Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorthesis

Jawar Mehrabanian

Standard Service Architekturen mit Testen & Monitoring von Web-Architekturen anhand des Beispiels Selenium WebDriver/Grid mit Google Cloud Platform

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Jawar Mehrabanian

Standard Service Architekturen mit Testen & Monitoring von Web-Architekturen anhand des Beispiels Selenium WebDriver/Grid mit Google Cloud Platform

Bachelorthesis eingereicht im Rahmen der Bachelorprüfung
im Studiengang Informations- und Elektrotechnik
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr.-Ing. Sebastian Rohjans
Zweitgutachter : Prof. Dr. Ulrike Steffens

Abgegeben am 17. November 2017

Jawar Mehrabanian

Thema der Bachelorthesis

Standard Service Architekturen mit Testen & Monitoring von Web-Architekturen anhand des Beispiels Selenium WebDriver/Grid mit Google Cloud Platform

Stichworte

Testautomatisierung, Selenium WebDriver, Selenium Grid, Überwachung, Cross-Browser-Testing, TestNG, Jenkins, Google Cloud Platform

Kurzzusammenfassung

Die vorliegende Bachelorthesis widmet sich der Implementierung eines qualitätssichernden automatisierten Testvorgangs unter anderem basierend auf das Selenium-Framwork und der Überwachung von Daten mithilfe von Diensten der Google Cloud Platform. Dabei werden Standardformate eines serviceorientierten Paradigmas technisch realisiert.
Die Umsetzung wird auf das Stadtportal hamburg.de angewandt.

Jawar Mehrabanian

Title of the paper

Testing and monitoring of a web architecture in a service-oriented architecture by using Selenium WebDriver/Grid and Google Cloud Platform

Keywords

Automated Testing, Selenium WebDriver, Selenium Grid, Monitoring, Cross-Browser-Testing, TestNG, Jenkins, Google Cloud Platform

Abstract

The present thesis describes an implementation of test automation as well as monitoring data by using mainly different tools of the Selenium framework and services of the Google Cloud Platform. As an important part of service-oriented architecture this paper explains various technical ways of communication based on the possibilities of web services.
This kind of process will be applied on the offical website of hamburg (hamburg.de).

Inhaltsverzeichnis

1 Einleitung

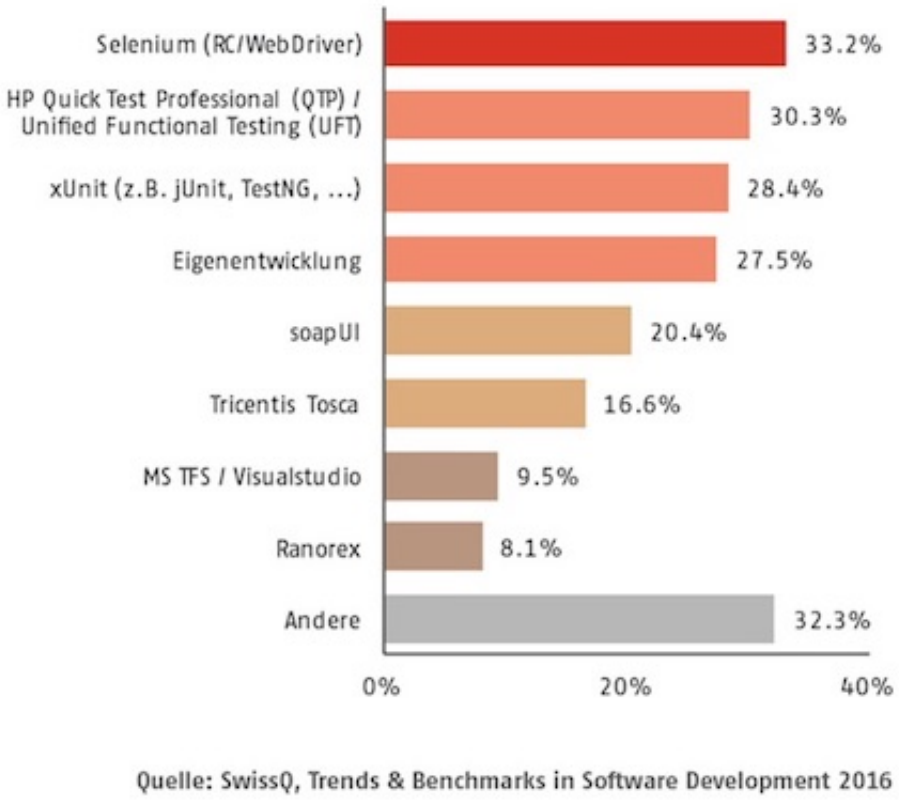
Diese Thesis behandelt die Nutzung von Testautomatisierung am Beispiel von dem Hamburger Online-Stadtportal „hamburg.de“. Die Aspekte der Standard Service-Architektur im Zusammenhang mit Testautomatisierung und Monitoring von Web-Architekturen werden erstmals in dieser Form beleuchtet. Dieses Szenario wird präzesiert anhand moderner Servicekomponenten wie Selenium WebDriver/Grid und den Diensten der Google Cloud Platform. Diverse Modelle bringen dem Leser die Präsenz im Alltag und die Funktionsweisen der Prozesse moderner IT-Unternehmensstruktur näher.
Spezifische Tests belegen die Realisierung einer Qualitätssicherung durch automatisierende Testprozesse an der etablierten Website „hamburg.de“.

1.1 Motivation

Standard Service Architektur oder vielmehr bekannt unter dem Begriff Serviceorientierte Architektur (Abk. **SOA**) gewinnt in der modernen Unternehmenswelt zunehmend an Bedeutung[1] und eröffnet durch Einbindung der Informationstechnik, besonders durch die Automatisierung, neue sowie effiziente Wege zur Handhabung von Arbeitsprozessen.
Ein relevanter Begriff im Zusammenhang mit zeitgemäßen SOAs sind Services, insbesondere Web Services. Sie ermöglichen eine technische Implementierung eines serviceorientierten Paradigmas. Dies beeinhaltet vor allem standardisierte Formate, wie beispielsweise **XML** oder **HTTP**.[2] Diese Einschränkungen festzulegen ist der Grundstein für eine serviceorientierte Umgebung, die zahlreiche sowie umfangreiche Ideen bietet. Über deratige Kern-Standards resultieren Möglichkeiten zur Interaktion zwischen Systemen, wie beispielsweise **Client-Server-Architekturen** oder **Hub-and-Spoke**-Konzept. All diese Wege zur Übertragung von Informationen und Kommunikation zwischen Systemen sind bedeutend für die Thematik dieser vorliegenden Thesis.
Das kostenfreie und quelloffene Selenium-Framework greift durch seine Vielzahl an verfügbaren Tools, wie **Selenium WebDriver** und **Selenium Grid** auf solche Mittel zurück.

[1] [vgl. Pla09, S.6]
[2] [vgl. Jos08, S.366]

Speziell Selenium Grid beansprucht ein Hub-and-Spoke-, auch bezeichnet als **Hub-and-Node**-, Konstrukt und ermöglicht dadurch parallele sowie plattformunabhängige Konnektivitäten.
Der vermehrte Gebrauch von Testautomatisierungen, vorrangig durch das Selenium-Framework, ist in Zukunft unausweichlich und es besteht eine hohe Wahrscheinlichkeit, dass es manuelles Testen vollständig ablöst.
Immer mehr User steigen auf Apps und mobile Versionen um, was auch das Selenium-Framework unterstützt und dementsprechend neue Opportunitäten hinsichtlich automatisiertes Testen bietet.



[3]

Abbildung 1.1: *Trends der Softwareentwicklung 2016*

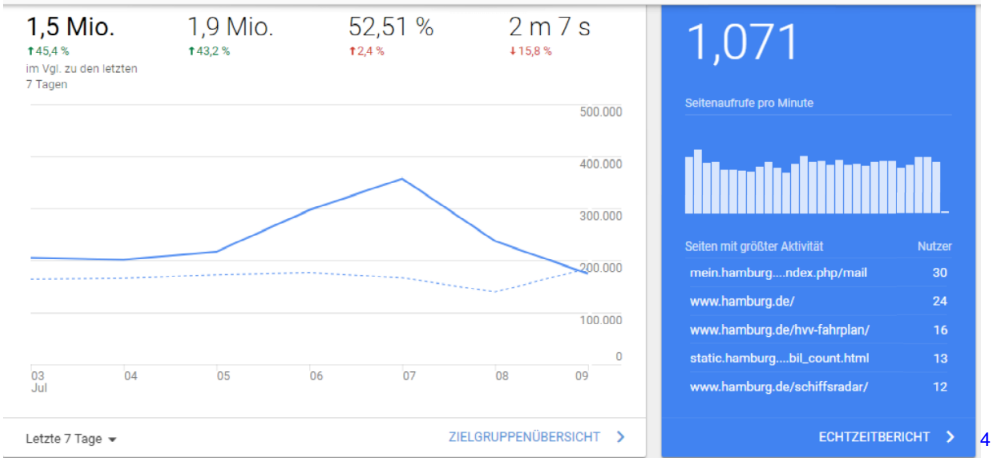
Auch hamburg.de wird zukünftig eine responsive Version ihrer Webseite veröffentlichen.
Um diese Thesis als Grundstein zur Entwicklung dieses Sachverhaltes anzusehen, wird sich vor allem auf Testing, weniger auf Monitoring, fokussiert. Monitoring, in Zusammenhang mit

[3] [Pic17b]

der **Google Cloud Platform**, wird erstmals in der Theorie beleuchtet. Die praktische Umsetzung konnte aufgrund des Umfangs dieser Arbeit nur im Ausblick umrissen werden und spielt aufgrunddessen eine sekundäre Rolle.

1.2 Zielsetzung

Mithilfe einer automatisierten Testsoftware soll die stetig gleichbleibend hohe Qualität des Online- Stadtportals „hamburg.de “ gewährleistet werden.
Das offizielle Online-Stadtportal „hamburg.de“ beschäftigt sich aussschließlich mit Themen in und um die Freie Hansestadt Hamburg. Das Stadtportal wurde im Jahre 2000 gegründet und wurde 2007 erstmals von einem Gesellschafter aus der Medienbranche, der Axel Springer AG, übernommen. Täglich gibt es über 1000 Seitenaufrufe pro Minute. Ein Beispiel liefert folgende Grafik durch die Analyse von Google Analytics, die die Echtzeitaktivität der Website „hamburg.de“erfasst und visuell darstellt.



[4]

Abbildung 1.2: *Seitenaufrufe pro Miniute von „hamburg.de“(12.07.2017)*

Das Analyse-Tool von Google Analytics wurde hier genutzt um die Echtzeitaktivitäten der Website „hamburg.de “zu erfassen und grafisch darzustellen.
Das Online-Stadtportal ist eine Public Private Partnerschaft (PPP) und wird von vier Gesellschaftern Axel Springer SE, Freie und Hansestadt Hamburg, Hamburger Sparkasse und Sparkasse Harburg-Buxtehude, getragen. [5] Informationen über Tourismus, Kultur, Jobs,

[4] Ausgabe aus Google Analytics
[5] [vgl. Lud] (Stand Juli 2012)

Wohnen, Erlebnis, Verkehr, Politik und Verwaltung - all diese Themen rundum Hamburg werden im Portal vermittelt und täglich aktualisiert.
Dementsprechend ist die Pflege und Wartung von so einer bedeutenden und sich täglich verändernden Webpränsenz besonders wichtig. Dazu gehört, wie auch schon anfangs erwähnt, ein ausgiebiges genaues Testen, was gemäß der Zeit am besten automatisiert verläuft.
Das Ziel dieser Arbeit ist es, die Komponenten der Testautomatisierung erfolgreich in ein wirtschaftlich wachsendes Unternehmen einzuführen. Dieses aber ohne die aktuelle Qualität zu mindern, sondern vielmehr sie möglichst zu steigern. Der Sinn der Implementierung dieser automatisierten Testprozesse ist vor allem, Kosten und Testzeiten zu minimieren. Außerdem soll regelmäßig nach Fehlern gesucht werden, um so eine Stabilität insbesondere bei Wachstum der Website gewährleisten zu können. Somit zeigt diese Thesis auf, wie die theoretischen Überlegungen in tatsächlich bestehende Kontexte projiziert werden können. Das automatisierte Testen soll zukünftig fester Bestandteil sein. Die Arbeit ist die Grundlage für zukünftige Entwicklungen, um Zeit als auch Kosten zu sparen. Vor allem wenn es darum geht, gezielt nach Fehlern zu suchen, die möglicherweise nach Updates und neuen Deployments entstehen können.

1.3 Gliederung

Die vorliegende Thesis ist in acht Kapitel unterteilt. Diese werden im Folgenden aufgelistet und knapp beschrieben, um für einen ersten Überblick zu sorgen.

Kapitel 1: Einleitung
In der Einleitung wird die Motivation und die Inspiration zur Verfassung dieser Thesis beschrieben. Anschließend wird das grundlegende Ziel in der Zielsetzung verdeutlicht.

Kapitel 2: Grundlagen SOA und Web-Architektur
Elementare Gesichtspunkte einer serviceorientierten Architektur (SOA) sowie einer Web-Architektur werden beleuchtet. Angefangen mit Definitionen aus unterschiedlichen Sichtweisen über Eigenschaften bis hin zu berücksichtigende Faktoren werden ausführlich geschildert.

Kapitel 3: Grundlagen der Testautomatisierung
Hier wird zunächst eine allgemeine Definition des Begriffs Testautomatisierung formuliert. Anhand des V-Modells nach Boehm werden die Merkmale einzelner Testaktivitäten

deklariert. Anschließend werden die signifikanten Qualitätskriterien nach ISO 25010 wiedergegeben.

Kapitel 4: Arbeitsablauf und Grundlagen verwendeter Softwarekomponenten
Anhand eines ersten erstellten Workflows in abstrakter Form wird ein Lösungsweg beschrieben und wie das Ziel eines automatisierten Testprozesses erreicht werden kann.[6] Nach dem Ablauf werden die einzelnen Softwarekomponenten und ihre Zusammenhänge im Hinblick auf den Workflow ausführlich erläutert.

Kapitel 5: Anforderungsanalyse
In der Anforderungsanalyse werden zunächst die Anforderungen, sowohl aus funktionaler als auch aus technischer Hinsicht, beschrieben. Diese werden daraufhin untersucht und nach Realisierbarkeit bewertet.

Kapitel 6: Implementierung
Die Implementierung behandelt die automatisierte Testprozedur anhand expliziten Fallbeispielen. Der Aufbau des Quellcodes wird beispielhaft erläutert. Die angewandten Testprozesse werden zudem für ein besseres Verständnis auf unterschiedliche Weise modelliert.

Kapitel 7: Testauswertung
Die Testauswertung basiert auf der Software Jenkins, ein open source CI-Server, mit diversen Erweiterungen. Anhand der bei Entwicklern beliebten Standardsoftware sollen Testergebnisse strukturiert aufgebaut und visualisiert werden.

Kapitel 8: Fazit und Ausblick
Kapitel 8, das letzte Kapitel der Thesis, resümiert den Sachverhalt der Thesis und leitet einen abschließenden Ausblick auf mögliche Weiterentwicklungen der hier vorgestellten Testarchitektur ein.

[6] [vgl. Jos08, S.366]

2 Grundlagen SOA und Web-Architektur

Die Problematik hinsichtlich der Begrifflichkeit von SOA besteht darin, dass keine offizielle und präzise Definition existiert. Vielmehr ist eine Vielzahl von Definitionen einer Service-orientierten Architektur vorhanden.[7] Dennoch versucht Thomas Erl einen grundlegenden und allübergreifenden Sachverhalt zur Begrifflichkeit von SOA in der Veröffentlichung *SOA - Studentenausgabe: Entwurfsprinzipien für serviceorientierte Architektur* zu definieren:

„*Serviceorientierte Architektur ist eine Form von Technologiearchitektur, die speziell zur Unterstützung serviceorientierter Lösungslogik geschaffen wurde und aus Services und Servicekompositionen besteht.*“[Erl08, S.55]

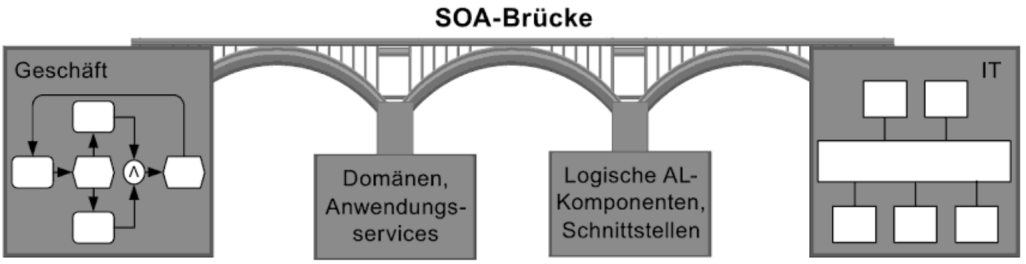
Desweiteren konkretisert Nicolai M. Josuttis den Bergiff SOA in seinem Werk *SOA in der Praxis: System-Design für verteilte Geschäftsprozesse* und erwähnt weitere essentielle Eigenschaften der SOA:

„*Service-orientierte Architektur (SOA) ist ein Architektur-Paradigma (Denkmuster) für den Umgang mit Geschäftsprozessen, die über eine große Landschaft von existierenden und neuen heterogenen Systemen verteilt werden [...]*“[Jos08, S.10]

Dieses Paradigma steigert im Kernpunkt die Anpassungsfähigkeit von großen Systemen. Flexibilität ist in einer SOA von großer Bedeutung für eine kontinuierliche und fristgerechte Lieferung von Lösungen, die zugeschnitten sind auf adäquate Qualitätsmetriken.[8]
Dabei spielen sowohl technische als auch nichttechnische Faktoren eine bedeutende Rolle. Einerseits werden „[...] *IT-Lösungen benötigt, um die Daten zu speichern und zu verwalten und die die dazugehörigen Prozess (Geschäftsprozesse) zu automatisieren, die diese Daten nutzen.*“[Jos08, S.16]
Andererseits werden strukturierte „[...] *Organisation,* [..] *Rollen* und [...] *Prozesse*“benötigt „[...] *damit die Flexibilität nicht zu Chaos führt.*“[Jos08, S.17]

[7] [vgl. Jos08, S.15]
[8] [vgl Jos08, S.15]

Mithilfe einer SOA werden also sowohl die fachliche als auch die technische Perspektive in einem Unternehmen auf einer flexiblen Weise zusammengeführt[9], wie auf der folgenden Abbildung 2.1 dargestellt wird.
Diese Zusammenführung wird in dem Werk *Quasar Enterprise: Anwendungslandschaften serviceorientiert gestalten* in Form von einer Brücke visualisiert und verdeutlicht:



[10]

Abbildung 2.1: *SOA-Brücke*

Um diese Brücke zu realisieren, werden nach Quasar Enterprise sogenannte Artefakte jedem Baustein der Brücke zugewiesen. Diese Projektion auf die einzelnen Komponenten der SOA-Brücke sorgt für ein besseres Verständnis von serviceorientierten Architektureigenschaften.[11]

In der geschäftlichen Komponente, als Startpunkt, geht es um ein strategisches Vorgehen hinsichtlich einer Geschäftsarchitektur und wie diese serviceorientiert umgesetzt werden kann.[12]
Zu den Artefakten der Geschäftsstrategie zählen:

- Geschäftsarchitektur insbesondere Geschäftsservices, -prozesse, -objekte sowie Organisation uvm.

Die IT wird unterteilt in Informationssysteme und Technische Infrastruktur und bildet den Endpunkt der Brücke.[13] Um diese SOA-Brücke zu ermöglichen, werden bestimmte Artefakte berücksichtigt.

- Domänen und Anwendungsservices, die aus der Business Architecture hervorgehen und auf der konzeptionellen Ebene stattfinden

[9] [vgl. Hum08, S.105]
[10] [Hum08, S.101]
[11] [vgl. Hum08, S.104]
[12] [vgl. Hum08, S.104]
[13] [vgl. Hum08, S.104]

- „[...] *Logische Anwendungslandschaftskomponenten (AL-Komponenten)* [...]“, die sich „[...] *auf der logischen Ebene* [...] befinden. Sie erbringen die Leistung eines Anwendungsservice“[Hum08, S.104] und aus den Domänen und Anwendungsservices resultieren

Diese beschriebenen Artefakte gehören der Katergorie Informationssysteme an. Alle weiteren Artefakte der IT, wie physische AL-Komponenten sowie Schnittstellen (Informationssysteme, physische Ebene), und die vollständige technische Infrastruktur wird in der IT integriert.[14]

Die flexible Weise einer gelungenen eingeführten SOA zeichnet sich durch eine schnelle und komfortable Integration und Anpassung der Anforderungen aus.[15] Wie bereits in der Erklärung nach Thomas Erl erwähnt, sind sogenannte Services ein im Vodergrund stehender Aspekt und ein grundlegender Bestandteil einer SOA.

„*Eine Servicekomposition besteht aus Services, die zusammengestellt wurden, um die zur Automatisierung eines spezifischen Geschäftsvorfalls oder -prozesses erforderliche Funktionalität zu liefern.*“[Erl08, S.55]

Eine explizitere Erläuterung sowie grundlegende Merkmale eines Services liefert Josuttis:
„*Services sind eine IT-Repräsentation von fachlicher Funktionalität, die durch eine [...] Schnittstelle beschrieben wird. Services sollten außerdem in sich abgeschlossen sein (autark sein und für sich selbst stehen).*“[Jos08, S.39]

Sowohl die Anwender als auch die Prozesse können über derartige Dienstschnittstellen auf existierende Systeme beispielsweise Anwendungsprogramme oder Datenbanken, die durch eine SOA vorhanden sind, zugreifen, von einem zum nächsten Prozess nahtlos miteinander interagieren oder Daten austauschen.
Um eine Verbindung zwischen den Services herstellen zu können, werden Web Services benötigt.[16]

Zu den weiteren technischen Merkmalen der SOA gehören nach Josuttis **Interoperabilität**, realisiert durch ein **Enterprise-Service-Bus** (ESB), sowie **lose Kopplung**.[17]
Eine hohe **Interoperabilität** zeichnet sich durch die möglichst simple und schnelle Realisierung einer Interaktion zwischen verschiedenen Systemen, beispielsweise mehrere

[14] [vgl. Hum08, S.105]
[15] [FOK12]
[16] [vgl. FOK12]
[17] [vgl. Jos08, S.10]

Plattformen und Programmiersprachen, aus. Eine SOA strebt eine derartige Eigenschaft an.[18] Technisch verwirklicht wird eine Interoperabilität durch ein ESB, die Infrastruktur einer SOA. In erster Linie soll ein **ESB** eine Kommunikation zwischen verschiedenartige Systeme mithilfe von Services schaffen, was die Interoperabilität steigert.[19]
Darunter fallen weitere technische Aufgaben wie „[...] *Datentransformation, (intelligentes) Routing, der Umgang mit Sicherheit und Zuverlässigkeit, die Verwaltung von Services (Service-Management), Monitoring und Logging.*“[Jos08, S.23]

Eine **ESB** kann basierend auf einem Protokoll oder einer API fungieren.
Bei einem durch Protokoll bedingtem ESB werden Anforderungen an Service-Anbieter als auch an Service-Nutzer gestellt, die für den Aufruf von Services zu berücksichtigen sind. Der Anbieter sowie der Nutzer müssen dabei jedoch das Protokoll auf die eigenen konfigurierten Schnittstellen zuschneiden. Somit wird kein gemeinschaftlicher Quellcode verwendet und außerdem dem Enterprise-Service-Bus keinerlei Bibliotheken offeriert. Dadurch sind Anbieter und Nutzer verpflichtet bei jeglicher Modifikation des Protokolls, die Codes für die Schnittstellen zu aktualisieren.[20] Bei einem ESB, die durch eine API interagiert, werden den „[...] *Anbietern und Nutzer plattformspezifische Schnittstellen zur Verfügung* [...]“gestellt, „[...] *um Services zu implementieren und aufzurufen.*“[Jos08, S.79]
Die Anpassung eines Protokolls wird aufgrund dessen evident. Bibliotheken sowie eventuell benötigte Codes werden Service-Anbieter und Service-Nutzer zur Verfügung gestellt und können anschließend eingebunden werden.[21]

Bei den zahlreichen Aufgaben einer ESB muss das Konzept der **losen Kopplung** beachtet werden. Eine SOA sollte flexibel, skalierbar als auch tolerant gegenüber Fehler sein vor allem, wenn es darum geht mehr neue Services zeitnah einzugliedern ohne die Qualität zu mindern.[22]
Hierfür bietet sich der Einsatz einer losen Kopplung an, was elementar ist für SOA. Sie dezimiert die Verbindung voneinander zu stark abhängigen Systeme.

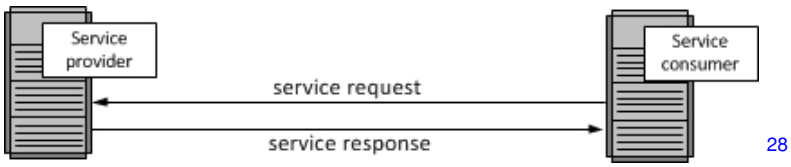
„*Je geringer die Abhängigkeiten, desto geringer die Auswirkungen von Veränderungen und Fehlverhalten. Je geringer [...] die Auswirkungen von Fehlern sind, desto fehlertoleranter sind wir [...], desto flexibler sind wir.*“[Jos08, S.22]
Jedoch kann die Einführung einer losen Kopplung eine SOA verkomplizieren. Beispiele

[18] [vgl. Jos08, S.358]
[19] [vgl. Jos08, S.358]
[20] [vgl. Jos08, S.79]
[21] [vgl. Jos08, S.79]
[22] [vgl. Jos08, S.22]

hierfür sind die Realisierung einer platformunabhängigen Umgebung oder zeitlich organisierte Deployments, im Sinne von keine gleichzeitige Verteilung bei eventuellen Softwareänderungen.[23]
Dementsprechend sollte situationsbedingt über ein angemessenen Grad der Reduktion von Abhängigkeiten mehrerer Systeme entschieden werden.[24]
Neben der bereits erwähnten Infrastruktur (ESB) sind weitere Komponenten, wie **Architektur** und **Prozesse** in einer SOA essentiell.
Das Ziel einer Architektur innerhalb einer serviceorientierten Landschaft ist die Eingrenzung von zahlreichen Opportunitäten einer SOA für einen besseren Fokus auf einen annehmbaren Bereich, beispielsweise Festlegung von Policies oder die Gliederung von verschiedenartigen Services. Infolgedessen kann das System einfacher gepflegt werden und effektiver agieren.[25]
Prozesse ordnen den Akteuren Verantwortlichkeiten zu, um Anforderungen leichter in die Praxis umsetzen zu können. Das sorgt wiederum für ein strukturiertes Vorgehen, „[...] *um ein Problem zu lösen oder ein Ziel zu erreichen. [...] Das eigentliche Ziel besteht darin, Geschäftsprozesse zu implementieren.*“[Jos08, S.362]
Derartige Prozesse inkludieren die Erstellung von Geschäftsprozessen durch BPMN[26] (engl. business process modeling), MDSD (engl. model-driven software) oder ein sogenannter Lifecycle eines Services, was den Durchlauf einzelner Zustände „[...] von der Identifizierung über Design und Implementierung bis hin zur Inbetriebnahme und Außerbetriebnahme“schildert.[Jos08, S.25]
Gesteuert werden die Prozesse, aber auch Infrastruktur sowie Architektur, durch die Governance, der „*Metaprozess aller Prozesse* [...]“repräsentiert durch das „[...] *SOA Competence Center*“.[Jos08, S.362]

Im Zusammenhang mit einer serviceorientierten Umgebung fällt in diversen Erläuterungen der Begriff Web-Service, wodurch ein potentieller Weg einer praktischen Umsetzung der technischen Komponente von SOA dargelegt wird.[27]
Nach Erl zufolge heißt es:
„[...] *Zeitgemäße SOA* [...] besteht aus [...] *Services, die als Web-Services implementiert werden.*“[Erl05, S.55 ff.]

[23] [vgl. Jos08, S.58, S.60]
[24] [vgl. Jos08, S.62, S.359]
[25] [vgl. Jos08, S.25]
[26] [vgl. Jos08, S.124]
[27] [vgl. Jos08, S.28]

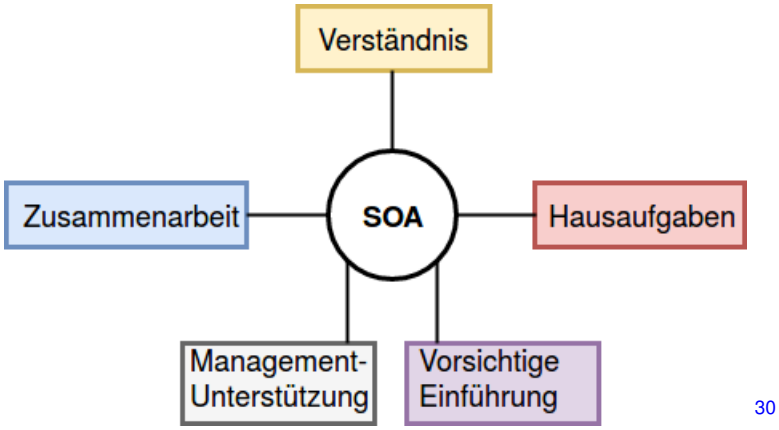


[28]

Abbildung 2.2: *Vereinfachte Darstellung von SOA*

Web-Services basieren auf Standards, die die Konnektivität zwischen Systemen realisierbar macht. Derartige Standards, wie **XML** (engl. eXtensible Markup Language), **HTTP**(S) (engl. Hypertext Transfer Protocol (Secure)), WSDL (engl. Web Services Description Language), SOAP (engl. Simple Object Access Protocol) oder UDDI (engl. Universal Description, Discovery and Integration) sind auf Interfaces beruhende Formate, als auch Kommunikationsprotokolle, zur Übertragung von Daten.[29]

Die Thesis beschränkt sich auf die Interaktion über XML und HTTP.
Ein letzter Aspekt definiert den Erfolg einer SOA, der sich durch folgende Gesichtspunkte nach Josuttis auszeichnet:



[30]

Abbildung 2.3: *SOA- Erfolgsfaktoren nach Nicolai M. Josuttis*

Der erste Faktor, der wichtig ist für eine gelungene SOA, ist das Verständnis. Bei der Einführung einer SOA sollte sich das Unternehmen darüber bewusst sein, worum es sich bei einer serviceorientierten Architektur handelt. Desweiteren sollten alle Beteiligten des Unternehmens bei einer SOA als Team agieren und gemeinsam vorgehen. Auch zeitliche und finanzielle Unterstützungen seitens des Managements sind notwendig, um den Erwartungen

[28] [Bar13, S.18]
[29] [vgl. Jos08, S.260/261]
[30] [Jos08, S.348/349]

einer erfolgreichen SOA gerecht zu werden. Der zeitliche Faktor spielt eine bedeutende Rolle, denn „zur Einführung einer SOA sollte man einen Zeitrahmen von drei oder mehr Jahren vorsehen.“[Jos08, S.348]
Je mehr Zeit zur Verfügung steht, desto bewusster können Entscheidungen getroffen werden und Prozesse sowie Richtlinien abgestimmt werden.[31] Um einen Qualitätsstandard setzen zu können, müssen genug Mühe und Fleiß in die einzelnen Elemente der SOA aufgewendet werden.[32]

Ein weiterer Begriff, aus der Kategorie der Softwarearchitektur, der im Laufe dieser Thesis zunehmend an Bedeutung gewinnt, ist die „Web-Architektur“.

Der Erfinder vom „WWW“(engl. World Wide Web) definiert das Web[33] folgendermaßen:

„*Web's major goal was to be a shared information space through which people and machines could communicate.*“[Cer98, S.302]

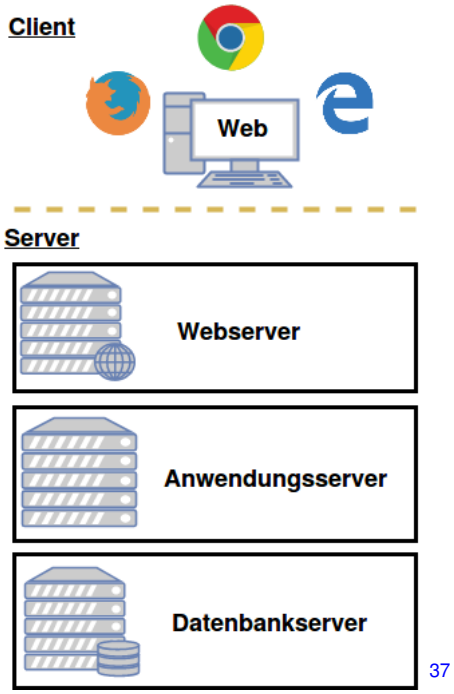
Im Wesentlichen besitzt das Web drei grundlegende Eigenschaften, die sich auch die Konzeption von webbasierten Anwendungssystemen bezieht:

- Datenübertragung durch **HTTP**[34]
- Ressourcen adressieren und lokalisieren durch Adressierungsstandards, wie **URL** (engl. Uniform Resource Language). URL gehört zu den Einzelkonzepten von URI (engl. Uniform Resource Identifier).[35]

  „In einem URI werden alle zur Identifikation notwendigen Informationen innerhalb einer Zeichenkette kodiert.“[Mei04, S.21]
- Repräsentationsformate, die in einer Web-Architektur als Standard gelten.
  Dazu gehören **HTML** (engl. Hypertext Markup Language), zuständig für eine Mensch-Maschinen-Interaktion, oder **XML** (engl. Extensible Markup Language) für eine Kommunikation zwischen mehreren Maschinen[36]

[31] [vgl. Jos08, S.348]
[32] [vgl. Jos08, S.349]
[33] [vgl. McP10, S.5]
[34] [vgl. Fin12]
[35] [vgl. Kar98]
[36] [vgl. Fin12]

Eine Web-Architektur basiert auf dem Fundament einer Client-Server-Architektur. Dabei werden „*logische Schichten bzw. Subsysteme auf Clients und ein oder mehrere Server verteilt [...]*“[Bal11, S.196]
Diese Schichten werden in der folgenden Abbildung verdeutlicht:



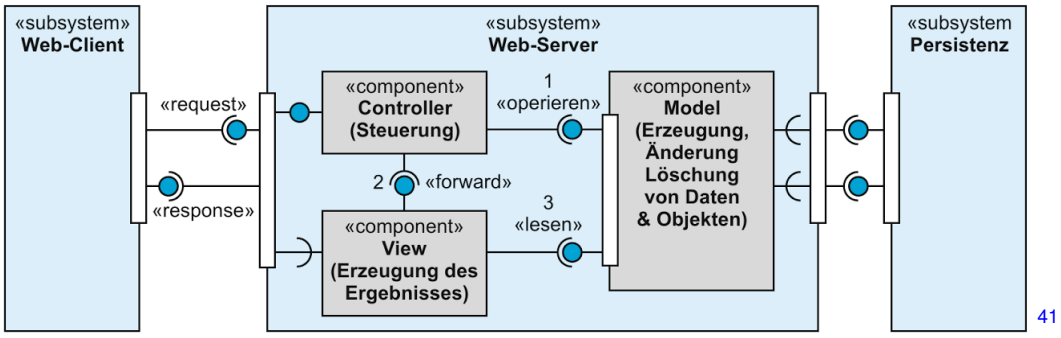
[37]

Abbildung 2.4: *Web-Architektur*

Der Web-Client enthält GUI-Elemente, die auf Webbrowser angewandt werden, wie auch auf der Darstellung 2.4 zu betrachten ist.[38]
Über HTTP empfängt der Webserver, der sich serverseitig befindet, Nachrichten vom Web-Client, bearbeitet diese und sendet eine entsprechende Rückmeldung aus, siehe folgende Abbildung 2.5.[39]
Diese Übertragung, die aufgebaut wird, sobald eine Website aufgerufen werden soll, basiert auf einer TCP-Verbindung (engl. Transmission Control Protocol). Sobald eine Antwort von dem Webserver an den Web-Client übertragen wurde, wird die auf TCP basierende Konnektivität terminiert und ist somit nicht durchgehend aktiv.[40]

[37] Erstellt mit draw.io
[38] [vgl. Bal11, S.197]
[39] [vgl. Bal11, S.197]
[40] [vgl. Bal11, S.197]

Dieser obig beschriebene Vorgang kann nach Helmut Balzert durch das auf MVC (engl. Model View Controller) basierte Modell visualisiert werden:



[41]

Abbildung 2.5: *Web-Architektur modelliert nach Konzept Model View Controller*

Damit der Webserver eine Rückmeldung aussenden kann, sind zusätzliche Informationen erforderlich, die ein Anwendungsserver (engl. application server)zur Verfügung stellt.[42]
Darin enthalten sind Anwendungsprogramme, „*wie Textverarbeitung , Tabellenkalkulation, Adressverwaltung, Kalender und die grafischen Programme [...]*“.[Lip]
Der Web-Server fungiert in diesem Schritt als Client und „[...] stellt eine Anfrage an den Anwendungsserver.“[Doh02, S.14]
Gleiche Interaktion wird ausgeführt bei einem Anwendungsserver mit einem Datenbankserver, welches die letzte Schicht der Serverseite bildet und weitere Informationen für den Applikationsserver bereitstelle. Auch hierbei funktioniert der Applikationsserver als Client und greift über Anfrage auf notwendige Daten des Datenbankservers zu.[43]

[41] [Bal11, S.197]
[42] [vgl. Doh02, S.14]
[43] [vgl. Doh02, S.14]

3 Grundlagen einer Testautomatisierung

Das vorliegende Kapitel Grundlagen beschäftigt sich mit der Begrifflichkeit der Testautomatisierung. Hierbei werden die einzelnen Aktivitäten eines Testprozesses geschildert sowie die typischen Testphasen einer Software erklärt.
Im Weiteren werden Anforderungen hinsichtlich der Qualität an einer Software definiert und dabei auf Normen und Kennzahlen eingegangen.
Abschließend werden die Testphasen eines Entwicklungsprozesses an der Grafik des V-Modells nach Boehm verdeutlicht.

3.1 Definition

Testautomatisierung, besonders im Zusammenhang mit Softwaretests, ist laut der Abhandlung von Seidl:

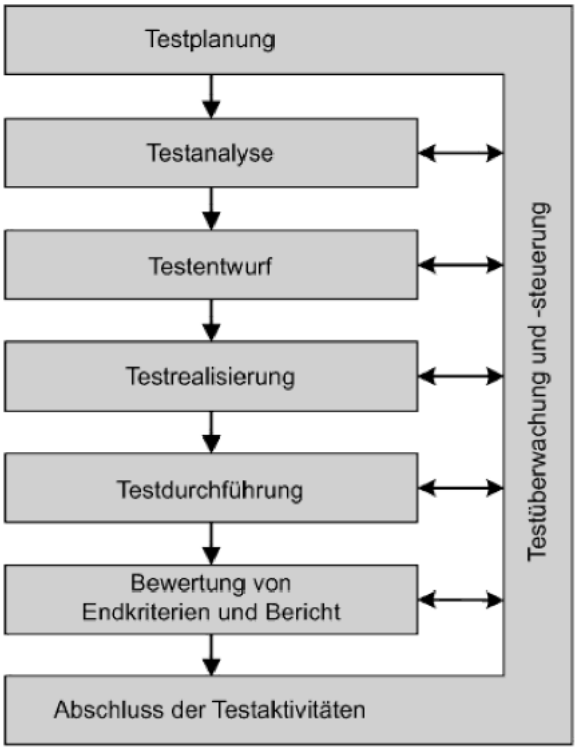
„[...] *die Durchführung von ansonsten manuellen Testtätigkeiten durch Automaten. Diese Tätigkeiten umfassen die Überprüfung der Softwarequalität im Entwicklungsprozess, in den unterschiedlichen Entwicklungsphasen und Teststufen sowie die entsprechenden Aktivitäten von Entwicklern, Testern, Analytikern oder auch der in die Entwicklung eingebundener Anwender.*“[Sei15, Kap. 1.4]

Während bei der manuellen Testdurchführung die Fehlerquote prozentual deutlich höher ist, kann diese durch die Nutzung von Testautomatisierungen dezimiert oder sogar eleminiert werden. Laut der Publikation von dem Fraunhofer IESE Institut ist „Maschinenzeit kostengünstiger als menschliche Arbeitszeit.“[Ami05] Da die Testautomatisierung parallel und im Hintergrund ablaufen kann, laufen diese deutlich schneller ab als manuelle Tests- es besteht also auch eine Zeitersparnis.[44]

[44] [Ami05]

3.2 Testaktivitäten

Die Testaktivitäten können nach Spillner & Linz durch ein strukturiertes übersichtliches Entwicklungs-bzw. Vorgehensmodell zusammengefasst werden. Diese ist angelehnt an das für Tester entwickeltes Konzept nach ISTQB (engl. International Software Testing Qualifications Board) Certified Tester. Dieses Modell wird im Folgenden abgebildet:



[45]

Abbildung 3.1: *„Der fundamentale Testprozess nach ISTQB“*

- **Testplanung:** Der Testprozess beginnt mit der Planung. In der Testplanung werden unter anderem Teststrategien, Ziele, Priorisierungen, Testbedingungen, Testarten, Ressourcen als auch Testwerkzeuge,-frameworks und umgebungen festgelegt bzw. definiert.
  Die Testplanung wird in ein Testkonzept geschrieben. Die sogenannte Testbasis, das zu testende Objekt, wird durch Dokumente wie beispielsweise Pflichten-oder Lastenheft beschrieben. Darin enthalten sind zum Beispiel Use Cases (zu Dt. Anwendungsfall) oder auch User-Stories (zu Dt. Anwendungserzählung).[46]

[45] [Sei15, Kap.2.1]
[46] [Sei15, Kap.2.1]

- **Testanalyse:** Die in der Testplanung entwickelten Dokumente werden für die Testanalyse benötigt.
  In der Testanalyse werden Testbasis sowie Testkonzept analysiert. Daraus bilden sich Testbedingungen, die für den Testentwurf verwendet werden.[47]
- **Testentwurf:** Im Testentwurf befinden sich ausschließlich nur logische Testfälle. Diese zeichnen sich dadurch aus, dass diese keine konkreten Testdaten enthalten.[48]
- **Testrealisierung:** In der Testrealisierung werden die logischen Testfälle konkretisiert und gefüllt mit expliziten Daten.[49]
- **Testdurchführung:** In dieser Phase werden Testprotokolle verfasst und gegebenfalls Abweichungsberichte geschrieben.[50]
- **Bewertung von Endkriterien und Bericht:** Die Testbewertung verwendet die vorher verfassten Testprotokolle um einen Testbericht zu erzeugen. Dieser Testbericht wird für einen Testzyklus, aber auch als Gesamtbericht für das komplette Projekt kreiiert.[51]
- **Testüberwachung-und steuerung:** „*Die Testüberwachung- und steuerung laufen parallel zu den Testaktivitäten im Projekt*“.[Sei15, Kap.2.1.1]
  In der Testüberwachung-und steuerung spielen sogenannte Kennzahlen eine signifikante Rolle. Kennzahlen dienen dem Testmanager die Tests zu steuern. Dabei wird geprüft, ob die einzelnen Ausführungen von dem Testkonzept abweichen. Der Testmanager ist somit in der Lage gegenzusteuern
  und das Testkonzept anzupassen. Die Kennzahlen, die sich aus dem Testentwurf, der Testrealisierung und der Testdurchführung bilden, werden meistens in einer Datenbank gespeichert.
  Dieser Teil des fundamentalen Testprozesses kann alle anderen Testaktivitäten dementsprechend beeinflussen.[52]

- **Abschluss der Testaktivitäten** Schlussendlich wird mithilfe des Testberichts der Testabschluss durchgeführt. Im Testabschluss werden erneut alle Testaktivitäten überprüft, ob diese erfolgreich abgeschlossen sind. Außerdem werden Testfälle reproduzierbar und wiederholbar gemacht.[53]

[47] [Sei15, Kap.2.1.1]
[48] [Sei15, Kap.2.1.1]
[49] [Sei15, Kap.2.1.1]
[50] [Sei15, Kap.2.1.1]
[51] [Sei15, Kap.2.1.1]
[52] [Sei15, Kap.2.1.1]
[53] [vgl. Sch15]

In einer Datenbank werden die gesamten Testmittel aufbereitet und archiviert, sodass sie wiederverwendbar für weitere nächste Projekte sind.[54]

3.3 Qualitätsanforderungen an Software-Produkte

Viele Unternehmen greifen hinsichtlich der Qualitätanforderungen an Software-Produkte auf bestehende Modelle zurück. Diese Modelle beinhalten Qualitätsmerkmale, die für eine Software essentiell sind. Das bekannteste Modell ist das folgend dargestellte ISO 25010:

[55]

Abbildung 3.2: *Qualitätsmerkmale nach ISO 25010*

- Funktionalität (engl. Functional Suitability):
  Functional Suitability beschreibt die funktionalen Anforderungen, die festgelegt werden

[54] [vgl. hil]
[55] [Vir16]

müssen, um die gewinnbringende Erwartungen einer Software zu erfüllen.
Unterkategorien: Korrektheit, Vollständigkeit, Angemessenheit[56]

- **Effizienz (engl.Performance Efficiency):**
  Performance Efficiency verdeutlicht das Potenzial im Verhältnis zum Verbrauch hinsichtlich der verfügbaren Ressourcen.
  Unterkategorien: Kapazität, Ressourcenverbrauch, Zeitberhalten[57]

- **Kompatibilität (engl. Compatibility):**
  Bei Compatibility wird die Konnektivität zwischen Systemen bewertet. Die Interaktion soll sowohl zwischen plattformunabhängigen als auch homogenen Komponenten funktionieren
  Unterkategorien: Interoperabilität, Ko-Existenz[58]

- **Benutzbarkeit (engl. Usability):**
  Usability bestimmt die Benutzertauglichkeit einer Software gemessen anhand von Usern, die eine Applikation auf einem möglichst verständlichen Weg effizient bedienen können.
  Unterkategorien: Erlernbarkeit, Bedienbarkeit, Schutz vor Fehlern des Benutzers, Barrierefreiheit[59]

- **Zuverlässigkeit (engl. Reliability):**
  In einem festgelegten Zeitrahmen sollen spezfische Qualifikationen hinsichtlich der Leistung erfüllt werden.
  Unterkategorien: Verfügbarkeit, Wiederherstellbarkeit, Reife, Fehlertoleranz[60]

- **Sicherheit (engl. Security):**
  Security beschäftigt sich mit dem Schutz von Daten, Erfassung als auch Eliminierung von Risiken oder Angriffen und sorgt für eine gefahrenfreie Umgebung.
  Unterkategorien: Integrität, Vertraulichkeit, Nachweisbarkeit, Ordnungsmäßigkeit, Authentizität[61]

- **Wartbarkeit (engl. Maintainability):**
  Das Kriterium Maintainability beschreibt die Anpassungsfähigkeit, die besonders bei Erneuerungen von neuen Kriterien, Modifikationen oder Fehlerkorrekturen unter Beweis gestellt wird. Unterkategorien: Modularität,Wiederverwendbarkeit, Änderbarkeit, Testbarkeit, Analysierbarkeit[62]

[56] [vgl. Tie14]
[57] [vgl. Tie14]
[58] [vgl. Tie14]
[59] [vgl. Tie14]
[60] [vgl. Tie14]
[61] [vgl. Tie14]
[62] [vgl. Tie14]

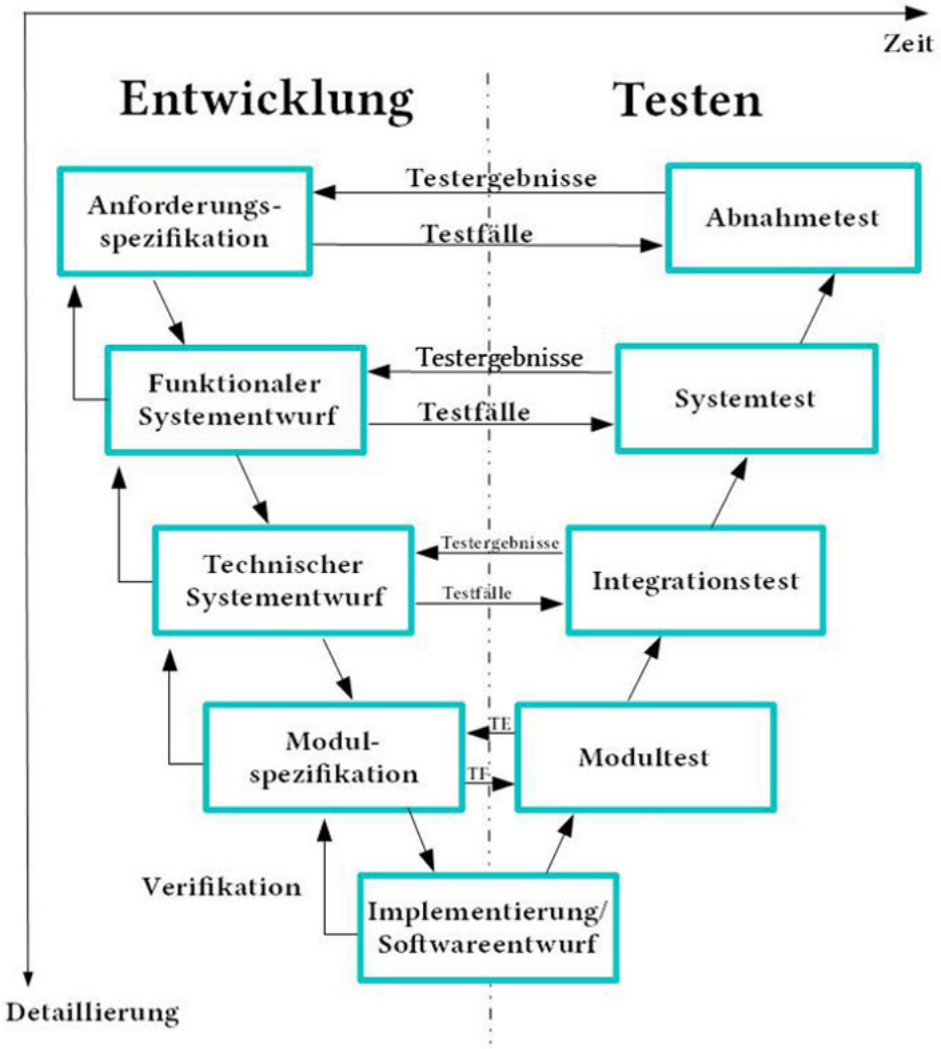
- **Portabilität (engl. Portability):**
  Portabilität ist die Verwendung einer Software auf unterschiedlichen Plattformen. Dieser Vorgang soll dabei umgänglich und unkompliziert vonstattengehen.
  Unterkategorien: Installierbarkeit, Austauschbarkeit, Anpassbarkeit[63]

Diese Norm gilt als Ablösung der verbreiteten Norm ISO 9126. Im Unterschied zur ISO 9126 wurde in der ISO 25010 die Kategorien Security und Compatibility als Hauptmerkmal aufgenommen und gilt somit nicht mehr als Unterkategorie, wie ursprünglichen ISO 9126.[64]

3.4 Testebenen

Softwaretests finden in unterschiedlichen Testphasen statt, die den Entwicklungsprozess beschreiben.
Diese können anhand von Vorgehensmodellen, die „[...] *für die Projektplanung, -abwicklung und -kontrolle* [...]“verwendet werden, „[...] *um systematisch ein IT-Projekt in vordefinierten Phasen durchzuführen.*“[Wie08, S.64]
Grafisch kann dies beispielsweise durch das populäre V-Modell nach Boehm verdeutlicht werden. Das V-Modell gehört kategorisch zu den inkrementellen Vorgehensmodellen und ist eine Erweiterung des ebenfalls bekannten Wasserfallmodells.[65]
Hierbei werden den einzelnen Testebenen eine entsprechende Testart zugewiesen:

[63] [vgl. Tie14]
[64] [vgl. Gno16]
[65] [vgl. Wie08, S.70]



[66]

Abbildung 3.3: *V-Modell von 1979 nach Barry Boehm*

Beschreibung der Testebenen entsprechend der Abbildung 3.3 von unten nach oben:

1. **Modultest:** Der Modultest, auch bekannt als Komponententest oder Unit-Test, befindet sich am untersten Punkt der Testebene. Sie überprüft Units, kleine Testfragmente, wie Klassen oder Methoden, einer Software.[67] Modultests bedienen sich am Fundament

[66] [aka15]
[67] [vgl. Ehm11, S.8/9]

der Datenkapselung. Nur über bestimmte Interfaces können auf die Daten zugegriffen werden.[68] Modultests zählen eher zu White-Box-Tests.[69]

2. **Integrationstest:** Um einen Integrationstest durchführen zu können, müssen mehrere Units vereint werden und jeweilige Interaktion, wie gemeinsamer Speicher oder Datenaustausch durch Nachrichten gleichzeitig ablaufen. Dieser Vorgang ist insofern sinnvoll, da Fehler, die in Schnittstellen einzeln getesteter Units auftreten, beseitigt werden können.[70] Für Integrationstests wird meist das Konzept der Gray-Box-Tests beansprucht.[71]

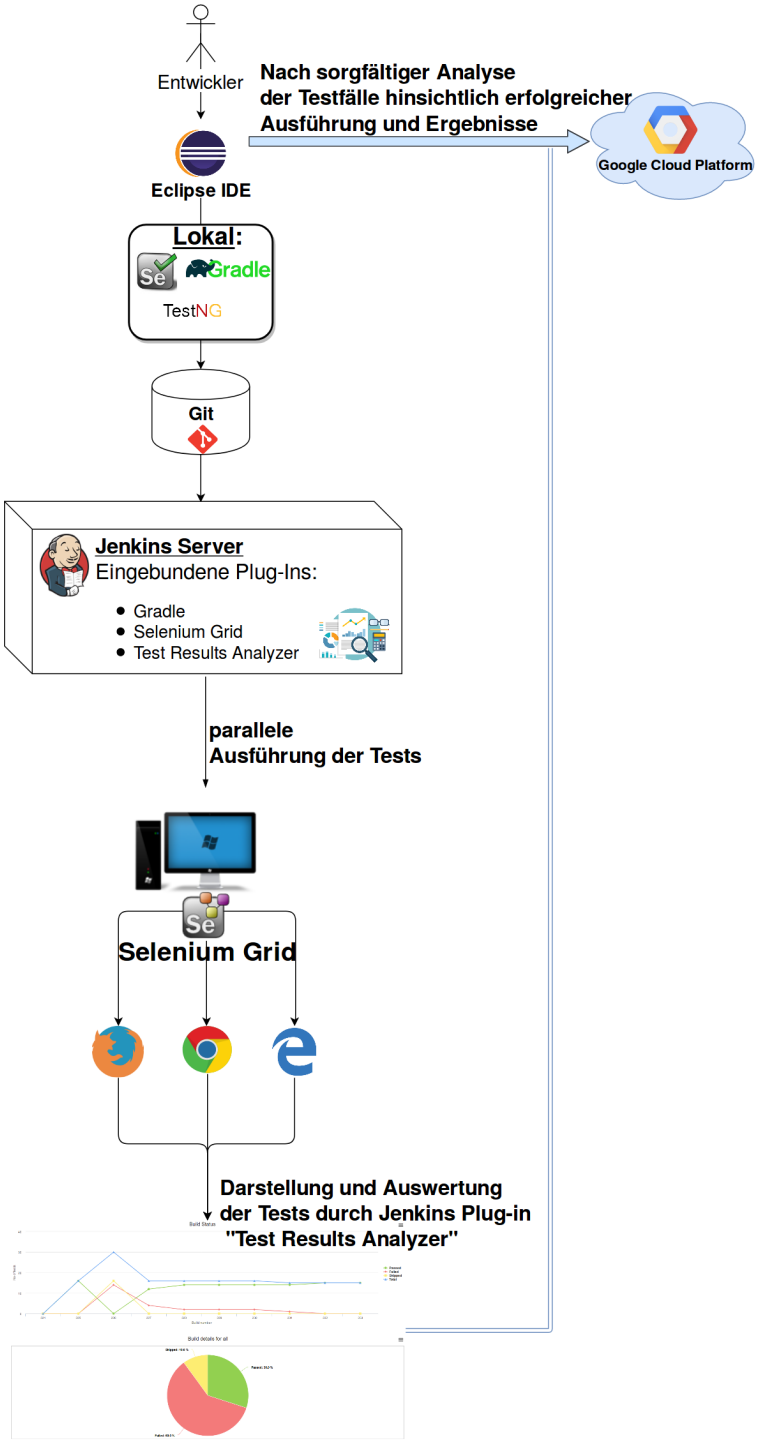
3. **Systemtest:** Systemtests entsprechen den Black-Box-Tests. Dies bedeutet, dass Entwicklungsanforderungen, die ein Anwender aus Benutzeranforderungen selektiert hat im Gesamtsystem realisiert werden.[72]

4. **Abnahmetest:** Bei einem Abnahme- oder Akzeptanztest wird sichergestellt, dass Kriterien des Benutzers erfüllt werden und Systeme einwandfrei ablaufen. Relevante Gesichtspunkte zur Überprüfung sind verständliche und einfache GUIs. Ebenso werden Anforderungen der User berücksichtigt und der erwartete Leistungsumfang realisiert.[73] Auch derartige Tests bedienen sich am Modell der Black-Box-Tests. Akzeptanztests finden dementsprechend nicht, wie gewohnt, in einer Testumgebung statt, sondern in einer entwicklungsexternen Umgebung und werden mit dem Kunden zusammen durchgeführt.[74]

[68] [vgl. Lig09, S.412]
[69] [vgl. Wit16, S.78]
[70] [vgl. Ehm11, S.10]
[71] [vgl. Kle09b, S.277]
[72] [vgl. Kle09b, S.277]
[73] [vgl. Ehm11, S.10]
[74] [vgl. Kle09b, S.277]

4 Arbeitsablauf und Grundlagen verwendeter Softwarekomponenten

Der folgende Workflow stellt die für das Projekt relevanten Bausteine in einen anschaulichen Überblick dar. Diese werden in abstrakter Form illustriert, um für eine leichtere Verständlichkeit zu sorgen.
Für jede Komponente des Arbeitsablaufs wurde ein Unterkapitel entworfen. In diesen Abschnitten werden die Komponenten definiert und näher beleuchtet, auch im Hinblick auf das Gesamtprojekt.
Dieser nachfolgende Workflow weist ausschließlich eine Art der Testautomatisierungsprozedur, die durch Jenkins ausgeführten Testfälle, auf. Eine weitere Möglichkeit wird im Kapitel 6.1.1 beleuchtet.



[75]

Abbildung 4.1: *Workflow*

[75] Anmerkung: Der abgebildete Arbeitsablauf wurde mit draw.io erstellt.

Lokal werden in der Entwicklungsumgebung „Eclipse IDE“die Selenium- Testskripte und Gradle-Build-Projekte, basierend auf Java, geschrieben. Bereits hier werden diese durch ein Framework, bekannt als „TestNG“, erstmals ausgewertet, bevor die Tests in den weiteren Prozessverlauf einbezogen werden. Im Kapitel 4.3 wird der Sachverhalt des Frameworks näher beleuchtet.
Die Tests werden anschließend durch ein in Eclipse IDE integriertes Plug-in in ein firmeninternes Repository unter Git gespeichert und verwaltet. Näheres dazu folgt im Kapitel 4.4.
Der CI-Server Jenkins (engl. Continuous Integration), greift auf das Repository zu und startet die Testfälle durch eingebundene Erweiterungen, wie Gradle- und Selenium-Plug-in, auf einen externen Computer parallel aus. Die Einbindung sowie Erläuterung der Plug-ins in Jenkins folgen im Kapitel 6.
Nähere Informationen zu Gradle sowie zu Selenium Grid folgen in den Kapiteln 4.7 und 4.2.
Gradle, ein sogenanntes Build-Management-Werkezeug, bietet sich für Jenkins als Plug-in hervorragend an, da dieses vollständig integriert ist und eine aktive Community für die Weiterentwicklung verfügt. Dieses Plug-in ermöglicht das Bauen von Gradle basierten Quellcodes.[76]

Die parallele Ausführung der Testfälle wird durch das Tool Selenium Grid, erläutert im Kapitel 4.2, ebenfalls durch Jenkins als eingegliedertes und optimiertes Plug-in, realisiert. Diese Erweiterung wandelt das Jenkins-Raster in eine Selenium 3 Grid-Umgebung um, sodass Selenium-Testfälle durchgeführt werden können. [77]
Mithilfe von Selenium Grid wird ein Hub-Node System bereitgestellt, mit dem Testfälle gleichzeitig auf unterschiedlichen Browser und Betriebssystemen ausgeführt werden können.
Nähere Informationen folgen im Kapitel 4.2
Durch das Plug-in in Jenkins, Test Results Analyzer, werden die Testergebnisse in grafischer Form dargestellt und ausgewertet. Die Testergebnisse können zusätzlich als CSV-Datei exportiert und für den weiteren Analyseprozess untersucht werden.
Nach Abschluss des Analyseprozesses werden die erfolgreich ausgeführten und untersuchten Tests in die Google Cloud Platform gelagert, die in Form eines Tools mit Eclipse IDE synchronisiert ist.
Google Cloud Platform bietet eine Reihe von Produkten an, die sich an der Infrastruktur von Google bedienen. Dadurch lässt sich unter Anderem das Problem der Sicherung von Daten lösen, sodass der Entwickler sich ausschließlich auf sein eigenes Projekt fokussieren kann.[78]
Das Thema Google Cloud Platform wird im Kapitel 4.8 ausführlicher behandelt.

Hier sei noch einmal hervorgehoben, dass die Testskripte auf den Browsern Mozilla Firefox, Google Chrome und Microsoft Edge unter dem Betriebssystem Windows 10 betrieben

[76] [vgl. Wol17]
[77] [vgl. Kaw17]
[78] [vgl. Gee17, S.2]

wurden, wie auch auf dem abgebildeten Workflow dargestellt ist.

4.1 Selenium WebDriver

Die folgenden Unterkapitel befassen sich mit dem Tool Selenium WebDriver. Es werden die Funktionen und Bedeutung im Hinblick auf die Automatisierung von Tests anhand der Historie, Architektur und des Prozessablaufs erläutert.

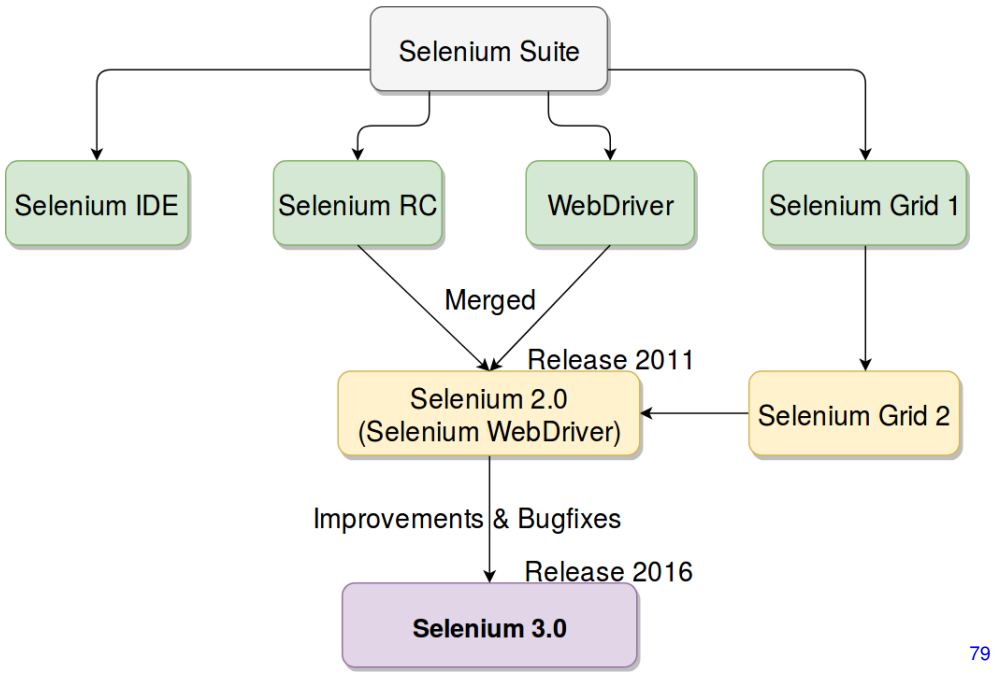
4.1.1 Allgemein

Selenium ist ein open-source Framework, welches aus verschiedenen Software Tools besteht, die der Testautomatisierung dienen.
Das Tool Selenium WebDriver wird laut „Webmasterpro.de“, dem Online-Magazin für professionelle Webentwicklung und -gestaltung, als

„mächtiges Werkzeug zur Erstellung anwendungsorientierter Tests [...]“ bezeichnet, *„[...] um Tests auf mehrere Threads und Rechner aufzuteilen.“*[Sch]

In dieser Arbeit wird insbesondere der Teil Selenium WebDriver behandelt, der in diesem Abschnitt beschrieben wird.

Folgendes Modell zeigt die Entstehungsphase von Selenium bestehend aus den verschiedenen Modulen:



[79]

Abbildung 4.2: *Zusammensetzung der einzelnen Software-Tools von Selenium*

- **Selenium IDE:** Selenium IDE ist eine Erweiterung des Webbrowsers Mozilla Firefox, welches Browserinteraktionen aufzeichnet und abspielt.[80]
- **Selenium RC:** Selenium Remote Control, auch unter den Namen Selenium 1.0 oder Selenium RC bekannt,
  „wird zur skriptgesteuerten Bedienung von Webbrowsern eingesetzt, um Tests von Web-Oberflächen und Web-Applikationen zu automatisieren. Durch den Einsatz von JavaScript zur Steuerung von Browsern, soll eine hohe Kompatibilität zu möglichst vielen Webbrowsern erreicht werden.“[Kle09a]
  Aufgrund der Client-Server-Architektur des Tools wird die Erstellung von Tests in verschiedenen Programmiersprachen gewährleistet.[81]
- **WebDriver:** WebDriver ist ein webbasiertes Testframework.
  WebDriver verfolgt einen ähnlichen Ansatz wie das Selenium-Framework bezüglich der Steuerung eines Browsers mit dem signifikanten Unterschied, dass das Framework nicht über JavaScript funktioniert.[82]

[79] Modifizierte Darstellung durch draw.io basierend auf dem Selenium Suite-Modell aus [Gar14, S.33]
[80] [vgl. Sch13]
[81] [vgl. Kle09a]
[82] [vgl. Bur10, S.182]

Die Kommandos werden vom Betriebssystem an den Browser übergeben, sodass keinerlei Einschränkungen durch die JavaScript-Sandbox vorhanden sind.
Zusätzlich ist die Einstellung eines Proxy-Servers, anders als bei Selenium RC, nicht nötig. Das ermöglichte die direkte Kommunikation zwischen einem Testskript und einem Browser, welche beide auf dem gleichen Rechner sind.[83]

- **Selenium 2.0 (Selenium WebDriver):** Die Erläuterung zur Verschmelzung von beiden Frameworks erfolgt im Kapitel 4.1.2.
- **Selenium Grid:** Selenium Grid 2, welches mittlerweile als Selenium Grid bekannt ist. Seit der Veröffentlichung von Selenium Grid 2 wurde Selenium Grid 1 ersetzt und gilt als überholt. Die Unterschiede dieser beiden Versionen werden im Kapitel 4.2 behandelt. Selenium Grid erlaubt die parallele Ausführung von Testfällen auf mehreren Serven durch ein Hub-Node-Network.
  Dabei können die „Testsuites"(engl. Testsammlungen) gleichzeitig auf unterschiedlichen Browsern und verschiedenen Betriebssystemen ausgeführt werden.[84]
  Seit dem Release von Selenium 2.0 wurde die Funktionalität von Selenium Grid 2 in Selenium WebDriver eingebunden.
- **Selenium 3.0:** Informationen zu Selenium 3.0 hinsichtlich Verbesserungen und Fehlerbehebungen folgen am Ende dieses Abschnitts 4.1.1.

Die Analyse der vorliegenden Thesis beschränkt sich auf die beiden Komponenten Selenium WebDriver und Selenium Grid.

Ein sogenanntes Testframework ist ein Programmiergerüst, was zur Ausführung von Softwaretests dient. Derartige Frameworks werden bei Entwicklungsumgebungen eingesetzt, die sich in einer Testphase befinden. Frameworks erleichtern hauptsächlich die Arbeit eines Entwicklers durch bestehende Funktionen, die aufgegriffen werden können, um sich Zeit für die Entwicklung der eigentlichen Tests und Mühe zu sparen. Selenium basiert auf der Programmiersprache Java und lässt sich dementsprechend auf jedem System, welches eine aktuelle JRE verfügt, realisieren.[85]
Durch die Einbindung von der objektorientierten WebDriver API wird ein Browser, lokal oder ferngesteuert mithilfe des verfügbaren Selenium-Server, automatisiert bedient.[86] Der Begriff API wird am Ende dieses Abschnitts 4.1.1 definiert.
So wird die Interaktion eines menschlichen Benutzers, der einen Webbrowser bedient, imitiert.[87]

[83] [vgl. Bur10, S.182]
[84] [vgl. Pre15]
[85] [vgl. Sch]
[86] [vgl. Steb]
[87] [vgl. Gos15]

Diese API ist dabei an keine Testframeworks gebunden und ist sowohl für Unit-Tests als auch die klassische main-Methode nützlich. Die WebDriver API, basierend auf einem Web-Interface[88] ermöglicht die Fernsteuerung eines Webbrowsers.[89] Selenium WebDriver oder auch bezeichnet als Selenium 2, die Fusion aus Selenium RC und WebDriver, wird zu Zwecken von automatisierten Browsertests eingesetzt.
Die Erläuterung zur Verschmelzung dieser Komponenten folgt in der Historie, im nachfolgenden Kapitel 4.1.2

Das Tool wird genutzt, um Webanwendungen automatisiert zu testen und zu überprüfen, ob diese wie erwartet funktionieren. Besonders, wenn es um die Ausführung von repetitiven Aufgaben geht, wird das Testtool häufig eingesetzt, um diese automatisch auszuführen. Selenium WebDriver unterstützt die bekanntesten Webbrowser. Dazu zählen Mozilla Firefox, Google Chrome, Windows Internet Explorer bzw. Microsoft Edge, Apple Safari, Opera etc.
Die Testskripte werden in den üblichen Programmiersprachen, Java, C# , Ruby, Python, PHP, JavaScript, Perl, geschrieben. Das plattformunabhängige Werkzeug funktioniert auf verschiedenen Betriebsystemen, wie z.B. Microsoft Windows, Apple OS und Linux. [90]
Seit Oktober 2016 wurde eine neue Version von Selenium, Selenium 3.0, veröffentlicht und bringt Neuerungen mit sich:

- Minimale Version von Java ist Java 8[91]
- Bugfixes für WebDriver und Selenium Grid
- Die Implementierung von dem Selenium Core wurde vollständig durch WebDriver ersetzt. Die bisherigen Selenium RC-APIs wurden in ein Legacy-Package verschoben und sind damit obsolet.[92]
- Durch die Implementierung der WebDriver-Schnittstelle wurden die Änderungen in Mozilla Firefox angepasst, das den Webbrowser stabilisiert und sicherer macht. Daraus folgt, dass der bisherige Firefox-Driver nicht mehr funktioniert ab der Mozilla Firefox Version 48. Der von Mozilla bereitgestellte Geckodriver wird stattdessen verwendet. [93]
- Unterstützung der Webbrowser Safari von Apple und Edge von Microsoft durch den verfügbaren SafariDriver und Microsoft Edge Driver[94]

[88] [Pre15]
[89] [vgl. Kir12]
[90] [vgl. Tim]
[91] [vgl. Jen15]
[92] [vgl. Mon16]
[93] [vgl. Blo16]
[94] [vgl. Aga17]

Die aktuellste Version, Selenium 3.7.1, ist seit November 2017 verfügbar.[95]

API steht für Application Programming Interface (zu Dt. Schnittstelle zur Anwendungsprogrammierung) und ist praktisch ein Nachrichtenübermittler, der Anfragen eines Users annimmt und dem System, in dem Fall eine Applikation, Datenbank oder ein Gerät, übermittelt. Die API erklärt also einem System, was der User tun möchte. Anschließend wird die Antwort des Systems zurück zum User geliefert.[96]

4.1.2 Historie

Im Jahre 2004 wurde Selenium von dem Entwickler Jason Huggins veröffentlicht. Er testete eine interne Applikation bei der IT-Firma ThoughtWorks.Zu dem Zeitpunkt galt Microsoft Internet Explorer als beherrschender Browser, doch ThoughtWorks arbeitete bereits mit den alternativen Webbrowser, vor allem mit Mozilla Firefox, welcher auch anfangs der Standard-Browser für das Framework war. Anders als bei den restlichen Browser, wie Microsoft Internet Explorer (IE) oder Google Chrome, war die Installation eines zustätzlichen Treibers für Mozilla Firefox nicht nötig. Bisherige Open Source Test-Tools wurden nur für einzelne Webbrowser, primär für Microsoft IE, entwickelt.
Die Problematik dieser kommerziellen Tools bestand darin, dass deren Kosten enorm hoch und dementsprechend für kleinere Projekte unbezahlbar waren.[97]

Dadurch das alle testenden Browser JavaScript unterstützen mussten, entwickelte er dementsprechend eine Javascript Bibliothek, die Interaktionen mit der Website startete und ihm erlaubte Testfälle automatisch auf multiplen Browsern zu wiederholen. Jason und sein Team entwickelten das Testtool Selenium, auch als Selenium Core bekannt, und veröffentlichten dieses unter einer Apache 2 Lizenz. Das Tool wurde in Javascript geschrieben und dementsprechend gab es Sicherheitseinschränkungen hinsichtlich der Browser.[98]
„Selenium musste auf dem gleichen Server wie die Applikation laufen, um die Browser Sicherheitsbestimmungen zu umgehen.“[tin12] Die Testsoftware war daher limitiert aufgrund des sog. Sandbox-Prinzips von JavaScript. Die Sandbox regelt durch Nutzung eines begrenzenten Bereichs den Zugriff eines Java-Programms auf Systemressourcen wie Festplatten, Arbeitsspeicher oder Windows-Funktionen. Dieses wurde konstruiert, um zu verhindern, dass bösartige Programmcodes auf Clienten-Computer zugreifen können.[99]

[95] [vgl. HQ08]
[96] [vgl. Vid15]
[97] [vgl. tin12]
[98] [vgl. tin12]
[99] [vgl. Mü03]

Wenn ein User also von HTTP zu HTTPS wechseln würde, was ein Normalfall während einem Log-IN-Prozess zur Authentifikation und Authorisierung des Benutzers wäre, würde der Browser die Aktion blockieren, dadurch dass der User nicht mehr in der gleichen Same-Origin-Policy ist. Um zu verhindern, dass Websites auf Inhalte anderer Quellen zugreifen können, wurde die Same-Origin-Policy, auch bezeichnet als SOP, entwickelt.[100]
Um dieses Problem zu lösen, wurde ein HTTP Proxy geschrieben, wodurch sich Einschränkungen seitens der SOP von den Webbrowsern umgehen ließen. Dieses Muster gestattete den Testern das Schreiben von Selenium-Anbindungen für einige Programmiersprachen, und diese über eine HTTP-Anfrage an eine bestimmte URL zu versenden.[101]
Das war der Anfang einer neuen Selenium Version, die unter dem Namen Selenese oder Selenium Remote Control (Selenium RC) bekannt wurde. Zeitgleich arbeitete Simon Stewart, der auch bei ThoughtWorks tätig war, an einem weiteren Projekt der Browser-Automatisierung namens WebDriver. Die erste Version von WebDriver wurde 2007 veröffentlicht.[102] Dieses Projekt wurde für Nutzer entworfen, die ihre End-to-end-Tests von dem Testtool isolieren wollten, was durch die Verwendung von Adapter möglich war. „Ein Entwurfsmuster, die die Schnittstelle einer Klasse an eine andere, von ihren Klienten erwartete Schnittstelle anpasst. Somit können Klassen miteinander kommunizieren, die zueinander inkompatible Schnittstellen besitzen.“[Kal01] Zwischen Selenium RC und WebDriver gab es ein paar gravierende Unterschiede:

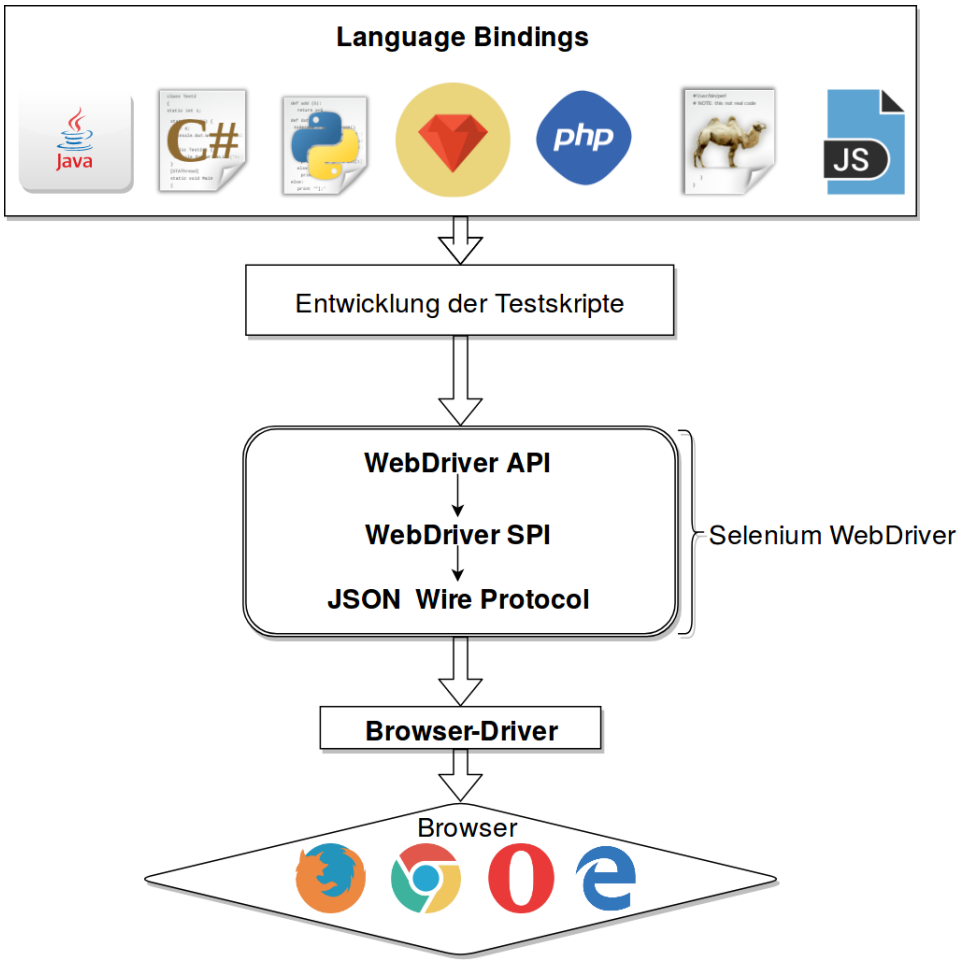
- Selenium RC besaß eine wörterbuch-basierte API, die alle Methoden für eine Klasse beanspruchte, während WebDriver eine objektorientierten API besitzt[103]
- Selenium 1 unterstützte viele Programmiersprachen, während WebDriver ausschließlich nur auf Java basiert[104]
- Selenium Remote Control war technisch gesehen eine auf JavaScript basierte Applikation, die in der Security-Sandbox des Webbrowsers lief. WebDriver hat das Ziel sich in den Browser zu integrieren und die sinnvollen Sicherheitsrichtlinien des Browsers bezüglich JavaScript zu umgehen, was hinsichtlich des Entwicklungsaufwands sehr herausfordernd war[105]

Im Jahre 2009 beschlossen Jason Huggins und Simon Stewart diese beiden Projekte zu verbinden. Daraus resultierte Selenium WebDriver (Selenium 2.0) . Neben den typischen Desktop-Browser wurden auch erstmals mobile Webbrowser für Android und iOS angeboten.[106]

[100] [vgl. Kar12]
[101] [vgl. tin12]
[102] [vgl. tin12]
[103] [vgl. Ava14, S.13, 14]
[104] [vgl. Ava14, S.13, 14]
[105] [vgl. Ava14, S.13, 14]
[106] [vgl. tin12]

4.1.3 Architektur

Die Architektur von Selenium WebDriver setzt sich aus fünf unterschiedlichen Teilen zusammen:



[107]

Abbildung 4.3: *Architektur von Selenium WebDriver*

- **Language Bindings**: Viele verschiedene Programmiersprachen werden von Selenium WebDriver unterstützt, um Testcodes bzw. Testskripte zu entwickeln. Diese sind in der Abbildung 2.3 verdeutlicht.[108]

[107] [vgl. Bur12, S.65]
[108] [vgl. Bur12, S.65 ff.]

- **WebDriver API**: Die WebDriver API übersetzt den in den Tests eingebundenen Programmcode für die SPI, Stateless Programming Interface, die zustandlos ist.[109]

  Der benutzerspezifische Applikationstext wird mithilfe des Stateless-Zustandsmodells ausschließlich für die Dauer eines Request-Response-Zyklus verwendet. Eine Anwendung zustandslos durchzuführen wird definiert, dadurch dass bei jedem Request und nach jeder Response jegliche Informationen gelöscht werden. Eine weitere Definition für zustandslos ist also gedächtnisfrei. Sobald ein Request ausgeführt wurde und der Browser eine Response empfangen hat, werden alle Ressourcen wie Instanzen, Seitenattributwerte und Anwendungsklassen gelöscht.[110]
  Auch das Übertragungsprotokoll HTTP, was im folgenden Abschnitt **WebDriver SPI** erklärt wird, befindet sich in einem zustandslosen Status.[111]
  Die Anfrage eines Clients enthält alle Zustandsinformationen, die ein Server benötigt. Im Unterschied dazu halten Server keine Zustandinformationen über den Client.[112]
  Um jede neue Anfrage des Clients zu gewährleisten, muss ein neuer Verbindungsaufbau und neue Datenbeschaffung generiert werden. [113] gesorgt werden muss.

- **WebDriver SPI**: Wenn der Programmcode den SPI erreicht, wird ein Mechanismus aktiviert, welcher die Elemente des Codes in kleine Teile bricht und sich dann die Kommandos auswählt, die relevant sind, beispielsweise eine im Code genutzte ID:[114]

  Aus **driver.findElement(By.id(„x“))** , die in der API steht,
  resultiert **findElement(using=„id“, value=„x“)** in der SPI[115]

  Zur Kommunikation mit den Webbrowser und Datenübertragung wird HTTP genutzt. Für jedes Kommando wird eine HTTP-Anfrage erstellt und über das JSON Wire Protocol an den Browser Driver verschickt.

- **JSON Wire Protocol**: JavaScript Object Notation, JSON, ist ein Format zur Übertragung von Daten zwischen Server und Client.[116]
  Um einen Austausch von Informationen zwischen Programmen, die unterschiedliche Programmiersprachen nutzen, herzustellen, wird JSON verwendet. Dieser Vorgang

[109] [vgl. Bur12, S.65 ff.]
[110] [vgl. Neu, S.1]
[111] [vgl. Neu, S.1]
[112] [vgl. Sro17]
[113] [vgl. Neu, S.3]
[114] [vgl. Bur12, S.66]
[115] [vgl. Bur12, S.66]
[116] [vgl. Bur12, S.66]

wird auch als Serialisierung von Daten betitelt.[117]
Das JSON Wire Protocol dient als Transportmittel und überträgt über REST alle benötigten Elemente zu dem Code.
REST definiert sich durch ein Architekturmodell mit webbasierten Anwendungssystemen, die über HTTP kommunizieren.[118]

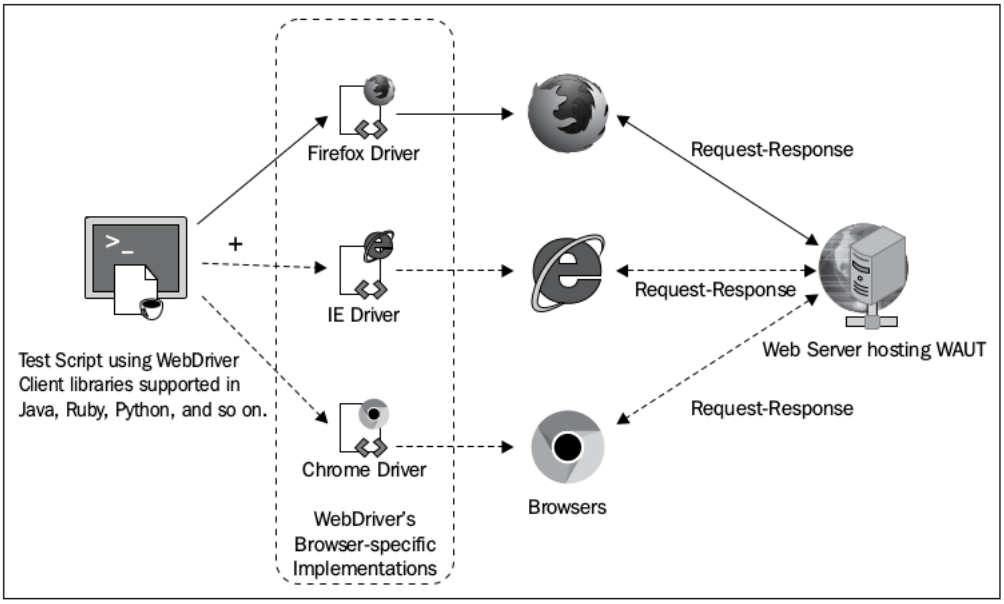
- Selenium Server, **Browser-Driver**: Der Selenium Server nutzt die Befehle vom JSON Wire Protocol und filtert sich die JSON Objekte heraus. Der Server gibt diese Kommandos an den entsprechenden Browser Driver weiter. Jeder Browser enthält einen eigenen Driver. Diese sind im Modell 2.3 sichtbar.
  Der Treiber startet und steuert den jeweiligen Webbrowser, direkt und nativ, und nicht über JavaScript.[119]

  Wenn ein Browser-Driver gesetzt wird, werden die vom WebDriver empfangenen Kommandos auf dem jeweils verantwortlichen Browser ausgeführt und in Form von HTTP als Antwort zurückgegeben.[120]

[117] [vgl. Lip13a]
[118] [vgl. Lip16]
[119] [vgl. Pre15]
[120] [vgl. Pre15]

4.1.4 Prozess

Das vorliegende Unterkapitel beschreibt die Prozedur von Selenium WebDriver. Verdeutlicht wird diese durch das folgende Modell aus der Veröffentlichung „Selenium WebDriver Practical Guide“von Satya Avasarala:



[121]

Abbildung 4.4: *Vorgang von Selenium WebDriver anhand Mozilla Firefox, Google Chrome und Windows Internet Explorer*

Der Prozess, visualisiert durch die Abbildung 4.4, lässt sich in drei signifikante Schritte gliedern:

1. Zunächst werden die vom Tester entwickelten Testskripte dazu verwendet WebDriver bestimmte Aktionen auf dem WAUT, Web Application Under Test, eines bestimmten Browsers ausführen zu lassen. Durch die vom WebDriver enthaltenen Bibliotheken oder Language Bindings können die Befehle eines Users durchgeführt werden.[122] Diese Bibliotheken unterstützen die im Kapitel 4.1.3 erwähnten Programmiersprachen.

2. Die Bibliotheken dienen dem Entwickler dazu die browserspezifischen Implementierungen des WebDrivers aufzurufen. Dazu gehören der Firefox Driver, IE Driver, Chrome Driver usw. Diese Browser Driver werden für die Interaktion mit dem WAUT

[121] [Ava14, S.13]
[122] [vgl. Ava14, S.14]

benötigt, um eine Übertragung zwischen des jeweiligen Browsers und Webserver herzustellen. Die browserspezifischen Einbindungen des WebDrivers kommunizieren mit dem jeweiligen Browser und führen Befehle von außerhalb der Browserumgebung aus, exakt wie der Umgang eines Users mit einer Applikation.[123]

3. Nach der Ausführung dieser Kommandos liefert WebDriver die Testergebnisse an das Testskript zurück beispielsweise realisiert durch ein Testautomatisierungs-Framework, wie TestNG, bereits beschrieben in den Kapiteln 4.3 und 4.4.
   Entwickler können anschließend die Testresultate analysieren, auswerten und gezielt die Fehler beheben.[124]

Mithilfe der Ergänzung durch Selenium Grid, können nicht nur einzelne, sondern mehrere Prozesse gleichzeitig auf einem Webserver abgearbeitet werden.
Dieses Konzept wird im nachfolgenden Abschnitt 4.2 behandelt.

[123] [vgl. Ava14, S.14]
[124] [vgl. Ava14, S.14]

4.2 Selenium Grid

Selenium Grid, ein weiterer wichtiger Bestandteil des Selenium- Frameworks, erlaubt dem Benutzer Testfälle auf unterschiedlichen Systemen mit verschiedenen Webbrowser parallel auszuführen, auch bezeichnet als distributed test execution[125] Durch diese Methodik wird essentiell viel Zeit für das Absolvieren einer Testsammlung gespart.
Diese Plattformen, auch bekannt als Nodes, werden von einem zentralen Punkt aus gesteuert. Dieser zentrale Kern wird als Hub bezeichnet. Im Hub werden alle wichtigen Informationen, Betriebssystem, Name des Browsers, Version, über die Nodes gespeichert. Der Hub veranlasst die Ausführung der Testskripte an die registrierten Nodes sobald eine Anfrage der Testfälle ansteht.
Folgendes Modell veranschaulicht diesen Vorgang:

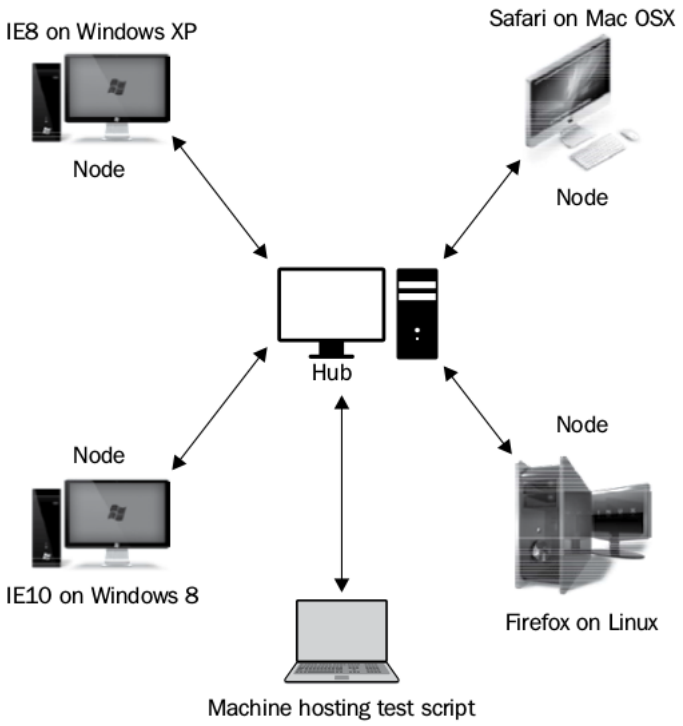


Abbildung 4.5: *Beispielhafte Darstellung von Selenium Grid*

[125] [vgl. IS]
[126] [Ava14, S.170]

In der obigen Abbildung 2.5 wird ein Hub im zentralen Punkt mit vier Nodes abgebildet. Diese vier Nodes sind unterschiedliche Rechner mit ‚entsprechend des Beispiels, verschiedenen Betriebssystemen und Webbrowser. Ein Testskript, gespeichert und illustriert durch ein Notebook, kommuniziert mit dem Hub und fordert eine Ausführung eines Testfalls auf einem der registrierten Nodes. Der Hub leitet diesen Befehl an dem zugeordneten Node weiter. Anschließend führt der gewünschte Node den Testfall aus, überträgt das Ergebnis an den Hub zurück und der Hub widerum leitet das Testergebnis weiter an das hosting Testskript.

Mittlerweile sind zwei Versionen von Selenium Grid verfügbar. Die Unterschiede sind in der folgenden Tabelle erfasst:

Selenium Grid 1	**Selenium Grid 2**
Server mit eigenem Fernsteuerungssystem, das sich von dem Selenium RC, auch bekannt als Selenium 1, Server unterscheidet	Selenium Grid- und Selenium RC Server zusammengefasst in einer JAR-Datei. Dabei handelt es sich um einen Standalone-Server, der einfach eingebunden werden kann. Ein Standalone-Server ist ein Programm, welches unabhängig von einem physikalischen Server fungiert und eigene serverspezifische Funktionalitäten verwendet.[127] Untersucht wird dieser Sachverhalt in Kapitel 6.
Vorinstallation und Konfiguration von Apache Ant, ein Tool zum Kompilieren von Programmiercode in Java, benötigt um Grid 1 zu starten	Keine Einbindung von Apache Ant nötig
Unterstützt ausschließlich Selenium RC Befehle	Akzeptiert sowohl Selenium RC- als auch Selenium WebDriver-Kommandos
Nur ein Webbrowser kann über die Fernsteuerung gestartet werden	Ein Server kann über den Fernsteuerungsmechanismus bis zu fünf Webbrowser parallel automatisiert aufrufen

Tabelle 4.1: *Vergleich der Selenium Grid Versionen*

[128]

Anhand der Tabelle 4.1 wird deutlich, dass die erste Version von Selenium Grid nach der Entwicklung und Veröffentlichung von Selenium Grid 2 überholt ist. Durch die Einbindung des

[128] [vgl. Gur17]

Selenium RC-Servers war es dem Nutzer möglich parallele Testfälle ferngesteuert auf unterschiedlichen Rechnern mit verschiedenen Betriebssystemen und Webbrowser ausführen zu können, welches der grundlegende Vorteil der zweiten gegenüber der ersten Version ist.
Selenium Grid 2 wurde einerseits entwickelt, um sicherstellen zu können, dass eine Testapplikation mit unterschiedlichen Kombinationen aus Webbrowser und Betriebssystem vollständig kompatibel ist. Andererseits ist die Zeitersparnis hoch durch die gleichzeitige Ausführung mehrerer Testsammlungen.[129]

4.3 TestNG

TestNG ‚Test next generation, ist einer der meist benutzten und gängigsten Testautomatisierungs-Framework in Java und inspiriert von JUnit und NUnit. Das Framework kann genutzt werden für Unit-‚funktionale-‚Integrations- und End-to-End Tests. TestNG nutzt Java-Annotationen, um Testmethoden zu schreiben und zu konfigurieren. Entwickelt wurde das Testautomatisierungs-Framework von Cedric Beust. Sein Grundgedanke bestand darin die Defizite von JUnit zu beseitigen. Die Vorteile, besonders gegenüber JUnit, werden im Folgenden aufgelistet und dann ausführlicher erläutert:

- Viele unterschiedliche Arten von *@Before-/After-* Annotationen sowie *@Before-/After Suite* und *@Before-/After Group*, die einfach zu verstehen und hilfreich sind um Testfälle zu lenken .
  Beispielsweise können Tests durch Setzen von Annotationen parallel ausgeführt, übersprungen oder priorisiert werden[130]
- Auf XML-basierte Testkonfigurationsdateien können genutzt werden um eine Testsammlung, auch als Testsuites bekannt, zu kreieren, die Klassen, Testmethoden, Pakete als auch TestNG Gruppierungen nutzen[131]
- Testmethoden können durch eine Konfiguration in der XML-Datei parametesiert werden. Innerhalb des Tests werden Parameter dabei an Testklassen und -methoden übergeben[132]
- Dependency Tests ermöglichen eine besondere Technik, die erlaubt, dass eine Testmethode sich abhängig von einer einzelnen oder Gruppe von Testmethoden macht.[133]
  Beispielsweise ist hierbei konfigurierbar, ob der Dependent Test im Falle eines Fehlers einer vorherigen Testmethode noch ausgeführt oder abgebrochen werden soll

[129] [vgl. Sha17a]
[130] [vgl. Men13, S.52]
[131] [vgl. Edl07, S.36]
[132] [vgl. Edl07, S.36]
[133] [vgl. Men13, S.105]

- Auch sogenannte Data-driven-Tests, erlauben einem Benutzer gleiche Testmethoden basierend auf unterschiedlichen Testdaten mehrmals auszuführen. Derartige Testdaten werden aus Dokumenten, wie aus CSV-Dateien, ausgelesen, wodurch der User die jeweilige Applikation mit unterschiedlichen Eingabeparameter verifizieren kann[134]
- Mehrere Testmethoden können gruppiert werden
- Multithreaded Execution ist möglich. Die Tests werden parallel ausgeführt, um Zeit zu sparen oder so ein multithreaded Testszenario zu prüfen[135]
- Umfassende HTML- und XML- Reports werden standardgemäß intern generiert bezüglich der Ausführung von Tests und des Builds[136] Auch eigene modifizierte Berichte können dem Framework hinzugefügt werden
- TestNG bietet Erweiterungen in Form von APIs an. Dadurch besteht die Möglichkeit eigene angefertigte Erweiterungen oder Plug-ins in das Framework einzubinden[137]

  TestNG wurde lokal mithilfe des in Eclipse IDE eingebundenen Plug-ins TestNG verwendet, die zu betrachten ist auf der Abbildung 4.1). Weitere Informationen zu diesem Plug-in werden im nachfolgenden Kapitel 4.4 aufgeführt.

4.4 Eclipse IDE

Eclipse IDE, *Integrated Development Environment*, ist eine integrierte Entwicklungsumgebung für Java, die im Jahre 2001 von einer Open Source Community entwickelt wurde. Durch die vielen verfügbaren Erweiterbarkeiten und zusätzlichen Softwarekomponenten oder Plug-ins gilt Eclipse als einer der wichtigsten Programmierwerkzeuge für die Entwicklung von Software.[138]
Zu den aus dem integrierten „Eclipse Marketplace“erworbenen Erweiterungen gehören:

- **Buildship Gradle Integration 2.1.1** : Unterstützt das Bauen von Software durch Gradle[139]. Nähere Informationen folgen im Kapitel 4.7

[134] [vgl. Edl07, S.54 ff.]
[135] [vgl. Edl07, S.30]
[136] [vgl. Edl07, S.28]
[137] [vgl. Edl07, S.17]
[138] [vgl. Foua]
[139] [vgl. Foub]

- **Google Cloud Tools for Eclipse 1.2.0** : Ein von Google bereitgestelltes Plug-in, das die Google Cloud Platform unterstützt. Das Tool ermöglicht dem User das Entwickeln, Importieren, Editieren, Bauen, Starten, „Debugging und Deploying“von auf Java basierten Applikationen ohne die Entwicklungsumgebung zu verlassen.[140]
  Weitere Details folgen im Kapitel 4.8)

- **TestNG** Das Plug-in führt TestNG-Tests, beispielsweise Testsammlungen, Gruppen und Methoden mit TestNG-Annotationen, aus. Bei der Auswertung werden die Testfälle in einem externen Reiter farbig (abhängig von Erfolg oder Fehler) markiert sowie die Testzeiten einzelner Tests angegeben. Zusätzlich wird der Ausgang eines durchgeführten Tests in Form von einem Statusbericht über die Konsole angezeigt. Daran kann sich ein Tester ebenfalls orientieren, um sich zu vergewissern, dass die Testresultate den Erwartungen oder dem Soll-Zustand entsprechen. Die Ergebnisse können effizient für nachfolgende Analysen genutzt werden.

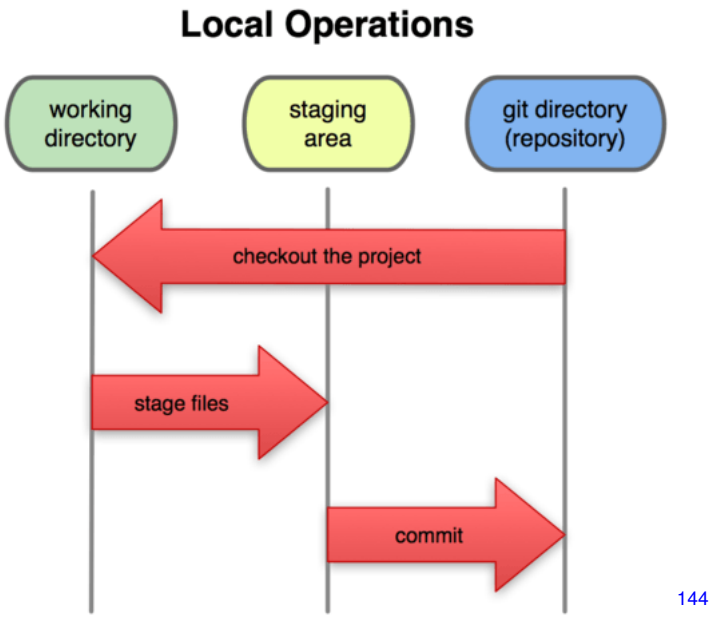
  Diese Erweiterung wurde ausschließlich für die Überprüfung von lokalen Testfällen genutzt, da bereits Einbindungen von TestNG-Plug-ins in Jenkins vorhanden sind. Diese nennen sich TestNG Results Plugin und Test Results Analyzer.
  Weitere Informationen folgen im Kapitel 5 Testauswertung.[141]

- **EGit**: Mithilfe von EGit wird die Integration von Git ermöglicht. Im Kapitel 4.5 wird das Thema Versionsmanagement durch Git aufgegriffen.
  Durch das in Eclipse integrierte Plug-in werden alle Testfälle in ein Repository gelagert und anschließend von einem eingerichteten Jenkins-Server aufgenommen bzw. verarbeitet.[142]

[140] [vgl. Fou17]
[141] [vgl. Fouc]
[142] [vgl. Fou16]

4.5 Versionsmanagement durch Git

Software entwickelt sich inkrementell. Bei Versuchen muss der Weg zurück möglich sein. Hierfür ist ein Versionsmanagement essentiell für einen Entwickler.
Git ist ein quelloffenes Versionverwaltungssystem . Durch ein Versionskontrollsystem, in diesem Fall Git, werden Dateien zentral im Repository verwaltet.
Ein Repository kann man als Ablage von Modellen und deren Bestandteile verstehen, es handelt sich somit um eine spezielle Datenbank. Der Grundgedanke befasst sich mit Speicherung, Versionskontrolle und Unterstützung beim Abrufen von Daten. Da mehrere User Zugriff auf die Dateien haben, dient es ebenso dazu verschiedene Versionen einzusehen hinsichtlich der Änderungen und diese zu verwalten.[143]
Sobald ein Entwickler eine zu ändernde Software auscheckt, erhält dieser eine lokale Kopie zur Bearbeitung. Durch ein derartiges Versionskontrollsystem besitzt jeder Entwickler eine Kopie von allen aktualisierten Programmcodes und ist in der Lage darauf zurückzugreifen.
Ein Git Projekt besteht aus drei Hauptbereichen: „Git Verzeichnis, Arbeitsverzeichnis und die Staging Area“.[Cha14, S.11]
Der Arbeitsablauf wird durch folgendes Modell verdeutlicht:



[144]

Abbildung 4.6: *Git Arbeitsprozess*

[143] [vgl. Ley93]
[144] [Cha14, S.11]

1. Git Verzeichnis, engl. git directory: Im Git Verzeichnis, auch als Repository bezeichnet, werden die Git Metadaten und die lokale Datenbank für das jeweilige Projekt gespeichert.[145]

2. Arbeitsverzeichnis, engl. working directory: Im Arbeitsverzeichnis können Dateien, spezifiziert durch Versionierung, über einen Zugriff auf die Datenbank auf eine Festplatte gesichert werden. Diese stehen dem Anwender anschließend für weitere Bearbeitungen oder Modifikationen zur Verfügung. Dieser Vorgang wird auch als Checkout bezeichnet.[146]

3. Staging Area: In der Staging Area wird durch Vormerkung die Erneuerungen eines anstehenden Commits festgelegt.

   Der Prozess verläuft, wie auf der Abbildung 4.6 erkennbar, folgendermaßen:
   Zunächst werden die Dateien im Arbeitsverzeichnis bearbeitet. Danach müssen die Dateien für den nächsten Commit markiert werden, dadurch dass sogenannte Snapshots der Staging Area hinzugefügt werden.
   „Ein Commit ist eine Freischaltung einer individuellen Änderung einer Datei.“[fat17] Ein Snapshot, zu Dt. Schnappschuss, spiegelt den momentanen Zustand aller Dateien des Projektes wieder, sprich den vollständigen Inhalt aller Dateien. „Damit Speicherplatz gespart werden kann, speichert Git allerdings Dateien, die in mehreren Snapshots inhaltsgleich sind, nur einmal.“[yA]
   Schließlich wird der Commit angelegt. Dadurch wird eine Veränderung der lokalen Datenbank dauerhaft im Git-Verzeichnis sichtbar und steht demnach für alle Entwickler zur Verfügung.[147]

[145] [vgl. Cha14, S.11]
[146] [vgl. Cha14, S.11]
[147] [vgl. Cha14, S.11]

4.6 Jenkins

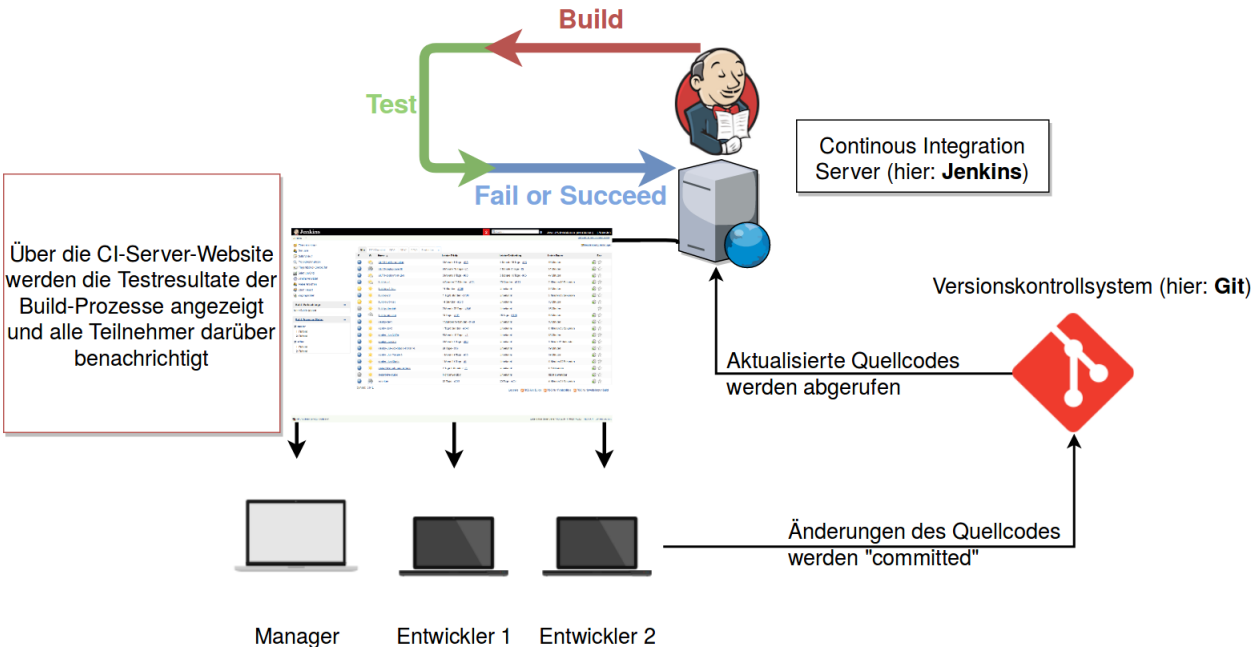
4.6.1 Allgemein

Jenkins ist ein Open Source Integration Server, der in Java geschrieben ist. Jenkins, ehemals auch als Hudson bekannt, gehört zu den bekanntesten und benutzerfreundlichsten Continuous Integration Tools in Bezug auf Entwicklung und Testen von Projekten. Das Ziel von Continuous Integration (CI) besteht darin die Softwarequalität zu steigern durch einen kontinuierlichen Integrationsprozess einzelner Komponenten der Software. Die Erstellung und das Testen einer Software wird wiederholend in einem bestimmten Zeitzyklus durchgeführt. Das sorgt für eine frühzeitige Erfassung von Problemen während des Integrationsprozesses und fehlerhaften Testfällen. Die Voraussetzung für eine Einführung eines CIs ist ein Versionskontrollsystem, in dem die Programmcodes gesichert und verwaltet werden, zuvor beschrieben im Kapitel 4.5.[148]

[148] [vgl. Kü13]

4.6.2 Continuous Integration-Server

Typischer Prozessablauf eines CI-Servers



[149]

Abbildung 4.7: *Prozessablauf eines CI-Servers*

[149] Das Diagramm wurde mit draw.io erstellt.

Wie in der obig aufgeführten Abbildung 4.7 werden zunächst durchgeführte Änderungen der Programmcodes in ein Respository committet. Im CI-Server wird durch Angabe einer VCS (engl. Version Control System) spezifischen URL gegenüber dem VCS, in dem Fall Git, authentifiziert und mitgeteilt, wo sich das Projekt innerhalb des VCS befindet. In regelmäßigen Zeitabständen wird das Versionskontrollsystem vom CI-Server abgefragt und prüft das Projekt auf Veränderungen nach. Bei erfassten Änderungen stößt der CI-Server einen CI-Build an. Bei einem solchen Build wird im ersten Schritt die dem CI-Server gehörende lokale Arbeitskopie, erhalten durch das Versionskontrollsystem, aktualisiert. Anschließend wird mit der durch ein Update aktualisierte Datei gebaut, das heißt automatisierte Tests werden kompiliert und ausgeführt. Der Build-Prozess wird dabei nicht durch den CI-Server durchgeführt, sondern mithilfe eines Build-Tools, wie beispielsweise Apache Ant, Apache Maven, Gradle etc.
Voraussetzung dafür ist sowohl die vollständig automatisierte Einstellung des Bauvorganges, als auch die Verwaltung der Build-Tool-spezifischen Dateien im VCS.
Zu den auf Build-Tool basierenden Dateien gehört unter anderem Mavens POM (engl. Project Object Model), eine XML-Datei, die Informationen über Projekt- und Konfigurationsdaten von Maven enthält und zum Bauen des Projektes genutzt wird.[Sch17a]
Nach Aufruf der Build-Tools klassifiziert der CI-Server den gesamten CI-Build und gibt über die eigene Website den Status an:

- Erfolgreich: Kein Fehlerfund beim Kompilieren und automatisierten Testen[150]
- Warnung: Der Kern des Kompilieren und der Tests ist erfolgreich. Nichtsdestotrotz existieren vereinzelnde Probleme, Warnungen und fehlgeschlagene Tests.[151]
- Fehler: Kritische Fehler sind vorhanden, die den CI-Build als gescheitert ansehen beispielsweise durch Abbruch des Compilers oder einer zu hohen Fehlerquote[152]

Der CI-Server ist aufgrund des Zugriffs auf das VCS in der Lage zu erfassen, welche Teilnehmer des Entwicklerteams Änderungen vorgenommen haben, die dementsprechend den neuesten CI-Build angestoßen haben.[153]

Abschließend kann die Software beispielsweise auf einem Stage-System zur Verfügung gestellt werden, worauf alle Teilnehmer benachrichtigt werden bezüglich des Build-Prozesses als auch die Möglichkeit besteht Build-Ergebnisse, Logs und Analysen zu betrachten. Die Resultate und Daten aller Builds können auf dem CI-Server in Form von einem Bericht gespeichert und auf der CI-Server-Website abgerufen werden. Bei fehlerhaften Commits wird

[150] [vgl. Feu12]
[151] [vgl. Feu12]
[152] [vgl. Feu12]
[153] [vgl. Feu12]

ein Entwickler zeitnah durch den CI-Server benachrichtigt. Zusätzlich stehen beispielsweise Projektleitern und Kunden aktuelle Informationen bezüglich Entwicklungsstand und ein Testsystem zur Verfügung. [154]
Möglich ist außerdem die Einstellung einer E-Mail Benachrichtigung über das Build-Resultat an die Entwickler und die dazugehörige Auskunft darüber, ob ihre Änderungen zu Problemen ausgelöst haben.
Der beschriebene typische CI-Ablauf kann durchaus verfeinert und modifiziert werden:

- Nightly Build, also das Auslösen von Builds nur zu bestimmten Zeiten mit umfangreicheren und zeitintensiveren Tests[155]
- Modular aufgebaute Projekte können „*nach jedem CI-Build eines Teilmoduls alle im Abhängigkeitsbaum nachgelagerten Teilmodule bauen, auch wenn diese selbst keine Änderungen aufweisen*“[Feu12]
- Die Möglichkeit besteht mehrere miteinander vernetzte CI-Server zu beschäftigen, um komplexere Tests zu parallelisieren oder auf unterschiedlichen Betriebssystemen ausführen zu können [156]

Somit lässt sich sagen, dass Continuous Integration Tools den Integrationsprozess einer Software durch automatisiertes Bauen und Testen validieren, um Probleme der Applikation zu erfassen und eine frühzeitige Rückmeldung zu geben, sodass Beteiligte des Entwicklungsprojektes auf diese Probleme rechtzeitig reagieren können.
Dadurch werden Risiken reduziert, die Produktqualität immens verbessert und der Build-Prozess umfassend dokumentiert. [157]

[154] [vgl. Kü13]
[155] [vgl. Feu12]
[156] [vgl. Feu12]
[157] [vgl. Sch11b, S.38]

Einbindung von CI-Server Messgrößen

Zu beachten gilt die Einbindung von CI-Server Messgrößen, um verlässliche Aussagen über den Zustand der Software und die Qualität des Quellcodes treffen zu können. Zu den verwendeten Messgrößen gehören beispielsweise:

- formale Eigenschaft des Programmcodes
  ↬ Überschreitung von Funktionen hinsichtlich Komplexität und Umfang, Einhaltung der Code-Konventionen etc.
- Testabdeckung
  ↬ Vollständigkeit der Tests messen, d.h. während der Ausführung von Tests wird erkannt, „[...]*welche Teile der Codebasis durch die Tests auch tatsächlich ausgeführt werden* [...]“[Hof16],
  Ermittlung von überflüssigen Teilen des Programmcodes, wenn Teile des Programmcodes die Akzeptanztests nicht erreichen. Akzeptanztests, auch bezeichnet als Abnahmetests, werden nach Anforderungen eines Benutzers überprüft. Die Software soll dadurch die Erwartungen eines Users erfüllen.
- Fehlermuster
  ↬ Nicht korrekt initialisierte Variablen, mehrfach identischer Code etc.
- architekturelle Anforderungen ↬ Unerwünschte Abhängigkeiten zwischen Teilen des Programmcodes, Annäherung zur Zielarchitektur etc.
- Performance
  ↬ Dauer der Request-Bearbeitung auf der Zielinfrastruktur, Überlastung des Systems beispielsweise durch zu viele Nutzer [158]

4.6.3 Vorteile von Jenkins

Jenkins unterstützt diverse Versionskontrollsysteme wie StarTeam, Subversion, CVS, Git, AccuRev etc und baut Projekte mithilfe von Build-Tools wie Apache Ant, Apache Maven oder Gradle. Das Besondere bei Jenkins ist die Erweiterbarkeit durch diverse Plug-ins. Jenkins bietet eine Reihe von Plug-ins aus unterschiedlichen Bereiche an wie VCS, SSH (Secure Shell) ist ein Unix-spezifisches Netzwerkprotokoll und ermöglicht den sicheren Zugriff auf einen entfernten Computer über ein unsicheres Netzwerk, wie dem Internet[159], Build-Tools, Benachrichtigungssysteme, Analysetools, Library Plug-ins und viele weitere.

[158] [vgl. Feu12, S.2]
[159] [vgl. Rou14]

Das für die vorliegende Thesis primär benutzte Selenium Plug-in sowie die Einbindung des Test Results Analyzer wurden im Kapitel 4 geschildert.

Außerdem liefert Jenkins einen Überblick für Entwickler, Admins oder Projektmanager, indem es alles hinsichtlich des Projekts in einer übersichtlichen Weboberfläche zusammengefasst und den jeweiligen Status anzeigt.[160]

4.7 Gradle

Gradle ist ein Build-Management-Werkzeug, dessen Konzept ähnlich wie von den weitverbreiteten Build-Management-Tools Apache Maven und Ant ist, jedoch auf der objektorientierten Programmier-und Skriptsprache Groovy basiert. Durch Gradle kann eine Software automatisiert gebaut, getestet und ausgeliefert werden.
Gradle wird, wie auch in der vorliegenden Arbeit, für das Bauen von auf Java basierten Selenium-Projekten verwendet, was im Kapitel 6 thematisiert wird. Für das professionelle Testen wird Selenium genutzt.

4.7.1 Grundsätzliche Aufgaben von Build-Management-Werkzeugen

Primär geht es bei Build-Management-Werkzeugen um

„[...] die Übersetzung von Quelltexten in der richtigen Reihenfolge und der darauf folgende Aufruf des Linkers, um ein ausführbares Programm oder eine Bibliothek zu erzeugen.“[Bau13, S.4]

Zu den weiteren Aufgaben eines modernen Build-Management-Systems gehört auch die Erstellung einer lokalen Kopie des Quelltextes durch ein Versionsverwaltungssystem, Checkout eines Programmcodes. Das Auslagern kann anschließend in einem Build-Server, wie z.B. Jenkins, stattfinden. Ebenso ist die Ausführung von unterschiedlichen Arten von Tests wichtig. Abhängig von der Testart müssen diese im Ablauf des Builds aufgeführt werden. Zu den Testarten hierfür gehören
„*Unit-Tests für einzelne Klassen, Modul-oder Komponententests, Integrationstests, funktionale als auch nichtfunktionale Systemtests und automatisierte Akzeptanztests*“.[Bau13, S.4]

[160] [vgl. Mei14]

Auch das Ausgeben von ausführlichen Testreports, die die Testresultate auf einer übersichtlichen Weise zusammenfassen ist ein Bestandteil von Build-Management-Werkzeugen. Außerdem ist das Packen der Ergebnisse eine weitere Funktion. In einer auf Java basierten Umgebung handelt es sich abhängig von der Anwendung oder Bibliothek dabei um Dateien in den Formaten JAR, WAR oder EAR.[161] Die Ergebnisse werden zu Testsystemen transferiert und nach jeweiliger Art des Tests ausgeführt, üblicherweise übernommen durch andere Werkzeuge. Auch polyglotte Projekte lassen durch moderne Build-Management-Werkzeuge realisieren .
Diese zeichnen sich durch die mögliche Verwendung von multiplen Sprachen und Nutzung auf mehr als eine Entwicklungsumgebung aus. Ebenso können Dokumentationen sowie Release Notes, erstellt werden, was ebenfalls zu den Aufgaben von neuartigen Build-Management-Werkzeugen gehört.[162]

4.7.2 Vergleich zu Apache Ant und Maven

Gradle vereint die postiven Eigenschaften sowie Funktionalitäten von Maven und Ant, die nachfolgend beschrieben werden.
Ant (engl. Another neat tool) galt anfangs als wegweisend für das im Jahre 1995 erschienene Java-Umfeld. Besonders die Erzeugung eines Java-Programms durch das Beschreiben einzelner Schritte in Form von einer XML-Datei (seit Version 1.1 erschienen in 2001) als auch die Möglichkeit die Funktionalitäten durch Erweiterungen (engl. Extensions) machte ältere komplexere Build-Management-Werkzeuge, wie beispielsweise *make*, überflüssig. Der Schwachpunkt bei Ant besteht darin, dass keine Abhängigkeits-und Versionsverwaltung für Java-Archive vorhanden waren. Außerdem fehlte die Idee eines Build-Prozesses, der sich aus unterschiedlichen, klar getrennten Phasen zusammensetzt, um die nötigen Schritte für das Resultat nachzuvollziehen und umzusetzen.
Jedoch gab es durch das im Jahre 2006 implementierte Apache-Werkzeug namens Ivy, welches inzwischen ein Subprojekt von Ant ist, eine Abhängigkeits- und Versionverwaltung.

Das darauf folgende Build-Management-System Maven fokussierte sich auf die fehlenden Funktionalitäten bei Ant und definiert einen Prozess in Form eines Lebenszyklus mit aufeinanderfolgenden Phasen als auch ein Versionverwaltungssystem von Java-Archiven, seit der veröffentlichten Version 1.0 in 2004. Maven folgt anders als bei Ant einen deklarativen Ansatz, nämlich statt der Frage nach dem Wie etwas durchgeführt werden soll, genügt es aus zu sagen, **was** umgesetzt werden soll. Dieses geschieht durch die Verwendung von einer XML-Datei , die als POM (engl. Project Obeject Model) bezeichnet wird. Dafür müssen

[161] [vgl. Bau13, S.4]
[162] [vgl. Bau13, S.4]

Konventionen gegeben sein, um die Ausführung der Schritte zu ermöglichen.
Die Erweiterung durch Plug-ins ist ebenfalls möglich, die allerdings

„[..] ihre jeweils eigenen Konventionen mitbringen.“[Bau13, S.3]

Problematisch für das Build-Management sind die äußerst strengen Vorgaben bei Maven. Durch derartig rigide Vorgaben bezüglich der Prozesse sowie bei den Konventionen sehen Maven-Projekte identisch aus.
Somit kann es vorkommen, dass komplexe Projekte oder

„[..] individuelle Anpassungen von Zielen beim Erstellprozess“[Moh16]

mit dem Build-Management kollidieren.

Auf der einen Seite steht Ant mit dem implementierten Werkzeug Ivy zur Verfügung, welches zwar durch keinerlei angewiesenen Vorgaben eine große Freiheit bietet, allerding keine Prozesse unterstützt und eine imperative Beschreibung aller Schritte benötigt. Auf der anderen Seite ist Maven vorhanden, was zwar einen Prozess definiert und ein deklaratives Paradigma verfolgt, aber durch die unflexiblen Vorgaben das Build-Management auf Dauer problematisch macht.
Gradle kombiniert die Vorteile aus Ant und Maven und adressiert gezielt deren fehlenden Eigenschaften. Gradle verfolgt ein deklaratives Muster mit leicht veränderlichen Konventionen. Durch die Einbindung von Maven- und Ivy-Repositories wird ein hervorragendes Abhängigkeits-und Versionverwaltungssystem geboten. [163]

4.7.3 Prinzipien von Gradle

Gradle verfolgt hauptsächlich zwei Ansätze:

↪**„Convention over Configuration“:**
Hierbei handelt es sich um ein einfaches Softwarekonzept, welches durch vorkonfigurierte Vorgabewerte das Programmieren erleichtert und die Produktivität steigert, anstatt eigene Konfigurationen zu entwickeln. Das Ziel von Convention over Configuration ist es den Entwickler dabei zu unterstützen leichter Entscheidungen zu treffen und die Komplexität hinsichtlich der Konfiguration als auch der Programmierung zu nehmen. Dadruch ist die Realisierbarkeit und Umsetzung von vielen Aufgaben in kürzerer Zeit gegeben. Mithilfe

[163] [vgl. Bau13, S.1-3]

derartiger Methodik wird für

„[..] *einen deklarativen Stil des Build-Skripts*“[Bau13, S.6]

gesorgt.

↪**„Don’t Repeat Yourself (DRY)“:**
Gradle vermeidet das Prinzip des Wiederholens. Das sorgt dafür, dass

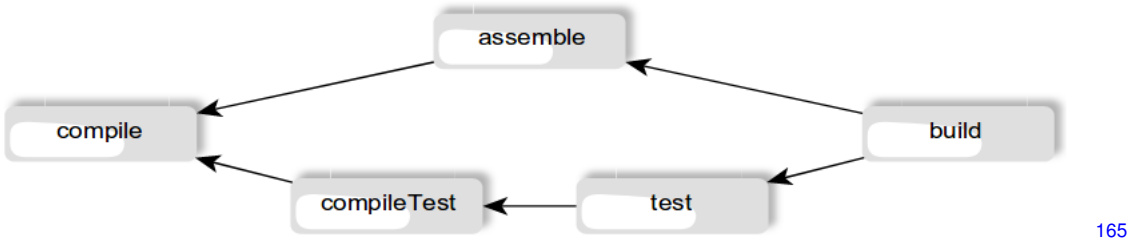
„*überflüssige Redundanzen und potentielle Fehlerquellen*[...]“verringert werden. Auch für eine „[...] *präzise, kurze Syntax*“in den Build-Codes wird gesorgt. [Bau13, S.6]

Diese Herangehensweisen ermöglichen das Beschreiben von kurzen Build-Skripten in komplexeren Entwicklungsumgebungen, wodurch sich die Fehlerrate verringert und die Wartbarkeit der Quellcodes einfacher wird. So können wiederum andere Entwickler die Skripte leichter nachvollziehen und modifizieren.
Gradle nutzt, im Gegensatz zu Mavens statischem Modell, ein dynamischen Workflow. Bei solch einem dynamischen Zyklus hat der Anwender Einfluss auf die unterschiedlichen Phasen des Build-Prozesses. Es können Entscheidungen über das Anfügen von selbsterstellten Phasen und die Relevanz einzelner Phasen getroffen werden.
Generell besteht Gradle aus zwei Verarbeitungsphasen, um den dynamischen Workflow zu erreichen:

1. Konfigurationsphase: In der Konfigurationsphase wird das Build-Skript verarbeitet. Zusätzlich wird der Inhalt des Skriptes interpretiert und gemäß der Anweisungen, die im Skript beinhaltet sind, am Modell angepasst.
2. Ausführungsphase: Bei der Ausführungsphase geht es um die Abarbeitung „[...] einzelner Schritte des Builds anhand des erzeugten Modells.“[Bau13, S.6]

Gradle bedient sich an einem Abhängigkeitsgraphen, um das Modell darzustellen. In diesem „[...] gerichteten azyklischen Graph[...]“[Bau13, S.6], auch als DAG bezeichnet (engl. directed acyclic graph), werden die Ziele bzw. Tasks von Gradle im Form von Knoten dargestellt. Abhängigkeiten werden durch Kanten angezeigt, die die einzelnen Knoten verbinden. Die Kanten stellen dar, welcher Task welchem anderen Task verfolgen kann.[164]
Folgendes Modell veranschaulicht einen beispielhaften gerichteten azyklischen Graph:

[164][vgl. Bau13, S.6]



[165]

Abbildung 4.8: *Beispiel eines vereinfachten Abhängigkeitsgraphen*

Bei obiger Abbildung wird eine topologische Sortierung angewandt, die typischerweise für die Verwaltung von Zielen oder Knoten bei Gradle verwendet wird.
Beispielsweise folgt die Bearbeitung des Knotens *test* erst dann, wenn der Knoten *compileTest* erfolgreich absolviert wurde, welcher wiederum abhängig von der erfolgreichen Durchführung des Knotens *compile* ist.
Durch solch einer topologischen Sortierung lässt sich die Ausführungsreihenfolge äußerst simpel bestimmen.[166]

Die Nutzung und Einbindung von dem Build-Management-Werkzeug Gradle wird ausführlich im Kapitel 6 behandelt.

4.8 Google Cloud Platform

Das vorliegende Kapitel thematisiert die Google Cloud Platform, ein wichtiger Bestandteil dieser Bachelorthesis.

4.8.1 Einblick

Google Cloud Platform, erschienen am 6.Oktober 2011 und veröffentlicht von Google, ist eine Sammlung von „Public-Cloud-Services“, welche auf das Konzept des Cloud Computings basieren. Die verfügbaren Dienste und Produkte der Google Cloud Platform decken Kategorien wie Computing, Storage, Big Data, Machine Learning ab.[167] Die GCP ermöglicht das

[165] [Bau13, S.7]
[166] [vgl. Bau13, S.7]
[167] [vgl. Rou16b]

Betreiben von Applikationen und Daten durch die von Google eigene bereitgestellte Infrastruktur.[168]

4.8.2 Einschub: Cloud Computing

Cloud Computing ist „[...] *die Bereitstellung und die Nutzung* [...]“[Bau11, S.29] von IT-Ressourcen über das Internet oder, meist bei größeren Unternehmen, ein lokales Network (Intranet)

„[...] *auf Basis einer Cloud* [...]“.[Sei10, S.3]

Die Cloud beschreibt die hinter den Diensten, bereitgestellt durch einen Provider, stehende Hardware und Systeme. Cloud Computing stellt eine IT-Infrastruktur zur Verfügung, die

„[...] *abstrahiert, netzwerk-zentriert und leicht skalierbar und eine vom Provider verwaltete Struktur ist, die meist auf Virtualisierung mehrerer Maschinen auf einer physischen Maschine beruht.*“[Sei10, S.4]

Softwarepakete, einzelne Applikationen, Rechnerleistung und Speicherplätze - all diese Arbeitsbereiche werden aus einer externen virtuellen Umgebung bezogen statt auf der eigenen Festplatte oder auf firmeninternen Rechner.
Abhängig vom Bedarf des Services kann der zuständige Provider dem Anwender einfach und schnell

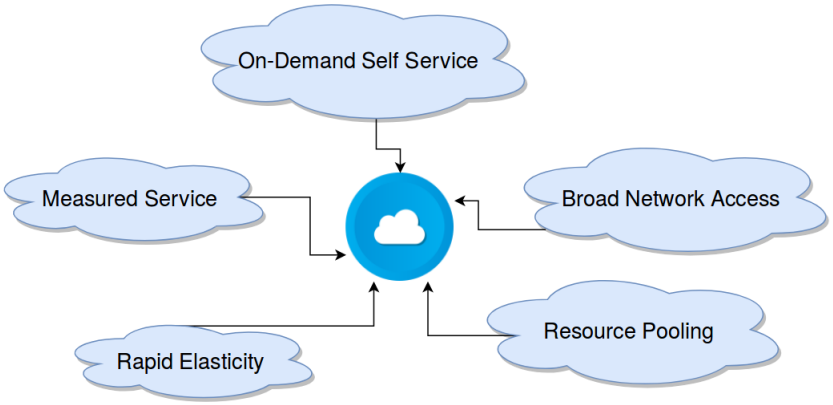
„[...] *theoretisch unlimitierte Ressourcen zur Verfügung stellen.*“[Sei10, S.4]

Ausschließlich wird nur für die Dienste vom Kunden entlohnt, die für ein Projekt verwendet wurden bzw. basierend auf Bedarf und Verbrauch relevant waren. Dieses Konzept wird auch als Pay-per-Use Preismodell bezeichnet.[169]

[168] [vgl. Hol]
[169] [vgl. Mic11]

4.8.3 Merkmale

Cloud Computing lässt sich nach NIST (National Institute of Standards and Technology) durch fünf wesentliche Merkmale identifizieren und definieren, die im folgenden grafisch abgebildet sind:



[170]

Abbildung 4.9: *Fünf signifikante Merkmale von Cloud Computing*

Laut dem „National Institute of Standards and Technology“(NIST) lässt sich Cloud Computing durch fünf signifikante technologische Eigenschaften definieren. Im vorherigen Kapitel 4.8.2 wurden bereits einige davon erwähnt, jedoch werden diese anhand der Abbildung 4.9 und den dazugehörigen Stichpunkten verdeutlicht und näher erläutert.

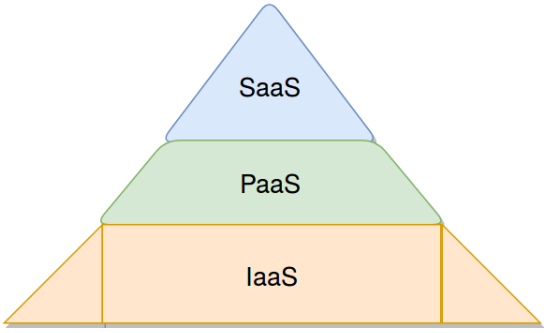
- On-Demand Self Service: Verbraucher können die IT-Kapazitäten selbstständig zusammenstellen ohne Einbindung des Anbieters in den Prozess.[171]
- Broad Network Access: Über das Internet bzw.
  „[...] *Zugriff auf ein standardbasiertes Netz*[...]“sind die Services verfügbar. Der Verbraucher kann aufgrund dessen seine Arbeit „[...] *unabhängig von Standort und Zeit erledigen.*“[Buc12]
- Resource Pooling: Jede Ressource, die gespeichert wird, wird im Anschluss gebündelt und multimandatenfähig zur Verfügung gestellt. Dieses wird nach Bedarfsfall kategorisiert. Dadurch wird sichergestellt, dass jeder Nutzer alle Ressourcen inklusive Verwaltung, Business Continuity und technische Instandhaltung erhält, ohne eine aktive Rolle einnehmen zu müssen.[172]

[171] [vgl. Buc12]
[172] [vgl. Buc12]

- Rapid Elasticity: Die Kapazitäten der Cloud sind dynamisch und werden jedem Nutzer rapide zur Verfügung gestellt. Außerdem gibt es eine Funktion, die es jedem Nutzer erlaubt, Serviceeinheiten zu skalieren, sie also nach Bedarfsfall abzurufen, hinzuzufügen oder zu entfernen.[173]
- Measured Service: Bei der nutzungsgerechten Abrechnung werden die verwendeten Ressourcen automatisch kontrolliert und optimiert durch ein Messsystem (engl. Metering). Dadurch wird eine Kostentransparenz zwischen Provider und Verbraucher sichergestellt.[174]

4.8.4 Ebenen

Das folgende Pyramidendiagramm bildet die hierarchische Service-Ebenen im Cloud Computing ab:



[175]

Abbildung 4.10: *Cloud-Service Ebenen*

- Infrastructure as a Service (IaaS): IaaS spiegelt die „[...] *reine Infrastruktur wie Leistung und Speicher* [...]“wieder. Hauptsächlich werden hierbei Speicher- und Rechenleistung auf virtuellen Servern als auch Netzwerkinfrastruktur abhängig vom Gebrauch zur Verfügung gestellt. Der große Vorteil bei IaaS ist „[...] *die Skalierbarkeit*[...]“.[Buc12]
- Platform as a Service (PaaS): Bei der mittleren Ebene PaaS handelt es sich um die Entwicklungs-Dienste. Die Anwender haben die Möglichkeit ihre eigenen Anwendungen zu entwickeln und den Programmcode auszuführen.[176]

[173] [vgl. Lab12, S.4]
[174] [vgl. Buc12]
[175] Erstellt mit draw.io
[176] [vgl. Buc12]

- Software as a Service (SaaS): Auf der Spitze des Ebenen-Modells ist SaaS. Hier hat der Vebraucher den Anspruch auf die einzelnen Applikationen, die vom Provider in der Cloud bereitgestellt werden.[177]

4.8.5 GCP-Infrastruktur

Die Infrastruktur von Google besteht grundsätzlich aus zwei Ebenen- einer physikalischen und einer abstrakten. Die **physikalische** Ebene beinhaltet:

- **Data Centers:** In den Rechenzentren der GCP, die als hocheffizient und sicher gelten, werden „über mehrere physikalischen Festplatten “[Bü12] die Daten berechnet, gelagert und gespeichert. Visualisiert und beschrieben wurde IaaS bereits im Abschnitt 4.8.4.
  Aktuell werden die Rechenzentren in zwölf Regionen betrieben. Weltweit sind drei davon in Europa (London, Belgien, Frankfurt). Weiterhin sind im Laufe der nächsten zwölf Monate eine Erweiterung um fünf RZs geplant.[178]
- **Backbone Network:** Der Backbone, auch als „Rückgrat eines Netzwerks“bekannt, ist eine zentrale Leitung, die jeweils zwei Standorte verbindet. Die Summe der Verbindungen stellt ein vernetztes Data Center dar.
  Dabei verbinden sich lokale Leitungen mit diesen Hauptleitungen und addieren ihre Daten.
  Grundsätzlich sind Backbones wegen der hohen Übertragunskapazität mit Glasfaserleitung ausgestattet.[179]
  Das Backbone Network von Google gilt als sehr durchdachtes, globales, redundantes und durch Glasfasernetz realisiertes Network.[180]
- **Points of Presence:** Point of Presence ist der Punkt an dem zwei oder mehrere unterschiedliche Networks oder Geräte eine Verbindung aufbauen, um miteinander zu kommunizieren. Dabei bezieht sich der PoP auf einen Access Point (zu Dt. Zugangspunkt), Ort oder einer Einrichtung, die eine Verbindung mit dem Internet herstellt.[181]
  Google verfügt über 100 Edge PoPs in 33 Ländern, welche das eigene Network mit

[177] [vgl. Buc12]
[178] [vgl. Sim16]
[179] [vgl. Rou16a]
[180] [vgl. Sim16]
[181] [vgl. Sim16]

den restlichen Networks des Internets via peering verbindet.[182] Bei peering findet ein Austausch von Daten zwischen Internetdienstanbieter statt.[183]

- **Edge Caching:** Bei Edge Caching handelt es sich grundsätzlich um Zwischenspeicherung oder Pufferung von Daten, um eine schnellere Interaktion mit den Kunden zu ermöglichen.[184]

[182] [vgl. Gri]
[183] [vgl. Rou07]
[184] [vgl. Sim16]

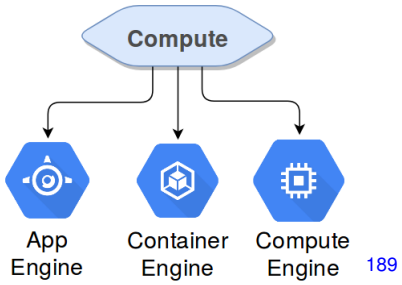
Die **abstrakte** Stufe der Infrastruktur ist unterteilt in:

- **Globale Ressourcen**:
  Ressourcen, die global sind, umfassen alle verschiedenen Regionen. Wenn beispielsweise eine IP Adresse für eine Web-Applikation oder ein Network erstellt wird, handelt es sich um eine globale Ressource. Dabei werden Rechenzentren einer Zone mit dem Network verbunden. Die Rechenzentren können mit allen anderen Rechnern interagieren und kommunizieren, die mit diesem globalen Network verbunden sind. Google stellt Load Balancer, zu Dt. Lastverteiler, standortübergreifend zur Verfügung, also regionale Lastverteiler, die sich auf die einzelnen Zonen konzentrieren. Das optimiert die Verfügbarkeit, Performance und Effizienz der Ressourcen.[185]
- **Regionale Ressourcen**:
  Regionen sind geografische Areale, die mehrere Zonen enthalten. Ein Beispiel für einer der Dienste, die auf der regionalen Ebene stattfindet, ist Cloud Load Balancer.
  „*Google Cloud Load Balancing ist ein vollständig verteilter, softwarebasierter verwalteter Dienst für den gesamten Traffic und dient als skalierbarer Lastenausgleich.*“[Picl]
  Wenn ein Cloud Load Balancer eingestellt ist, werden Anfragen an alle verschiedenen Zonen versendet, sodass die Gesundheit aller Rechner kontrolliert werden kann. Bei Aufall eines dieser Computer werden keine Anfragen übermittelt und die anderen Computer der restlichen Zonen diesbezüglich informiert.[186]
- **Zonale Ressourcen**:
  Eine Zone ist ungefähr gleichzusetzen mit einem Data Center. Untersucht wurde dieser Aspekt im Abschnitt 4.8.5.
  In Zonen befinden sich unter anderem die Rechenleistungen und Festplatten. Wenn ein System, wie die GCP, eine hohe Verfügbarkeit liefern soll, müssen Ausfallmaßnahmen einer Zone berücksichtigt werden. Dementsprechend ist die Verteilung der Rechenleistung und des Speichers auf diversen Zonen notwendig, um bei Abruf in jedem Fall verfügbar zu sein und für einen Ausgleich sorgen zu können. In einer Zone sind die Ressourcen der Cloud mit den verfügbaren GCP-Diensten verknüpft. Beispielweise findet der GCP-Dienst Compute Engine, der die Bereitstellung skalierbarer virtueller Hochleistungsmaschinen (VMs)[187]

[185] [vgl. Sim16]
[186] [vgl. Sim16]
[187] [vgl. Sim16]

4.8.6 Verfügbare Dienste

Die von der Google Cloud Platform bereitgestellten Kernprodukte und -dienste lassen sich kategorisieren in „Compute, Storage, Big Data und Machine Learning“:[188]



[189]

Abbildung 4.11: *Google Cloud Plattform* - Compute

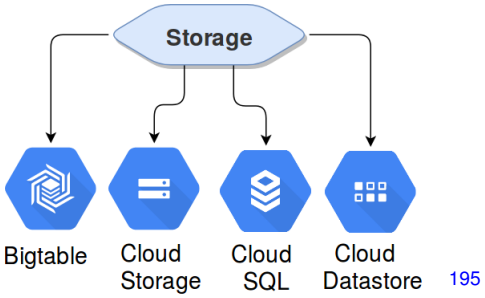
Bei **Compute** wird eine Palette an globalen lastausgleichenden Diensten bis hin zu skalierbaren einzelnen virtuellen Maschinen zur Verfügung gestellt. In dieser Kategorie sind drei Optionen von großer Wichtigkeit.[190]

Die **App Engine**, ein Dienst der PaaS-Ebene, bereits erwähnt im Kapitel 4.8.4, ermöglicht den Anwendern den Zugriff auf die skalierbaren Hosting-Angebote von Google.[191] Die Plattform dient zur Erstellung von Web-und Mobileapplikationen, bietet eine vertraute Umgebung in gängigen Programmiersprachen, wie Node.js, C#, Ruby, Python, PHP und Java, und skaliert die Infrastruktur automatisch, sodass sich der Entwickler vollständig auf das Programmieren der eigenen Software konzentrieren kann. Google kümmert sich dabei auch um die Softwareaktualisierung, Patching, Skalierung.[192]

Container Engine „[...] *ist ein leistungsfähiges Clusterverwaltungs-und Orchestrierungssystem zur Ausführung von Docker-Containern* [...]“[Picp], basierend auf Kubernetes, ein quelloffenes Management-System von Google.
„Kubernetes automatisiert Deployment, Skalierung, Wartung und Betrieb verschiedener Container-Anwendungen in unterschiedlichen Clustern und Nodes.“[Rou16d]

[188] [vgl. Sim16]
[189] Erstellt mit draw.io
[190] [vgl. Sim16]
[191] [Rou16c]
[192] [vgl. Pick]

Die **Compute Engine** stellt skalierbare VMs, virtuelle Maschinen, zur Verfügung und dient zur Auslagerung von leistungsintensiven Prozessen und Anwendungen.[193] Google verwaltet bzw. kümmert sich dabei um die Hardware.[194]



[195]

Abbildung 4.12: *Google Cloud Plattform* - Storage

In der Kategorie Storage werden Datenbanken verwaltet und Objekt-sowie Archivspeicher angeboten für unterschiedliche Anwendungen.[196]

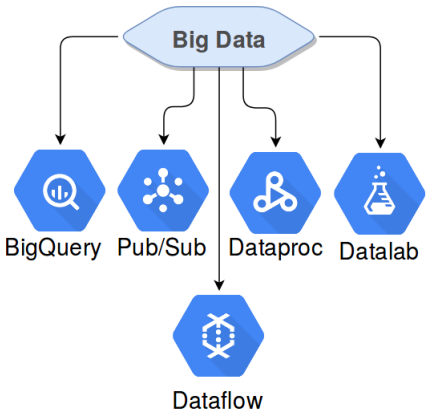
Bigtable ist *„[...] eine skalierbare, vollständig verwaltete, spaltenorientierte NoSQL-Datenbank, die sich sowohl für den Echtzeitzugriff als auch für Analysearbeitslasten eignet.“*[Pics]

Die **Cloud Storage** ist verantwortlich für das Speichern von „[...] *großen, unstrukturierten Datenmengen* [...]“(z.B. Bilder ,Videos etc.).[Rou16c]
Dieser skalierbare, langlebige und vollständig verwaltete Blob- (engl. Binary Large Objects), Multimedia-Objekte z.B. Audioaufzeichnung oder Bilddatei, und Objektspeicher stellt Live-daten bereit, wie beispielsweise Streamen von Multimedia-Inhalten. Außerdem sichert und archiviert dieser Dienst die Daten und zeichnet sich durch seine Kosteneffizienz aus.[197]

Der Dienst des **Cloud SQLs** ist eine auf Google's starke und zuverlässige Infrastruktur basierter „MySQL- und PostgreSQL-Datenbankdienst“.[198]

Der **Cloud Datastore** ist als „[...] vollständig verwaltete und skalierbare NoSQL-Dokumentendatenbank [...]“zuständig für Mobil-und Webanwendungen.[Pics]

[193] [vgl. Hol]
[194] [vgl. Picl]
[195] Erstellt mit draw.io
[196] [vgl. Sim16]
[197] [vgl. Pics]
[198] [vgl. Pics]



[199]

Abbildung 4.13: *Google Cloud Plattform* - Big Data

Mithilfe von Google's Big Data können Daten aufgenommen, verarbeitet, gelagert und analysiert werden durch ein „[...] vollständig verwaltetes Data Warehouse."[Picc]

Das Data Warehouse, auch bezeichnet als DW oder DWH, ist eine Sammlung von Unternehmensdaten, bestehend aus verschiedenen Quellen[200], die vor allem „[...] *zwecks Analyse und betriebswirtschaftlicher Entscheidungshilfe dauerhaft gespeichert wird.*"[Cor]

BigQuery ist ein Analysedienst für riesige Mengen von Daten, auch als Big Data bezeichnet, im Terra- und Petabytebereich. Die Datenmengen können über einer Web-Oberfläche, REST-API oder via Kommandozeile durch SQL-Befehle untersucht werden. Beispielsweise können „*Logfiles analysiert werden oder Trends in Verkaufszahlen erkannt werden*"[Ihl12]. Daten werden als CSV-Format hochgeladen und in tabellarischer Form gesichert. Die Kosten dieses Dienstes orientieren sich an das PAYG (Pay-As-You-Go) - Modell oder Pay-per-Use, beschrieben im Abschnitt 4.8.2 oder bei vorhersehbaren Kosten besteht die Möglichkeit[201], „[...] *einen monatlichen Pauschalpreis* [...]"zu wählen.[Picc]

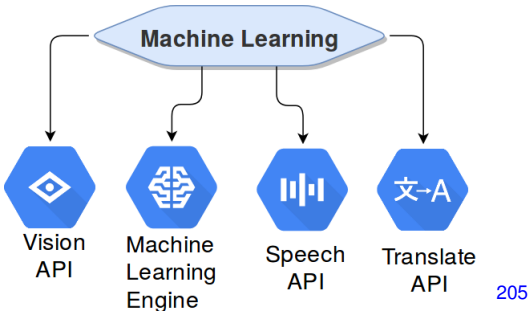
Pub/Sub ist ein Dienst um Nachrichten zwischen unabhängigen Anwendungen in Echtzeit zu senden und zu empfangen."[202]

[199] Erstellt mit draw.io
[200] [vgl. Cor]
[201] [vgl. Picc]
[202] [vgl. Picj]

Dataflow ist ein Datenverarbeitungsservice zum automatischen Kreiieren von Workflow-Daten. In derartigen Workflow werden Daten erfasst, bearbeitet und „[...] *sowohl im Batch-als auch im Streaming-Modus*“[Neu14] analysiert.
Mit dem Dienst gelingt es dem Anwender leichter Erkenntnisse aus Daten zu ziehen und praktische Lösungen abzuleiten.[203]

Bei **Dataproc** handelt es sich um eine Plattform, bei der „[...] *Apache Hadoop-, Apache Spark-, Apache Pig- und Apache Hive-Dienste* [...]“[Picd] verwaltet werden.
Aufgrund der anpassbaren und schnellen Erstellung von Cluster sowie Verarbeitung von Daten kann sich der Verbraucher auf „[...] *die Analyse anstatt auf die Infrastruktur konzentrieren.*“[Picd]

Mithilfe des Entwicklertools **Datalab** können Daten untersucht, analysiert und visualisiert werden.[204]



[205]

Abbildung 4.14: *Google Cloud Plattform* - Machine Learning

Die Kategorie Machine Learning bietet moderne Dienste im Bereich „[...] *maschinelles Lernen mit bereits trainierten Modellen* [...]“[Pice] an.
Der User hat zusätzlich die Option, Modelle nach seinen Wünschen zu personalisieren.[206]

Die **Vision API** wird zur Bildanalyse verwendet„[...] *um auf maschinellem Lernen basierende Bilderkennung in ihre Anwendungen zu integrieren.*“[Gre16]
Dabei werden Gesichter sowie Objekte in den Bildern erkannt und die Bilder in Kategorien

[203] [vgl. Neu14]
[204] [vgl. Lar15]
[205] Erstellt mit draw.io
[206] [vgl. Hol]

klassifiziert.[207]

Die **Machine Learning Engine** wird genutzt, um auf einer einfachen Weise eigene Modelle für maschinelles Lernen zu entwicklen. Diese Modelle können mit Daten jeglicher Größe und jeden Typs arbeiten.[208]

Die **Speech API** wandelt „[...] *gesprochene Wörter in digitalen Texten* [...]“[Picg] um. Dieser automatische Spracherkennungsdienst erkennt mehr als 80 Sprachen.[209]

Mithilfe der **Translation API** können auf einer schnellen und dynamischen Weise beliebige Strings automatisiert in über 100 unterstützende Sprachen übersetzt werden. Die integrierte Spracherkennung kann bei Nichterkennung der Quellsprache genutzt werden.[210]

4.8.7 Überwachung durch Stackdriver Monitoring

Google Stackdriver ist eine „[...] *Cloud-Monitoring- und Diagnose-Plattform* [...]“von Google.[Wü17]
Sie ermöglicht sowohl die Überwachung (engl. Monitoring) als auch das Diagnostizieren und Protokollieren „[...] *für Anwendungen, die auf der GCP und in Amazon Web Services ausgeführt werden.*“[Pict]
Verbraucher erlangen dadurch umfassende Performance Metriken und Metadaten über ihre Anwendung und virtuellen Maschinen, die auf der GCP gespeichert und betrieben werden.[211]
Der Google Stackdriver besitzt mehrere grundlegende Eigenschaften.
Die für die Thesis relevante Funktion ist das Betriebszeitmonitoring (Stackdriver Monitoring).
Laut der Homepage der GCP heißt es:
„*Stackdriver Monitoring bietet Endpunktprüfungen für Webanwendungen und andere über das Internet zugängliche Dienste, die über [..] Cloudumgebung ausgeführt werden. Betriebszeitprüfungen können konfiguriert werden, die mit bestimmten URLs, Gruppen oder Ressourcen, wie etwa Instanzen oder Lastenausgleichsmodulen, verknüpft sind.*“[Picm]

Zu den Dienste der Google Cloud Plattform, die mithilfe von Stackdriver Monitoring automatisch überwacht und erkannt werden können, zählen beispielsweise Cloud SQL, App

[207] [vgl. Clo15]
[208] [vgl. Picf]
[209] [vgl. Picg]
[210] [vgl. Pich]
[211] [vgl. Rou16e]

Engine,Compute Engine, **BigQuery** uvm.
Das Stackdriver Monitoring überwacht standardgemäß Rechnerressourcen für „[...] *jede Instanz, wie etwa CPU-Auslastung und die Ein-und Ausgänge des Netzwerks.*“[Pic17a]
Auch Drittanwendungen, wie beispielsweise Apache-Web-Server, MySQL, Tomcat etc., lassen sich durch Konfiguration überwachen.[212] Durch verfügbare Dashboards, die die Benutzer personalisieren können, können Messwerte sowie Metadaten einbezogen werden. Mithilfe von Darstellungstools können zusätzlich diese Daten übersichtlich visualisiert,je nach Anforderung justiert und die „[...] *Messwerte in Echtzeit erstellt* [...]“werden. Dadurch lassen sich wiederum„[...] *potentielle Probleme rechtzeitig erkennen* [...]“und verhindern.[Pict]
Ein weiteres Merkmal des Stackdriver Monitorings ist das Benachrichtigungssystem.
Benutzer können über konfigurierbare Benachrichtigungen per E-Mail, SMS-Nachricht oder weiteren „[...] *Instant-Messaging-Diensten (wie z.B. Slack)* [...]“[Pict]
informiert werden, wenn ein Performance-Problem erkannt wurde.[213]

Diese Thesis ist spezialisiert auf Google Cloud BigQuery, ein Produkt der Google Cloud Platform aus der Kategorie Big Data, in Verbindung mit der Überwachungsfunktion vom Stackdriver Monitoring.

4.8.8 Vor- und Nachteile der GCP für Unternehmen

Die Einführung der Google Cloud Platform (GCP) in einem Unternehmen weist viele Vor-und Nachteile auf.[214]
Durch die zahlreichen Dienste ist die GCP in allen drei Service-Ebenen des Cloud Computings vertreten.[215]
Der Grundgedanke hinter der GCP liegt darin, dass sich ein Verbraucher auf eigene Projekte konzentrieren kann während Google die Verwaltung der Infrastruktur, die Bereitsstellung von Server und die Netzwerkkonfiguration übernimmt.[216]
Die Einbindung der GCP sowie bei anderen Cloud-Anbietern in ein Unternehmen ist dennoch mit Vorsicht zu genießen. Langfristig muss analysiert werden, wie mit firmenbezogene Daten umgegangen wird. Dadurch dass Datenmengen in die Obhut eines Cloud-Dienstleisters, in dem Fall Google, überlassen werden, begibt sich der Verbraucher in einer unweigerlichen Abhängigkeit. Auch hinsichtlich der IT-Kompetenzen, wenn diese unzureichend oder durch

[212] [Pic17a]
[213] [Pict]
[214] [vgl. Min17]
[215] [vgl. Min17]
[216] [vgl. Min17]

zu wenige Fachkräfte besetzt ist, ist das Unternehmen, zu einem Großteil, angewiesen auf dem Cloud-Anbieter.[217]

Die moderne und als zukunftssicher geltende **Infrastruktur** von Google zeichnet sich unter Anderem durch die Compute-Engine Instanzen aus.
Die Umstellung auf VM sorgt für einen Ausgleich der Lasten und Wartbarkeit der Hardware. Die Instanzen, die einen SSD-Arbeitsspeicher von bis zu 1,5 Terrabyte besitzen, können selbst bei extremer Auslastung auf physikalische Maschinen verschoben werden. Das Bewegen einer VM zwischen zwei Hosts z.B. Server geschieht dabei ohne Unterbrechung der Verbindung mit dem Nutzer.
Die Nutzbarkeit von lokalen SSDs sind mit einer Lesegeschwindigkeit von 680000 IOPS (engl. Input Output Per Second) ausgestattet. Je höher die Geschwindigkeit desto schneller wird die Applikation beschleunigt und pro Instanz sind mehr Nutzer möglich.
Der virtuelle CPU sowie die Arbeitslast ist konfigurierbar.
Darüber hinaus ist eine Archivwiederherstellung gegeben für eine schnelle Verfügbarkeit der Daten und hohen Durchsatz für direkte Datenwiederherstellung.[218]
Durch den globalen Lastenausgleich sorgt für eine bessere Leistung der Anwendungen. Das globale Network ist seit 2007 CO_2-neutral und damit äußerst umweltfreundlich als auch energiesparend. Für 37% der betrieblichen Prozesse werden erneuerbare Energien genutzt.
Google's effiziente Netzwerkinfrastruktur wird durch sein globales Glasfasernetzwerk gegeben. Die zahlreichen weltweit verteilten POPs „[...] *sorgen für eine hohe Redundanz. Die Daten werden auf Speichergeräten an mehreren Standorten gespiegelt.*“[Picn] Zusätzlich durch die Edge-Caching-Dienste und das softwarebasierte Network ist eine „[...] *schnelle globale Vernetzung* [...]“[Picn] gegeben und somit eine geringe Latenz und eine hohe Verfügbarkeit gegeben. Aufgrund der „[...] *transparenten Wartung durch integrierte Live-Migrationstechnologie müssen für Wartungsmaßnahmen die VMs nicht heruntergefahren werden. Hosts werden ohne Ausfallzeiten gepatcht und die Rechenzentren gewartet.*“[Picn]

[217] [vgl. Min17]
[218] [vgl. Picn]

[219]

Abbildung 4.15: *Cloud-Rechenzentren der GCP unterteilt in Regionen*

Hinsichtlich der Anzahl von Rechenzentren weltweit befindet sich Google noch in der Ausbauphase. Derzeit ist Google lediglich „[...] in zwölf Regionen [...]“, siehe Abbildung 4.15,„[...] mit einem Cloud-Data Center vertreten, davon drei bzw. zukünftig fünf Standorte in Europa, und bietet seine Dienste in 30 Zonen an [...]“[Red17, S.2], während beispielsweise Amazon Web Services (AWS) in 16 Regionen und momentan in 44 Zonen verfügbar ist.[220]
Auch der Auswahl an Diensten betreffend liegt die Angebotsbreite bei der GCP bei aktuell 64 Services, wovon sich über fünf in einer Betatestphase befinden und damit kleiner als bei dem größten Cloud-Service Anbieter AWS mit über 120 Services, die vollständig integriert sind.[221]

Das **Sicherheitsmodell** von Google baut auf einen über 15 Jahre langen durch Erfahrung geprägten Prozess auf, was die hohe Sicherheit begründet.[222]
Dementsprechend können die Daten und Anwendung von Speicherstil profitieren. Neben das für die Informations-,Anwendungs- und Netzwerksicherheit verantwortliche Team, bestehend aus 750 Experten, ist auch eine physische Sicherheit wie Alarmanlagen„[...] *hochauflösenden Innen- und Außenkameras, Laserstrahlen* [...]“[Picr] vorhanden, welches die Rechenzentren jederzeit schützt. Die Server sind nach Vorgaben von Google identisch aufgebaut. Durch „Homogenität in Kombination mit der Kontrolle über den gesamten Software-

[219] [Pici]
[220] [Red17, S.2]
[221] [vgl. Red17, S.2]
[222] [vgl. Min17]

stack besteht eine geringere Angriffsfläche und die Möglichkeit rechtzeitig bei Bedrohungen zu reagieren.“[Picr]
Dennoch besteht eine gravierende Gefahr bezüglich der Sicherung der Daten im Ausland.
„*Die Datenspeicherung auf einem Server in den USA, etwa bei Google, unterliegt nämlich nicht den deutschen oder europäischen Datenschutzrichtlinien.*“[Min17]

Die leistungsstarken **Daten-und Analysedienste**, besonders Big Query und Datalab aus der Kategorie Big Data, sprechen ebenfalls für eine Einführung der GCP.
Die Abfragen von Daten in Petabyte-Größen erfolgen ohne lange Wartezeiten. Die Back-End-Infrastruktur wird hinsichtlicher der Verwaltung vollständig übernommen. Die Analyseservices werden automatisch skaliert, sodass die Anwender sich auf die eigenen geschäftlichen Prozesse fokussieren kann.[223]
Die Datenmengen können effektiv für die Programmcodes verwendet werden, um kontextbezogene und optimale Ergebnisse zu erhalten.[224]
Die Daten ist für das gesamte Unternehmen zugänglich. Durch Dienste wie BigQuery und Cloud Datalab werden Daten an die gewünschten Personen bzw. die für das Projekt relevanten Verbraucher übermittelt.[225]
„*Um die Einsatzmöglichkeiten des maschinellen Lernens auszuweiten, hat Google kürzlich TensorFlow, seine Bibliothek für Maschinenintelligenz, als Open Source zur Verfügung gestellt und Cloud Machine Learning-Produkte veröffentlicht.*“[Picb] Die Verarbeitung der Daten kann bereits unter Anwendung der auf Maschinenintelligenz basierten Dienste ausgeführt werden.[226]

Die GCP wird **serverlos** verwaltet. Die Server arbeiten im Hintergrund als physikalischer Host bzw. virtuelle Ressource, wo Quellcodes und Daten verarbeitet und gelagert werden. Anders als bei PaaS übernimmt der Cloud-Service die Arbeit mithilfe der Blackbox. Dabei wird der Programmcode hochgeladen und die Cloud-Dienste wie beispielsweise die AppEngine übernehmen administrative Aufgaben und stellen ausreichende Ressourcen für die Anwendungen bereit. „*Der Cloud-Service kümmert sich hierbei u.a. um die automatische Skalierung der Server-Infrastruktur, des Speichers, Netzwerks und anderer Ressourcen und übernimmt somit eigenständig das Kapazitätsmanagement.*“[Bü16]
Durch ein serverloses Modell müssen Faktoren wie die notwendige Zeit zum Hochfahren der Server, das Optimieren der Netzwerkeinstellungen und der allgemeine Aufwand für die Technologien nicht berücksichtigt werden. Zeit und Kosten werden somit gespart.[227]

[223] [vgl. Picb]
[224] [vgl. Picb]
[225] [vgl. Picb]
[226] [vgl. Picb]
[227] [vgl. Pico]

Das **Preiskonzept** gilt als kundenfreundlich. Generell werden keine Vorauszahlungen benötigt. Bei konstanter Nutzung der Dienste wie beispielsweise Compute Engine, Cloud SQL etc. wird automatisch ein 30%-iger Rabatt verrechnet. Das bereits mehrfach erwähnte Pay-As-You-Go-Prinzip, die minutengenaue Abrechnung sowie der Coldline Speicher sprechen ebenfalls für die GCP.[228] Coldline zeichnt sich durch seine langlebige Datensicherung aus. Dabei handelt es sich um Daten, auf die nur in seltenen Fällen zugegriffen werden.[229] Außerdem ist die GCP für viele Computing- Arbeitslasten, wie Amazon Web Services oder Microsoft Azure, zu 60% preisgünstiger als andere Plattformen.[230]
Während AWS sich an dem Pay-Per-Hour- Modell bedient, also bereits nach zehnminütiger Nutzung auf eine Stunde aufrundet und Gebühren abrechnet, nutzt die GCP das Pay-Per-Minute- Modell. Auf einer langen Frist betrachtet rentiert sich das Preiskonzept von der GCP im Vergleich zu dem von AWS deutlich, auch durch die zusätzlichen Rabatte. AWS verpflichtet Benutzer an längerfristige Verträge ohne jegliche Kostenentlastungen.[231]

[228] [vgl. shu17]
[229] [vgl. Pica]
[230] [vgl. shu17]
[231] [vgl. shu17]

5 Anforderungsanalyse

In diesem Abschnitt werden die funktionalen sowie technischen Anforderungen bezüglich des Testverfahrens, die aus dem Workflow des vorherigen Kapitels 4.1 resultieren, dargelegt und anschließend bewertet.

Grundlegend geht es bei der Einführung einer Testautomatisierung bzw. Teil-Testautomatisierung speziell bei hamburg.de darum, dass einige oder alle Aktivitäten eines Testprozesses vor jedem Deployment über Jenkins hauptsächlich automatisiert verlaufen. Deployment ‚zu Dt. Softwareverteilung, „ist die gezielte Übertragung von von Software oder -Updates an alle mit dieser Software arbeitenden Rechner oder an alle an einem betrieblichen Prozess beteiligten Rechner.“(deployment, 10.08.2005, ITWissen.info, http://www.itwissen.info/deployment-Softwareverteilung.html, Zugriff 09.11.2017 um 11:40) Für die Verifikation von Fehlern wird ein manueller Test nachträglich durchgeführt, um die Existenz des jeweiligen Fehlers sicherstellen zu können.

5.1 Anforderungen

5.1.1 Funktionale Anforderungen

Funktionale Anforderungen widmen sich grundlegend der Frage, was ein System bzw. eine Software-Komponente leisten soll.[232] Sie definieren die Vorgehensweise zur Verarbeitung von Daten für eine Softwarekomponente sowie das Ein- und Ausgabeverhalten.[233]
Diese werden nachfolgend aufgelistet und sind zugeschnitten auf die Thematik der Thesis:

- Bei Änderungen oder Erneuerungen des Stadtportals hamburg.de sowie den zukünftigen responsiven Seiten soll ein automatisierter Testvorgang starten, der sowohl explizit die neu eingebauten Features testet als auch prüft, ob die ursprünglichen Funktionen intakt sind

[232] [vgl. Sch11a, Folie 9]
[233] [vgl. Fra07, S.25]

- Die Testfälle sind für Cross-Browser-Testing ausgelegt. Die Ausführung kann parallel auf unterschiedlichen Webbrowser mit mehreren Instanzen realisiert werden
- Die Testprozesse werden überwacht. Wenn ein Testprozess zwischendurch abbricht, wird der Anwender darüber informiert
- Alle Entwickler und Tester werden über Erfolg oder Misserfolg nach der Testdurchführung benachrichtigt und entscheiden anschließend über den weiteren Prozess
- Testfälle werden im Einzelnen betrachtet. Dabei wird detailliert angezeigt, welcher Testfall erfolgreich bzw. fehlerhaft war. Dieses ist für die Auswertung der Testfälle essentiell dadurch, dass dokumentiert, welche Methode des jeweiligen Testfalls fehlerhaft ist
- Die Testergebnisse werden in grafischer Form dargestellt. Die Testdaten können als CSV-Format heruntergeladen und anschließend ausgewertet werden
- Bei auftretenden Bugs wird der jeweilige Testfall erneut lokal automatisiert, damit die mögliche Fehlerursache ermittelt werden kann.[234]
- Nach Abschluss der Testauswertung werden die funktionierenden Tests in einer Datenbank gelagert

  Der Begriff funktionierende Tests bedeutet in dem Fall, dass sichergestellt wurde, dass der jeweilige Testfall immer erfolgreich ausgeführt wurde und Resultate erbringt, die den Tester und Entwickler für ihre Analysen weiterhelfen oder gegebenfalls rechtzeitig auf Fehler aufmerksam machen

5.1.2 Technische Anforderungen

Technische Anforderungen gehören kategorisch zu den *nicht-funktionalen Anforderungen*. Bei nicht-funktionalen Anforderungen steht die Frage, **wie** ein System funktionieren soll, im Vordergrund.[235]
Sie „[...] *beschreiben das allgemeine Verhalten einer Software-Komponente.*“[Fra07, S.25]
Im Folgenden wird sich auf die technische Anforderungen beschränkt:

- Implementierung sowie Ausführung der Tests funktionieren plattformunabhängig

[234] [vgl. Lig09, S.211]
[235] [vgl. Sch11a, Folie 9]

- Alle Testfälle werden in der Programmiersprache Java geschrieben. Aufgrund der Tatsache, dass das Selenium-Framework „in Java geschrieben ist, kann es problemlos in eigene Java basierte Projekte eingebunden werden“[Gro10]
- Als Entwicklungsumgebung wird Eclipse IDE mit integrierten Plug-ins, zu betrachten im Abschnitt 4.4, verwendet
- Selenium WebDriver wird als Bibliothek hinzugefügt.
  Heruntergeladen wird diese Bibliothek sowie weitere benötigte Packages und Treiber für die Webbrowser auf der Website: http://www.seleniumhq.org/download/
- Umsetzung des Hub-Node-Networks erfolgt durch den Jenkins-Server, als Hub fungierend, und einer Windows 10-Maschine als ein konfigurierter Node
- Jenkins-Server ist mit notwendigen Plug-ins eingerichtet
- Verwendete Webbrowser: Google Chrome, Mozilla Firefox und Microsoft Edge
- Alle Testdaten werden in der Google Cloud Platform gespeichert und durch BigQuery mit Stackdriver Monitoring analysiert, überwacht und visualisiert

5.2 Analyse

In dem vorliegenden Abschnitt werden ausschließlich die funktionalen Anforderungen analysiert. Die technischen Anforderungen und deren Realisierbarkeiten wurden zum größten Teil bereits im Kapitel 4 beschrieben. Die Zusammenhänge der einzelnen Software-Komponenten werden zusätzlich im Kapitel 6 validiert. Die technischen Anforderungen sind also umsetzbar.

- **Funktionale Anforderung 1**: Die erste Anforderung wird mithilfe von dem Jenkins-Server als CI-Werkzeug vor jedem Deployment ermöglicht. Der Jenkins-Server führt die Selenium-Tests automatisiert durch.

- **Funktionale Anforderung 2**: Durch die Einbindung des TestNG- Frameworks mit Selenium WebDriver ist eine parallele Ausführung der Testfälle auf verschiedenen Webbrowser realisierbar. Hiefür sollten die Fallbeispiele in den Kapiteln 6.2 und 6.3 in Betracht gezogen werden.
  Die Webbrowser werden ebenfalls durch die Initialisierung des Nodes über die Kommandozeile parametrisiert. Dabei können unter Anderem der Webbrowser (***-browser*** <***params***>) oder die Anzahl der parallel ausgeführten Sitzungen (***-maxSession 5***, standardgemäß gesetzt auf 5, festgelegt werden. Sitzungen unterscheiden sich von Instanzen (***-maxInstances***) insofern, dass ersterer Parameter die maximale Anzahl der parallel ausgeführten Webbrowser, unabhängig von Typ oder Version, auf einem Node festlegt, während Instanzen die Anzahl eines Browsertyps verwalten.[236]

  Beispiel: Ein Node ist mit Mozilla Firefox 50 und Google Chrome konfiguriert. Wenn der Parameter *maxInstances=5* gesetzt wird, werden insgesamt 10 Instanzen der unterschiedlichen Webbrowser, also jeweils fünf von Firefox 50- und fünf Chrome-Instanzen, parallel aufgerufen. Der Parameter *maxSession* überschreibt die Funtion des Parameters *maxInstances* und limitiert die Anzahl der parallel ausführbaren Browser-Instanzen. Wird *maxSession=1* definiert, dann wird ausschließlich die Ausführung eines einzigen Webbrowsers erzwungen.[237]

- **Funktionale Anforderung 3**: Die Überwachung eines Testprozesses erfolgt über die Jenkins-Weboberfläche. Teilnehmer können den laufenden Testprozess mitverfolgen. Zusätzlich wird die ungefähre verbleibende Zeit eines Prozesses bis zum Abschluss

[236] [vgl. Ste15]
[237] [vgl. Mal12]

angezeigt.

- **Funktionales Anforderung 4**: Es bestünde die Möglichkeit ein Benachrichtigungssystem über E-Mail einzustellen durch Einbindung eines Plug-ins, bezeichnet als Email-ext plugin. Dieser schaltet sich ein, sobald ein Build-Prozess abgeschlossen ist und versendet an alle Projektteilnehmer, die konfigurierbar sind, eine Mitteilung.[238]
  Möglich ist ebenfalls beispielsweise ein auf Groovy basiertes Template zu erstellen, die über die Jenkins API mit dem E-Mail-Plug-in interagiert.[239]
  Da dieses noch nicht im Betrieb eingeführt wurde, beschränke ich mich auf weiterhin auf die Einsicht über die Weboberfläche von Jenkins.

- **Funktionale Anforderung 5**: Diese Anforderung wird sowohl durch das in Jenkins integrierten Plug-in testng-plugin als auch durch das Test Results Analyzer Plugin ermöglicht, ebenfalls aufgeführt im Kapitel 7.

- **Funktionale Anforderung 6**: Die Umsetzung dieser funktionalen Anforderung erfolgt durch das Test Results Analyzer Plugin. Das Plug-in wurde im Kapitel 7.3 erläutert.

- **Funktionale Anforderung 7**: Dieser Aspekt wird anhand eines praktischen Beispiels anhand des Kapitels 6.2 beleuchtet und beschreibt die Implementierung.

- **Funktionale Anforderung 8**: Das Ziel dieser Anforderung ist prinzipiell durch Dienste der Google Cloud Plattform, in dem Zusammenhang BigQuery und Stackdriver Monitoring, zu erreichen. Aus zeitlichen Gründen konnte diese nur bedingt in die Praxis umgesetzt werden. Ein erläuternder Beitrag bezüglich dieses Themas folgt im Kapitel 8.

[238] [vgl. vL16]
[239] [vgl. Fre]

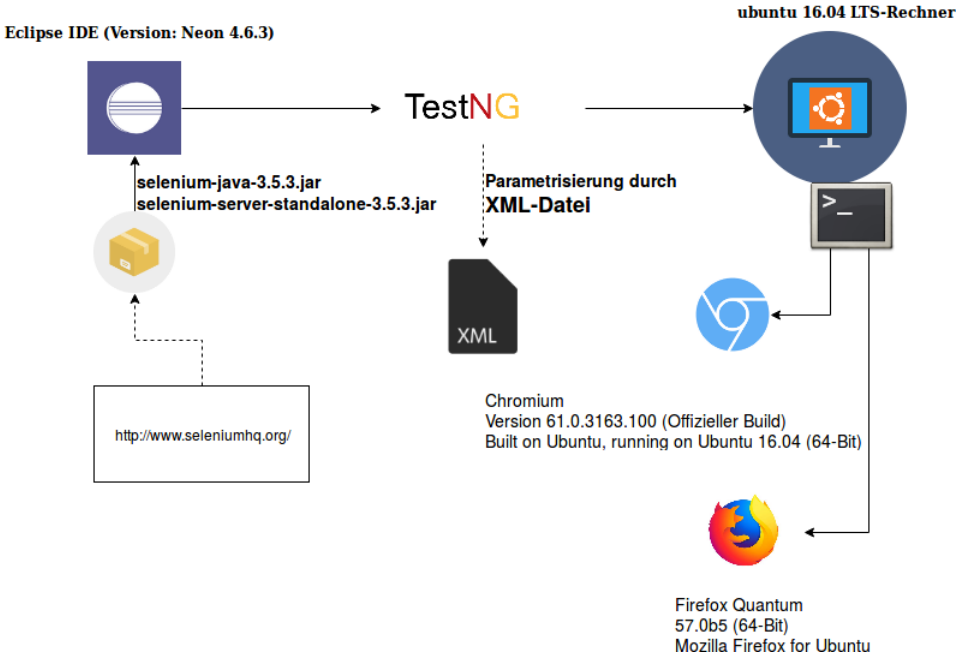
6 Implementierung

In diesem Kapitel geht es um die Umsetzung eines Lösungsansatzes anhand eines Fallbeispiels.
Anhand eines Vorgehensmusters wird der Lösungsansatz verdeutlicht.
Dabei werden zwei Prozesse unterschieden, einerseits die lokal automatisierten Test und andererseits die durch den CI-Server Jenkins ausgeführten Tests. Anschließend werden zu beiden Prozessen jeweils ein Fallbeispiel erläutert.
Auf besondere Aspekte hinsichtlich Herausforderungen und Problematiken wird ebenfalls eingegangen.

6.1 Vorgehensmodell

6.1.1 Lokal automatisierte Tests

Java Project: LocalSeleniumTestHamburg
Java Version: 8
Libraries: JRE System Library JavaSE-1.8, TestNG
Referenced Libraries: selenium-java-3.5.3.jar,selenium-server-standalone-3.5.3.jar
Packages: startseite, suchen, login, upload, mobile, de.hamburg



[240]

Abbildung 6.1: *Ablauf von lokal automatisiert ausgeführten Testfällen in abstrakter Darstellung*

Zuerst werden die für das Testtool Selenium WebDriver benötigten Bibliotheken aus der Website www.seleniumhq.org in den Dateiformaten ZIP und JAR heruntergeladen.
Diese zwei Dateien wurden in der Entwicklungsumgebung Eclipse IDE eingebunden. Durch die Option „Configure Java Build Path“lassen sich externe JAR- und ZIP-Dateien in das gewünschte Java-Projekt einbinden.
Um die Dateien ausführen zu können, wird die JRE auf (engl. Java Runtime Environment) vorausgesetzt, die ebenfalls in der Form einer Bibliothek eingebunden werden kann (JRE System Library JavaSE-1.8).

[240] Erstellt durch draw.io

Der Nutzer ist dadurch in der Lage die Objekte (Klassen) der jeweilig eingebundenen Bibliothek zu erzeugen und aufzurufen.[241] In der externen Bibliothek selenium-server-standalone-3.5.3.jar sind Java-Klassen enthalten, die für das Testen mit Selenium unentbehrlich sind. Die wichtigsten sind in dem Paket org.openqa.selenium.* enthalten, die in den Testfällen importiert werden und dementsprechend in diesen Tests auftauchen, weiter beschrieben in Kapitel 9. In diesem Paket sind wiederum weitere Pakete und auf Selenium WebDriver basierte Implementationen enthalten, die beispielsweise einen bestimmten Webbrowser wie Chromium lokal steuern: *Class ChromeDriver, package org.openqa.selenium.chrome*[242] oder Seitenelemente auffinden durch unterschiedliche Lokatoren[243], die sowohl im nachfolgenden Fallbeispiel, Kapitel 6.2, erklärt als auch auf der beigefügten CD-ROM verwendet werden.
Durch die Einbindung des TestNG-Frameworks, bereits erwähnt in Kapitel 4.3, als weitere Bibliothek in Eclipse werden die Testfälle parametrisiert mit verschiedenen Webbrowserwerten durch die Nutzung einer XML-Datei, sodass eine parallele Ausführung von Testskripten auf unterschiedlichen Webbrowsern, wie Firefox Quantum und Chromium, gewährleistet werden kann. Für weitere Informationen sollte das folgende Fallbeispiel in Kapitel 6.2 in Betracht gezogen werden.

[241] [vgl. Die16, S.25]
[242] [Stea]
[243] [Stea]

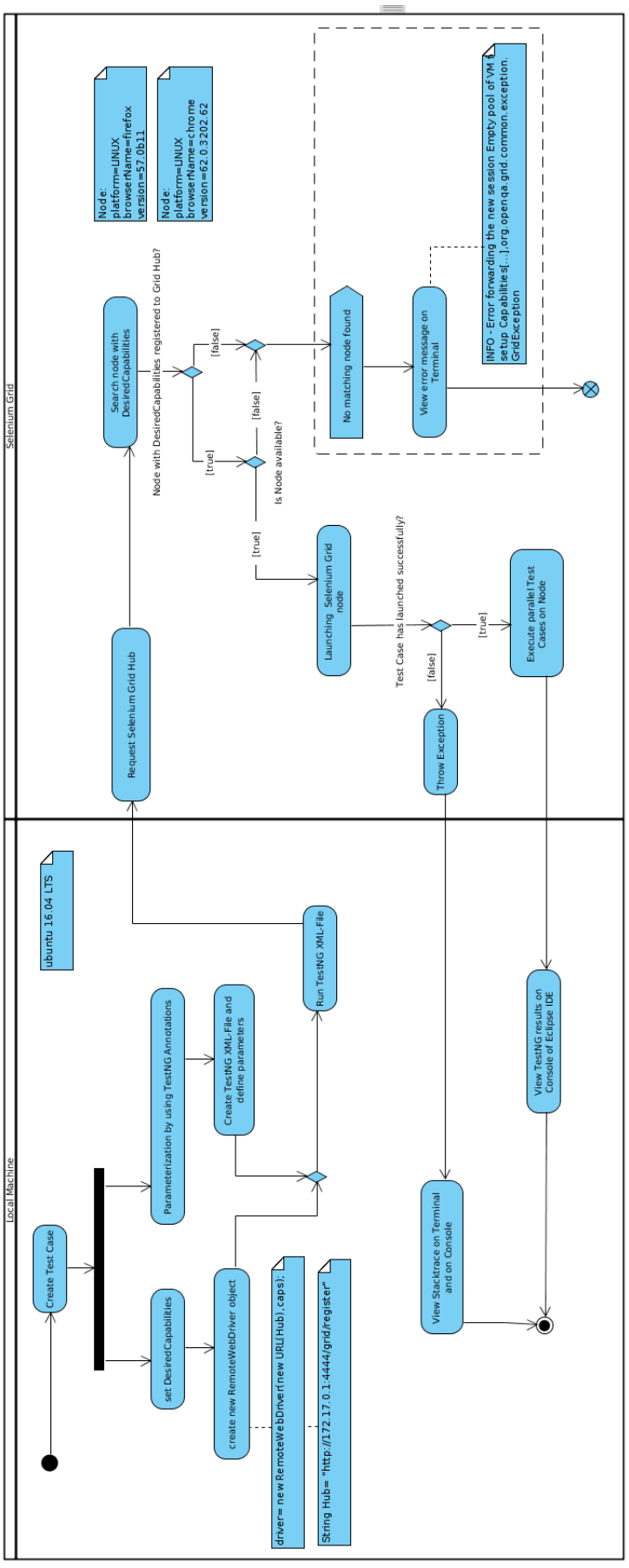


Abbildung 6.2: *Aktivitätsdiagramm- Lokal automatisierte Tests mithilfe von „TestNG“*[244]

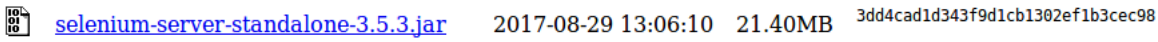
Der Ablauf eines Testfalls über Selenium Grid wird durch ein Aktivitätsdiagramm 6.2 verdeutlicht. Hierbei beschränke ich mich auf Grundelemente, da bereits zu Beginn des Kapitels 6.1.1 viele Komponenten berücksichtigt wurden:

- Im Testfall werden „[...] *Anforderungen im erstellten DesiredCapabilities-Objekt* [...]“festgelegt[Gie12, S.45]
- Zusätzlich werden TestNG- Annotationen notiert, um Werte von der auf XML basierten Testkonfigurationsdatei erhalten zu können
- Durch die Erstellung eines *RemoteWebDriver*-Objekts können IP-Adresse des eingerichteten Selenium Grid Hubs sowie die gesetzten Anforderungen aus Punkt 1 übergeben werden
- In der XML-Datei werden die Parameter (*platform,browser,url,version*) konfiguriert. Innerhalb der Tests werden diese an den Testklassen und -methoden übergeben, schon beschrieben im Kapitel 4.3
- Die TestNG XML-Datei wird ausgeführt
- Der Selenium Grid Hub erhält eine Anfrage
- Der Hub analysiert die vorhandenen Nodes nach den gewünschten Anforderungen und der Verfügbarkeit.
- Sobald ein Node mit den entsprechenden Fähigkeiten registriert wurde und verfügbar ist, können die Testfälle ausgeführt werden. Anschließend wird die Durchführung der Testfälle auf Erfolg oder Misserfolg untersucht und eine entsprechende Ausgabe erzeugt:
  - Bei erfolgreichem Start eines Testfalls werden die Testresultate nach abschließender Durchführung in den TestNG-Ergebnisse der Entwicklungsumgebung aufgelistet
  - Bei Misserfolg werden Exceptions geworfen, beispielsweise *RemoteDriverServerException* oder*WebDriverException*[245]. Die Stacktraces werden über der Konsole von Eclipse, die TestNG-Ergebnisse oder das Terminal angezeigt
- Wenn kein Node registriert werden konnte, wird eine Exception mit einer Information oder Fehlermeldung auf dem Terminal sichtbar, wie beispielsweise im Aktivitätsdiagramm 6.2 in Form von einer Notiz abgebildet:
  INFO - Error forwarding the new session Empty pool of VM for setup Capabilities[...],org.openqa.grid.common.exception.GridException

[244] Erstellt mit Visual Paradigm, Orientierung an Bachelorthesis [Gie12, S.46]
[245] [Stea]

Die Programmierung der Testskripte als auch die Ausführung der Testsammlungen erfolgen hierbei auf einem ubuntu 16.04 LTS-Rechner.
Mithilfe von Selenium Grid wird hierbei ein lokales Hub-Node-Konzept aufgesetzt und konfiguriert.

Einstellungen eines lokalen Hubs:
Für das Aufsetzen des Hubs wird die Datei, selenium-server-standalone-3.5.3.jar, in diesem Beispiel Version 3.5.3, die unter der Website http://www.seleniumhq.org/download/ heruntergeladen werden kann, vorausgesetzt.



246

Abbildung 6.3: *Download von „Selenium Standalone Server“in Dateiformat JAR*

Die heruntergeladene JAR-Datei wird in einem seperaten Ordner gespeichert. Der Pfad dieses Ordners muss bei der Einstellung des Hubs über das Terminal berücksichtigt werden.

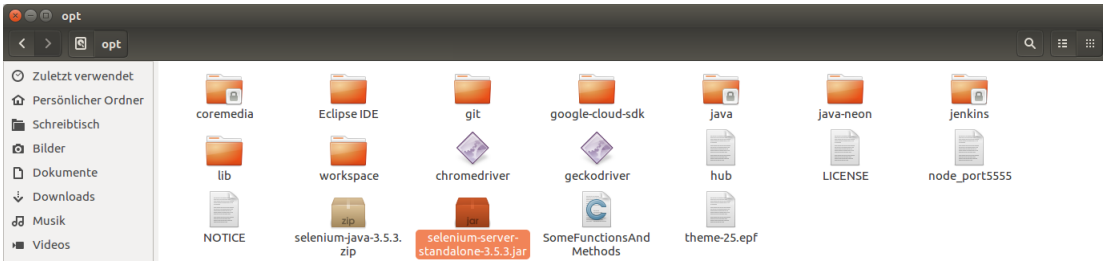


Abbildung 6.4: *Heruntergeladene Datei gelagert in einem Ordner „opt“*

Durch den folgenden dargestellten Befehl wird der Hub gestartet:

```
jawar@jawar-Entry-Tower-Workstation:/opt$ java -jar selenium-server-standalone-3.5.3.jar -role hub
```

247

Abbildung 6.5: *Starten eines Selenium Grid Hubs durch ausgeführten Befehl via Terminal*

246 [vgl. Steb]

Folgende optionale Parameter sind erlaubt, die in Abbildung 6.7 verwendet werden:
„*platform=*{ *WINDOWS,LINUX,MAC* }
browserName={ *android, chrome, firefox, htmlunit, internet explorer, iphone, opera* }
version={ *browser version* }
Dwebdriver.{*Driver*}*.driver=*{ *„Pfad zur binären Browser-Driver-Datei“*[248]}
maxInstances={ *maximum number of browsers of this type* }[249]“

Wenn der Befehl aus Abbildung 6.5 erfolgreich ausgeführt wurde, erscheint die folgenden Informationen:

```
jawar@jawar-Entry-Tower-Workstation:/opt$ java -jar selenium-server-standalone-3.5.3.jar -role hub
14:47:02.977 INFO - Selenium build info: version: '3.5.3', revision: 'a88d25fe6b'
14:47:02.978 INFO - Launching Selenium Grid hub
2017-10-11 14:47:03.855:INFO::main: Logging initialized @2012ms to org.seleniumhq.jetty9.util.log.StdErrLog
14:47:03.914 INFO - Will listen on 4444
2017-10-11 14:47:03.958:INFO:osjs.Server:main: jetty-9.4.5.v20170502
2017-10-11 14:47:04.025:INFO:osjs.session:main: DefaultSessionIdManager workerName=node0
2017-10-11 14:47:04.025:INFO:osjs.session:main: No SessionScavenger set, using defaults
2017-10-11 14:47:04.036:INFO:osjs.session:main: Scavenging every 660000ms
2017-10-11 14:47:04.043:INFO:osjsh.ContextHandler:main: Started o.s.j.s.ServletContextHandler@16022d9d{/,null,AVAILABLE}
2017-10-11 14:47:04.097:INFO:osjs.AbstractConnector:main: Started ServerConnector@5e57643e{HTTP/1.1,[http/1.1]}{0.0.0.0:4444}
2017-10-11 14:47:04.098:INFO:osjs.Server:main: Started @2255ms
14:47:04.098 INFO - Nodes should register to http://172.17.0.1:4444/grid/register/
14:47:04.098 INFO - Selenium Grid hub is up and running
```

[250]

Abbildung 6.6: *Aktivierung des Selenium Grid Hubs*

Besonders zwei Informationen sind von größerer Bedeutung:

- ***INFO - Nodes should register to http://172.17.0.1:4444/grid/register***
  Die obige Information, die automatisch nach einem registrierten Hub ausgegeben wird, beinhaltet die notwendige IP-Adresse des Rechners, auf dem der Hub aufgesetzt wurde und der „Port des Dienstes auf der Selenium-Hub-Maschine (Default ist Port 4444)“.[Pre15] Der Port ist konfigurierbar durch eine zusätzliche Angabe via Terminal („-port xxxx“)

- ***INFO - Selenium Grid hub is up and running***
  Der Selenium Hub ist erfolgreich gestartet und wartet nun auf Testanweisungen, die an folgende konfigurierte Nodes verteilt werden können.

[247] [vgl. Sha17b]
[248] Benutzung der Parameter [Pre15]
[249] https://github.com/SeleniumHQ/selenium/wiki/Grid2
[250] [vgl. Sha17b]

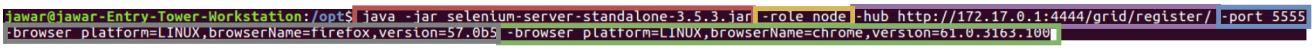
Zur Überprüfung kann über der Konsole http://172.17.0.1:4444/grid/console, allgemein *http:///IP/Name des Rechners:Port/grid/console*, die Existenz des Hubs verifiziert werden. Die Konfiguration der Nodes erfolgt über ein weiteres Terminalfenster durch eine Kommandozeile.

Konfiguration von zwei Nodes über das Terminal:

java -jar selenium-server-standalone-3.5.3.jar -role node -hub http://172.17.0.1:4444/grid/register/ -port 5555 -browser platform=LINUX,browserName=firefox,version=57.0b11 -browser platform=LINUX,browserName=chrome,version=62.0.3202.62

Der Aufruf des Befehls inklusive Parameter wird in Abbildung 6.7 visualisiert.

Die Versionen der Webbrowser können in den Testfällen mit den angezeigten Versionen auf Abbildung 6.7 variieren aufgrund von bereits vorhandenen Aktualisierungen. Die dort angegebenen Versionen sollen ausschließlich zum besseren Verständnis dienen und haben keinerlei Auswirkung auf den weiteren Testprozess.



251

Abbildung 6.7: *Befehl für Registrierung von zwei Nodes zum bereits konfigurierten Hub-Server*

Der Befehl für die Registrierung von zwei Nodes über das Terminal, illustriert durch Abbildung 6.7, umfasst unterschiedliche Parameter:

- Angabe des Pfads der bereits heruntergeladenen Selenium Standalone Server-Datei, wie in Abbildung 6.3 zu betrachten
- Spezifizierung bzw. Identifizierung der Rolle (in dem Fall Node)
- „IP-Adresse oder Hostname mit Angabe des Ports von dem eingestellten Selenium Grid-Hub“
- Vergabe eines Ports „[...] *unter dem der Node* [...]“[Pre15] registriert wird
- Der erste Browser wird initiiert und kann anschließend mit zusäzlichen Parametern deklariert werden. Wie aus der Abbildung 6.7 zu erkennen, wird eine Mozilla Firefox-Instanz eingestellt.

[251] [vgl. Sha17b]

- Der zweite Browser wird initiiert und kann anschließend mit zusäzlichen Parametern deklariert werden. Wie aus der Abbildung 6.7 zu erkennen, wird eine Google Chrome-Instanz eingestellt.

Bei den aufgelisteten optionalen Parameter handelt es sich nur um die Parameter, die für die vorliegende Arbeit im Vordergrund stehen. Es gibt weitaus mehr einzusetzende Parameter. Der vierte Parameter aus obiger Auflistung, der für den „Typen des Browser-Treibers“[vgl. Pre15] zuständig ist, wie in Kapitel 4.1.3 verdeutlicht, wurde bereits in den Testfällen implementiert, sodass dieser Parameter nicht im Befehl der Registrierung für die Nodes integriert wurde.

252

Abbildung 6.8: *Abschluss des Registrierungsvorgangs der eingestellten Nodes*

Nach Betätigung des Befehls für die Konfiguration der Nodes, wie in Abbildung 6.7 veranschaulicht, über der Kommandozeile werden einem User einige Informationen aufgelistet, die aus der Abbildung 6.8 entnommen werden können.
Diese informieren den Anwender darüber, dass Nodes eingeschaltet sind und dem Hub zugeordnet wurden. Testanweisungen können ab fortan automatisiert über den erstellten Selenium Grid-Hub weitergeleitet werden an die jeweiligen implementierten Nodes.

[252] Konzept nach: [Sha17b]

[253]

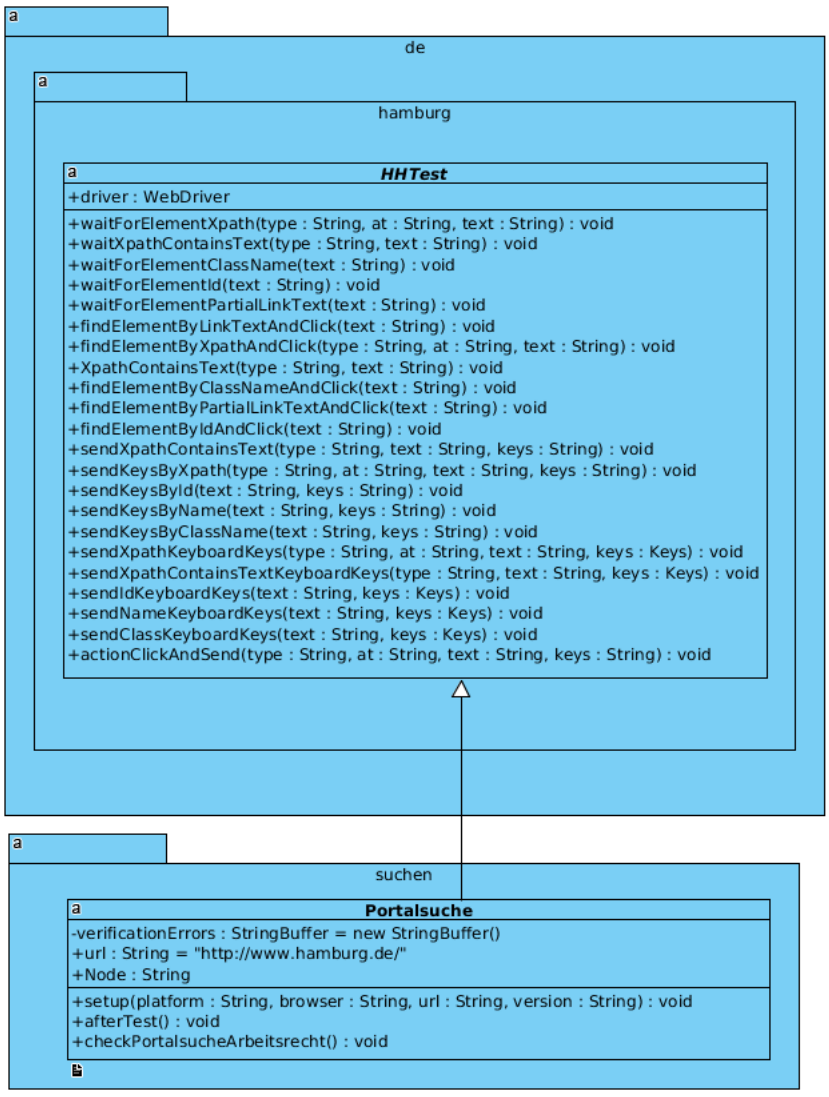
Abbildung 6.9: *Anzeige der verfügbaren Nodes auf einem Hub über die Selenium Grid-Konsole*

Die verfügbaren Nodes samt eingestellter Parameter sind nach erfolgreicher Registrierung auf der Grid-Konsole sichtbar.
Es besteht auch die Möglichkeit Nodes durch eine JSON zu konfigurieren. Dieser Fall wird jedoch hierfür nicht berücksichtigt.

6.2 Fallbeispiel

In dem vorliegenden Testfall wird auf dem Stadtportal www.hamburg.de in der Portalsuche nach dem Begriff **Arbeitsrecht** gesucht. Nachdem Suchergebnisse erfolgreich angezeigt werden, werden diese nach Einträgen aus dem Branchenbuch untersucht.
Inhalte des Branchenbuches werden als Extrakt auch in die übergreifende Protalsuche gefeedet. Die Portalsuche auf der Basis von Core Media ist über eine lose Kopplung mit IASON verbunden. Bedingt durch die lose Kopplung der Systeme ist die Überprüfung der Verbindung zwischen beiden Systemen relevant. Das Branchenbuch gehört zu einen der wichtigsten Suchmasken von www.hamburg.de. Zahlreiche User können auf diese Datenbank zurückgreifen und werden mit geforderten Informationen versorgt. Bei derartigen Informationen handelt es sich meist um Firmenkontaktdaten.[254]
Dieser Testfall erfolgt nach einem *IASON*-Update.
IASON ist der Produktname der Basissoftware für das bei hamburg.de realisierten Branchenbuchs.

[253] http://localhost:4444/grid/console
[254] [vgl. Fuc]



[255]

Abbildung 6.10: *Vereinfachtes Klassendiagramm der Klasse DesktopPortalsuche*

Das vereinfachte Klassendiagramm, bestehend aus einer Ober-und Unterklasse, umfasst alle Parameter sowie Methoden, die für die Umsetzung des Testfalls benötigt und verwendet wurden.
Dabei handelt es sich bei HHTest.java um die Oberklasse und bei Portalsuche.java um die Unterklasse. Die Oberklasse beinhaltet alle refaktorisierten Methoden, die für die Untersuchung eines Elements essentiell sind. „Mit Refactoring bezeichnet man die Überarbeitung der Struktur einer Software, ohne dass sich deren Verhalten nach außen ändert.“[Lip13b]

[255] Erstellt mit Visual Paradigm

Die Unterklasse *Portalsuche.java* erbt von der Oberklasse *HHTest.java*:

Die Objektvariable ***driver*** vom Typ WebDriver ist ein wichtiger Bestandteil für die Realisierung des Testfalls. *WebDriver* ist ein Interface, welches verwendet wird für das Testen. Die beeinhalteten Methoden dieses Interfaces lassen sich kategorisieren in Kontrolle eines Webbrowsers, Auswählen eines *WebElements* (Interface stellt ein HTML-Element dar) und Debugging. Mithilfe des Zugriffs auf ein WebElement werden weitere wichtige Methoden freigeschaltet. Darunter fallen die Methoden *get (String[256])*, welches verwendet wird, um eine neue Website zu laden oder *findElement(By[257])*, um HTML-Elemente auf einer Website zu finden.
Diese Objektvariable steht in den Methoden
public void setup(String platform, String browser, String url, String version)
public void checkPortalsucheArbeitsrecht() im Vordergrund.
Diese Methoden werden im Programmcode 128 (Listing 6.1) kommentiert und erklärt.

- **waitForElementXpath(type:String,at:String,text:String):void**
  Innerhalb der refaktorisierten Methode wird eine Anweisung aufgerufen ⇒ wait.until(ExpectedConditions.presenceOfElementLocated(By.xpath(...))) Dieser Aufruf wartet auf die Präsenz eines Elements auf dem DOM einer Website, lokalisiert anhand von *XPath*. Dies bedeutet jedoch nicht gezwungenermaßen, dass das Element sichtbar ist, wie bei der Methode *visibilityOfElementLocated*. Visibility bedeutet in dem Zusammenhang, dass das Element nicht nur sichtbar, sondern eine Höhe und Weite von größer als 0 besitzt.“[258] XPath ist eine Sprache, die die hierachische Struktur eines DOM (Document Objekt Model) einer Webseite traversiert. Besonders durch die Flexibilität von XPath lässt sich jedes Element identifizieren. Der Methode werden String-Parameter übergeben. Diese können beim Methodenaufruf angepasst werden und mit Daten des DOMs gefüllt werden, weshalb auch diese Methode refaktorisiert wurde. Das Refaktorisieren ist im Quellcode 128 unterlegt und kommentiert.

- **sendKeysById(text:String, keys:String):void**
  Ein Element wird zunächst anhand der ID identifiziert. Der ID-Parameter ist eine „case-sensitive“Zeichenkette, die ein einzigartiges Attribut eines Elements repräsentiert.[259] Diesem Element wird nach der erfolgreichreichen Lokalisierung ein String übergeben, was dazu führt, dass die Suchleiste des Stadtportals befüllt wird mit Daten, in dem Fall mit der Zeichenkette *Arbeitsrecht*. Folgende beinhaltete Objektreferenzen sind dafür zuständig:[driver.findElement(By.id(text)).sendKeys(keys);].

[256] Einsetzen einer Zeichenkette
[257] By locator, Einsetzen eines Lokators
[258] [vgl. Stea]
[259] [vgl. Lev17]

- **findElementByXpathAndClick(type:String, at:String, text:String):void**
  Ein Element wird durch Verwendung von *XPath* lokalisiert. Nach erfolgreichem Finden des HTML-Elements wird jenes Element fokussiert und angeklickt. Voraussetzungen für das Anklicken sind die Sichtbarkeit und die Größe[260] des HTML-Elements. Zu beachten ist ebenfalls, dass nach Ausführung der *click()*-Methode, die Methode nicht auf das nächste Laden der Website wartet. Hiefür eignet sich die Methode *Thread.sleep(long millis):void* oder *wait.until(ExpectedConditions.presenceOfElementLocated(By locator)*, die in allen refaktorisierten Methoden aufgerufen werden und nach einer gewünschten Zeit abwarten oder auf das Erscheinen eines bestimmten Webelements warten.

- **waitForElementId(text:String):void** Dieser Methodenaufruf ist nahezu identisch wie bei dem ersten Beispiel der Methode 6.2 mit dem Unterschied, dass diesmal der ID-Lokator verwendet wird und anhand dieses Kriteriums identifiziert und auf die Präsenz des Elements gewartet wird.

Die Methoden der Unterklasse *Portalsuche* werden in dem nachfolgenden Quellcode 128 ausreichend kommentiert.

[260] Höhe und Weite müssen größer als 0 betragen

```
package suchen;

import java.net.MalformedURLException;
import java.net.URL;

import org.openqa.selenium.By;
import org.openqa.selenium.chrome.ChromeOptions;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.testng.Assert;
import org.testng.annotations.AfterTest;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Parameters;
import org.testng.annotations.Test;

import de.hamburg.HHTest;

/**
 * Selenium Test für Portalsuche, Suche nach Arbeitsrecht
 *
 * @author jawar
 *
 */
public class Portalsuche extends HHTest {
        private StringBuffer verificationErrors = new StringBuffer();
        public String url = "http://www.hamburg.de/";

        /**
         * Die Methode setup() wird zur Initialisierung der Browser benötigt, damit
         * der entsprechende Browser gestartet werden kann. Folgende Parameter
         * werden dieser Methode übergeben:
         *
         * @param platform
         * @param browser
         * @param url
         * @param version
         *
         *            Durch System.setProperty() mit Angaben des Browser-Drivers
         *            (hier innerhalb der if-else-Schleife) wird der Aufruf eines
         *            Webbrowsers ermöglicht. Angegeben wird der jeweilige
         *            Browser-Driver und der Pfad des benötigten ausführbaren
         *            Treibers festgelegt für die Interaktion zwischen Webbrowser
         *            und Selenium Wire Protocol (Quelle:
         *            https://stackoverflow.com/questions/35285698/
         *            why-do-we-need-to-set-the-system-property-
         *            for-chrome-and-ie-browser-and-not-for)
         *
         *            Kapitel 4, Selenium WebDriver, Architektur
         * @throws MalformedURLException
         */
        @Parameters({ "platform", "browser", "url", "version" })
        @BeforeTest(alwaysRun = true)
        public void setup(String platform, String browser, String url, String version)
            throws MalformedURLException {
                DesiredCapabilities caps = new DesiredCapabilities();
                caps.setPlatform(org.openqa.selenium.Platform.LINUX);
                System.setProperty("java.net.preferIPv4stack", "true");
                /*
                 * Kommunikation nur über IPv4 möglich, deshalb Eigenschaft auf 'true'
                 * setzen (Quelle: https://docs.oracle.com/javase/7/docs/api/java/
                 * net/doc-files/net-properties.html)
                 */
                caps.setCapability("SeleniumTests", "redhat5 && amd64");
                /*
                 * Spezifizierung der Architektur (Quelle:
                 * https://wiki.jenkins.io/display/JENKINS/Selenium+Plugin)
                 */
                if (browser.equalsIgnoreCase("firefox")) {
                        System.out.println("Executing on Firefox");
                        String Hub = "http://172.17.0.1:4444/wd/hub";
                        caps = DesiredCapabilities.firefox();
                        caps.setBrowserName("firefox");
                        System.setProperty("webdriver.gecko.driver",
                            "/opt/geckodriver.exe");
```

```
                        driver = new RemoteWebDriver(new URL(Hub), caps);
                        driver.manage().window().maximize();
                        driver.navigate().to(url);

                } else if (browser.equalsIgnoreCase("chrome")) {
                        System.out.println("Executing on Chrome");
                        String Hub = "http://172.17.0.1:4444/wd/hub";
                        caps = DesiredCapabilities.chrome();
                        caps.setBrowserName("chrome");
                        ChromeOptions options = new ChromeOptions();
                        System.setProperty("webdriver.chrome.driver",
                            "/opt/chromedriver.exe");
                        caps.setCapability(ChromeOptions.CAPABILITY, options);

                        options.addArguments("--start-maximized");
                        driver = new RemoteWebDriver(new URL(Hub), caps);
                        driver.navigate().to(url);
                }
        }

        /**
         * Die Methode afterTest() dient dem erfolgreichen Schließen des jeweiligen
         * Browserfensters
         */
        @AfterTest
        public void afterTest() {
                driver.quit();
                String verificationErrorString = verificationErrors.toString();
                if (!"".equals(verificationErrorString)) {
                        Assert.fail(verificationErrorString);
                }

        }

        /**
         * Ablauf: Es wird zunächst abgewartet bis der Suchbutton auftaucht.
         * Anschließend wird in der Portalsuchleiste der Begriff Arbeitsrecht
         * eingegeben. Nach der angezeigten Trefferliste, werden die Einträge des
         * Branchenbuchs gefilter und auf der Konsole angezeigt.
         *
         * @throws InterruptedException
         */
        @Test(description = "Calculating Test")
        public void checkPortalsucheArbeitsrecht() throws InterruptedException {
                waitForElementXpath("button", "type", "submit");
                sendKeysById("header-search-input", "Arbeitsrecht");
                waitForElementXpath("button", "type", "submit");
                findElementByXpathAndClick("button", "type", "submit");
                waitForElementXpath("div", "class", "result-link");
                String results = driver.findElement(By.partialLinkText
                                ("/branchenbuch/hamburg")).getText();
                System.out.println(results);

        }
}
```

Listing 6.1: *Portalsuche nach Arbeitsrecht*

Der folgende Ausschnitt illustriert den zum Quellcode 128 (Listing 6.1) gehörenden XML-Datei. Die Kommentare in dem XML-Code sind in grau unterlegt und tragen zum Verständnis bei.

```
<?xml version="1.0" encoding="utf-8"?>
<!--Set thread-count = 3 to execute test parallel in 3 max browsers at at
```

```
        time. You can increase it -->
<suite name="Parallel Tests" verbose="1" thread-count="2" parallel="tests">
        <tests>
                <!--Set test parameters to execute test in firefox browser on Windows
                    platform. -->
                <test name="Linux+firefox Test1">
                        <parameters>
                                <parameter name="platform" value="linux" />
                                <parameter name="browser" value="firefox" />
                                <parameter name="url" value="http://hamburg.de/" />
                                <parameter name="version" value="57.0b5" />
                        </parameters>
                        <classes>
                                <class name="suchen.Portalsuche" />
                        </classes>
                </test>

                <!--  Set test parameters to execute test in chrome browser on Windows
                    platform.-->
                <test name="Linux+chrome Test1">
                        <parameters>
                                <parameter name="platform" value="linux" />
                                <parameter name="browser" value="chrome" />
                                <parameter name="url" value="http://hamburg.de/" />
                                <parameter name="version" value="61.0.3163.100" />
                        </parameters>
                        <classes>
                                <class name="suchen.Portalsuche" />
                        </classes>
                </test>


        </tests>
</suite>
```

Listing 6.2: *Parametrisierung durch XML-Datei*

Um gewährleisten zu können, dass die Funktionalität der Testfälle und deren Elemente auf unterschiedlichen Webbrowser erfolgreich ist, bietet sich ein sogenanntes Cross-Browser-Testing. Dadurch können Kompatibiltätsprobleme vermieden werden, die aufgrund der Tatsache auftauchen, dass der „[...] *standardisierte Quellcode individuell interpretiert* [...]“wird von den verschiedenen Webbrowsern.[Reb16]

Zu Beginn startet die XMl-Datei mit der Bezeichnung $<$*suite*$>$ und wird abgeschlossen mit $<$ /*suite*$>$.
Daraufhin folgen die Parameter:
name - Name der Testsammlung
verbose - „[...] *Logging-Level von TestNG* [...]“ist konfigurierbar.[Edl07, S.29/30]
Der Log gibt Informationen über den Testausgang aus bezüglich der Testklassen und Methoden. Je höher die Zahl definiert wird (maximal Level 10) wird, desto detaillierter, wird darüber informiert, „[...] welche Klassen und Methoden aufgerufen und welche Verzeichnisse für den Report erstellt werden. Am beliebtesten ist der Defaultmode 1, da er lediglich den Namen der Suit, die Runs, die Fehler und die übersprungenen Tests mit Skip ausgibt.“[Edl07, S.29/30]
parallel - Eine Boolesche Variable, die *true* oder *false* übergibt. Sobald *true* angegeben

wird, werden mehrere verschiedene Threads aktiviert, wodurch alle Testfälle durchgeführt werden können. Dieser Fall beschleunigt die Ausführung. Eine explizite Angabe bei **parallel** parallelisiert den enstprechenden XML-Test-Tag.[261]
thread-count - Durch die Angabe eines Integers wird die Anzahl der zu aktivierenden Threads definiert. „*Wird bei parallel true angegeben, aber kein thread-count, so wird hier per Default mit fünf Threads gearbeitet.*“[Edl07, S.29/30]
Weitere optionale XML-Konfigurationen wurden für den Testfall nicht berücksichtigt.

Die XML-Datei enthält die nötige Test-Suite-Beschreibung (*class name=„suchen.Portalsuche“*, greift durch den Befehl auf den Testfall *Portalsuche* im Package *suchen* zu und kann die Ausführung des Testfalls starten. Der Testfall wird parallel auf Mozilla Firefox und Google Chrome ausgeführt, codiert im Programmcode 128 (Listing 6.1) und parametrisiert bei 35 (Listing 6.2).

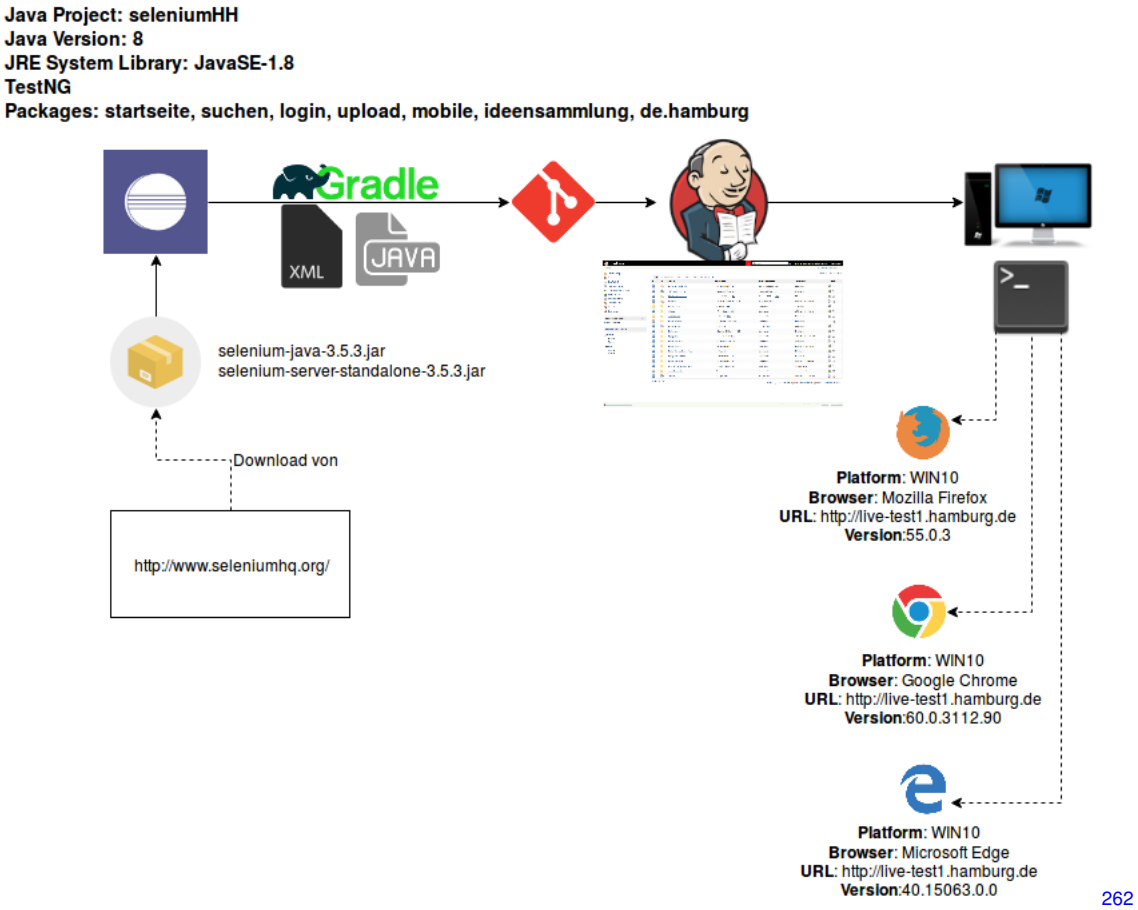
6.2.1 Durch Jenkins gestartete Tests

Der Prozess eines durch Jenkins ausgeführten Testfalls ähnelt dem aus Kapitel 6.1.1 behandelten Ablauf, weist jedoch ein paar gravierende Unterschiede auf:

- Jenkins Server fungiert als Selenium Grid Hub durch das eingebundene Selenium Plugin, welches bereits erwähnt wurde in Kapitel 4
- Einsatz von einem auf Gradle basierten Build-Script *build.gradle*
- Alle für den automatiserten Testprozess relevanten Dateien, insbesondere Java-Klassen, XML-Dateien und *build.gradle*, werden in einem eingerichteten Git Verzeichnis gelagert, findet Erwähnung im Kapitel 4.5). Auf die Dateien kann der Jenkins anhand der angegebenen URL für das Repository erfassen.

Der abstrakt dargestellte Algorithmus auf Abbildung 6.11 wurde vorwiegend im Kapitel 4 behandelt.

[261] [Edl07, S.29/30]



[262]

Abbildung 6.11: *Ablauf für durch Jenkins ausgeführte Testfälle in abstrakter Darstellung*

Im Jenkins wurden folgende Konfigurationen gespeichert, um den Workflow aus Abbildung 6.11 zu realisieren:

- Version der JDK, welches für das Projekt *seleniumHH* verwendet wurde: Java 8
- Source-Code-Management auf Git eingestellt:
  - Repository URL *http://cheyenne-trac.hh.int:8113/hh/selenium.git*
- Buildverfahren erfolgt durch den Aufruf eines Gradle Build-Script (Gradle Version 3.5)
- Als Post-Build-Aktionen wurde der Pfad der auf XML basierten Testberichte festgelegt: *build/test-results/test/TEST-*.xml*

[262] Erstellt mit draw.io

Dieser befindet sich auf dem Jenkins erstellten Arbeitsbereich *selenium*.
Ein kurzes Bespiel folgt im Kapitel 10.

Für ein besseres Verständnis wurde ein folgendes BPMN-Modell 6.12 mit der Software Signavio erstellt, um den komplexeren Worflow auf eine einfache Weise zu visualisieren. Die folgende Beschreibung eines durch Jenkins automatisierten Testvorgangs bezieht sich auf die Abbildung 6.12.

Die organisatorische Einheit, auch als Pool bezeichnet[263], wird hier durch die Qualitätssicherung symbolisiert, die einen automatisierten Testprozess behandelt. Es existieren drei Lanes (Prozessbeteiligte)[264] , in dem Fall Tester, Jenkins-Server und Entwickler, die jeweils einen Akteur repräsentieren und gezielte Verantwortungen tragen. Zunächst werden die Testfälle von einem Tester programmiert und auf das Selenium-Framework zugeschnitten durch Einbindung von Selenese-Befehlen. Anschließend werden den auf Eclipse IDE entwickelten Tests eine erste lokale Überprüfung mithilfe von TestNG unterzogen, siehe hierfür Abschnitt 6.1.1.
Die lokale Überprüfung fokussiert sich hierbei auf die erfolgreiche Durchführung als auch auf die Testresultate. Durch eine Fallunterscheidung, dargestellt durch ein exklusives Gateway, wird über den weiteren Prozessverlauf entschieden.[265] Wenn die Ergebnisse den Soll-Werten entsprechen, werden diese erfolgreichen Testdateien in das git-Repository commitet. Andererseits, bei aufkommenden Fehlern in der Testdurchführung, wird die Ursache der Fehler mithilfe von durch TestNG angegebenen Stacktraces ermittelt.
Die erfolgreichen Testdateien, die nun im Repository gespeichert sind, werden bei der Ausführung eines Jobs, der Erstellungsprozess, von dem Jenkins-Server aufgegriffen. Ausgeführt werden diese auf unterschiedlichen Nodes, hier beispielhaft durch ein Windows 10-Rechner abgebildet. Die Testfälle sind auf Cross-Browser-Testing ausgelegt und werden parallel auf den Webbrowsern Google Chrome, Mozilla Firefox sowie Microsoft Edge durchgeführt.
Sobald der Jenkins Job beendet ist, wird der Testabschluss auf Weboberfläche von Jenkins angezeigt und gleichzeitig (paralleles Gateway) ist es möglich die Resultate der Testdaten auf TestNG als auch über Test Results Analyzer in visueller Form zu betrachten, siehe folgendes Kapitel 7. Anschließend werden die Projektteilnehmer, die über Jenkins konfiguriert wurden, benachrichtigt, beispielsweise über E-Mail.
Im nächsten Task werden die Testdaten ausgewertet. Darin sind ebenfalls die anfangs beschriebenen fehlgeschlagenen Testdateien enthalten. Der Tester analysiert und wertet die vorhandenen Testdaten aus. Dabei entscheidet dieser nach den festgelegten Soll-Richtlinien

[263] [vgl. Mey17, S.258]
[264] [vgl. Mey17, S.256]
[265] [vgl. Mey17, S.279]

oder gemäß den Anforderungen, ob der jeweilige Testfall als erfolgreich gilt. Wenn dies der Fall ist, werden die verifizierten Testfälle in einen Datenspeicher gespeichert, hierbei dargestellt durch die Google Cloud Platform.
Wenn die Auswertung des Testers negativ ausfällt, gilt es als nächsten Schritt die Fehlerursache zu erfassen, die in eine weitere Falluterscheidung mündet. Handelt es sich um einen Testfehler, wird der Testfehler mithilfe vom Stacktrace, aber auch Online-Ressourcen, wie Stackoverflow oder Selenium-Community, behoben. Daraufhin finden nächste lokal automatisierte Überprüfungen der korrigierten Tests statt.
Wenn ein Produktfehler diagnostiziert wurde, wird der Entwickler alarmiert und darauf aufmerkam gemacht. Der existierende Fehler wird zwischen dem Tester und Entwickler kommuniziert. Infolgedessen handelt der Entwickler entsprechend und versucht den scheinbaren Produktfehler zu korrigieren. Nach der Korrektur kann über ein Deployment der Fehler bereinigt werden. Abschließend wird der Tester benachrichtigt, der einen erneuten Testprozess des jeweiligen Testfalls einleitet. Die Benachrichtigung wird hierbei durch ein Message-Event gekennzeichnet.

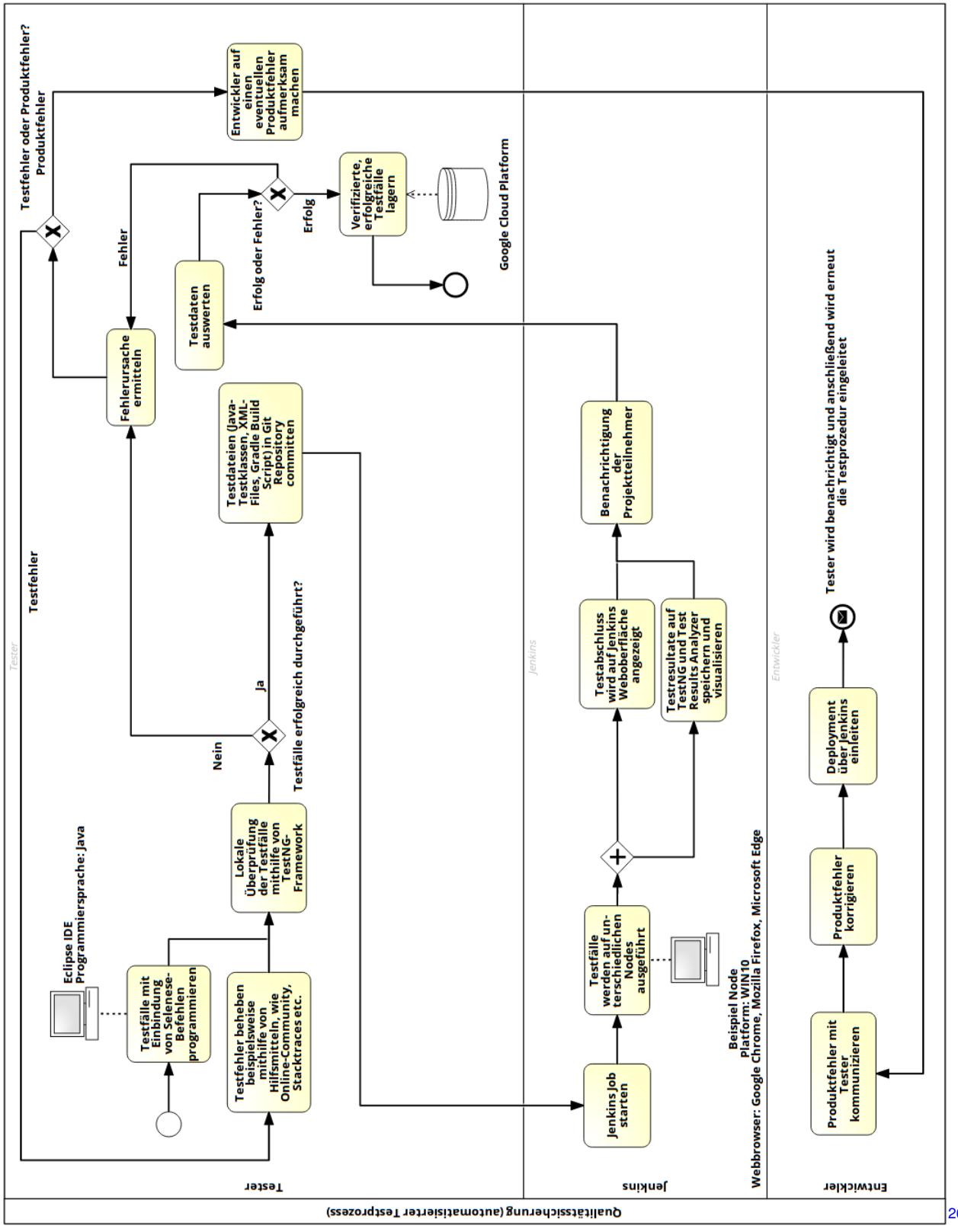
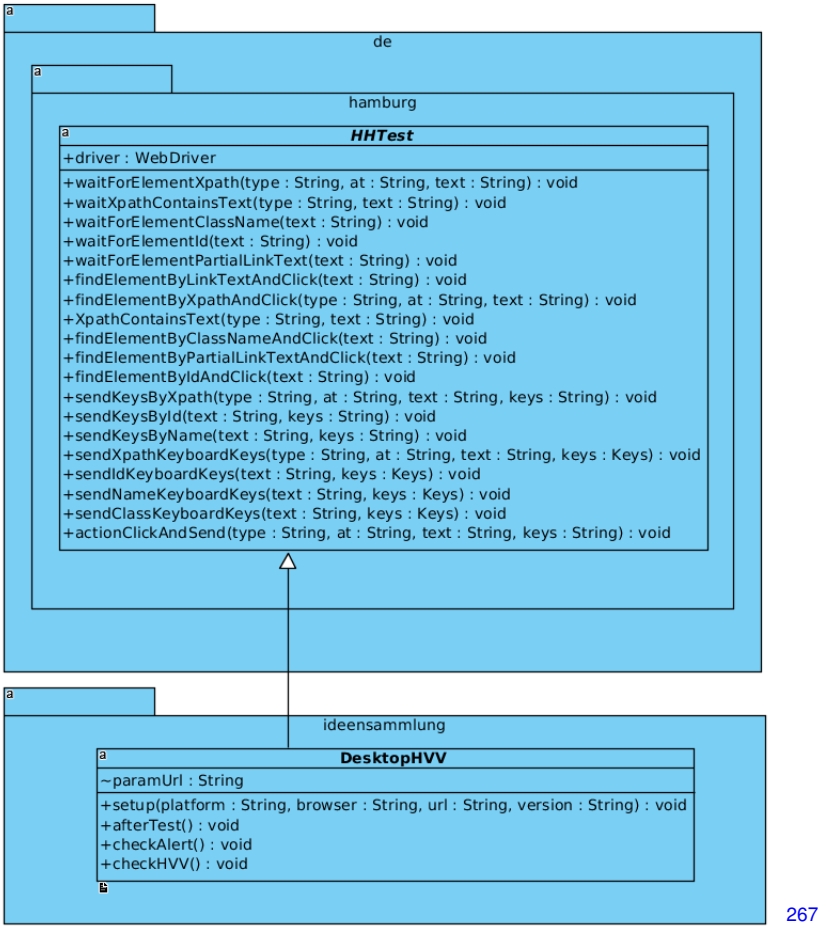


Abbildung 6.12: *Modellierung des durch Jenkins automatisierten Testprozesses durch BPMN*

6.3 Fallbeispiel



267

Abbildung 6.13: *Vereinfachtes Klassendiagramm der Klasse DesktopHVV*

Die illustrierte Oberklasse auf der Abbildung 6.13 wurde bereits im Kapitel 6.10 ausführlich beschrieben.
Die Methoden der Unterklasse *DesktopHVV* werden in dem nachfolgenden Quellcode 121 ausreichend kommentiert.

[266] Erstellt mit Signavio
[267] Erstellt mit Visual Paradigm

```
package ideensammlung;

import java.net.MalformedURLException;

import org.openqa.selenium.Keys;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.testng.annotations.AfterTest;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Parameters;
import org.testng.annotations.Test;

import de.hamburg.HHTest;
import de.hamburg.browser.Chrome;
import de.hamburg.browser.Edge;
import de.hamburg.browser.Firefox;

/**
 * Selenium Test: Suche nach einer HVV-Verbindung
 * @author jawar
 *
 */
public class DesktopHVV extends HHTest {

        String paramUrl;
/**
 *
 * @param platform
 * @param browser
 * @param url
 * @param version
 * @throws MalformedURLException
 * Die Initialisierungen inklusive der notwendigen Eigenschaften
 *  der Webbrowser wurden ausgelagert in separte Java-Klassen
 */
        @Parameters({ "platform", "browser", "url", "version" })
        @BeforeTest(alwaysRun = true)
        public void setup(String platform, String browser, String url, String version)
            throws MalformedURLException {
                paramUrl = url;
                DesiredCapabilities caps = new DesiredCapabilities();
                caps.setPlatform(org.openqa.selenium.Platform.WIN10);
                System.setProperty("java.net.preferIPv4stack", "true");
                caps.setCapability("SeleniumTests", "redhat5 && amd64");

                if (browser.equalsIgnoreCase("firefox")) {
                        driver = Firefox.initFirefox(url, driver);

                } else if (browser.equalsIgnoreCase("chrome")) {
                        driver = Chrome.initChrome(url, driver);

                } else if (browser.equalsIgnoreCase("edge")) {
                        driver = Edge.initEdge(url, driver);

                } else {
                        throw new IllegalArgumentException("The Browser Type is
                            undefined");
                }
        }

        /**
         * Die Methode afterTest() dient dem erfolgreichen Schließen des jeweiligen
         * Browserfensters
         */
        @AfterTest
        public void afterTest() {
                driver.quit();

        }


/**
 *
 * @throws InterruptedException
 * Ablauf: -Warten auf das Element Bus & Bahn und anschließend
 *                      bei erfolgreicher Lokalisierung anklicken
 *                 -Warten auf Abfahrt-Suchleiste der Suchmaske HVV-Verbindung
```

```
 *                      und anklicken
 *                  -Senden der Zeichenkette 'Hallerstraße' bei Abfahrt
 *                  -Warten auf angezeigte Vorschläge und auswählen (Keys.ARROW_DOWN),
 *                      bestätigen durch Eingabetaste
 *                  - Gleicher Vorgang bei Ankunft-Suchleiste
 *                  - Vorgegebene Zeit, bestehend aus 6 Zeichen, löschen
 *                  - Zeit '19:00' als Zeichenkette übersenden
 *                  - Warten auf Bestätigungsbutton, Button: HVV-Suche
 *                       und bei erfolgreicher Lokalisierung betätigen
 *                  - Warten auf Ergebnismaske der personalisierten
 *                  - HVV-Verbindung
 */
        @Test(description = "Calculating Test")
        public void checkHVV() throws InterruptedException {
                waitForElementXpath("a", "data-click-link-target",
        "http://site.hamburg.de/hamburg/hamburg/s?HVV_Link_Header&ns_type=clickout");
                findElementByXpathAndClick("a", "data-click-link-target",
        "http://site.hamburg.de/hamburg/hamburg/s?HVV_Link_Header&ns_type=clickout");
                waitForElementId("starting-point");
                sendKeysById("starting-point", "Hallerstraße");
                waitForElementXpath("div", "class", "title-wrapper");

                sendIdKeyboardKeys("starting-point", Keys.ARROW_DOWN);
                sendIdKeyboardKeys("starting-point", Keys.ENTER);

                waitForElementId("destination");
                sendKeysById("destination", "Alter Teichweg");
                waitXpathContainsText("a", "Alter Teichweg, Hamburg");

                sendIdKeyboardKeys("destination", Keys.ARROW_DOWN);
                sendIdKeyboardKeys("destination", Keys.ENTER);

                waitForElementId("departure");
                findElementByIdAndClick("departure");
                sendIdKeyboardKeys("departure", Keys.ARROW_LEFT);
                waitForElementXpath("input", "name", "time");
                for (int i = 0; i <= 5; i++) {
                        sendNameKeyboardKeys("time", Keys.BACK_SPACE);
                }
                sendKeysByName("time", "19:00");
                waitForElementXpath("button", "class", "hvv-submit");
                findElementByXpathAndClick("button", "class", "hvv-submit");
                waitForElementClassName("hvv-info-ticket");

        }
}
```

Listing 6.3: *HVV-Suche „hamburg.de“*

Die Initialisierungen der Webbrowser wurden in Form von Java-Klassen ausgelagert und mit notwendigen Eigenschaften, wie Name des Webbrowsers, Version sowie Bezeichnung und Pfad des jeweiligen Webbrowsers definiert. Dies sorgt für eine bessere Übersicht.

```
package de.hamburg.browser;

import java.net.MalformedURLException;
import java.net.URL;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.remote.RemoteWebDriver;

public class Firefox {
```

```
        public static WebDriver initFirefox(String URL, WebDriver driver) throws
            MalformedURLException {
                DesiredCapabilities caps;
                System.out.println("Executing on Firefox");
                String Hub = "http://cheyenne-trac.hh.int:4444/wd/hub";
                caps = DesiredCapabilities.firefox();
                caps.setBrowserName("firefox");
                caps.setVersion("57.0b12");
                System.setProperty("webdriver.gecko.driver",
                    "P:\\DEV\\geckodriver.exe");

                driver = new RemoteWebDriver(new URL(Hub), caps);

                driver.navigate().to(URL);
                driver.manage().window().maximize();

                return driver;
        }
}
```

Listing 6.4: *Zusätzliche Initialisierung für Mozilla Firefox*

```
package de.hamburg.browser;

import java.net.MalformedURLException;
import java.net.URL;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.remote.RemoteWebDriver;

public class Chrome {

        public static WebDriver initChrome(String url, WebDriver driver) throws
            MalformedURLException {
                DesiredCapabilities caps;
                System.out.println("Executing on Chrome");
                String Hub = "http://cheyenne-trac.hh.int:4444/wd/hub";
                caps = DesiredCapabilities.chrome();
                caps.setBrowserName("chrome");
                System.setProperty("webdriver.chrome.driver",
                    "P:\\DEV\\chromedriver.exe");
                ChromeOptions o = new ChromeOptions();

                o.addArguments("--start-maximized");
                o.addArguments("load-extension=C:\\Users\\jenkins\\AppData"
                        + "\\Local\\Google\\Chrome\\User Data\\Default"
                        + "\\Extensions\\gighmmpiobklfepjocnamgkkbiglidom\\3.18.0_0");

                caps.setCapability(ChromeOptions.CAPABILITY, o);
                driver = new RemoteWebDriver(new URL(Hub), caps);

                driver.navigate().to(url);
                String originalHandle = driver.getWindowHandle();
                for (String winHandle : driver.getWindowHandles()) {
                        if (!winHandle.equals(originalHandle)) {

                                driver.switchTo().window(winHandle);
                                driver.close();

                        }

                }
                driver.switchTo().window(originalHandle);

                return driver;
        }
}
```

Listing 6.5: *Zusätzliche Initialisierung für Google Chrome*

```
package de.hamburg.browser;

import java.net.MalformedURLException;
import java.net.URL;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.remote.RemoteWebDriver;

public class Edge {
        public static WebDriver initEdge(String URL, WebDriver driver) throws
            MalformedURLException {
                DesiredCapabilities caps;
                System.out.println("Executing on Firefox");
                String Hub = "http://cheyenne-trac.hh.int:4444/wd/hub";
                caps = DesiredCapabilities.edge();
                caps.setBrowserName("edge");
                caps.setVersion("40.15063.674.0");
                System.setProperty("webdriver.edge.driver",
                    "P:\\DEV\\MicrosoftWebDriver.exe");

                driver = new RemoteWebDriver(new URL(Hub), caps);

                driver.navigate().to(URL);
                driver.manage().window().maximize();

                return driver;
        }
}
```

Listing 6.6: *Zusätzliche Initialisierung für Microsoft Edge*

Die Parameter für die im Testfall 121 vorkommenden TestNG-Annotationen werden aus der folgenden XML-Datei aufgegriffen.

```
<?xml version="1.0" encoding="utf-8"?>
<!--Set thread-count = 3 to execute test parallel in 3 max browsers at at
        time. You can increase it -->
<suite name="Parallel Tests" verbose="1" thread-count="2" parallel="tests">
        <tests>
                <!--Set test parameters to execute test in Firefox browser on
                    Windows platform. -->
                <test name="Linux+firefox Test1">
                        <parameters>
                                <parameter name="platform" value="WIN10" />
                                <parameter name="browser" value="firefox" />
                                <parameter name="url"
                                    value="http://www.hamburg.de/" />
                                <parameter name="version" value="57.0b12" />
                        </parameters>
                        <classes>
                                <class name="ideensammlung.DesktopHVV" />
                        </classes>
                </test>

                <!--  Set test parameters to execute test in Chrome browser on
                    Windows platform.-->
                <test name="Linux+chrome Test1">
                        <parameters>
                                <parameter name="platform" value="WIN10" />
                                <parameter name="browser" value="chrome" />
                                <parameter name="url"
                                    value="http://www.hamburg.de/" />
                                <parameter name="version" value="62.0.3202.62" />
                        </parameters>
                        <classes>
                                <class name="ideensammlung.DesktopHVV" />
                        </classes>
                </test>

                <!--  Set test parameters to execute test in Edge browser on
                    Windows platform.-->
                <test name="Linux+edge Test1">
                        <parameters>
                                <parameter name="platform" value="WIN10" />
                                <parameter name="browser" value="chrome" />
                                <parameter name="url"
                                    value="http://www.hamburg.de/" />
                                <parameter name="version" value="40.15063.674.0" />
                        </parameters>
                        <classes>
                                <class name="ideensammlung.DesktopHVV" />
                        </classes>
                </test>
        </tests>
</suite>
```

Listing 6.7: *Parametrisierung durch XML-Datei*

Wie bereits in Kapitel 6.1.1, wird auch hierbei im nachfolgenden Gradle Build-Script, der Pfad der auszuführunden TestNG-XML-Datei festegelegt.

```
apply plugin: 'java'
apply plugin: 'eclipse'

jar {
        version  '1.0'
        baseName 'SeleniumStarter'
        extension '.jar'
}

sourceCompatibility = 1.8
targetCompatibility = 1.8

description = ""

repositories {

    mavenCentral()
    mavenLocal()
}

ext.seleniumVersion = '3.5.3'

dependencies {
    compile group: 'org.seleniumhq.selenium', name: 'selenium-java',
        version:seleniumVersion
    compile group: 'org.seleniumhq.selenium', name: 'selenium-server',
        version:seleniumVersion
    compile group: 'org.seleniumhq.selenium', name: 'selenium-edge-driver',
        version:seleniumVersion
    compile group: 'org.seleniumhq.selenium', name: 'selenium-firefox-driver',
        version:seleniumVersion
    compile group: 'org.seleniumhq.selenium', name: 'selenium-chrome-driver',
        version:seleniumVersion
        compile group: 'org.seleniumhq.selenium', name: 'selenium-api',
            version:seleniumVersion
        compile group: 'org.seleniumhq.selenium', name: 'selenium-java',
            version:seleniumVersion

    compile group: 'org.uncommons', name: 'reportng', version:'1.1.4'
    testCompile group: 'junit', name: 'junit', version:'4.12'
    testCompile group: 'org.testng', name: 'testng', version:'6.11'
}

test {
     useTestNG {

        suites 'src/test/resources/ParallelTest.xml'

     }
 }


eclipse {
  classpath {
  containers 'org.springsource.ide.eclipse.gradle.classpathcontainer'}
}

// A custom task to show report on tests that have run
task viewResults(dependsOn: ['test'] , type:Exec) {
                workingDir './build/reports/tests'
                commandLine 'cmd', '/c', 'start index.html'
}

task wrapper(type: Wrapper) {
    gradleVersion = '2.10' //we want gradle 2.10 to run this project
}
```

Listing 6.8: *build.gradle für das Projekt „seleniumHH“*

6.4 Herausforderungen im Bereich der Implementierung

Bis zum 26. Februar 2017 wurde die JDK7 Version der Selenium Standalone Server als im Jenkins integrierten Plug-in nicht aktualisiert. Seit Selenium 3 laufen die Bibliotheken allerdings unter JDK8.
Dadurch bestand zunächst ein Kompatibilitätsproblem, welcher weiterhin auftaucht, jedoch durch ein Vorgehen im folgenden Unterkapitel gelöst werden konnte.

6.4.1 Workaround zur Versionsproblematik

Mihilfe des Entwicklerforums Stack Overflow, in dem Hilfestellungen von Entwicklern rundum die Thematik der Softwareentwicklung angeboten werden und jegliche Fragen beantwortet werden können, konnte das Problem umgegangen werden. Die folgenden Veränderungen waren an den zur Verfügung gestellten Quellcodes, XML und Java, nötig und wurden auf der Entwicklungsumgebung vorgenommen. Alle vorliegenden Dateien befinden sich im Projektordner **selenium-plugin**, welcher über GitHub heruntergeladen werden kann.

- Dateiname: **maven-metadata-local.xml**
  Pfad: dist-server-standalone/local_m2/org/seleniumhq/selenium/htmlunit-driver-standalone/maven-metadata-local.xml
  Bei diesem XML-Code wurde ausschließlich das Release-Datum angepasst und aktualisiert. Dieser Teil ist optional, beschrieben im Anhang 9.
- Dateiname: **selenium-server-standalone-3.1.0.pom**
  Pfad: selenium-plugin/dist-server-standalone/local_m2/org/seleniumhq/selenium/selenium-server-standalone/3.1.0/selenium-server-standalone-3.1.0.pom
  In der POM-Datei wurde die im Maven Projekt angegebene Versionsnummer mit der aktuellen Version des heruntergeladenen Selenium Standalone Servers angeglichen. In dem vorliegenden Beispiel lautet die *3.1.0*, jedoch kann diese je nach Belieben angepasst werden, vorausgesetzt sie stimmt mit der heruntergeladenen Version überein. Belege hierfür befinden sich im Anhang 9.
- Dateiname: **maven-metadata-local.xml**
  Pfad: selenium-plugin/dist-server-standalone/local_m2/org/seleniumhq/selenium/selenium-server-standalone/maven-metadata-local.xml
  Ebenso wie in der vorherigen Datei, können in dieser XML-Datei Versionsnummer sowie Release-Datum verändert werden. Die Relevanz der Versionsnummer ist an dieser Stelle höher als das Release-Datum, welches optional ist und ebenfalls augeführt wird im Anhang 9.

- Dateiname: **pom.xml**
  Pfad: dist-server-standalone/pom.xml
  Erneut müssen hier die Selenium Versionen mit der gewünschten heruntergeladenen Standalone Server-Version übereinstimmen. Diese Informationen sind gelistet im Anhang 9.

- Dateiname: **pom.xml**
  Pfad: pom.xml
  In der sogenannten Parent POM, die von Maven gelesen wird, befinden sich alle Informationen sowie Eigenschaften des Projekts über die Versionen der *artifactID,dependencies,groupID* etc. Die Parent POM ist ein zentraler Punkt, in der die wichtigen Informationen geschrieben sind, die für das Bauen des Projekts benötigt wird.
  Auch die *htmlunit-driver-standalone* wird in dieser POM aufgeführt. „HtmlUnit"ist ein Browser für auf Java basierte Programme, der keine GUI verwendet. HtmlUnit dient primär zur Modellierung von HTML Dokumenten. Außerdem liefert es eine API, die einem User beispielsweise erlaubt Seitenaufrufe durchzuführen. Üblicherweise wird HtmlUnit zu Testzwecken verwendet.[268]
  Die aktuelle Version der HtmlUnit lautet *2.2.7-with-dependencies*[269].

- Klasse: **HubLauncher.java**
  src/main/java/hudson/plugins/selenium/HubLauncher.java
  In der Klasse *HubLauncher.java* wird der Grid Hub des Selenium Grid Plug-ins konfiguriert. Um eine eigene neue Konfiguration des Hubs vornehmen zu können, wird ein neues Objekt vom Typ *GridHubConfiguration* erzeugt. Das neu erzeugte Objekt *gridconfig* wird dem JCommander. JCommander ist ein Java Framework, welches mithilfe von Annotationen einem Objekt Daten aus einer Kommandozeile übergibt.[270] und, nach der Zuweisung von *port, capabilityMatcher*, dem erstellten Hub übergeben. Weitere Informationen folgen in Form von Kommentaren im Programmcode in Kapitel 9.

- Klasse: **HubParamsCallable.java**
  src/main/java/hudson/plugins/selenium/HubParamsCallable.java
  Diese Klasse ist verantwortlich für das Aufrufen von Parametern, die für den Hub notwendig sind.
  Die Einbindung der Konfiguration wurde im Aufruf berücksichtigt durch *getConfiguration()*, sodass sowohl der Host als auch der Port des Hubs Änderungen annehmen können.

- Klasse: **RemoteControlLauncher.java**
  src/main/java/hudson/plugins/selenium/RemoteControlLauncher.java

[268] http://htmlunit.sourceforge.net/
[269] http://www.seleniumhq.org/download/
[270] [Dzi12]

In dieser Klasse wird durch die Einbindung des neu erzeugten Objektes *nodeconfig* vom Typ *GridNodeConfiguration* die Konfiguration eines ferngesteuerten Nodes über den Jenkins-Server gewährleistet.
Weitere Details werden im Quellcode im Kapitel 9 kommentiert.

- Klasse: **RemoteStopSelenium.java**
  src/main/java/hudson/plugins/selenium/callables/RemoteStopSelenium.java
  Diese Klasse widmet sich dem Stoppen des ferngesteuerten Nodes.
  Um den konfigurierten Node zu ermitteln, greift der Parameter *srr* vom Typ *SelfRegisteringRemote* auf den eingestellten Host sowie Port zu.
  Diese beiden Parameter *host,port* werden anschließend in Form von einem String zurückgegeben.

- Klasse: **SeleniumProcessUtils.java**
  src/main/java/hudson/plugins/selenium/process/SeleniumProcessUtils.java
  Bei „SeleniumProcessUtils.java“geht es um die Versionierung des „Selenium Standalone Servers“. Dadurch, dass sich bei der JAR-Datei mittlerweile um die dritte Generation handelt, wurde das Paket *org.openqa.grid.selenium.GridLauncherV3* importiert und anschließend in der Klasse zurückgegeben. Diese Änderung ist im Progammcode des Anhangs in Kapitel 9 zu betrachten.

Nach den Veränderungen folgt das Bauen des Projekts im Ordner *selenium-plugin* durch den Befehl

mvn clean install -U -Dmaven.test.skip=true

über das Terminal.

Im Ordner *selenium-plugin/target* befindet sich die Datei *selenium.hpi* nach dem erfolgreichen Build-Prozess von Maven.
Diese Datei lässt sich anschließend durch Zugriff auf den Jenkins-Server über eine SSH-Verbindung ablegen. Der Befehl dazu lautet: *ssh jenkins@cheyenne-trac.hh.int*. Zu berücksichtigen ist die Umbenennung der Endung von **hpi** zu **jpi**, wodurch der Jenkins-Server die Datei als Plug-in erfassen kann.
Zusätzlich muss auch der Ordner *selenium*, auffindbar durch Dateipfad: *selenium-plugin/target/*, im Jenkins-Server abgelegt werden. Dieser Ordner beinhaltet wiederum einen *META-INF*- sowie einen *WEB-INF* -Verzeichnis, die notwendig sind für das Plug-in. META-INF- Verzeichnisse beinhalten Manifest-Dateien zu einer „Jar-Datei (*Java-Archiv-Datei*“).[Lan] Das WEB-INF-Verzeichnis „enthält Konfigurations-Dateien und die benötigten

Bibliotheken und Java-Klassen der Anwendung. Dieses Verzeichnis ist für jede Web-Anwendung obligatorsich.“[Rit05]

Abschließend muss der Jenkins-Server neugestartet werden.
Das Kommando über das Terminal lautet: *sudo /etc/init.d/jenkins restart*
Für die Verifizierung der Version kann über die Konsole *cheyenne-trac.hh.int:4444/grid/console* die Versionsnummer nachgeschaut werden.
Die Lösungsvorschläge vom Stack Overflow werden im Anhang aufgeführt.

6.4.2 Probleme beim Cross-Browser-Testing

Zu den häufig auftretenden Problemen hinsichtlich Cross-Browser-Testing, die auch während der Testphase teilweise aufgetreten sind, zählen

- Abweichung in der Darstellung der Schriftgröße (engl. font-size)
- Unterschiedlich JavaScript Implementationen
- CSS-,HTML-Fragmente können hinsichtlich der Validierung unterschiedlich sein
- Seitenlayout und Größe der Elemente der Website können variieren
- Ausrichtung eines Bildes kann sich unterscheiden
- Kompatibilitätsprobleme zwischen Webbrowser und Betriebssystem[271]

Dementsprechend ist es immens wichtig Versionen der eingebundenen Selenium-Pakete, der Webbrowser sowie des verwendeten Betriebsystems zu notieren, um ggf. Kompatibilitätsprobleme zu ermitteln und andere Problemursachen leichter ausschließen zu können. Dadurch wird der nachfolgende Testprozess weniger beeinträchtigt.

[271] [vgl. Cas14, S.503/504]

7 Testauswertung

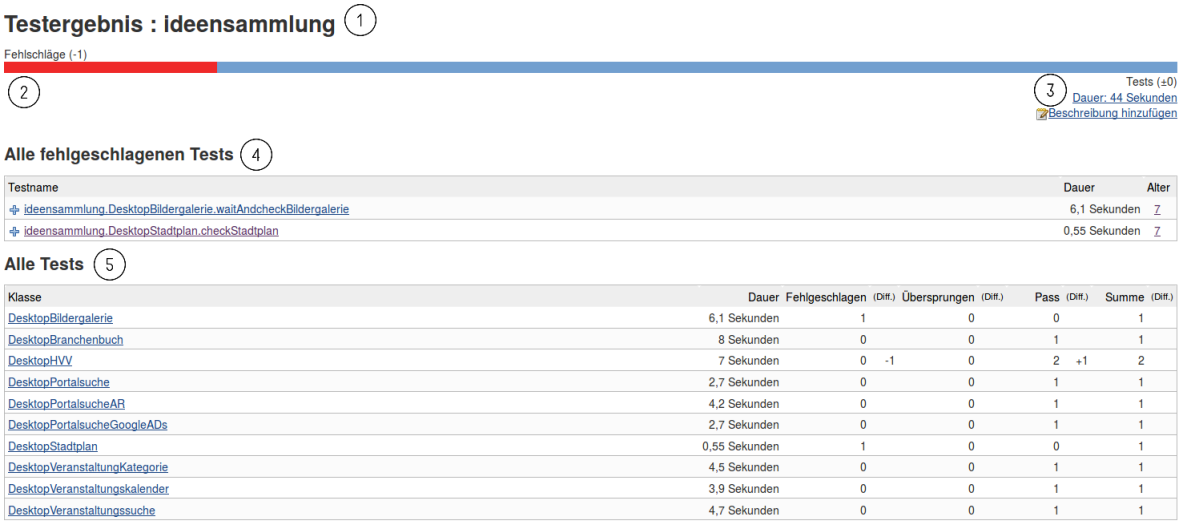
Das Kapitel Testauswertung widmet sich den Ergebnissen, die aus den Testfällen hervorgehen und mithilfe der eingebundenen Plug-ins anschaulich dargestellt werden können. Dabei spezialisiere ich mich auf die durch Jenkins gestarteten Tests.
Zunächst wird der Aufbau derartiger Auswertungen beschrieben. Daraufhin folgt eine Erläuterung zur Handhabung von aus den automatisierten Tests resultierenden Fehlern.
Abschließend werden die Testergebnisse in grafischer Form abgebildet und analysiert.

7.1 Aufbau

Nach Abschluss der eingestellten Testfälle wird der Tester über die Jenkins-Weboberfläche benachrichtigt, weiteres dazu in Kapitel 4.6.2.
Hierfür wurde ein eigenes Projekt namens *selenium* erstellt.
Im Build-Verlauf werden alle durch Jenkins angestoßenen Tests samt Status, Testnummer, Datum und Uhrzeit angezeigt. Im jeweiligen Test Suite werden zusätzliche Informationen über Änderungen im eingestellten *git-Repository*, in Kapitel 4.5 behandelt, seit dem letzten ausgeführten Test sowie die Revision, Name des Users, Zeitdauer der Ausführung als auch die Rubrik Testergebnis aufgelistet, die durch das installierbare testng-plugin zur Auswahl steht.

Bei der Auswahl Testergebnis wird eine Zusammenfassung von allen durchgeführten Testfällen abgebildet und nach Erfolg und Fehlschlag sortiert. Im Titel inbegriffen ist der Name des Pakets, zu betrachten auf Abbildung 7.1, Punkt (**1**), auf den die Testklassen zugreifen, der in der *build.gradle*-Datei durch die Angabe der XML-Datei gegeben ist. Das Gradle Build-Script wurde im Abschnitt 66 veranschaulicht .
Darunter befindet sich eine Leiste, welche die gesamten Testresultate nach Erfolg und Fehlschlag farblich sortiert. Diese wird auf der folgenden Grafik 7.1, Punkt (**2** abgebildet. Die Gesamtdauer einer Testsammlung wird neben der Leiste angegeben, ebenfalls auf Abbildung 7.1, Punkt (**3**) angegeben.
Die fehlgeschlagenen Tests werden mit Hinweis auf die fehlerverursachende Methode in tabellarischer Darstellung aufgeführt, zu betrachten auf Abbildung 7.1, Punkt (**4**). Zusätzlich

wird die Dauer des einzelnen Testfalls als auch das Alter, also seit wievielen durch Jenkins ausgeführten Testdurchläufen der jeweilige Fehler existent ist. Für weitere Informationen können Anwender durch das Klicken auf die jeweilige Methode Fehlerdetails sowie Stacktraces entfaltet werden, die die Fehlerursache explizit angeben.
Stacktraces dienen als Hilfsmittel für die Ermittlung von Fehlern. Stacktraces, die „[...] *über die Stacktrace-Eigenschaft des Exception-Objekts erreicht, liefert Informationen zu der Methode, zum Konstruktor oder zur Eigenschaft, in der die Ausnahme aufgetreten ist* [...]“(Visual C# 2008: Windows-Programmierung mit dem .NET Framework 3.5, Jürgen Bayer, 2008, Markt+Technik Verlag, S. 454)
In der darunterliegenden Tabelle werden alle Klassen einschließlich mit Dauer, Fehlschlag, Übersprungen, Pass (Dt. Bestanden) und die Summe, also Anzahl der Ausführungen, aufgelistet. Die Auflistung ist auf der folgenden Abbildung 7.1, Punkt (**5**) dargestellt.
Im Folgenden wird beispielhaft eine derartiger Testaufbau illustriert:



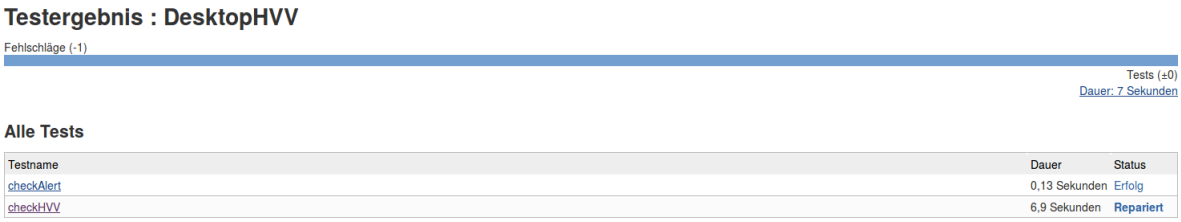
272

Abbildung 7.1: *Screenshot einer ausgeführten Testsammlung aus dem Projektordner selenium, Package: ideensammlung*

Hierfür wurde bewusst ein Testresultat eines Test Suites ausgewählt, welches Fehler aufwies, um die Situation eines Fehlerfalls abbilden zu können.
Wenn ein Testfall im vorherigen Durchlauf fehlgeschlagen ist, jedoch im nächsten Testzyklus erfolgreich ausfällt, wird, wie bei dem Testfall *DesktopHVV* zu erkennen, ein Punkt in der

[272] Screenshot aus Jenkins-Weboberfläche, Projekt *selenium*, Testergebnis vom Package *ideensammlung*

Spalte *Fehlgeschlagen* subtrahiert und der Kategorie *Pass* gutgeschrieben. Dieses Verfahren wird ebenfalls über der Leiste dargestellt auf 7.1, Punkt (**2**)).
Dadurch kann ein Tester eindeutig verifizieren, dass ein überarbeiteter Tesfall erfolgreich repariert wurde, welches in folgender Abbildung betrachtet werden kann.



[273]

Abbildung 7.2: *Testklasse DesktopHVV in der Detailansicht*

Außerdem besteht die Möglichkeit, wie bereits im Kapitel 6.2.1 geschildert, auf XML basierende Testberichte zuzugreifen:

```
<?xml version="1.0" encoding="UTF-8"?>
<testsuite name="ideensammlung.DesktopHVV" tests="2" skipped="0" failures="0"
    errors="0" timestamp="2017-10-30T19:59:48"
    hostname="jawar-Entry-Tower-Workstation" time="8.356">
  <properties/>
  <testcase name="checkAlert" classname="ideensammlung.DesktopHVV" time="0.038"/>
  <testcase name="checkHVV" classname="ideensammlung.DesktopHVV" time="8.318"/>
  <system-out><![CDATA[alert is not present
]]></system-out>
  <system-err><![CDATA[]]></system-err>
</testsuite>
```

Listing 7.1: *XML-Testbericht anhand Testklasse DesktopHVV*

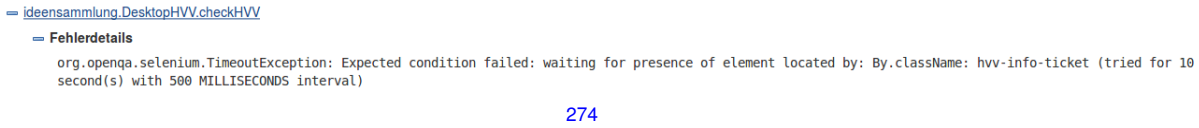
Dabei werden folgende wichtige Informationen angegeben:
name - Name des Packages sowie der zugegriffenen Testklasse, in diesem Fall *„ideensammlung.DesktopHVV“*
test - Anzahl der vohandenen Testmethoden hier: *„2“*
skipped,failures, errors - Resultate der Tests, die als übersprungen, fehlerhalft oder abgebrochen gewertet wurden
timestamp - Genaue Datum-/Zeitangabe eines abgeschlossenen Testfalls
time - Zeitdauer eines Testfalls

[273] Screenshot aus Jenkins-Weboberfläche, Projekt *selenium*, Testergebnis vom Package *ideensammlung*

Anschließend folgen Details bezüglich der einzelnen Testmethoden („testcases“) bestehend aus Bezeichnung der Methode (*name*), Name der Testklasse (*classname*) sowie eine Angabe der Zeitdauer einer Methode (*time*)

7.2 Fehlerprüfung

Die Phase der Fehlerprüfung erfolgt durch eine explizite Fehlermeldung in den Fehlerdetails, die anhand eines folgenden Ausschnitts erläutert wird. Dadurch kann direkt die fehlerhafte Methode, die innerhalb der Testresultate angezeigt wird, verfolgt werden, siehe vorherigen Abschnitt 7.1.
Abbildung 7.3 veranschaulicht solche eine Fehlermeldung anhand der getesteten Methode *checkHVV* der Klasse *DesktopHVV*.



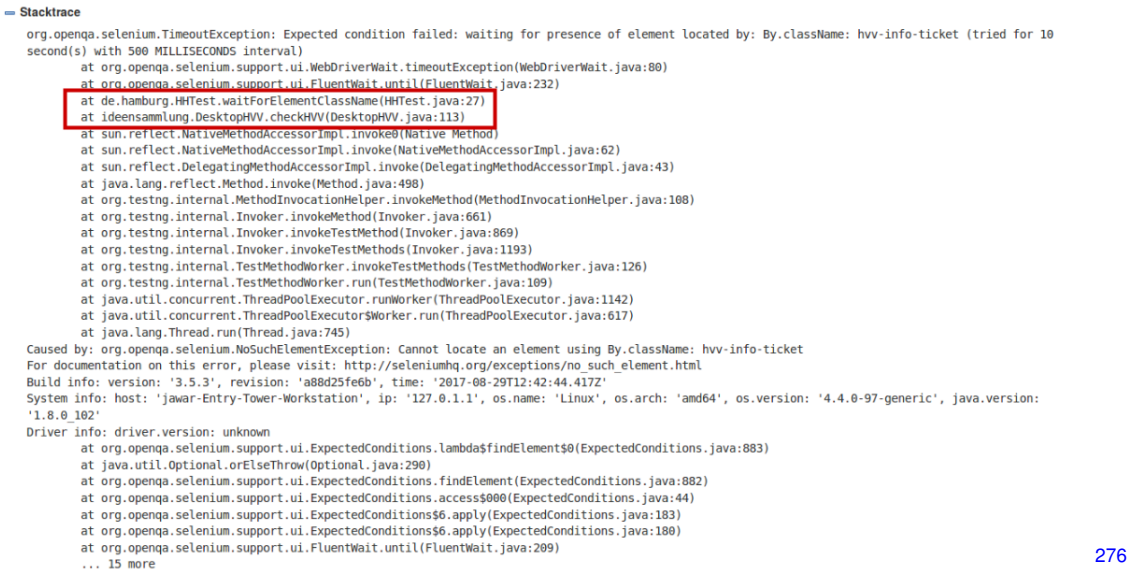
274

Abbildung 7.3: *Fehlerdetails der Klasse DesktopHVV, Methode checkHVV*

Anhand der Fehlerdetails auf Abbildung 7.3 steht folgende Anmerkung im Vordergrund:
Expected condition failed: waiting for presence of element located by: By.className: hvv-info-ticket

Nach 10 sekündiger Wartezeit[275] auf das Element **className=„hvv-info-ticket“** durch die Methode wird eine *TimeoutException* geworfen aufgrund das im DOM fehlende Element. (*wait.until(ExpectedConditions.presenceOfElementLocated(By.className(text))*); Eine alternative Methode wäre an dieser Stelle*wait.until(ExpectedConditions.visibilityOfElementLocated (By.className(text)));*
,die auf die tatsächliche Sichtbarkeit des Elements wartet.

[274] Screenshot aus Jenkins-Weboberfläche, Projekt *selenium*, Testergebnis vom Package *ideensammlung*
[275] *WebDriverWait wait = new WebDriverWait(driver, 10);*



276

Abbildung 7.4: *Stacktrace der Klasse DesktopHVV, Methode checkHVV*

Der in Abbildung 7.4 ausgegebene Stacktrace zeigt ebenfalls die obige Exception (Dt. Ausnahme) mit einer Fehlermeldung an, bekannt aus Abbildung 7.3, und gibt gleichzeitig Auskunft über die Fehlerursache beziehungsweise vielmehr über den Ort, an dem der jewilige Fehler existiert, siehe rote Markierung auf Abbildung 7.4). Stacktraces werden dementsprechend für Debugging-Zwecke verwendet, um eine Fehlerursache, der während der Ausführung auftaucht, diagnostizieren zu können.[277]
Dadurch kann ein Tester zeitnah den Fehler im jeweiligen Testfall nachverfolgen und spart letztendlich Zeit hinsichtlich der Fehlersuche.

Eine weitere häufig auftretene Exception lautet:
org.openqa.selenium.WebDriverException: Element is not clickable at point [...]. Other element would receive the click: [...][278]

Diese Exception deutet auf ein Element hin, welches nicht angewählt werden kann beziehungsweise nicht anklickbar ist aufgrund der Tatsache, dass es durch ein anderes Element verdeckt oder gestört wird und somit die Testklasse nicht erfolgreich erfüllt werden kann.
In der Praxis, insbesondere durch die eigenen automatisierten und manuell ausgeführten Testfälle und Informationen aus Stack Overflow, zeigt sich, dass dieses Problem

[276] Screenshot aus Jenkins-Weboberfläche, Projekt *selenium*, Testergebnis vom Package *ideensammlung*
[277] [vgl. Jan]
[278] Ermitteltes Testergebnis aus Jenkins, TestNG

durch Werbe-Pop-ups verursacht wird, mit der Browser-Kompatibilität zusammenhängt oder ein Problem in Verbindung mit der WUI (Web User Interface, zu Dt. Webschnittstelle) auftreten kann, also „[...] *ein Bug in einer bestimmtem Betriebssystem/Browserversions-Kombination.*“[Rup17] erfasst wird.
Ein Pop-up ist eine GUI (engl. graphical user interface), welches typischerweise in Form von einem kleinen Browserfenster automatisch und abrupt im Vorderund auftaucht.[279] Ein Pop-up wird beispielsweise durch ein auf JavaScript basiertes *window*-Objekt mit der zugreifbaren Methode *open()* erzeugt.[280]
Ohne, dass die Navigation eines Users abbricht, besteht die Möglichkeit dieses „[...] *Fenster durch Anklicken wieder zu schließen.*“[Sch17b, S.299]
Derartige fehlerhafte Tests wurden hinterher lokal verifiziert und bestätigt.

7.3 Test Results Analyzer

Durch den Test Results Analyzer lassen sich die durch Jenkins ausgeführten Testklassen visualisieren. Die Darstellung der Grafiken können personalisiert und als Datei in CSV-Format heruntergeladen werden.
Im Folgenden werden eine Reihe von abgeschlossenen Build-Prozessen in unterschiedlichen Darstellungsarten abgebildet.

[279] [vgl. Rou05]
[280] [vgl. Aps]

New Failures	Chart	See children	Build Number ⇒ Package-Class-Testmethod names ⇓	292	291	290	289	288	287	286	285	284	283
	☐	⊖	ideensammlung	PASSED	PASSED	PASSED	FAILED	PASSED	PASSED	PASSED	FAILED	N/A	PASSED
	☐	⊖	DesktopBranchenbuch	PASSED	PASSED	PASSED	FAILED	PASSED	PASSED	PASSED	FAILED	N/A	N/A
	☐	⊖	DesktopHVV	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	N/A	PASSED
	☐		checkAlert	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	N/A	PASSED
	☐		checkHVV	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	N/A	PASSED
	☐	⊖	DesktopPortalsuche	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	N/A	PASSED
	☐		checkPortalsucheSenat	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	N/A	PASSED
	☐	⊖	DesktopPortalsucheAR	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	N/A	PASSED
	☐		checkPortalsucheArbeitsrecht	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	N/A	PASSED
	☐	⊖	DesktopPortalsucheGoogleADs	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	N/A	PASSED
	☐		checkGoogleADs	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	N/A	PASSED
	☐	⊖	DesktopStadtplan	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	N/A	PASSED
	☐		checkStadtplan	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	N/A	PASSED
	☐	⊖	DesktopVeranstaltungKategorie	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	N/A	PASSED
	☐		checkCatagorizedEvents	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	N/A	PASSED
	☐	⊖	DesktopVeranstaltungskalender	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	N/A	PASSED
	☐		checkCurrentEvents	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	N/A	PASSED
	☐	⊖	DesktopVeranstaltungssuche	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	N/A	PASSED
	☐		checkCurrentEvents	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	N/A	PASSED

[281]

Abbildung 7.5: *Alle Testergebnisse des Test-Package ideensammlung in tabellarischer Form*

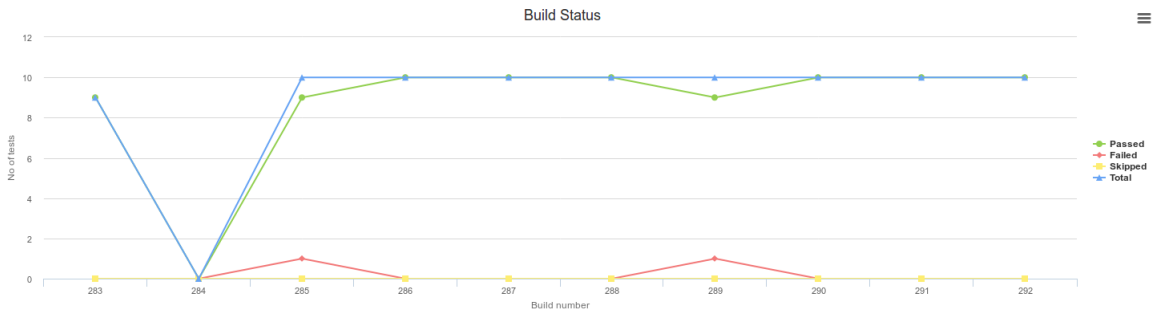
Wie in der Abbildung 7.5 verdeutlicht, werden die Ergebnisse der einzelnen Testklassen- sowie methoden in einer Tabelle aufgelistet, nach den jeweiligen abgeschlossenen Build-Prozessen (Dt. Erstellungsprozesse) sortiert und farblich je nach Erfolg (*PASSED*, grün) bzw. Fehlschlag (*FAILED*, rot) markiert.

Der Vorteil an der Tabelle ist die Verknüpfung mit den weiteren Darstellungsformen. Wenn ein einzelner Testfall präziser betrachtet werden soll, filtern die darauffolgenden Darstellungsarten die Daten der bisherigen abgeschlossenen Build-Prozessen eines oder mehrerer ausgewählten Testfälle und werden auf diese zugeschnitten (⇒ siehe Abbildung 7.9).

Um Fehlerdetails oder den Stacktrace eines Testfalls zu verfolgen, wie in Kapitel 7.2 erläutert, ist es möglich über das Betätigen eines „FAILED“-Felds, veranschaulicht auf Abbildung 7.5, einer Testklasse auf weitere Informationen bezüglich des existierenden Fehlers zuzugreifen.

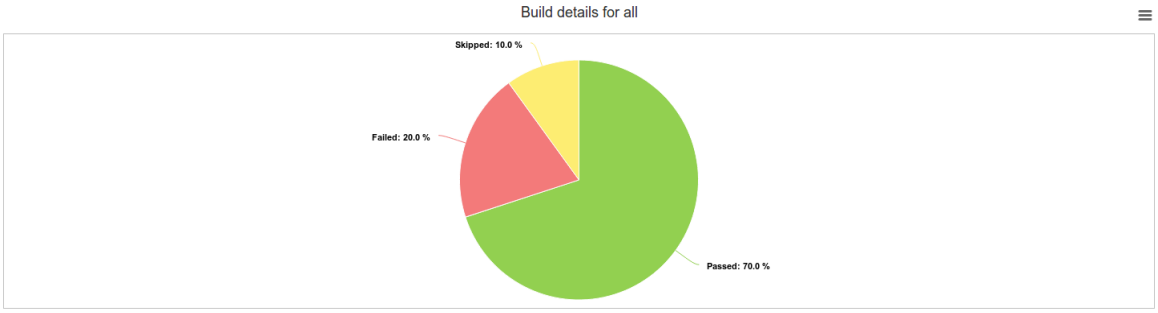
[281] Screenshot aus „Test Results Analyzer“-Plugin, Projekt selenium, Test-Package *ideensammlung*

Nachfolgend werden die **gesamten** Resultate aus den Testfällen in unterschiedlichen Darstellungsarten, wie Linien-/Kurvendiagramm, Kreisdiagramm und Balkendiagramm, illustriert.



[282]

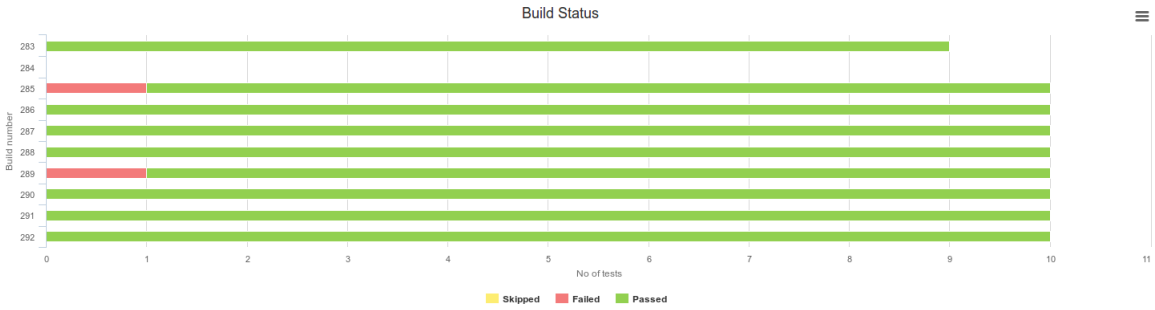
Abbildung 7.6: *Gesamtergebnisse aller Tests auf einem Liniendiagramm, Test-Package ideensammlung*



[283]

Abbildung 7.7: *Gesamtergebnisse aller Tests auf einem Kreisdiagramm, Test-Package ideensammlung*

[282] Screenshot aus „Test Results Analyzer"-Plugin, Projekt selenium, Test-Package *ideensammlung*
[283] Screenshot aus „Test Results Analyzer"-Plugin, Projekt selenium, Test-Package *ideensammlung*



284

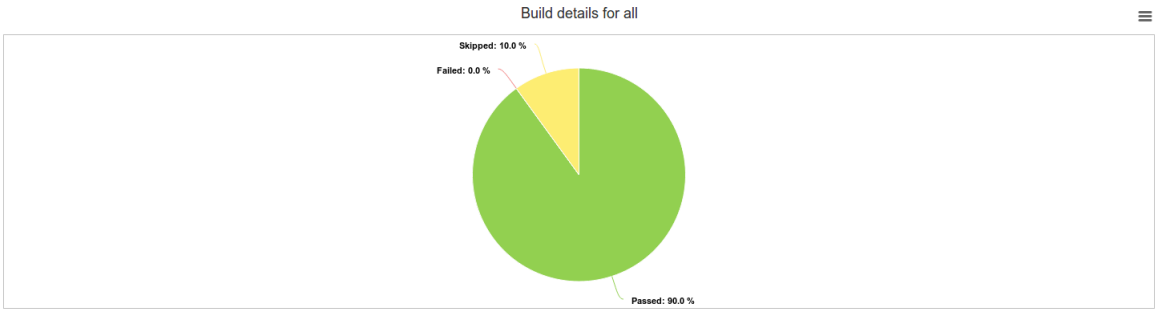
Abbildung 7.8: *Gesamtergebnisse aller Tests auf einem Balkendiagramm, Test-Package ideensammlung*

Beispielhaft wurden im Folgenden die Resultate der Testklassen *DesktopHVV* und *DesktopPortalsuche* ausgewählt und als ein Kreisdiagramm visualisiert.

☑	⊖	DesktopHVV	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	N/A	PASSED
☑		checkAlert	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	N/A	PASSED
☑		checkHVV	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	N/A	PASSED
☑	⊖	DesktopPortalsuche	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	N/A	PASSED
☑		checkPortalsucheSenat	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	N/A	PASSED

285

Abbildung 7.9: *Ausgewählte Testklassen aus den Testergebnissen anhand Tabelle*



286

Abbildung 7.10: *Ausgewählte Testklassen aus den Testergebnissen auf Kreisdiagramm*

[284] Screenshot aus „Test Results Analyzer“-Plugin, Projekt selenium, Test-Package *ideensammlung*
[285] Screenshot aus „Test Results Analyzer“-Plugin, Projekt selenium, Test-Package *ideensammlung*
[286] Screenshot aus „Test Results Analyzer“-Plugin, Projekt selenium, Test-Package *ideensammlung*

8 Fazit und Ausblick

Das letzte Kapitel schaut retroperspektiv auf die vorliegende Thesis und bewertet diese mit Hinblick auf zukünftige Schritte.
Aufgrund des begrenzten zeitlichen Rahmens konnten nicht alle theoretischen Aspekte in die Praxis umgesetzt werden, weshalb die künftigen Herangehensweisen hier diskutiert werden.

Zu Beginn dieser Thesis wurden diverse grundsätzliche Merkmale einer SOA im Zusammenhang einer Web-Architektur erklärt. Diese wurden auf qualitätssichernde Methoden der Testautomatisierung und der Datenüberwachung projiziert und anschließend durch Einbindung zahlreicher Software-Komponenten, insbesondere anhand von Testwerkzeugen des Selenium-Frameworks, realisiert.
Wichtige Standpunkte des Cloud Computings wurden im Hinblick auf das Thema Monitoring in der Theorie definiert für einen künftigen Bestandteil im Prozessverlauf.
Sowohl funktionale, als auch technische Anforderungen wurden hinsichtlich des angewandten Testvorgangs festgelegt und analysiert. Die Implementierung von praktisch umgesetzten Testprozessen wurden anhand von mehreren Diagrammen sowie expliziten Programmcodes verdeutlicht.
Abschließend wurden beispielhafte Testresultate, die aus angewandten Testfällen hervorgegangen sind, vorgestellt, visualisiert und weitere Vorgehensmaßnahmen hinsichtlich der Überprüfung von Fehlern kommentiert.

In der Thematik der Testautomatisierung wird das Selenium-Framework neue Wege in der Zukunft ermöglichen besonders durch die kontinuierlichen Aktualisierungen und der aktiven Community. Seit der Veröffentlichung von Selenium 3.0 sowie nach der Selenium Conference am 09. Oktober 2017 in Berlin wurde eine engere Zusammenarbeit mit dem W3C Consortium angekündigt. Der Fokus wird auf die WebDriver API gelegt, die standardisiert wird und von neuen Kompatibilitäten durch die Spezifikationen des W3C-Protokolls profitiert. Das W3C ist eine internationale Community, die von Tim Berners-Lee und Jeffrey Jaffe geleitet wird. Deren Ziel ist es mithilfe von Entwicklungen und Modifikationen der Übertragungsprotokolle aktiv Web Standards zu erweitern und dadurch das vollständige Potential des Webs zu nutzen. [287] Durch die Einbindung des W3C-Protokolls und Appium, ein open-

[287] [AZ17]

source Testautomatisierungs-Framework ausgelegt auf mobile Applikationen[288], können in Selenium zukünftig Tests auf mobile Webseiten und responsive Layouts einfacher automatisiert werden, da diese Art und Weise stärker ausgeprägt wird von W3C. Dieser Aspekt spielt eine zunehmend wichtige Rolle auch für die responsive Version von hamburg.de. Dazu gehören beispielsweise die verbesserte Integration von dem Safaridriver für iOS, modifizierte benutzerfreundlichere Lokatoren für Testfälle, tieferer Zugang zum Webbrowser (Shadow DOM)[289] oder die Einbindung von HTML5-Features für multimediale Elemente.[290] Auch Selenium Grid wird W3C *Capabilities* stärker bevorzugen als die in JSON Wire Protocol beeinhalteten *Capabilities*, was zudem die Plattformunabhängigkeit zwischen unterschiedlichen Systemen verstärkt.[291]

Desweiteren wird die Einbindung der Google Cloud Platform durch die Dienste des Stackdriver Monitorings in Verbindung mit BigQuery im Prozess zur Qualitätssicherung berücksichtigt.
Die notwendige Clientbibliothek kann über Gradle durch den folgenden Befehl installiert werden:

```
compile group: 'com.google.cloud', name: 'google-cloud-monitoring', version: '0.9.4-alpha'
```

[292]

Abbildung 8.1: *Installation Stackdriver Monitoring über Gradle*

Auch der auf Java basierte Quellcode bezüglich der Nutzung der Clientbibliothek sowie weitere erforderliche Dokumentationen über die API werden von der Google Cloud Plattform zur Verfügung gestellt [Picu]. Dabei handelt sich unter anderem um die verschiedenen möglichen Konfigurationen von Daten aus Stackdriver Monitoring.
Anhand von zahlreichen modifizierbaren Metriken kann die Performance der Software gemessen und dargestellt werden.
Diese Daten können über die „Stackdriver Monitoring API“[Picq] abgerufen werden. Überwachte Ressourcen werden in dem *MonitoredResourceDescriptor object* definiert. Das Objekt beschreibt das Konzept von einem MonitoredResource object, eine Ressource, die insbesondere zur Überwachung und Protokollierung verwendet wird.[293]

[288] http://appium.io/
[289] [Pro16]
[290] [Pal17, S.12]
[291] https://raw.githubusercontent.com/SeleniumHQ/selenium/master/java/CHANGELOG
[292] [Picu]
[293] https://hexdocs.pm/google_api_logging/api-reference.html

Künftig wird nach der Integration einer testautomatisierenden Umgebung, ebenso wie die Google Cloud Platform, auch die Anbindung an ein Monitoring-System genutzt. Beispielsweise können, anhand von Nachrichten RSS-Feeds, die Aktualitäten von Feed-Importer überprüft werden.

9 Anhang

```
<?xml version="1.0" encoding="UTF-8"?>
<metadata>
  <groupId>org.seleniumhq.selenium</groupId>
  <artifactId>htmlunit-driver-standalone</artifactId>
  <versioning>
    <release>2.20</release>
    <versions>
      <version>2.20</version>
    </versions>
-   <lastUpdated>20160408102537</lastUpdated>
+   <lastUpdated>20170223172754</lastUpdated>
  </versioning>
</metadata>
```

294

Abbildung 9.1: *Stackoverflow- Änderungen an htmlunit-driver-standalone/maven-metadata-local.xml*

```
<?xml version="1.0" encoding="UTF-8"?>
<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd" xmlns="http://m
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.seleniumhq.selenium</groupId>
  <artifactId>selenium-server-standalone</artifactId>
- <version>2.53.0</version>
+ <version>3.1.0</version>
  <description>POM was created from install:install-file</description>
</project>
```

295

Abbildung 9.2: *Stackoverflow- Änderungen an selenium-server-standalone-3.1.0.pom*

[294] [Mah17]
[295] [Mah17]

```
1  1  <?xml version="1.0" encoding="UTF-8"?>
2  2  <metadata>
3  3    <groupId>org.seleniumhq.selenium</groupId>
4  4    <artifactId>selenium-server-standalone</artifactId>
5  5    <versioning>
6     -    <release>2.53.0</release>
   6  +    <release>3.1.0</release>
7  7      <versions>
8     -      <version>2.53.0</version>
   8  +      <version>3.1.0</version>
9  9      </versions>
10    -    <lastUpdated>20160408102529</lastUpdated>
   10 +    <lastUpdated>20170223172753</lastUpdated>
11 11   </versioning>
12 12 </metadata>
```

296

Abbildung 9.3: *Stackoverflow- Änderungen an selenium-server-standalone/maven-metadata-local.xml*

[296] [Mah17]

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>fake</groupId>
    <artifactId>fake</artifactId>
    <packaging>pom</packaging>
    <version>${selenium.version}</version>

    <name>fake</name>

    <properties>
-       <selenium.short.version>2.53</selenium.short.version>
-       <selenium.version>${selenium.short.version}.0</selenium.version>
+       <selenium.short.version>3.1</selenium.short.version>
+       <selenium.version>3.1.0</selenium.version>
        <htmlunit.driver.version>2.20</htmlunit.driver.version>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-antrun-plugin</artifactId>
                <version>1.7</version>
                <executions>
                    <execution>
                        <phase>install</phase>
                        <goals>
                            <goal>run</goal>
                        </goals>
                        <configuration>
                            <target>
                                <get src="http://selenium-release.storage.googleapis.com/${selenium.short.version}/selenium
                                     dest="${project.build.directory}/selenium-server-standalone-${selenium.version}.jar"
                                     verbose="on"
                                     usetimestamp="true"
                                />
                                <get src="https://github.com/SeleniumHQ/htmlunit-driver/releases/download/${htmlunit.driver
                                     dest="${project.build.directory}/htmlunit-driver-standalone-${htmlunit.driver.version}
                                     verbose="on"
                                     usetimestamp="true"
                                />
                            </target>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
```

297

Abbildung 9.4: *Stackoverflow- Änderungen an dist-server-standalone/pom.xml*

[297] [Mah17]

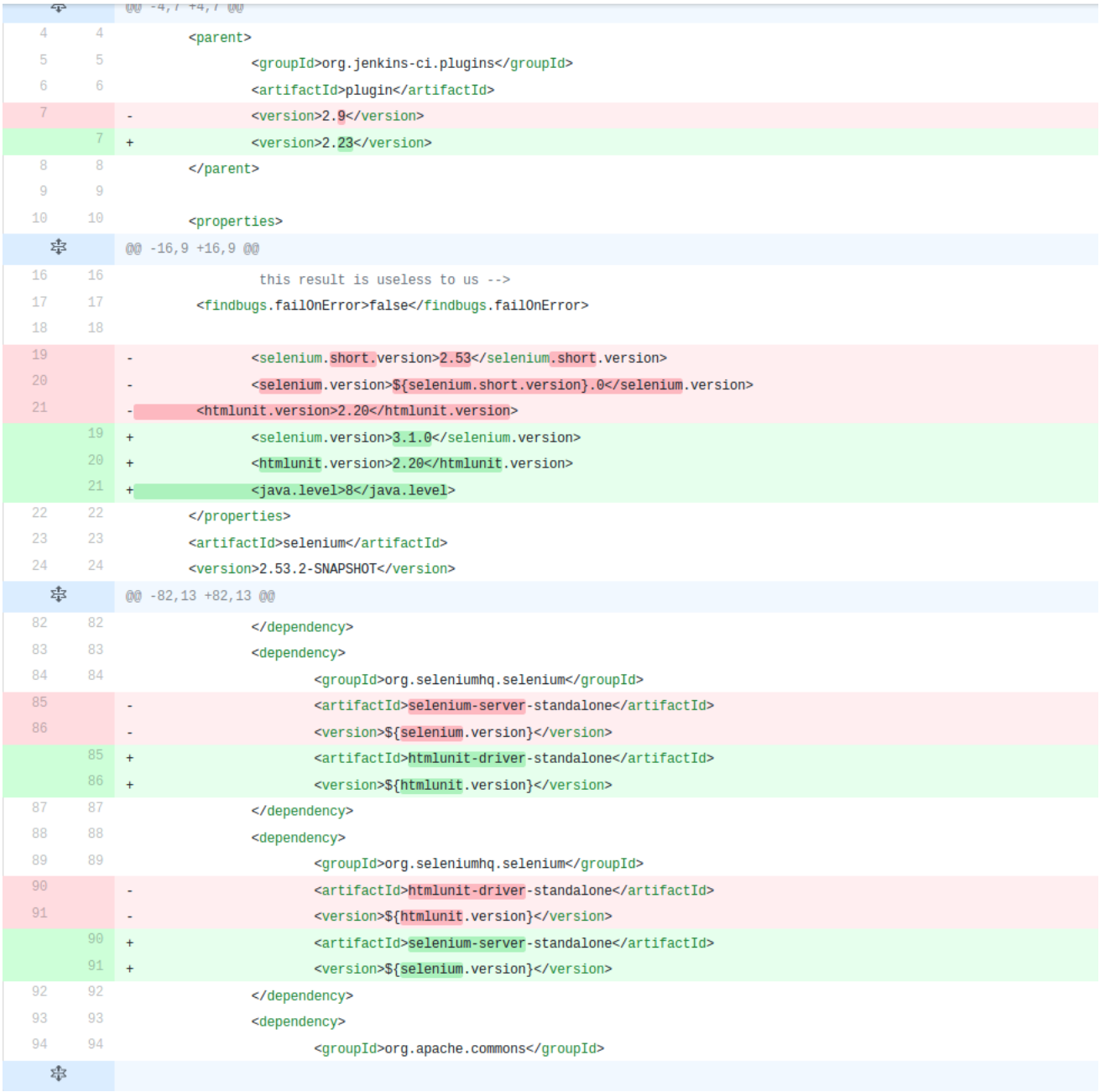


Abbildung 9.5: *Stackoverflow- Änderungen an pom.xml*

[298] [Mah17]

```
...  ...  @@ -1,9 +1,10 @@
1    1    package hudson.plugins.selenium;
2    2
     3   +import com.beust.jcommander.JCommander;
3    4    import hudson.remoting.Channel;
4    5    import jenkins.security.MasterToSlaveCallable;
5    6    import org.jfree.util.Log;
6        -import org.openqa.grid.internal.utils.GridHubConfiguration;
     7   +import org.openqa.grid.internal.utils.configuration.GridHubConfiguration;
7    8    import org.openqa.grid.web.Hub;
8    9
9    10   import java.util.logging.Level;
     @@ -38,10 +39,12 @@ public Void call() {
38   39               Logger log = Logger.getLogger(HubLauncher.class.getName());
39   40               log.log(Level.OFF, "Grid hub starting with log level " + logLevel.getName());
40   41               log.log(Level.OFF, "Grid Hub preparing to start on port " + port);
41       -           GridHubConfiguration c = GridHubConfiguration.build(args);
42       -           c.setPort(port);
43       -           c.setCapabilityMatcher(new JenkinsCapabilityMatcher(Channel.current(), c.getCapabilityMatcher()));
44       -           Hub hub = new Hub(c);
     42  +           GridHubConfiguration gridconfig = new GridHubConfiguration();
     43  +           new JCommander(gridconfig, args);
     44  +           gridconfig.port = port;
     45  +           gridconfig.capabilityMatcher = new JenkinsCapabilityMatcher(Channel.current(), gridconfig
     46  +               .capabilityMatcher);
     47  +           Hub hub = new Hub(gridconfig);
45   48               hub.start();
46   49               HubHolder.setHub(hub);
47   50
```

299

Abbildung 9.6: *Stackoverflow- Änderungen an selenium/HubLauncher.java*

[299] [Mah17]

```
...  ...  @@ -1,18 +1,19 @@
1    1    package hudson.plugins.selenium;
2    2
3    3    import jenkins.security.MasterToSlaveCallable;
4    4
5    5    class HubParamsCallable extends MasterToSlaveCallable<HubParams, Throwable> {
6    6
7    7            /**
8    8             *
9    9             */
10   10           private static final long serialVersionUID = -4136171068376050199L;
11   11
12   12           public HubParams call() throws Throwable {
13   13                   if (HubHolder.getHub() == null) {
14   14                           return new HubParams();
15   15                   }
16        -              return new HubParams(HubHolder.getHub().getHost(), HubHolder.getHub().getPort(), true);
     16   +              return new HubParams(HubHolder.getHub().getConfiguration().host, HubHolder.getHub().getConfiguration().port,
     17   +                      true);
17   18           }
18   19   }
```

300

Abbildung 9.7: *Stackoverflow- Änderungen an selenium/HubParamsCallable.java*

[300] [Mah17]

```
...  ...  @@ -1,14 +1,16 @@
1    1    package hudson.plugins.selenium;
2    2
     3   +import com.beust.jcommander.JCommander;
3    4    import hudson.plugins.selenium.callables.PropertyUtils;
4    5    import hudson.plugins.selenium.callables.SeleniumConstants;
5    6    import hudson.remoting.Channel;
6    7    import jenkins.security.MasterToSlaveCallable;
7    8
8    9    import org.openqa.grid.common.RegistrationRequest;
9    10   import org.openqa.grid.internal.utils.SelfRegisteringRemote;
     11  +import org.openqa.grid.internal.utils.configuration.GridNodeConfiguration;
10   12   import org.openqa.selenium.remote.DesiredCapabilities;
11       -import org.openqa.selenium.server.SeleniumServer;
     13  +import org.openqa.selenium.remote.server.SeleniumServer;
12   14
13   15   import java.util.logging.Level;
14   16   import java.util.logging.Logger;
     @@ -43,8 +45,10 @@ public RemoteControlLauncher(String nodeName, String[] args) {
43   45       // exception needs to be reported explicitly.
44   46       public Void call() throws Exception {
45   47           try {
46       -            RegistrationRequest c = RegistrationRequest.build(args);
47       -            for (DesiredCapabilities dc : c.getCapabilities()) {
     48  +            GridNodeConfiguration nodeconfig = new GridNodeConfiguration();
     49  +            new JCommander(nodeconfig, args);
     50  +            RegistrationRequest c = RegistrationRequest.build(nodeconfig);
     51  +            for (DesiredCapabilities dc : c.getConfiguration().capabilities) {
48   52                   JenkinsCapabilityMatcher.enhanceCapabilities(dc, nodeName);
49   53               }
50   54               SelfRegisteringRemote remote = new SelfRegisteringRemote(c);
```

301

Abbildung 9.8: *Stackoverflow- Änderungen an RemoteControlLauncher.java*

[301] [Mah17]

```
@@ -7,7 +7,6 @@

 import jenkins.security.MasterToSlaveCallable;

-import org.openqa.grid.common.RegistrationRequest;
 import org.openqa.grid.internal.utils.SelfRegisteringRemote;

 public class RemoteStopSelenium extends MasterToSlaveCallable<String, Exception> {
@@ -32,8 +31,8 @@ public String call() throws Exception {
     }

     private String getRemoteURL(SelfRegisteringRemote srr) {
-        String host = (String) srr.getConfiguration().get(RegistrationRequest.HOST);
-        Integer port = (Integer) srr.getConfiguration().get(RegistrationRequest.PORT);
+        String host = (String) srr.getConfiguration().host;
+        Integer port = (Integer) srr.getConfiguration().port;
         return "http://" + host + ":" + port;
     }
```

302

Abbildung 9.9: *Stackoverflow- Änderungen an RemoteStopSelenium.java*

[302] [Mah17]

```
@@ -13,7 +13,7 @@
 import hudson.util.ClasspathBuilder;
 import hudson.util.JVMBuilder;
 import jenkins.model.Jenkins;
-import org.openqa.grid.selenium.GridLauncher;
+import org.openqa.grid.selenium.GridLauncherV3;
 import org.openqa.selenium.htmlunit.HtmlUnitDriver;

 import java.io.*;
@@ -39,7 +39,7 @@ private SeleniumProcessUtils() {}
      * Locate the stand-alone server jar from the classpath. Only works on the master.
      */
     public static File findStandAloneServerJar() throws IOException {
-        return Which.jarFile(GridLauncher.class);
+        return Which.jarFile(GridLauncherV3.class);
     }

     /**
      * Locate the htmlunit driver jar from the classpath. Only works on the master.
      */
     public static File findHtmlUnitDriverJar() throws IOException {
         return Which.jarFile(HtmlUnitDriver.class);
     }

     /**
      * Launches Hub in a separate JVM.
      *
      */
     public static Channel createSeleniumGridVM(TaskListener listener) throws IOException {
         JVMBuilder vmb = new JVMBuilder();
         vmb.systemProperties(null);
         return Channels.newJVM("Selenium Grid", listener, vmb, new FilePath(Jenkins.getInstance().getRootDir()),
                 new ClasspathBuilder().add(findStandAloneServerJar()).add(findHtmlUnitDriverJar()));
     }

     /**
      * Launches RC in a separate JVM.
      *
```

303

Abbildung 9.10: *Stackoverflow- Änderungen an selenium/process/SeleniumProcessUtils.java*

[303] [Mah17]

Literatur- und Quellenverzeichnis

[Aga17] Meenakshi Agarwal. Selenium webdriver – download and install stable versions, 2017. techbeamers, http://www.techbeamers.com/selenium-webdriver-download-install/, Letzter Zugriff 14.11.2017 um 14:30.

[aka15] akazen. Software-tests mit vectorcast, 2015. Akazen GmbH, http://www.akazen.de/elektronik-und-softwareentwicklung/software-tests/, Letzter Zugriff am 10.10.2017 um 10:30.

[Ami05] Samir Amiry. Research lab rheinland-pfalz testen und testautomatisierung: Anforderungen an testwerkzeuge und marktstudie der fraunhofer iese. Technical report, Fraunhofer IESE, 2005.

[Aps] Matthias Apsel. Javascript/window. selfhtml, https://wiki.selfhtml.org/wiki/JavaScript/Window, Letzter Zugriff 15.11.2017 um 18:00.

[Ava14] Satya Avasarala. *Selenium WebDriver Practical Guide*. Packt Publishing, 2014.

[AZ17] Shadi Abou-Zahra. Webdriver, 2017. W3C, https://www.w3.org/TR/webdriver/, Letzter Zugriff 16.11.2017.

[Bal11] Helmut Balzert. *Lehrbuch der Softwaretechnik: Entwurf, Implementierung, Installation und Betrieb*. Spektrum Akademischer Verlag, 2011.

[Bar13] Douglas K. Barry. *Web Services, Service-Oriented Architectures, and Cloud Computing: The Savvy Manager's Guide*. Elsevier Inc., 2013.

[Bau11] Christian Baun. *Cloud Computing: Web-basierte dynamische IT-Services*. Springer-Verlag, 2011.

[Bau13] Joachim Baumann. *Gradle Ein kompakter Einstieg in momodern Build-Management*. dpunkt.verlag, 2013.

[Blo16] Official Selenium Blog. Selenium 3.0: Out now!, 2016. Official Selenium Blog, https://seleniumhq.wordpress.com/category/releases/, Letzter Zugriff 01.11.2017 um 15:00.

[Buc12] Dr. Stefan Bucher. Cloud computing als ganzheitliche geschäftsstrategie, 2012. CloudComputingInsider, https://www.cloudcomputing-insider.de/cloud-computing-als-ganzheitliche-geschaeftsstrategie-a-384309/index2.html, Letzter Zugriff 15.11.2017 um 13:30.

[Bur10] David Burns. *Selenium 1.0 Testing Tools Beginner's Guide*. Packt Publishing, 2010.

[Bur12] David Burns. *Selenium 2 Testing Tools Beginner's Guide*. Packt Publishing, 2012.

[Bü12] René Büst. Die google cloud platform, 2012. crispResarch, https://www.crisp-research.com/die-google-cloud-platform/, Letzter Zugriff 15.11.2017 um 14:00.

[Bü16] René Büst. „serverless infrastructure": Der schmale grat zwischen einfachheit und kontrollverlust, 2016. crisp Research, https://www.crisp-research.com/serverless-infrastructure-der-schmale-grat-zwischen-einfachheit-und-kontrollverlust/, Letzter Zugriff 15.11.2017 um 14:20.

[Cas14] Sven Casteleyn. *Web Engineering: 14th International Conference, ICWE 2014*. Springer, 2014.

[Cer98] Paul E. Ceruzzi. *A History of Modern Computing*. MIT Press, 1998.

[Cha14] Scott Chacon. *Pro Git*. Apress, 2014. https://git-scm.com/book/de/v1/Los-geht%E2%80%99s-Git-Grundlagen, Letzter Zugriff 14.11.2017 um 17:30.

[Clo15] Google Cloud. What is cloud vision api?, 2015. YouTube, https://www.youtube.com/watch?v=eve8DkkVdhI, Zugriff 09.11.17 um 12:30.

[Cor] Boris Cordes. Data warehouse. pmOne, https://www.pmone.com/wiki/data-warehouse/, Letzter Zugriff 15.11.2017 um 15:00.

[Die16] Prof. Henning Dierks. Software engineering, 2016.

[Doh02] Helmut Dohmann. *Die Praxis des E-Business. Technische, betriebswirtschaftliche und rechtliche Aspekte (IT-Professional)*. Vieweg Verlag, 2002.

[Dzi12] Tomasz Dziurko. Programmingjcommander - parsing command line parameters with ease, 2012. http://tomaszdziurko.com/2012/05/jcommander-parsing-command-line-parameters-ease/, Letzter Zugriff 15.11.2017 um 17:45.

[Edl07] Stefan Edlich. *Next Generation Testing mit TestNG*. entwickler.press, 2007.

[Ehm11] Florian Ehmke. Softwaretest, 2011. https://wr.informatik.uni-hamburg.de/_media/teaching/wintersemester_2010_2011/siw-1011-ehmke-tests-ausarbeitung.pdf, Letzter Zugriff 14.11.2017 um 12:15.

[Erl05] Thomas Erl. *Service-Oriented Architecture:concepts, technology, and design*. Prentice Hall, 2005.

[Erl08] Thomas Erl. *SOA - Studentenausgabe: Entwurfsprinzipien für serviceorientierte Architektur*. Addison-Wesley Verlag, 2008.

[fat17] fatamorgana. Was ist ein commit?, 2017. https://www.karteikarte.com/card/1681282/was-ist-ein-commit, Letzter Zugriff 14.11.2017 um 15:30.

[Feu12] Björn Feustel. Continuous integration in zeiten agiler programmierung, 2012. heise Developer, https://www.heise.de/developer/artikel/Continuous-Integration-in-Zeiten-agiler-Programmierung-1427092.html, Letzter Zugriff am 14.11.2017 um 18:05.

[Fin12] Andreas Fink, 2012. Enzyklopädie der Wirtschaftsinformatik Online-Lexikon, http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/is-management/Systementwicklung/Softwarearchitektur/Architekturparadigmen/Web-Architektur,Letzter Zugriff 12.11. um 15:00.

[FOK12] Fraunhofer FOKUS. Service-orientierte architekturen, 2012. YouTube, https://www.youtube.com/watch?v=JynwX6F7BCw, Letzter Zugriff 23.10.2017 um 13:45.

[Foua] The Eclipse Foundation. Desktop ides. The Eclipse Foundation, https://eclipse.org/ide/, Letzter Zugriff 14.11.2017 um 17:00.

[Foub] The Eclipse Foundation. Eclipse buildship: Eclipse plug-ins for gradle. The Eclipse Foundation, https://projects.eclipse.org/projects/tools.buildship, Letzter Zugriff 15.11.2017 um 17:00.

[Fouc] The Eclipse Foundation. Testng for eclipse. The Eclipse Foundation, https://marketplace.eclipse.org/content/testng-eclipse, Letzter Zugriff 14.11.2017 um 17:15.

[Fou16] The Eclipse Foundation. Egit - git integration for eclipse, 2016. The Eclipse Foundation, https://marketplace.eclipse.org/content/egit-git-integration-eclipse, Letzter Zugriff 14.11.2017 um 15:00.

[Fou17] The Eclipse Foundation. Google cloud tools for eclipse, 2017. The Eclipse Foundation, https://marketplace.eclipse.org/content/google-cloud-tools-eclipse, Letzter Zugriff am 14.11.2017 17:00.

[Fra07] Klaus Franz. *Handbuch zum Testen von Web-Applikationen: Testverfahren, Werkzeuge, Praxistipps*. Springer-Verlag, 2007.

[Fre] Jon Freedman. How can i write a jenkins email-ext template to display test results like the standard test report. Stack Overflow, https://stackoverflow.com/questions/11332956/how-can-i-write-a-jenkins-email-ext-template-to-display-test-results-like-the-st, Letzter Zugriff 15.11.2017 um 14:20.

[Fuc] Manuel Fuchs. Branchenbuch begriffserklärung und definition. SEO-Analyse, https://www.seo-analyse.com/seo-lexikon/b/branchenbuch/, Letzter Zugriff 15.11.2017 um 17:20.

[Gar14] Navneesh Garg. *Test Automation using Selenium WebDriver with Java*. AdactIn Group Pty Ltd., 2014.

[Gee17] JJ Geewax. *Google Cloud Platform in Action*. Manning, 2017.

[Gie12] Jan-Ole Giebel. Ressource-management für eine testautimationsfarm, 2012.

[Gno16] Harm Gnoyke. Iso, weshalb warum? ist software-qualität geschmackssache?, 2016. embarc Software Consulting GmbH, https://www.embarc.de/software-qualitaet-iso-25010/, Letzter Zugriff 10.10.2017 um 13:45.

[Gos15] Martin Gossen. Automation einer google-suchanfrage (professionelle browser-automation mit selenium webdriver, teil 1), 2015. IKS blog, https://blog.iks-gmbh.com/professionelle-browser-automation-mit-selenium-webdriver-teil-1/, Letzter Zugriff 14.11.2017 um 14:00.

[Gre16] Björn Greif. Google macht öffentliche beta seiner cloud vision api verfügbar, 2016. ZDNet, http://www.zdnet.de/88260586/google-macht-oeffentliche-beta-seiner-cloud-vision-api-verfuegbar/?inf_by=59afc412671db8bb0d8b4572, Letzter Zugriff 15.11.2017 um 15:25.

[Gri] Ilya Grigorik. Infrastructure. Google, https://peering.google.com/#/infrastructure, Letzter Zugriff 15.11.2017 um 14:10.

[Gro10] Christoph Groebke. Selenium framework von sogeti deutschland, 2010. Sogeti, https://www.sogeti.de/dienstleistungen/software-qualitatssicherung-und--test/Testprozessberatung/takt-testautomatisierung/blog-selenium-framework2/, Letzter Zugriff 15.11.2017 um 14:20.

[Gur17] Guru99. Selenium grid tutorial: Step by step guide with example, 2017. Guru99, https://www.guru99.com/introduction-to-selenium-grid.html, Letzter Zugriff 14.11.2017 um 16:30.

[hil] hilster. Testautomatisierung für eingebettete systeme. hilster, https://www.hilster.de/de/leistungen/testautomatisierung/, Letzter Zugriff 12.11.2017 um 00:00.

[Hof16] Marc Hoffmann. Wirklich alles testen: Testabdeckung messen mit eclemma, 2016. jaxenter, https://jaxenter.de/testen-eclemma-45460, Letzter Zugriff 14.11.2017 um 18:30.

[Hol] Florian Holz. Cloud platform. Cloudwuerdig, https://cloudwuerdig.com/google-cloud/cloud-platform/, Letzter Zugriff 15.11.2017 um 13:20.

[HQ08] Selenium HQ, 2008. Selenium Project. http://www.seleniumhq.org/download/, Letzter Zugriff 14.11.2017 um 14:40.

[Hum08] Bernhard Humm. *Quasar Enterprise: Anwendungslandschaften serviceorientiert gestalten*. dpunkt.verlag, 2008.

[Ihl12] Jens Ihlenfeld. Bigquery zur schnellen analyse großer datenmengen, 2012. golem.de, https://www.golem.de/news/google-bigquery-zur-schnellen-analyse-grosser-datenmengen-1205-91499.html, Letzter Zugriff 10.10.17, 13:45.

[IS] Luke Inman-Semerau. Selenium-grid. SeleniumHQ, http://www.seleniumhq.org/about/getting-involved.jsp, Letzter Zugriff 12.09.2017 um 13:45.

[Jan] Dale Janssen. Stack trace. techopedia, https://www.techopedia.com/definition/22307/stack-trace, Letzter Zugriff 15.11.2017 um 18:27.

[Jen15] Jakob Jenkov. Selenium webdriver and java 8, 2015. Codoid, http://codoid.com/selenium-webdriver-and-java-8/, Letzter Zugriff 14.11.2017 um 14:25.

[Jos08] Nicolai M. Josuttis. *SOA in der Praxis: System-Design für verteilte Geschaeftsprozesse*. dpunkt.verlag GmbH, 2008.

[Kal01] Ralf Kalmar. Adapter, 2001. Fraunhofer IESE, http://www.software-kompetenz.de/servlet/is/25749/?print=true, Letzter Zugriff 14.11.2017 um 15:10.

[Kar98] Besim Karadeniz. Das konzept uniform resource identifiers, 1998. http://www.netplanet.org/adressierung/uri.shtml,Letzter Zugriff 12.11.2017 um 16:30.

[Kar12] Helmut Karger. Same-origin-policy, 2012. JSONP.eu, http://jsonp.eu/sop.html, Letzter Zugriff 14.11.2017 um 14:30.

[Kaw17] Kohsuke Kawaguchi. Selenium plugin, 2017. Atlassian, https://wiki.jenkins.io/display/JENKINS/Selenium+Plugin, Letzter Zugriff 14.11.2017 um 13:20.

[Kir07] Karl Kirst. Standalone-server, 2007. ZUM-Wiki, https://wiki.zum.de/wiki/Standalone-Server, Letzter Zugriff 14.11.2017 um 14:30.

[Kir12] Christian Kirsch. Test-schnittstelle für browser soll standard werden, 2012. heise Developer, https://www.heise.de/developer/meldung/Test-Schnittstelle-fuer-Browser-soll-Standard-werden-1636635.html, Letzter Zugriff 14.11.2017 um 14:15.

[Kle09a] Prof. Dr. Stephan Kleuker. Werkzeuge für die qualitätssicherung, 2009. Hochschule Osnabrück, http://home.edvsz.fh-osnabrueck.de/skleuker/CSI/Werkzeuge/kombiQuWerkzeuge.html,http://home.edvsz.fh-osnabrueck.de/skleuker/CSI/Werkzeuge/Selenium/Selenium_RC.html Letzter Zugriff 14.11.2017 um 13:50.

[Kle09b] Stephan Kleuker. *Grundkurs Software-Engineering mit UML: Der pragmatische Weg zu erfolgreichen Softwareprojekten*. Vieweg + Teubner, 2009.

[Kü13] Torsten Kühn. Eine einführung in die continuous integration mit jenkins, 2013. entwickler.de, https://entwickler.de/online/eine-einfuehrung-in-die-continuous-integration-mit-jenkins-158667.html, Letzter Zugriff 14.11.2017 um 14:55.

[Lab12] Stine Labes. *Grundlagen des Cloud Computing - Konzept und Bewertung von Cloud Computing*. Universitätsverlag der TU Berlin, 2012.

[Lan] Prof. Dr. Hans Werner Lang. Jar-datei erzeugen. FH Flensburg, http://www.inf.fh-flensburg.de/lang/eclipse/jardatei.htm, Letzter Zugriff 15.11.2017 um 17:55.

[Lar15] Frederic Lardinois. Google launches cloud datalab, an interactive tool for exploring and visualizing data, 2015. TechCrunch, https://techcrunch.com/2015/10/13/google-launches-cloud-datalab-an-interactive-tool-for-exploring-and-visualizing-data/, Letzter Zugriff 15.11.2017 um 15:20.

[Lev17] Viktor Levytskyi. document.getelementbyid(), 2017. MDN web docs, https://developer.mozilla.org/en-US/docs/Web/API/Document/getElementById, Letzter Zugriff um 17:25.

[Ley93] Professor Dr. Frank Leymann. *Repository: Eine Einführung*. Oldenbourg Wissenschaftsverlag, 1993.

[Lig09] Peter Liggesmeyer. *Software-Qualität: Testen, Analysieren und Verifizieren von Software*. Spektrum Akademischer Verlag, 2009.

[Lip] Dipl.-Ing. Klaus Lipinski. Anwendungsserver. ITWissen.info, http://www.itwissen.info/Anwendungsserver-application-server.html, Letzter Zugriff 12.11.2017 um 17:45.

[Lip13a] Dipl.-Ing. Klaus Lipinski. Json (javascript object notation), 2013. ITWissen.info, http://www.itwissen.info/JSON-JavaScript-object-notation.html, Letzter Zugriff 11.11.2017 um 15:45.

[Lip13b] Dipl.-Ing Klaus Lipinski. Refactoring, 2013. ITWissen.info, http://www.itwissen.info/Refactoring.html, Letzter Zugriff 11.11.2017 um 15:45.

[Lip16] Dipl.-Ing. Klaus Lipinski. Rest (representational state transfer), 2016. ITWissen.info, http://www.itwissen.info/JSON-JavaScript-object-notation.html, Letzter Zugriff 15.10.17 um 15:25.

[Lud] Carsten Ludowig. hamburg.de, http://www.hamburg.de/impressum, Letzter Zugriff 15.11.2017 um 12:00.

[Mah17] Krishnan Mahadevan. Support selenium 3.0.1, github, https://github.com/jenkinsci/selenium-plugin/pull/101. 2017.

[Mal12] Anurag Mala. Selenium grid: Maxsessions vs maxinstances, 2012. Stack Overflow, https://stackoverflow.com/questions/13723349/selenium-grid-maxsessions-vs-maxinstances, Letzter Zugriff 15.11.2017 um 14:20.

[McP10] Stephanie Sammartino McPherson. *Tim Berners-Lee: Inventor of the World Wide Web*. USA Today, 2010.

[Mei04] Christoph Meinel. *WWW: Kommunikation, Internetworking, Web-Technologien*. Springer-Verlag, 2004.

[Mei14] Claudia Meindl. Jenkins, 2014. AlphaNodes, https://alphanodes.com/de/jenkins, Letzter Zugriff am 11.11.2017 um 19:00.

[Men13] Varun Menon. *TestNg Beginner's Guide*. Packt Publishing, 2013.

[Mey17] Andreas Meyer. *Signavio Process Manager*, 2017.

[Mic11] MicrosoftAT. Cloud computing einfach und schnell erklärt., 2011. YouTube, 01.04.2011 https://www.youtube.com/watch?v=_qGXKFJz71U, Letzter Zugriff, 08.10.2017 um 11:35.

[Min17] Sebastian Minnich. Die vorteile und nachteile des cloud-computing, 2017. heise Online, https://www.heise.de/download/blog/Die-Vorteile-und-Nachteile-des-Cloud-Computing-3713041, Letzter Zugriff 15.11.2017 um 16:00.

[Moh16] Dominik Mohilo. 8 build-tools im vergleich: Ant – buildr – maven – bazel – buck – gradle – pants – sbt, 2016. jaxenter, https://jaxenter.de/8-build-tools-im-vergleich-ant-buildr-maven-bazel-buck-gradle-pants-sbt-41627, Letzter Zugriff 15.11.2017 um 13:00.

[Mon16] Andreas Monschau. Selenium 3: Wie viel sich wirklich geändert hat, 2016. jaxenter, https://jaxenter.de/selenium-3-ist-da-48527, Letzter Zugriff 01.11.2017 um 14:30.

[Mü03] Burkhard Müller. Java: Mehr sicherheit durch das sandbox-prinzip, 2003. PC Welt, https://www.pcwelt.de/ratgeber/Java-Mehr-Sicherheit-durch-das-Sandbox-Prinzip-383671.html, Letzter Zugriff 14.11.2017 um 14:50.

[Neu] Dr. H. Neuendorf. Webappserver.

[Neu14] Alexander Neumann. Big data mit cloud dataflow: Google hat sich von mapreduce verabschiedet, 2014. heise Developer, https://www.heise.de/developer/meldung/Big-Data-mit-Cloud-Dataflow-Google-hat-sich-von-MapReduce-verabschiedet-2238697.html, Letzter Zugriff 08.11.2017 um 17:25.

[Pal17] Narayanan Palani. *Software Automation Testing Secrets Revealed: Revised Edition - Part 1*. Educreation, 2017.

[Pica] Sundar Pichai. Archival cloud storage: Nearline & coldline. Google, https://cloud.google.com/storage/archival/?hl=de, Letzter Zugriff 15.11.2017 um 14:20.

[Picb] Sundar Pichai. Big-data-lösungen. Google, https://cloud.google.com/solutions/big-data/?hl=de, Letzter Zugriff 15.11.2017 um 14:20.

[Picc] Sundar Pichai. Bigquery. Google, https://cloud.google.com/bigquery/?hl=de, Letzter Zugriff 15.11.2017 um 14:20.

[Picd] Sundar Pichai. Cloud dataproc. Google, https://cloud.google.com/dataproc/?hl=de, Letzter Zugriff 15.11.2017 um 14:20.

[Pice] Sundar Pichai. Cloud machine learning-dienste. Google, https://cloud.google.com/products/machine-learning/?hl=de, Letzter Zugriff 15.11.2017 um 14:20.

[Picf] Sundar Pichai. Cloud machine learning engine. Google, https://cloud.google.com/ml-engine/?hl=de, Letzter Zugriff 15.11.2017 um 14:20.

[Picg] Sundar Pichai. Cloud speech api. Google, https://cloud.google.com/speech/?hl=de, Letzter Zugriff 15.11.2017 um 14:20.

[Pich] Sundar Pichai. Cloud translation api. Google, https://cloud.google.com/translate/?hl=de, Letzter Zugriff 15.11.2017 um 14:20.

[Pici] Sundar Pichai. Cloudstandorte. Google, https://cloud.google.com/about/locations/?hl=de, Letzter Zugriff 15.11.2017 um 14:20.

[Picj] Sundar Pichai. Dokumentation zu google cloud pub/sub. Google, https://cloud.google.com/pubsub/docs/?hl=de, Letzter Zugriff 15.11.2017 um 14:20.

[Pick] Sundar Pichai. Google cloud app engine. Google, https://cloud.google.com/appengine/?hl=de, Letzter Zugriff 15.11.2017 um 14:20.

[Picl] Sundar Pichai. Google cloud load balancing. Google, https://cloud.google.com/load-balancing/?hl=de, Letzter Zugriff 15.11.2017 um 14:20.

[Picm] Sundar Pichai. Google stackdriver. Google, https://cloud.google.com/stackdriver/?hl=de, Letzter Zugriff 15.11.2017 um 14:20.

[Picn] Sundar Pichai. Innovationen fÜr rechenzentren. Google, https://cloud.google.com/about/data-centers/?hl=de, Letzter Zugriff 15.11.2017 um 14:20.

[Pico] Sundar Pichai. Keine server – nur code. Google, https://cloud.google.com/why-google/serverless/?hl=de, Letzter Zugriff 15.11.2017 um 14:20.

[Picp] Sundar Pichai. Kubernetes engine. Google, https://cloud.google.com/kubernetes-engine/, Letzter Zugriff 15.11.2017 um 14:20.

[Picq] Sundar Pichai. Monitored resource types. Google, https://cloud.google.com/monitoring/api/resources, Letzter Zugriff 16.11.2017 um 16:20.

[Picr] Sundar Pichai. Sicherheit in der google cloud platform. Google, https://cloud.google.com/security/?hl=de, Letzter Zugriff 15.11.2017 um 14:20.

[Pics] Sundar Pichai. Speicheroption auswählen. Google, https://cloud.google.com/storage-options/?hl=de, Letzter Zugriff 15.11.2017 um 14:20.

[Pict] Sundar Pichai. Stackdriver monitoring. Google, https://cloud.google.com/monitoring/?hl=de, Letzter Zugriff 15.11.2017 um 14:20.

[Picu] Sundar Pichai. Stackdriver monitoring-clientbibliotheken. Google, https://cloud.google.com/monitoring/docs/reference/libraries#client-libraries-install-java, Letzter Zugriff 16.11.2017 um 16:20.

[Pic17a] Sundar Pichai. Google cloud platform für rechenzentrumsexperten: Verwaltung, 2017. Google, https://cloud.google.com/docs/compare/data-centers/management?hl=de#monitoring, Letzter Zugriff 15.11.2017 um 14:20.

[Pic17b] Ralf Pichler. Testautomatisierung mit der selenium tool-familie, 2017. swissQ, https://swissq.it/de/testing/testautomatisierung-mit-selenium/, Letzter Zugriff 15.11.2017.

[Pla09] Jobst Planz. *WebServices und serviceorientierte Architektur*. GRIN Verlag GmbH, 2009.

[Pre15] Robert Preis. Automatisiertes testen: So funktioniert selenium grid, 2015. Schwarzer.de Blog , https://www.schwarzer.de/blog/automatisiertes-testen-funktioniert-selenium-grid/#prettyPhoto, Letzter Zugriff 14.11.2017 um 14:00.

[Pro16] Mitch Pronschinske. Selenium 3.0, 4.0, and 5.0 roadmap finally unveiled, 2016. TechBeacon, https://techbeacon.com/selenium-30-40-50-roadmap-finally-unveiled, Letzter Zugriff 16.11.2017 um 18:30.

[Reb16] Thomas Rebbe. Cross-browser-testing – browserübergreifende website-optimierung, 2016. 1&1, https://hosting.1und1.de/digitalguide/hosting/hosting-technik/cross-browser-testing/, Letzter Zugriff 15.11.2017 um 17:30.

[Red17] Bernd Reder. Google cloud platform soll aws paroli bieten, 2017. Computerwoche, https://www.computerwoche.de/a/google-cloud-platform-soll-aws-paroli-bieten,3229201,2, Letzter Zugriff 15.11.2017 um 14:20.

[Rit05] Wolfram Rittmeyer. Verzeichnisstruktur von java-webanwendungen, 2005. http://www.jsptutorial.org/content/directoryStructure, Letzter Zugriff 15.11.2017 um 17:55.

[Rou05] Margaret Rouse. pop-up, 2005. TechTarget, www.searchdatacenter.de/definition/pop-up, Letzter Zugriff 15.11.2017 um 14:05.

[Rou07] Margaret Rouse. peering, 2007. TechTarget, http://searchtelecom.techtarget.com/definition/peering, Letzter Zugriff 15.11.2017 um 14:05.

[Rou14] Margaret Rouse. Secure shell (ssh), 2014. SearchSecurity.de, http://www.searchsecurity.de/definition/Secure-Shell-SSH, Letzter Zugriff 12.11.2017 um 18:40.

[Rou16a] Margaret Rouse. Backbone, 2016. TechTarget, http://www.searchnetworking.de/definition/Backbone, Letzter Zugriff 15.11.2017 um 14:00.

[Rou16b] Margaret Rouse. Google cloud platform, 2016. TechTarget, http://www.searchenterprisesoftware.de/definition/Google-Cloud-Platform, Letzter Zugriff 15.11.2017 um 13:15.

[Rou16c] Margaret Rouse. Google cloud platform, 2016. TechTarget, http://www.searchenterprisesoftware.de/definition/Google-Cloud-Platform, Zugriff 15.11.2017 um 14:25.

[Rou16d] Margaret Rouse. Google kubernetes, 2016. TechTarget, www.searchdatacenter.de/definition/Google-Kubernetes, Letzter Zugriff 15.11.2017 um 14:05.

[Rou16e] Margaret Rouse. Google stackdriver, 2016. TechTarget, www.searchdatacenter.de/definition/Google-Stackdriver, Letzter Zugriff 15.11.2017 um 14:05.

[Rup17] Sven Ruppert. Parallele ui-tests mit selenium grid und docker, 2017. jaxenter, https://jaxenter.de/vaadin-selenium-grid-docker-60205, Letzter Zugriff 15.11.2017 um 18:00.

[Sch] Sebastian Schwarz. Automatisierters testen von web apps mit selenium. Webmaster.pro, http://www.webmasterpro.de/coding/article/werkzeuge-automatisierters-testen-von-web-apps-mit-selenium.html, Letzter Zugriff 14.11.2017 um 13:40.

[Sch11a] Dr.-Ing. Ina Schaefer. Anforderungsanalyse, 2011. Technische Universität Braunschweig, https://www.tu-braunschweig.de/Medien-DB/isf/sse/v2-re.pdf, Letzter Zugriff 15.11.2017 um 14:20.

[Sch11b] Stephan Schmidt. Continuous integration mit jenkins, 2011. https://de.slideshare.net/schst/continuous-integration-mit-jenkins, Letzter Zugriff 14.11.2017 um 18:15.

[Sch13] Gregor Schraegle. Aufbau eines frameworks zur testautomatisierung von webfrontends im e-commerce-bereich, 2013. jaxenter, https://jaxenter.de/aufbau-eines-frameworks-zur-testautomatisierung-von-web-frontends-im-e-commerce-bereich-2962, Letzter Zugriff 14.11.2017 um 13:45.

[Sch15] Maud Schlich. Der fundamentale testprozess, 2015. YouTube, https://www.youtube.com/watch?v=VTFsXcd72SE, Letzter Zugriff 23.10,2017 um 13:45.

[Sch17a] Robert Scholte. Introduction to the pom, 2017. Apache Maven Project, https://maven.apache.org/guides/introduction/introduction-to-the-pom.html, Letzter Zugriff 11.11.2017 um 18:00.

[Sch17b] Gertraud Schrattenecker. *Werbung: Eine Einführung*. UVK Verlagsgesellschaft mbH, 2017.

[Sei10] Daniel Seidl. *Cloud-Computing: Vom Hype zur Realität?* GRIN Verlag, 2010.

[Sei15] Richard Seidl. *Basiswissen Testautomatisierung: Konzepte, Methoden und Techniken*. dpunkt.verlag GmbH, 2015.

[Sha17a] Lakshay Sharma. Selenium grid, 2017. ToolsQA, http://toolsqa.com/selenium-webdriver/selenium-grid/, Letzter Zugriff am 23.10.2017 um 15:40.

[Sha17b] Lakshay Sharma. Selenium grid – how to easily setup a hub and node, 2017. ToolsQA, http://toolsqa.com/selenium-webdriver/selenium-grid-how-to-easily-setup-a-hub-and-node/, Letzter Zugriff 15.11.2017 um 17:15.

[shu17] shuhaibs. Amazon web services (aws) vs. google cloud platform (gcp), 2017. teamwave, https://blog.teamwave.com/2017/08/15/amazon-web-services-aws-vs-google-cloud-platform-gcp/, Letzter Zugriff 15.11.2017 um 14:20.

[Sim16] Simplilearn. Introduction to google cloud platform fundamentals certification, 2016. YouTube, https://www.youtube.com/watch?v=VUWB807TJEg&t=537s, Letzter Zugriff 15.11.2017 um 14:00.

[Sro17] Dirk Srocke. Was ist eine rest api?, 2017. CloudComputing Insider, https://www.cloudcomputing-insider.de/was-ist-eine-rest-api-a-611116/, Letzter Zugriff 15.09.2017.

[Stea] Simon Stewart. http://seleniumhq.github.io/selenium/docs/api/java/index.html, Letzter Zugriff 15.11.2017 um 14:20.

[Steb] Simon Stewart. Selenium webdriver. SeleniumHQ, http://www.seleniumhq.org/projects/webdriver/, Letzter Zugriff 14.11.2017 um 14:00.

[Ste15] Simon Stewart. Grid2, 2015. gitHub, https://github.com/SeleniumHQ/selenium/wiki/Grid2, Letzter Zugriff 15.11.2017 um 14:20.

[Tie14] Ernst Tiemeyer. *Handbuch IT-Projektmanagement: Vorgehensmodelle, Managementinstrumente, Good Practices*. Carl Hanser Verlag, 2014.

[Tim] The Economic Times. Definition of 'selenium web driver'. The Economic Times, http://economictimes.indiatimes.com/definition/selenium-web-driver, Letzter Zugriff 14.11.2017 um 14:20.

[tin12] tinatigertech. Automatisiertes testen von webseiten mit dem selenium webdriver, 2012. TIGER TECH TALK, http://www.tigertech.de/automatisiertes-testen-von-webseiten-mit-dem-selenium-webdriver/, Letzter Zugriff 14.11.2017 um 14:45.

[Vid15] MuleSoft Videos. What is an api?, 2015. Youtube, https://www.youtube.com/watch?v=s7wmiS2mSXY, Letzter Zugriff 13.10.2017 um 13:30.

[Vir16] Thilina Viraj. Quality assurance in agile software development, 2016. http://blingtechs.blogspot.de/2016/01/quality-assurance-in-agile-software.html, Letzter Zugriff 12.11.2017 um 15:30.

[vL16] David van Laatum. Email-ext plugin, 2016. Jenkins, https://wiki.jenkins.io/display/JENKINS/Email-ext+plugin, Letzter Zugriff 15.11.2017 um 17:00.

[Wie08] Hans W. Wieczorrek. *Management von IT-Projekten: Von der Planung zur Realisierung*. Springer-Verlag, 2008.

[Wit16] Frank Witte. *Testmanagement und Softwaretest Theoretische Grundlagen und praktische Umsetzung*. Springer Vieweg, 2016.

[Wol17] Stefan Wolf. Gradle plugin, 2017. Atlassian, https://wiki.jenkins.io/display/JENKINS/Gradle+Plugin, Letzter Zugriff 14.11.2017.

[Wü17] Marcel Wüthrich. Google erweitert stackdriver-cloud-monitoring, 2017. Swiss IT Magazine, http://www.itmagazine.ch/Artikel/65455/Google_erweitert_Stackdriver-Cloud-Monitoring_.html, Letzter Zugriff 15.11.2017 um 15:30.

[yA] youEngineering AG. Einführung in git. youEngineering, Liestal, https://www.youengineering.com/blog/startergit/, Letzter Zugriff 14.11.2017 um 17:30.

10 Abbildungsverzeichnis

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „– bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] – ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Dieses Blatt, mit der folgenden Erklärung, ist nach Fertigstellung der Abschlussarbeit durch den Studierenden auszufüllen und jeweils mit Originalunterschrift als letztes Blatt in das Prüfungsexemplar der Abschlussarbeit einzubinden.
Eine unrichtig abgegebene Erklärung kann -auch nachträglich- zur Ungültigkeit des Studienabschlusses führen.

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: _____

Vorname: _____

dass ich die vorliegende – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

- die folgende Aussage ist bei Gruppenarbeiten auszufüllen und entfällt bei Einzelarbeiten -

Die Kennzeichnung der von mir erstellten und verantworteten Teile der ist erfolgt durch:

_____	_____	_____
Ort	Datum	Unterschrift im Original