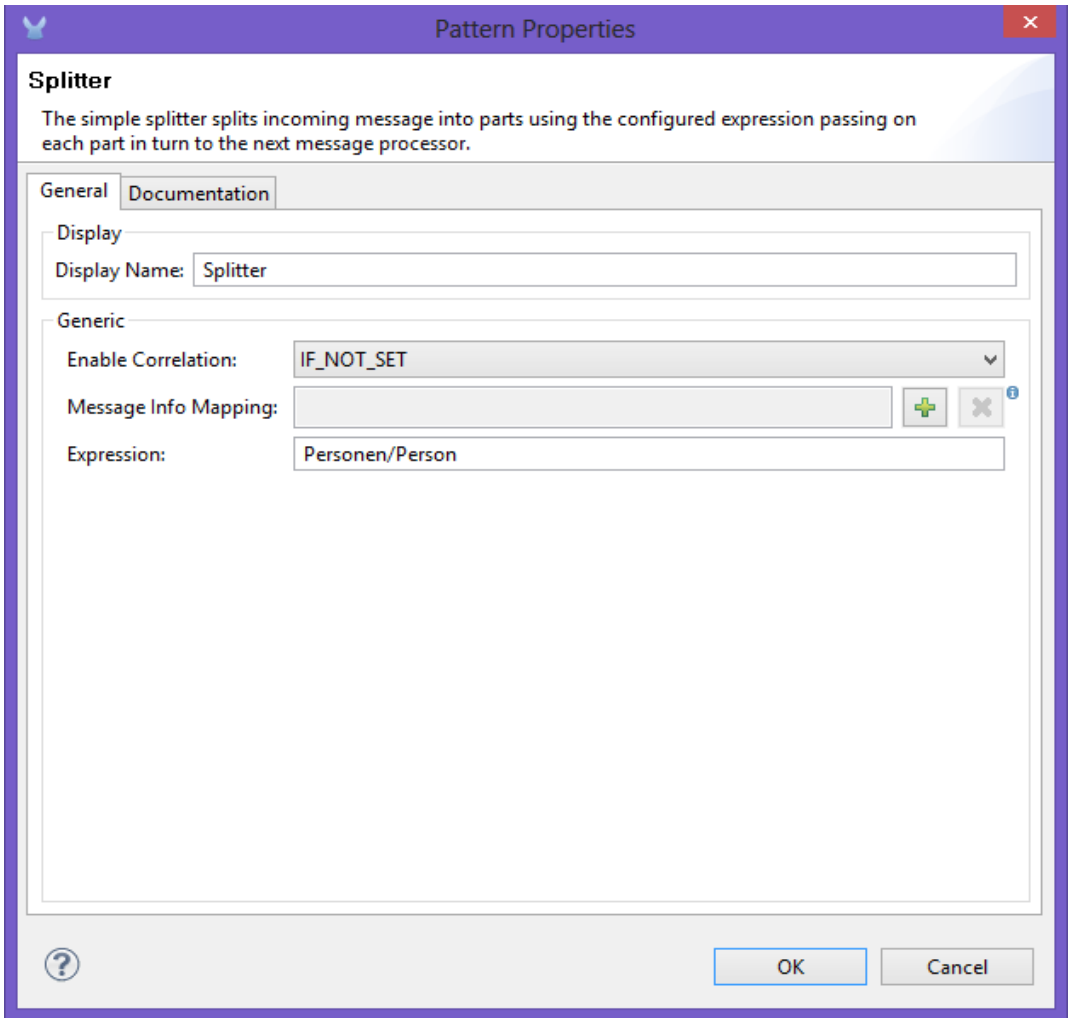


Abbildung 16: Konfigurationsassistent der Splitterkomponente in Mule

Wie in der Abbildung 17 zu erkennen ist, wird zunächst die CSV-Datei aus dem Ordner *InCSV* entnommen und von dem selbsterstellten Transformator *CSVStream2XML Transformer* entgegengenommen. Dieser verarbeitet den eintreffenden Datenstrom, generiert daraus eine XML-Datei und schickt einen Datenstrom aus dieser Datei an den Splitter. Der Splitter ist eine Standardkomponente des Mule ESB und generiert aus der erhaltenen XML-Datei anhand eines XPath-Ausdruckes pro Personendatensatz einen Datenstrom. Dieser Datenstrom wird von einem weiteren selbst erstellten Transformer namens *XMLStream2FileStreamTransformer* entgegengenommen, erneut in eine XML-Datei trans-

formiert und ein FileInputStream auf diese Datei an den Zielordner übergeben, der diese mittels Output Pattern als XML-Datei ablegt.

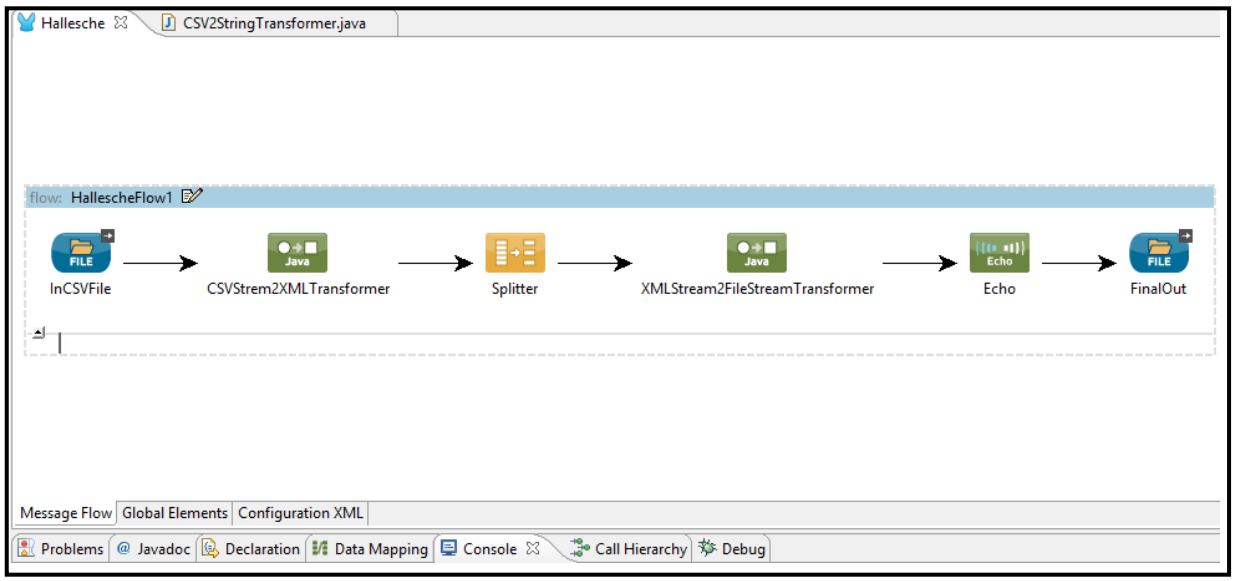


Abbildung 17: Workflow des Testfalls des Prototyps

Als Endergebnis werden im Ordner *FinalOut* folgende zwei XML-Dateien abgelegt:

```
<?xml version="1.0" encoding="UTF-8"?>
<Person>
	<Name>
		<Vorname>Adam</Vorname>
		<Nachname>Mustermann</Nachname>
	</Name>
	<Geburtsdatum>01.01.1960</Geburtsdatum>
	<Beruf>Architekt</Beruf>
</Person>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Person>
	<Name>
		<Vorname>Eva</Vorname>
		<Nachname>Musterfrau</Nachname>
	</Name>
	<Geburtsdatum>31.01.1970</Geburtsdatum>
	<Beruf>Zahnarzthelferin</Beruf>
</Person>
```

Abbildung 18: Aufbau der abgelegten XML-Dateien im Protoyp

5 Fazit

Das Ziel des Forschungsprojektes war es, auf Basis einer durchgeführten Marktanalyse, eine Empfehlung an den Kooperationspartner auszusprechen, inwiefern die elaborierten Open Source ESB Produkte eine adäquate Alternative zum kommerziellen IBM WebSphere ESB darstellen.

Zu diesem Zweck wurden mit den Begriffen Open Source, serviceorientierte Architektur und Enterprise Service Bus zunächst die theoretischen Grundlagen geschaffen. Anhand der recherchierten Aufgaben eines ESBs, wurden in Zusammenarbeit mit dem Kooperationspartner Anforderungen definiert und gewichtet, die das Fundament einer Nutzwertanalyse in Form eines Scoring-Modells bilden.

In dieser Nutzwertanalyse wurden der kommerzielle IBM WebSphere ESB und die Open Source Tools JBoss ESB, Mule ESB und Apache ServiceMix evaluiert und miteinander verglichen.

Aus diesem Vergleich geht, streng nach den Anforderungen des Kooperationspartners, der IBM WebSphere ESB als bestgeeignetes Produkt hervor. Dies liegt in der Tatsache begründet, dass sämtliche untersuchten Open Source ESB Lösungen das Musskriterium Routingfähigkeit - aufgrund der fehlenden Funktionalität des versionsgerechten Routings - nicht erfüllen. Wird von diesem Kriterium abgesehen, schneidet hingegen die Enterprise Version des Mule ESB am besten ab.

Aus dem Ergebnis der Marktanalyse wurden für den Kooperationspartner folgende zwei Handlungsempfehlungen abgeleitet:

Zum einen wird empfohlen, die Wichtigkeit des versionsgerechten Routings neu zu bewerten. Zum anderen sollte die strategische Ausrichtung des Unternehmens bei der Wahl des ESB Produktes berücksichtigt werden.

Abschließend bleibt festzuhalten, dass Open Source Produkte im Bereich der Anwendungsintegration durchaus eine attraktive Alternative zu kommerziellen Lösungen darstellen.

Anhang

Anhangverzeichnis

Anhang 1: Checkliste der Aktivitäten

Nr.	Aktivität	Verantwortlichkeit	Zeitraum	zusätzliche Infos	Stand
0	Kick-Off	Team	20.11.	- Themenauswahl	offen
1	Gliederung	Team	20.11.	- Termin mit Hr. Deininger zur Absprache am 29.11.	offen
2	Recherche der Quellen	Team	28.11.	- Bibliotheksbesuch	offen
3	Erste Kundenklärungen	Team	29.11.	- Termin mit Hr. Deininger & Hr. Kassel	offen
4	Verfassen des Kapitels zum Projektmanagement	Ferro	über das ganze Projekt	- Rollenaufteilung - Einsatz welche Tools – Begründung - Aktivitäten-Checkliste/ Zeitplan - grafische Darstellung der Qualitätssicherung (Meilensteinergebnisse)	offen
5	Einleitung	Ferro	28.11.		offen
6	Grundlagen				
6.1	OpenSource	Ferro	05.12.		offen
6.2	SOA	Satelmair Sikora	28.11. 29.11.		offen
6.3	ESB	Satelmair	05.12.		offen
7	konunerzielle Lösung (IBM Websfears)	Kolarzik	28.11. 29.11. 05.12.		offen
8	Open Source Lösungen (JBoss, Mule, Apache)	Thies	28.11. 29.11. 05.12.		offen
9	Hallesche Anforderungen	Team	06.12. 10.01.	- [redacted]	offen
10	Gegenüberstellung – Bewertung	Team	03.01. 08.01. 09.01. 10.01.	- abhängig von Status erledigt Nr. 9 - [redacted]	offen
11	Prototyp (Mule)				
11.1	Einarbeitung Tool	Satelmair	06.12.		offen
11.2	Ablauf Umsetzung	Satelmair	12.12.- 16.01.	- [redacted]	offen
12	Fazit	Team	15.01.	- abhängig von Status erledigt Nr. 10	offen
13	Ergebnisse				
13.1	Druck	?	16.01.	- abhängig von Status erledigt Nr. 12	offen
13.2	Präs. Erstellung	Team	15.–16.01.	- abhängig von Status erledigt Nr. 12	offen
13.3	Präsentation halten	Team	22.01.	- abhängig von Status erledigt Nr. 12	offen
14	Puffer	Team	17.01.		

Anhang 2: 10 Kriterien, die Open Source Software erfüllen muss

1. **Freie Weitergabe**

Die Lizenz darf niemanden in seinem Recht einschränken, die Software als Teil eines Software-Paketes, das Programme unterschiedlichen Ursprungs enthält, zu verschenken oder zu verkaufen. Die Lizenz darf für den Fall eines solchen Verkaufs keine Lizenz- oder sonstigen Gebühren festschreiben.

2. **Quellcode**

Das Programm muss den Quellcode beinhalten. Die Weitergabe muss sowohl für den Quellcode als auch für die kompilierte Form zulässig sein. Wenn das Programm in irgendeiner Form ohne Quellcode weitergegeben wird, so muss es eine allgemein bekannte Möglichkeit geben, den Quellcode zum Selbstkostenpreis zu bekommen, vorzugsweise als gebührenfreien Download aus dem Internet. Der Quellcode soll die Form eines Programms sein, die ein Programmierer vorzugsweise bearbeitet. Absichtlich unverständlich geschriebener Quellcode ist daher nicht zulässig. Zwischenformen des Codes, so wie sie etwa ein Präprozessor oder ein Übersetzer („Translator“) erzeugt, sind unzulässig.

3. **Abgeleitete Software**

Die Lizenz muss Veränderungen und Derivate zulassen. Außerdem muss sie es zulassen, dass die solcherart entstandenen Programme unter denselben Lizenzenzbestimmungen weitervertrieben werden können wie die Ausgangssoftware.

4. **Unversehrtheit des Quellcodes des Autors**

Die Lizenz darf die Möglichkeit, den Quellcode in veränderter Form weiterzugeben, nur dann einschränken, wenn sie vorsieht, das zusammen mit dem Quellcode so genannte „Patch files“ weitergegeben werden dürfen, die den Programmcode bei der Kompilierung veränder. Die Lizenz muss die Weitergabe von Software, die aus verändertem Quellcode entstanden ist, ausdrücklich erlauben. Die Lizenz kann verlangen, dass die abgeleiteten Programme einen anderen Namen oder eine andere Versionsnummer als die Ausgangssoftware tragen.

5. **Keine Diskriminierung von Personen oder Gruppen**

Die Lizenz darf niemanden benachteiligen.

6. **Keine Einschränkungen bezüglich des Einsatzfeldes**

Die Lizenz darf niemanden daran hindern, das Programm in einem bestimmten Bereich einzusetzen. Beispielsweise darf sie den Einsatz des Programms in einem Geschäft oder in der Genforschung nicht ausschließen.

7. **Weitergabe der Lizenz**

Die Rechte an einem Programm müssen auf alle Personen übergehen, die diese Software erhalten, ohne dass dafür diese die Notwendigkeit bestünde, eine eigene, zusätzliche Lizenz auszuüben.

8. Die Lizenz darf nicht auf ein bestimmtes Produktpaket beschränkt sein

Die Rechte an dem Programm dürfen nicht davon abhängig sein, ob das Programm Teil eines bestimmten Software-Paketes ist. Wenn das Programm aus dem Paket herausgenommen und im Rahmen der zu diesem Programm gehörenden Lizenz benutzt oder weitergegeben wird, so sollen alle Personen, die dieses Programm dann erhalten, alle Rechte daran haben, die auch in Verbindung mit dem ursprünglichen Software-Paket gewährt wurden.

9. Die Lizenz darf die Weitergabe zusammen mit anderer Software nicht einschränken

Die Lizenz darf keine Einschränkungen enthalten bezüglich anderer Software, die zusammen mit der lizenzierten Software weitergegeben wird. So darf die Lizenz z.B. nicht verlangen, dass alle anderen Programme, die auf dem gleichen Medium weitergegeben werden, auch quelloffen sein müssen.

10. Die Lizenz muss technologieneutral sein

Die Lizenz darf keine individuelle Technologien, Software-Art oder Schnittstelle voraussetzen.[142]

[142] Vgl. Harhoff, D. u. a. (2004), S. 20 f.

Anhang 3: Unterschied der JBoss Community- und Enterprise-Version

Die folgende Tabelle stellt nochmals detaillierter die JBoss Enterprise und die Community Version gegenüber.

Features	JBoss Community	JBoss by Red Hat
Open Source	x	x
Benefits from testing by worldwide Community	x	x
Recommended for Production Use		x
Patch Update & Service Pack Program		x
Security Errata Program		x
Automated Software Update & Alert Service		x
Detect & Feature Escalation & Prioritization Process		x
Developer Support		x
24/7 Production Support & Services		x
Platform Certifications & Training Certifications		x
Defined Support SLA and End-of-Life Policy		x
Out-of-the-Box Configured for Enterprise Use		x
Operations Management Tools		x
Platform testing & certification process		x
Redistribution of modified JBoss technologies		x
Red Hat Open Source Assurance (Legal Protection)		x

Tabelle 21: Funktionenübersicht JBoss Community und Enterprise Version[143]

[143] Vgl. JBoss Community and Enterprise Features (2010)

Anhang 4: Unterschiede der Mule Community und Enterprise Version

Im Folgenden sind sämtliche Unterschiede zwischen der Community und der Enterprise Version des Mule ESBs aufgeführt:

	Mule ESB Community	Mule ESB Enterprise
Leading Open Source ESB	✓	✓
Enterprise hardening with back-ported bug fixes		✓
Access to source code	✓	✓
Mule Studio (graphical tooling)	✓	✓
Mule Studio DataMapper		✓

AnyPoint Connectivity	Mule ESB Community	Mule ESB Enterprise
Community Connectors	✓	✓
Native WebSphere MQ Connector		✓
Premium JDBC Connector		✓
Enterprise Gateway for SAP		✓

Security	Mule ESB Community	Mule ESB Enterprise
Mule Enterprise Security		✓

Reliability & Performance	Mule ESB Community	Mule ESB Enterprise
Self-Healing Connections	✓	✓
High Availability		✓
Caching		✓

Performance management	Mule ESB Community	Mule ESB Enterprise
SLA Alerts		✓
Monitoring Framework Integration		✓
Runtime Performance Manager		✓
Performance KPI Tracking		✓

Problem resolution	Mule ESB Community	Mule ESB Enterprise
Service Flow Analyzer		✓
Root Cause Analysis		✓
Bottleneck Detection		✓

Operational Control	Mule ESB Community	Mule ESB Enterprise
Operational Dashboard		✓
Deployment Manager		✓
Integrated Application Repository		✓
ESB Remote Control		✓
Task Scheduler		✓
REST API for Management		✓

Support & maintenance	Mule ESB Community	Mule ESB Enterprise
Community forums	✓	✓
24/7 Premium support		✓
Knowledge base		✓
Service packs & hot patches		✓

Abbildung 19: Unterschiede der Mule ESB Community- und Enterprise-Version[144]

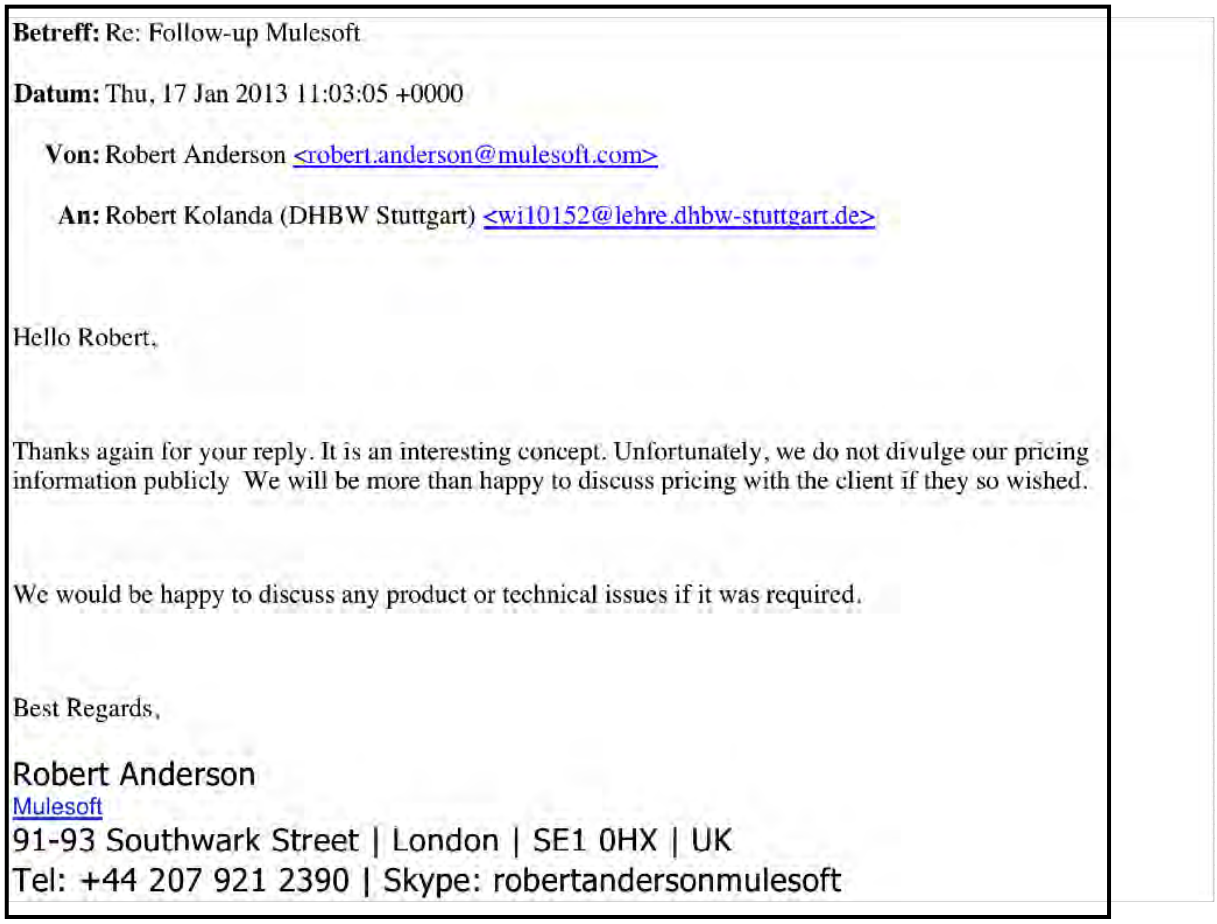
[144] Entnommen aus: MuleSoft (2012f)

Anhang 5: Scoring-Modell

Scoring-Modell ist als gesonderter Anhang in DIN A3 Format beigefügt.

Anhang 6: Aussage von Mulesoft über Kosten

Im Rahmen der Kostenrecherche konnten selbst nach **mehrmaligem** Schriftwechsel per E-Mail mit einem Vertreter von Mule keine Angaben in Erfahrung gebracht werden.



Betreff: Re: Follow-up Mulesoft

Datum: Thu, 17 Jan 2013 11:03:05 +0000

Von: Robert Anderson <robert.anderson@mulesoft.com>

An: Robert Kolanda (DHBW Stuttgart) <wi10152@lehre.dhbw-stuttgart.de>

Hello Robert,

Thanks again for your reply. It is an interesting concept. Unfortunately, we do not divulge our pricing information publicly We will be more than happy to discuss pricing with the client if they so wished.

We would be happy to discuss any product or technical issues if it was required.

Best Regards,

Robert Anderson
Mulesoft
91-93 Southwark Street | London | SE1 0HX | UK
Tel: +44 207 921 2390 | Skype: robertandersonmulesoft

Quellenverzeichnisse

Literaturverzeichnis

Braehmer, U. (2005): Projektmanagement für kleine und mittlere Unter- nehmen – Schnelle Resultate mit knappen Ressourcen; 1. Auflage; München: Carl Hanser Verlag

Chappell, D. A. (2004): Enterprise Service Bus; 1. Auflage; Sebastoplo: O'Reilly Media Inc.

Dangelmaier, W. u. a. (2002): Klassifikation von EAI-Systemen; in: Praxis der Wirtschaftsinformatik; 2002; Nr. 06/2002; S. 61-71

Davis, J. (2009): Open Source SOA; 1. Auflage; Greenwich: Manning Verlag

Dostal, W. u. a. (2005): Service-orientierte Architekturen mit Web Services; 1. Auflage; München: Spektrum Akademischer Verlag

Gramm, A. (2007): Service orientierte Architektur (SOA); 1. Auflage; Norderstedt: GRIN Verlag

Harhoff, D. u. a. (2004): Open-Source-Software, Eine ökonomische und technische Analyse; 1. Auflage; München: Springer-Verlag

Hiekel, S. (2007): Diplomarbeit, Bedeutung und Qualitätseigenschaften des Enterprise Service Bus im Kontext von serviceorientierten Architekturen; Magdeburg

Kischel, S. (2003): Der Enterprise Service Bus, Implementierung einer service-orientierten Architektur; in: OBJEKTSpektrum; 2003; Nr. 2/2003; S. 37-40

Lorenzelli-Scholz, D. (2005): Service-orientierte Integration mit einem „Enterprise Service Bus"; in: OBJEKTSpektrum; 2005; Nr. 6/2005; S. 18-22

Melzer, I. u. a. (2010): Service- orientierte Architekturen mit Web Services, Konzepte – Standards – Praxis; 4. Auflage; Heidelberg: Springer Verlag

Münz, S. (2008): <Webseiten professionell erstellen>, Programmierung, Design und Administration von Webseiten; 3. Auflage; München: Addison-Wesley Verlag

Scheiner, D. (2010): Herausforderungen bei der Einführung serviceorientierte Architekturen; 1. Auflage; Norderstedt: GRIN Verlag

Thiel, T. (2007): Design and Implementation of a Service-oriented Information System Architecture Based on a Case Study; 1. Auflage; München: GRIN Verlag

Younes, Y. (2010): Analyse und Konzept zur Integration eines Enterprise Service Bus in elastischen Infrastructure as a Service Umgebungen; 1. Auflage; Norderstedt: GRIN Verlag

Zalewski, S. (2009): Open Source - Der Weg in das Unternehmen; 1. Auflage; Norderstedt: GRIN Verlag

Zeppenfeld, K. / Finger, P. (2009): SOA und WebServices; Informatik und Fokus (Hrsg.: Günther, O.; Karl, W.; Lienhart, R.; Zeppenfeld, H.); 1. Auflage; Heidelberg: Springer Verlag

Verzeichnis der Internetquellen

Apache Software Foundation (2011a):	Apache ServiceMix; http://servicemix.apache.org/; Abruf 03.01.2012
Apache Software Foundation (2011b):	Apache ActiveMQ; http://activemq.apache.org/; Abruf 03.01.2013
Apache Software Foundation (2011c):	Apache Camel; http://camel.apache.org/; Abruf 03.01.2013
Apache Software Foundation (2011d):	Apache Karaf; http://karaf.apache.org/; Abruf 03.01.2013
Apache Software Foundation (2012):	The Apache Software Foundation; http://www.apache.org/; Abruf: 09.01.2013
Apache Software Foundation (2013a):	Apache ODE; http://ode.apache.org/index.html; Abruf: 11.01.2013
Apache Software Foundation (2013b):	Apache CFX; http://cxf.apache.org/index.html; Abruf: 11.01.2013
Apache Software Foundation (2013c):	X-Integrate; http://www.x-integrate.com/x-in-cms.nsf/id/home-de; Abruf: 03.01.2013
Apache Software Foundation (2013d):	OSGi Alliance, Technology; http://www.osgi.org/Technology/HomePage; Abruf: 14.01.2013
Bosanac, D. (2010):	Apache ActiveMQ – Enterprise messaging in action,;

http://de.slideshare.net/dejanb/apache-activemq-enterprise-messaging-in-action;
Abruf: 02.01.2013

Buchholz, A. / Hohloch, R. / Rathgeber, T. (2007): Vergleich von Open Source ESBs; elib.uni-stuttgart.de/opus/volltexte/2007/3376/pdf/FACH_0081.pdf; Abruf: 08.01.2013

Chappell, D. (2005): Using the ESB Service Container; http://onjava.com/pub/a/onjava/excerpt/esb_ch6/; Abruf: 02.12.2012

Commercial IT (o.J.): Serviceorientierte Architektur (SOA); http://www.commercial-it.de/dokumentation/glossar/Allgemein/SOA_Serviceorientierte_Architektur_.htm; Abruf: 06.12.2012

De Angelis, R. (2009): Fuse ESB das bessere ServiceMix?; http://www.soa-bpm-integration.com/2009-08-10-fuse-esb-bessere-servicemix; Abruf: 08.01.2013

Diller, J., Viola, J. (2011): JBoss ESB - Zuverlässiger Busverkehr; http://www.objectcode.de/fileadmin/user_upload/Presse___PR/JBossESB_SOA.pdf; Abruf: 13.12.2012

Enzyklopädie der Wirtschaftsinformatik (2012): Architekturparadigmen; http://www.enzyklopaedie-der-wirtschaftsinformatik.de/wi-enzyklopaedie/lexikon/is-management/Systementwicklung/Softwarearchitektur/Architekturparadigmen; Abruf: 05.12.2012

Fink, T. (2009): Die SOA-Plattform von JBoss; http://www.sigs.de/publications/js/2009/01/fink_JS_01_09.pdf;

Abruf: 10.12.2012

Golem (o. J.): Web Service;
http://www.golem.de/specials/webservice/;
Abruf: 08.01.2013

Hanson, J. J. (2005): ServiceMix as an enterprise service bus - Use ServiceMix 2.0 as a service-oriented message routing framework;
http://www.javaworld.com/javaworld/jw-12-2005/jw-1212-esb.html?page=3;
Abruf: 02.01.2013

IBM developerWorks (2012): IBM WebSphere Developer Technical Journal;
http://www.ibm.com/developerworks/websphere/techjournal/0602_tost/0602_tost.html;
Abruf: 02.01.2013

IBM Software Prices (2013): IBM Software Preise;
https://www-112.ibm.com/software/howtobuy/buyingtools/paexpress/Express?P0=E1&part_number=D58AKLL,D58AELL,D06UVLL,D04Q1LL,D04Q3LL,D0BA1LL,D0B3HLL,D0B5HLL,D0B2XLL,D0GALLL,D0GB0LL&catalogLocale=de_DE&Locale=de_DE&country=DEU&PT=html;
Abruf: 05.12.2012

IBM WebSphere Adapter Toolkit (2012): WebSphere Adapter Toolkit V6.2;
http://www.ibm.com/developerworks/websphere/downloads/wat/;
Abruf: 02.01.2013

IBM WebSphere ESB (2011): Using IBM WebSphere Enterprise Service Bus;
http://publibfp.dhe.ibm.com/epubs/pdf/c3472361.pdf;
Abruf: 05.12.2012

IBM WebSphere Integration Developer (2009): WebSphere Integration Developer;
http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/topic/com.ibm.wbit.620.help.prodovr.doc/pdf/prodovr.pdf;
Abruf: 04.12.2012

JBoss Community (2012):	JBoss Community; https://community.jboss.org/en/jbossesb; Abruf: 10.12.2012
JBoss Community and Enterprise Features (2010):	JBoss EntErprisE MiddlEwarE - Making software froM the open source coMMunity ready for the enterprise; http://www.redhat.com/rhecm/rest-rhecm/jcr/repository/collaboration/jcr:system/jcr:versionStorage/c00ea0300a0526025777c3f9662cbf19/956/jcr:frozenNode/rh:pdfFile.pdf; Abruf: 02.01.2013
JBoss Community or Enterprise (2012):	Die SOA-Plattform von JBoss; http://www.redhat.com/products/jbossenterprisemiddleware/community-enterprise/; Abruf: 06.12.2012
JBoss Docs (2012):	JBoss ESB Documentation Library; http://www.jboss.org/jbossesb/docs/index.html; Abruf: 28.11.2012
JBoss Downloads (2012):	JBoss ESB Downloads; http://www.jboss.org/jbossesb/downloads; Abruf: 04.12.2012
JBoss Enterprise Calculator (2011):	JBOSS COST COMPARISON CALCULATOR; http://www.redhat.com/promo/esb_calculator/; Abruf: 06.12.2012
JBoss Enterprise Middleware (2012):	JBoss Enterprise Middleware; http://www.redhat.com/products/jbossenterprisemiddleware/; Abruf: 10.12.2012
JBoss ESB Background (2012):	Background; http://www.jboss.org/jbossesb/history.html; Abruf: 06.12.2012

JBoss Getting Started Guide (2012): JBoss Community - Getting Started Guide; http://docs.jboss.org/jbossesb/docs/4.11/manuals/html/Getting_Started_Guide/index.html; Abruf: 05.12.2012

JBoss Messaging (2012): JBoss Messaging; http://www.jboss.org/jbossmessaging/; Abruf: 02.12.2012

JBoss Programmers Guide (2012): JBoss Community - Programmers Guide; http://docs.jboss.org/jbossesb/docs/4.11/manuals/html/Programmers_Guide/index.html#d0e53; Abruf: 20.12.2012

JBoss Registry Guide (2010): JBoss Enterprise SOA Platform 4.2 - SOA ESB Registry Guide; https://access.redhat.com/knowledge/docs/en-US/JBoss_Enterprise_SOA_Platform/4.2/html-single/SOA_ESB_Registry_Guide/index.html#id3112375; Abruf: 13.12.2012

JBoss Standards (2012): JBoss Standards; http://www.jboss.org/jbossesb/resources/product_overview/standards.html; Abruf: 12.12.2012

MuleSoft (2012 g): Customers & Case Studies; http://www.mulesoft.com/customers; Abruf: 10.01.2013

MuleSoft (2012a): What is Mule ESB?; http://www.mulesoft.org/what-mule-esb; Abruf: 10.01.2013

MuleSoft (2012b): Mule ESB The most widely used Enterprise Service Bus; http://www.mulesoft.com/downloads/mule-esb.pdf; Abruf: 02.01.2013

MuleSoft (2012c): Welcome to the Mule Community;
http://www.mulesoft.org/;
Abruf: 02.01.2013

MuleSoft (2012d): Mule ESB: Scalable, Available, Reliable;
http://www.mulesoft.com/high-availability-mule-esb;
Abruf:09.01.2013

MuleSoft (2012e): Introducing Mule Studio;
http://www.mulesoft.com/webinars/intoducing-mule-studio;
Abruf: 09.01.2013

MuleSoft (2012f): Mule ESB Enterprise;
http://www.mulesoft.com/mule-esb-enterprise;
Abruf: 10.01.2013

MuleSoft (2012h): Documentation for Mule ESB;
http://www.mulesoft.org/mule-documentation;
Abruf: 10.01.2013

MuleSoft (2012i): Welcome to Mule Enterprise;
http://blogs.mulesoft.org/welcome-to-mule-enterprise;
Abruf: 17.01.2013

MuleSoft (o. J.): Architecture ofMule and ServiceMix;
http://www.mulesoft.com/downloads/architecture-mule-service-mix.pdf;
Abruf: 03.01.2013

NetWays (2012): Standardfeatures;
http://www.netways.de/de/produkte/mule/features/;
Abruf: 04.01.2013

o.V. (2010): The Athena Smalltalk;
http://bergel.eu/athena/?Documentation:JSR_223;
Abruf: 10.01.2013

Savoir Technologies Inc. (2012): Savoir Technologies; http://www.savoirtech.com; Abruf: 11.01.2013

Gesprächsverzeichnis

Bereichsleiter SE und AE-PP (2012): Kooperationspartner Unternehmen; Stuttgart; persönliches Gespräch am 17.12.2012, persönliches Gespräch am 29.12.2012