Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorthesis

Kumar Gurditta

Beispielhafte Umsetzung einer Analyse von Service Desk Daten unter Verwendung von Big Data Technologien

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

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

Kumar Gurditta

Beispielhafte Umsetzung einer Analyse von Service Desk Daten unter Verwendung von Big Data Technologien

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. rer. nat. Hans-Jürgen Hotop
Zweitgutachter : Prof. Dr. rer. nat. Henning Dierks

Abgegeben am 03. März 2015

Kumar Gurditta

Thema der Bachelorthesis

Beispielhafte Umsetzung einer Analyse von Service Desk Daten unter Verwendung von Big Data Technologien

Stichworte

Big Data, Hadoop, Spark, MapReduce, Service Desk, Open Source, Analyse

Kurzzusammenfassung

Diese Arbeit umfasst ein beispielhaftes Konzept für den Service Desk. Dabei werden die Service Desk Daten analysiert, um den Service Desk Mitarbeiter im 1st Level Support die Hilfestellung anzubieten und nach dem gleichen bzw. ähnlichen Problem zu suchen. Abschließend erfolgt die Entwicklung und Umsetzung neuer Ideen.

Kumar Gurditta

Title of the paper

Exemplary Implementation of an analysis of service desk data using Big Data technologies

Keywords

Big Data, Hadoop, Spark, MapReduce, Service Desk, Open Source, Analyse

Abstract

This paper enfolds an exemplary draught for the Service Desk. The
Service Desk Datas are analysed for the Service Desk employee on the 1st Level Support to render assistance for the same or similar problem. Finally, the development and execution of new ideas occurs.

Hinweis

Drei Abbildungen sind aufgrund der Größe unter dem Verzeichnis „CD Bachelorarbeit\1.1 Bachelorarbeit Anhang“ auf der beiliegenden DVD/CD abgelegt. Die Abbildungen „Sequenzdiagramm Spelling.pdf“ und „Sequenzdiagramm WorkOnHDFSData.pdf“ wurden mit dem Programm Altova UModel 2015 erstellt, die dritte und letzte Abbildung „Projekt und Klassen UML.pdf“ wurde mit dem Programm Visual Paradigm 12.0 erstellt. Einerseits wurde für die erstellung des JAVA Source Codes Eclipse und anderseites für das Komplette System die VMware Workstation 10.0 auf dem Linux Ubuntu 12.1 istalliert ist verwendet.

Danksagung

Sehr lange habe ich auf diesen Moment hingefiebert und konnte es kaum abwarten, bis ich endlich mein Bachelorarbeit in Empfang nehmen konnte. Nun ist es geschafft und auch wenn ich die Universität sicherlich auch mit einem weinenden Auge verlassen werde, so schaue ich doch voller Vorfreude und Zuversicht auf die neuen Aufgaben, die das Berufsleben an mich stellen wird. Dass ich für diese Aufgaben so gut gewappnet bin, verdanke ich aber nicht nur meinen eigenen Leistungen, sondern auch sehr vielen anderen Menschen.

An dieser Stelle möchte ich mich bei Computercenter AG für die Ermöglichung dieser Bachelorarbeit sowie die gute Zusammenarbeit, und meinem Betreuer Herrn Dr. Peter Meitz für den sehr engagierten Einsatz bedanken. Mein Dank geht auch an die Mitarbeiter von Computacenter AG, die im Big Data Bereich tätig sind und mir durch ihr Fachwissen und ihre Anregungen sehr oft weitergeholfen haben.

Ebenfalls bedanke ich mich bei Herrn Professor Dr. rer. nat. Hans-Jürgen Hotop und an Herrn Professor Dr. rer. nat. Hennig Dierks, die mich bei meiner Bachelorarbeit unterstützt, mir mit wertvollen Hinweisen und Tipps zur Seite gestanden und als Gutachter meiner Arbeit fungiert haben.

Abschließend gebührt mein besonderer Dank meinen Freunden und meiner Familie, insbesondere meinen Eltern für ihre Geduld und Unterstützung.

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Veranlassung und Hintergrund

Big Data ist einer der letzten großen Hypes der IT in den letzten Jahren. 2010 nutzten Cloudera und Aster Data den Begriff erstmalig in ihrem Business Kontext. Noch im Jahre 2009 kam es zum umgehenden Löschen eines ersten Artikels auf Wikipedia mit der Überschrift Big DATA mit der Begründung, nur zwei Nomen in einen Kontext zu setzen, reiche noch nicht aus, um einen neuen Artikel zu rechtfertigen. 2011 folgte eine erste Gardner Studie zum gleichen Thema. Damit war Big Data aufseiten der Analysten und des Business geboren.[1]

Die rasche Geschwindigkeit, mit der sich das Themenfeld BIG DATA entwickelt hat, liegt u. a. auch in der zunehmenden Vernetzung der IT Systeme begründet. Seitdem Tim Berners-Lee für das World Wide Web im Jahr 1989 am CERN den Grundstein gelegt hat, ist es zu einem extremen Zuwachs des Datenvolumens in diesem Bereich gekommen. Im Grunde liegt die Begründung des großen Durchbruches des Internets im Jahr 1994 durch Marc Andersson und seinem Entwicklerteam, denen es gelang, einen benutzerfreundlichen Internet Browser „Netscape“ zu entwickeln und auf den Markt zu bringen. Mittlerweile hat die Zahl der Nutzer des Internets – womit heute das World Wide Web gemeint ist – explosionsartig zugenommen.[2]

Nicht nur die Informationen aus dem Bereich des Internets sind von Bedeutung, sondern auch andere Techniken, die uns im Alltag begleiten und wertvolle Rohdaten[3] enthalten. Hier ist z. B. zu nennen: *„das moderne Auto“* von heute, das ein komplexer Informationsspeicher auf vier Rädern ist. Ein weiterer Informationsspeicher ist: *„das Mobiletelefon*“ oder besser gesagt, „das *Smartphone*“, welches nicht mehr aus dem Alltag wegzudenken ist.

Inzwischen gibt es eine Vielzahl an Geschäftsmodellen von Internet-Suchmaschinen, Sozialen Netzwerken bis zu virtuellen Shops und Informationsbörsen jeglicher Art. Mit all diesen Angeboten ist das Datenvolumen des Internets gestiegen. Laut einer Studie des IDC[4] hat die Datenmenge im Internet in den letzten zwei Jahren um das Zweifache auf 2,8 Zettabyte zugenommen und steigt exponentiell an.[5]

---

[1] Vgl.: **Literatur: [11]** S. 8
[2] (Mitte 1998 zwischen 40 und 80 Millionen Benutzer weltweit – mit stark steigender Tendenz [vgl. Brockhaus 2000]).
[3] Rohdaten: Unbearbeitete und meist unformatierte Daten, z.B. ein Datenstrom, der noch nicht nach Befehlen oder Sonderzeichen gefiltert wurde; vgl. http://www.enzyklo.de/Begriff/Rohdaten, abgerufen am 02.01.2015,
[4] Anbieter von Marketinginformationen der Informationstechnologie und Telekommunikation.
[5] Vgl.: **Quelle: [22]**

Die Unternehmen nutzen Big Data Technologien, um aus diesen Daten wertvolle businessrelevante Erkenntnisse zu gewinnen. Ein Beispiel hierfür ist Google, welches ein fast vollständiges digitales Geschäftsmodell[6] hat. Dieses Geschäftsmodell basiert technologisch auf Big Data Applikationen. Neben businessrelevanten Anwendungszenarien sind selbstverständlich auch wissenschaftliche Analysen möglich. Während der H1N1-Krise in 2009 erwies sich das Google-System als hilfreicher und schnellerer Indikator als die Regierungsstatistiken mit ihren unabwendbaren Verzögerungen. Die Gesundheitsbehörde gewann mittels der Methode von Google einen wertvollen Informationsvorsprung im Gegensatz zu Regierungsstatistiken. Google sammelt dazu keine Gewebeproben ein oder wertet Berichte von Hausärzten aus. Stattdessen beruht die Methode auf einer Vielzahl von Daten, die Google über das Anfrageverhalten seiner Nutzer erhielt.

Auf Basis von Daten-Analysen unter Verwendung von Big Data Technologien ist es möglich, eine Vielzahl an Informationen auszuwerten, sodass neue Erkenntnisse, Güter oder Dienstleistungen von bedeutendem Wert zu gewinnen sind.[7]

Anwendungsbeispiele finden sich in allen Branchen, Wissenschaftsbereichen oder anderen, die ein sehr hohes Datenaufkommen haben und/oder deren Daten aufgrund ihrer Struktur nicht zu analysieren waren. Die Big Data Technologie ermöglicht hier neue Analysemöglichkeiten. Big Data basiert nicht auf einer singulären Technologie, sondern ist vielmehr das Resultat des Zusammenwirkens einer ganzen Reihe an Technologien und Innovationen auf verschiedenen Gebieten.[8] Es entstehen Tag für Tag neue Daten im Bereich des Service Desk Supports. Es kommt zu einer Speicherung einer großen Menge an Daten, dabei verbergen sich in diesen Daten wertvolle Informationen, wodurch ein Kundenproblem in kürzerer Zeit zu beheben ist. Denn für jedes behobene Problem kann ein Dienstleistungsunternehmen, das diesen Support anbietet, eine Marge verlangen. Die vorliegende Arbeit analysiert die Service Desk Daten und entwickelt einen Prototyp.

1.2 Zielsetzung

Ziel der nachfolgenden Arbeit ist es, zum einen Ideen und zum anderen einen Prototypen für die Analyse von Service Desk Daten zu entwickeln. Da konventionelle Techniken zur Persistenz und Analyse von Daten hier an ihre Grenzen stoßen, gilt es, mittels modernster

[6] Vgl.: **Literatur: [09]** , S. 273
[7] Vgl.: **Literatur: [23]** Kapitel 1
[8] Vgl.: **Quelle: [08]**

In-Memory Techniken, u. a. durch den Einsatz von Spark und grundlegenden Text Mining Algorithmen, den Big Data-Bestand von Service Desk zu untersuchen, um so einen Erkenntnisgewinn zu generieren, der die Problembehebung bei Service Desk Support erleichtert und beschleunigt. Die Big Data Technologien sind gut einsetzbar bei Unstrukturierte Daten die in großer Menge. Hinzu kommt das In-Memory-Technologie verwendet wird, bei großen Mengen an Daten werden Daten in den Hauptspeicher geladen um einen schnelleren Zugriff auf die Daten zu gewährleisten. Das Spektrum von Text Mining ist sehr groß; hier sind wenige Algorithmen Gegenstand der Betrachtung, denn sonst würde dies den Rahmen dieser Arbeit sprengen. Des Weiteren sind folgende Ziele einzuhalten. Der Mitarbeiter soll die Möglichkeit haben, über das Web-Interface eine Suche in einem Textfeld einzugeben und daraus eine Lösung vorgeschlagen zu bekommen. Falls keine Lösung vorhanden ist, aber das bestehende Problem schon einmal in der Vergangenheit aufgetreten ist, erhält der Mitarbeiter eine ID angezeigt, mit deren Hilfe er im Ticketsystem alle Informationen über das bereits behobene Problem ansehen kann. Ist das bestehende Problem noch nie aufgetreten, so erhält der Mitarbeiter keine Lösung angezeigt. Des Weiteren soll ein Administrator den Zugriff haben, Daten zu löschen, umzubenennen und einzufügen. Außerdem geht es darum, Rohdaten automatisch durch das Einfügen einer neuen Datei aufzubereiten. Ein erstelltes Testkonzept soll nicht nur die Entwicklungsabteilung nutzen und verstehen können, sondern auch die Fachabteilung, die kein Java programmieren kann. Fürs Testen ist ein Tool einzusetzen, das überprüfen soll, ob der Test eine große Testabdeckung hat und die wichtigen Klassen und Methoden getestet wurden. Im Weiteren ist eine Softwareumgebung aufzubauen, die es ermöglicht, die Software einfach weiterzuentwickeln. Dazu gelangen Tools zur Anwendung. Die Softwarekomponenten sind entkoppelt zu gestalten, das heißt, es soll für den Entwickler möglich sein, einzelne Komponenten aus der Software zu ersetzen.

1.3 Aufbau der Arbeit

Für die Bearbeitung der vorliegenden Bachelorarbeit „Beispielhafte Umsetzung einer Analyse von Service Desk Daten unter Verwendung von Big Data Technologien“ setzt als Erstes einen theoretischen Rahmen, gefolgt von einem praktischen Teil. Diese Bachelorarbeit ist insgesamt in sechs Hauptkapitel untergliedert.

Das **erste Kapitel** führt an das Thema dieser wissenschaftlichen Arbeit heran, es erfolgt eine Beleuchtung der Veranlassung und des Hintergrundes und es findet eine Darstellung der Zielsetzung statt.

Das **zweite Kapitel** enthält eine kurze, nur auf die relevanten Bereiche für den Inhalt dieser Bachelorarbeit fokussierte Darstellung des Themas Big Data Technologien und Text Mining. Zunächst liegt der Fokus auf Big Data und welche Technologien es im Big Data Bereich gibt. Anschließend stehen Text Mining und Algorithmen aus diesem Bereich im Mittelpunkt. Des Weiteren erfolgt eine Beschreibung der verwendeten Tools, die zum Einsatz kommen.

Das **dritte Kapitel** beinhaltet die Anforderungen, die Datenanalyse der Service Desk Daten sowie die Beschreibungen von Ideen für eine Suchoptimierung. Zunächst steht eine Beschreibung der Anforderungen für die Datenanalyse, der Systemarchitektur sowie der Softwarearchitektur an. Anschließend folgt eine Erläuterung der Grundlagen der Service Desk und der Ideen der Suchoptimierung. Die Suchoptimierung beleuchtet die Ideen der Stopp Wörter, der Rechtschreibkorrektur und die Ermittlung der Wichtigkeit eines Begriffes sowie die Suche nach aufbereiteten Daten.

Das **vierte Kapitel** beschreibt die für diese Arbeit angewandte Systemarchitektur.

Das **fünfte Kapitel** beinhaltet die Erklärung des Model View Controllers sowie des im System implementierten, modifizierten Model View Controllers. Es findet auch eine Beschreibung der Klassenstruktur sowie von zwei Funktionen anhand von Sequenzdiagrammen statt. Im Anschluss daran steht die Erläuterung der verwandten Tools; anschließend geht es um das Erstellen eines Testkonzepts.

Das **sechste Kapitel** erklärt das realisierte System anhand von Abbildungen für den Service Desk Mitarbeiter und den Administrator.

Im **siebten Kapitel** folgen das Fazit und der Ausblick.

2 Grundlagen

Dieses Kapitel beschreibt die Grundlagen: Für den weiteren Verlauf dieser Arbeit sind die Grundlagen ein wichtiger Bestandteil. Zunächst liegt der Schwerpunkt auf Big Data und den Technologien, anschließend auf Text Mining. Das letzte Unterkapitel geht auf das Service Desk ein; dabei sind die wichtigen Begriffe Gegenstand der Betrachtung.

2.1 „BIG DATA“ – Hadoop Framework

M. Cox und D. Ellsworth veröffentlichten 1997 einen Artikel, der die Herausforderungen von großen Datenmengen und deren Problematik für bestehende IT Systeme behandelt. Aus der

Problematik der wachsenden Datenmengen entstand der Begriff „Big Data“[9]. Die Definition von Big Data legte im Jahr 2001 das Marktforschungsunternehmen META Group vor. Die META Group veröffentlichte den folgenden Bericht „3-D Data Management: Controlling Data Volume, Velocity und Variety“, um die Problemlösungen durch das Data Warehousing und die Möglichkeiten darzustellen, diese Herausforderungen mittels relationaler Technologie zu lösen. Hieraus entwickelte sich die Definition der „3 Vs“[10].

Im Internet finden sich viele weitere Erklärungen für Big Data; dennoch legen die „3 Vs“ die Grundlage von Big Data laut dem Bericht von META Group. Der Autor, Pavlo Baron, der sein Werk „Big Data für IT-Entscheider“ im Jahre 2013 veröffentlichte, ist der Ansicht, diese Grundlagen seien nicht ausreichend für die jeweilige Problemstellung anzupassen.[11]

Im Weiteren veröffentlichte BitKom 2014 einen Leitfaden „Big-Data-Technologien Wissen für Entscheider“, deren Autoren – Jorg Bartel (IBM Deutschland GmbH), Georg Urban (Microsoft Deutschland GmbH) et al. – die Analyse als eine weitere Facette von Big Data sehen.[12]

Die „3 Vs“ und die Analyse bei Big Data bestehen aus:

Volume: Anzahl von Datensätzen und Files
Velocity: Datengenerierung in hoher Geschwindigkeit
Variety: Datenvielfalt durch Bilder, Videos, Maschinendaten u. v. m.
Analyse: Erkennen von Zusammenhängen, Bedeutungen, Muster

In Abbildung 1 sind weitere Merkmale von Big Data zu erkennen. Die wesentlichen Merkmale der „3 Vs“ in der Abbildung sind Datenmenge, Datenvielfalt, Geschwindigkeit und Analytics. Diese vier verschiedenen Merkmale finden nachfolgend Erläuterung.

---

[9] Vgl.: **Quelle: [12]**
[10] Vgl.: **Literatur: [07]** S. 11.
[11] Vgl.: **Literatur: [04]** S. 23.
[12] Vgl.: **Quelle: [08]** *S. 12.*

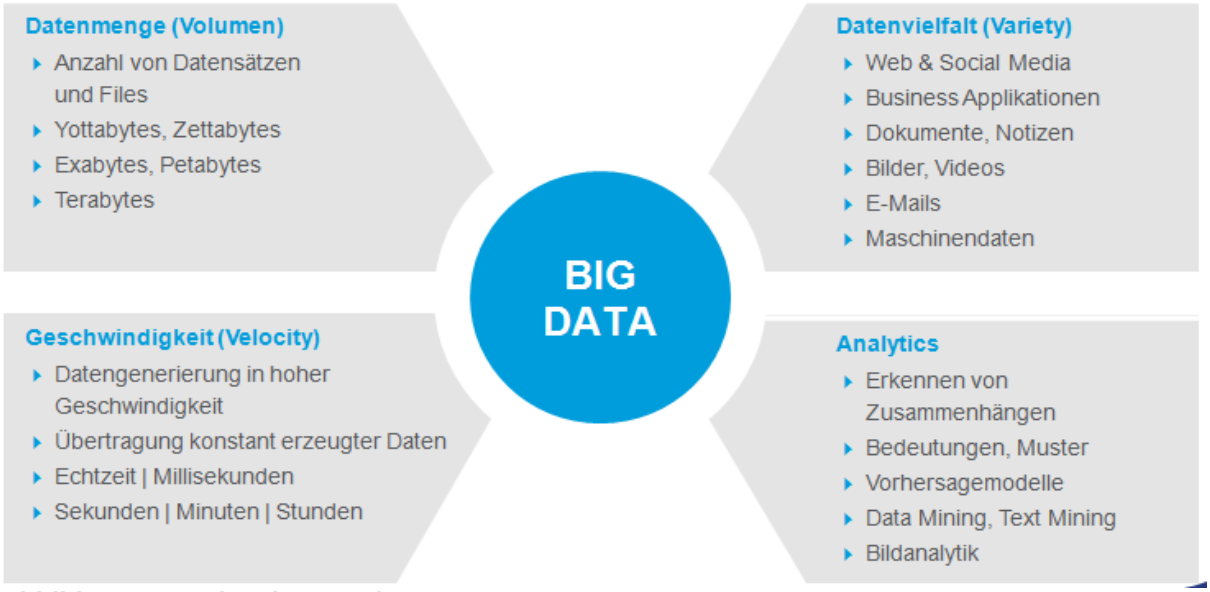

Abbildung 1: Merkmale von Big Data

Volume (Datenmenge)

Viele Unternehmen verfügen über Daten im Terabyte Bereich, die Informationen beinhalten, auf die das Unternehmen nicht in kürzester Zeit zugreifen kann. Diese Menge und die schnelle Verarbeitung der Daten ist der offensichtlichste positive Aspekt von Big Data. Vorherrschend ist nicht die Definition der großen Datenmenge, sondern das Erkennen im Hinblick auf die nicht vorhandene Speicherkapazität für große Datenmengen.[13]

Velocity (Geschwindigkeit)

Aufgrund der Notwendigkeit von schnellen Entscheidungen durch Führungskräfte gilt es, große Datenmengen immer schneller auszuwerten; dabei hat die Verarbeitungsgeschwindigkeit mit dem schnellen Datenwachstum Schritt zu halten. Folgende Herausforderungen sind damit verbunden: Analysen großer Datenmengen mit Antworten im Sekundenbereich, Datenverarbeitung in Echtzeit, Datengenerierung und Übertragung in hoher Geschwindigkeit.[14]

Variety (Vielfalt von Datenquellen und Datenformaten)

Den Unternehmen liegen immer mehr Quellen von Daten unterschiedlichster Art vor wie z. B. E-Mails, Internetquellen, unternehmenseigene Daten und Soziale Netzwer-

[13] Vgl.: **Quelle: [08]** *S. 12.* & Vgl.: **Literatur: [04]**
[14] Vgl.: **Quelle: [08]** *S. 12.* & Vgl.: **Literatur: [04]**

ke mit der Zuordnung in unstrukturierte, semistrukturierte und strukturierte Daten.[15] Die klassischen transaktionalen IT-Syteme, z. B. ERP, sind in der Regel nur in der Lage, strukturierte Daten zu verarbeiten. Big Data Technologien ermöglichen die Verarbeitung auch aller anderen Daten.

Analytics

Unter Analytics ist eine Vielzahl von Methoden zu verstehen zur möglichst automatisierten Erkennung und Nutzung von Mustern, Zusammenhängen und Bedeutungen. Zum Einsatz kommen u. a. statistische Verfahren und Text- und Bildanalytik.[16]

Mit den Jahren und der Entwicklung von Big Data haben sich weitere „Vs" zu den „3 Vs" und Analytics gebildet: **Value** (Wert), **Veracity** (Richtigkeit), **Volatility** (Flüchtigkeit).

Value (Wert)

Unternehmen archivieren oder zeichnen ihre Daten für eine spätere Verwendung auf. Da diese Daten vorliegen, können Unternehmen mittels des maschinellen Lernens und schnellerer Systeme neue Werte aus zuvor gespeicherten Daten gewinnen. Diese „neuen Werte" bedeuten das Voranbringen der Unternehmen hinsichtlich der Steigerung ihrer Einnahmen oder auch das Voranbringen eines Forschungsgebietes. Es sollte für ein informationsverarbeitendes System möglich sein, aus diesen Daten neue Werte zu erhalten und aus der Beantwortung auf Fragen neue Erkenntnisse zu gewinnen.[17]

Veracity (Richtigkeit)

Bevor aus Daten Schlüsse und Antworten zu ziehen sind, sollte dem Analysten ihre Korrektheit und Relevanz für das gestellte Problem bewusst sein.[18]

Volatility (Flüchtigkeit)

Bei Volatility handelt es sich um Daten, die nach einer gewissen Zeit aufgrund ihrer nicht erfolgten Aktualisierung nicht von Bedeutung sind. Diese Daten sind zu löschen, da sie die Analyse verfälschen und somit einen großen Schaden anrichten können.

---

[15] Vgl.: **Quelle: [08]** *S. 12.* & Vgl.: **Literatur: [04]**
[16] Vgl.: **Quelle: [08]** *S. 12.* & Vgl.: **Literatur: [04]**
[17] Vgl.: http://www.controllerverein.com/BigData.185074.html abgerufen am 12.01.2014 & Vgl.: **Literatur: [04]**
[18] Vgl.: **Literatur: [11]** S. 10 f & Vgl.: **Literatur: [04]**

Für diese Arbeit trifft dieser Aspekt zu. Es ist wichtig, die Daten in einem angemessenen Zeitrahmen zu pflegen, um ein anständiges und relevantes Ergebnis mittels der Analyse zu erhalten. Beim Entwurf des Systems fand dies Berücksichtigung.[19]

Hadoop

Bei der Betrachtung von Big Data Technologien ist in der Regel immer das Framework Hadoop beteiligt. Es bildet auch historisch gesehen die Basis der Big Data Technologien. Oftmals kommt es auch zur synonymen Anwendung der Begriffe Big Data und Hadoop Framework.

2002 arbeitete Doug Cutting im Rahmen eines Open-Source-Projekts an einer freien Suchmaschine mit dem Namen Nutch. Doug Cutting ist ein bekannter Entwickler in diesem Bereich, denn bereits im Jahr 1997 hat er die freie Suchmaschine Excite veröffentlicht. Das von Doug Cutting verfolgte Ziel war das Kopieren des kompletten World Wide Web. In 2004 konnte die Suchmaschine Nutch 100 Mio Webseiten indexieren, was aber nicht ausreichte. Doug Cutting und die Nutch-Coummnity suchten eine Möglichkeit, die Architektur noch erweiterbarer zu gestalten. Im selben Jahr hat Google eine wissenschaftliche Arbeit über MapReduce publiziert. Durch die Veröffentlichung von MapReduce und Google File System konnte die Nutch-Community ihre zugrunde liegende Architektur vorteilhafter skalieren. Im Jahr 2006 stellte Yahoo! Doug Cutting ein; er selbst und einige Mitarbeiter arbeiteten weiter an den von Google bereitgestellten Systemen und erweiterten ihre Nutch Architektur[20].

Das verteilte Dateisystem und MapReduce wurden aus Nutch extrahiert und in ein eigenes Framework überführt. Dadurch entstand Hadoop, welches Doug Cutting nach dem gelben Stoffelefanten seines Sohnes benannte. Seit 2008 ist Hadoop ein sogenanntes Top-Level-Projekt der Apache Software Foundation. Verschiedene große Firmen setzen auf Hadoop, darunter u. a. Facebook, IBM und Yahoo![21,22].

Die Anwender haben die Möglichkeit, auf Hadoop 1.x bzw. Hadoop 2.x als Basisversion von Apache Hadoop zuzugreifen. Es ist als Open Source kostenlos verfügbar, setzt jedoch beträchtliches Know-how voraus. Das betrifft insbesondere das Aufsetzen und Verwalten eines Hadoop-Server-Clusters. Es stehen etliche Tools und Frameworks bei „Apache Software

---

[19] Vgl.: **Literatur: [05] S. 45 f** & Vgl.: **Literatur: [04]**
[20] Vgl.: **Literatur: [26]** S. 9
[21] Vgl.: **Literatur: [21]**
[22] Vgl.: **Literatur: [24]**

Foundation“[23], die als Top Level Projekte eingestuft sind, zur Verfügung. Diese werden im Rahmen anderer Apache-Hadoop-Projekte entwickelt. Allerdings sind diese quasi von Hand zu implementieren und aufeinander abzustimmen.[24]

Unternehmen nutzen „Hadoop-Distributionen“, Pakete, die aus Basissoftware und den dazugehörigen Tools, die auf der Grundlage von Hadoop, dessen Eco-System, bestehen. Anbieter solcher Distributionen sind beispielsweise EMC und MapR. Im Vergleich zur Grundversion von Hadoop besteht der Vorteil dieser Anbieter darin, dass die Unternehmen ein integriertes Softwarepaket erhalten, welches sich einfacher konfigurieren lässt. Zu diesen Paketen bieten die Anbieter auch einen technischen Support für die Kunden an.[25] Die grundlegende Idee von Hadoop ist die Speicherung von großen Daten in einem „Cluster“ aus Standard-Hardware. Hadoop Distributed Files System (HDFS) und MapReduce sind die zwei Kernkomponenten von Hadoop 1.x. Der Unterschied zwischen Hadoop 2.x und Hadoop 1.x ist die Möglichkeit von Version 2, verschiedene weitere Tools sowie Frameworks ausführen zu können, wobei die Version 1 ausschließlich MapReduce unterstützt. Im Weiteren folgt das Beschreiben der Funktionsweise von Spark und es schließt sich die Begründung an, wieso es eine gute Alternative zu MapReduce ist. Spark läuft unter Hadoop 2.x. Diese Arbeit verwendet die Version Hadoop 2.x.

Architektur von Hadoop

Abbildung 2 stellt die Hadoop Architektur dar. Mit MapReduce und dem Hadoop Distributed File System (HDFS) ist es möglich, auf Hadoop sowohl die verteilte Speicher- als auch die Rechenfähigkeit zu nutzen. Im Wesentlichen besteht Hadoop aus einem Master- und mehreren Slave-Nodes, diese und die Erläuterung der Task Tracker, Job Tracker, Data Node und Name Node finden Beschreibung unter dem Punkt „HDFS und MapReduce“ im Abschnitt HDFS und MapReduce. Wie in Abbildung 2 beschrieben, sind das Framework MapReduce für „Distributed computation“[26] und HDFS für „Distributed storage“[27] zuständig.[28] Auf der Ebene der Distributed computation kommt es zur Verteilung bzw. Parallelisierung der Berechnung; zu diesem Zweck findet das Framework MapReduce Anwendung. Die darunterliegende Ebene Distributed storage dient der verteilten Speicherung der Daten, die durch das HDFS erfolgt.

---

[23] Vgl.: https://projects.apache.org/
[24] Vgl. Abbildung 5: Das Hadoop Ökosystem
[25] Vgl.: http://www.cio.de/a/hadoop-distributionen-im-kurzprofil,2963574
[26] Verteiltes Rechnen.
[27] Verteiltes Speichern von Daten.
[28] Vgl.: Abbildung 6: High Level Architektur von Hadoop.

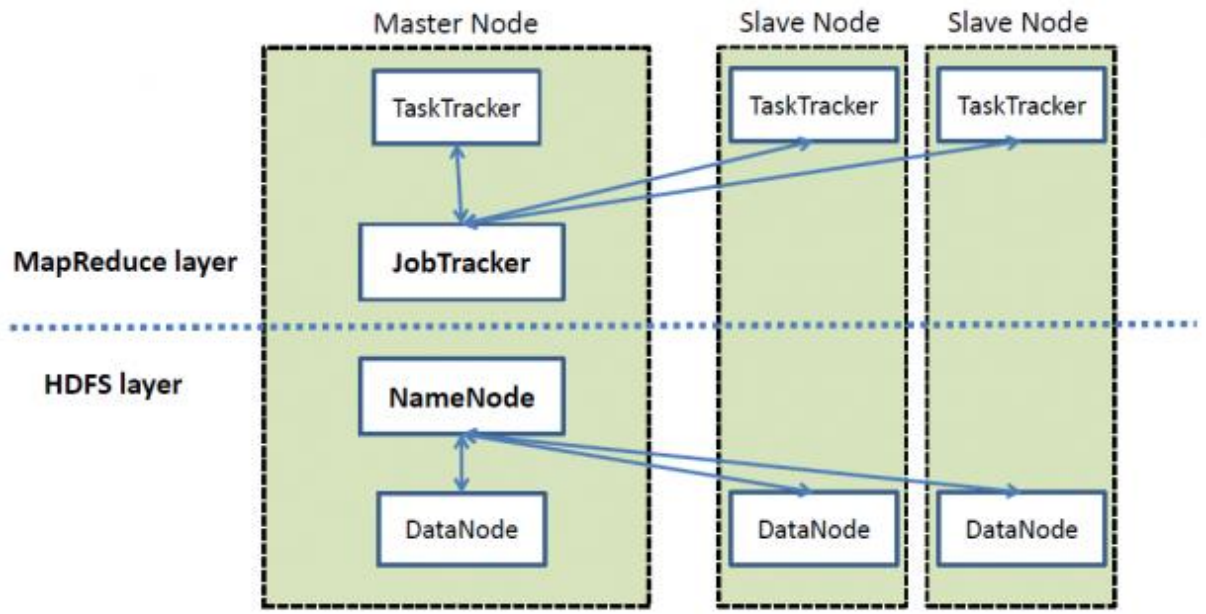

Abbildung 2: High Leven Hadoop Architektur [vgl. Literatur: [01]]

HDFS und MapReduce

Das HDFS ist ein verteiltes Dateisystem und besteht aus einem Name Node, dem sogenannten Master Node, und einem Data Node, d. h. dem Slave Node. Der Name Node ist der Admin der Data Nodes. Jeder dieser Data Nodes und der Name Node werden auf einem separaten Server im Cluster ausgeführt, wobei ein Name Node per Cluster existiert. Der Name Node enthält Informationen über die Leserechte der Daten, Speicherorte der Datenblöcke und Zugriffrechte auf die Daten. Ein Neustart des Name Node veranlasst Folgendes:

Der Name Node lädt die „fsimage“ Datei in den Speicher.

Der Name Node lädt die „edits“ Datei, die „Journal“[29]-File mit den aktuellen Dateioperationen.

Der Date Node „daemons“ sendet an den Name Node einen Bericht. Jeder Data Node beinhaltet einen Bericht über seine Blockdaten und die Funktionsfähigkeit.

[29] Das Journal beinhaltet alle Zugriffsrechte für die einzelnen Dateien, prüft ob der User dafür berechtigt ist und speichert alle Änderungen im Bezug auf die Daten ab.

Des Weiteren ist es möglich, die Daten in Blöcken aufzuteilen, wobei ein Block im Megabyte-Bereich liegen kann. Die Größe des Blocks ist konfigurierbar. Alle Änderungen gelangen zur Aufzeichnung in einem dafür reservierten Speicherbereich, dem sogenannten *„Journal“*.

Außerdem repliziert HDFS die Daten um einen konfigurierbaren Faktor, um die Ausfallsicherheit zu minimieren. Dabei erfolgt das Ablegen des replizierten Blockes in einem anderen Festplattenspeicher, denn bei Ausfall einer Festplatte ist ein Rückgriff auf replizierte Daten möglich. Jeder dieser individuellen Computer ist ein „Self-Contained Server“, der seinen eigenen Speicher, CPU, Plattenspeicher und installiertes Betriebssystem, vorzugsweise Linux, aufgesetzt bekommt.[30]

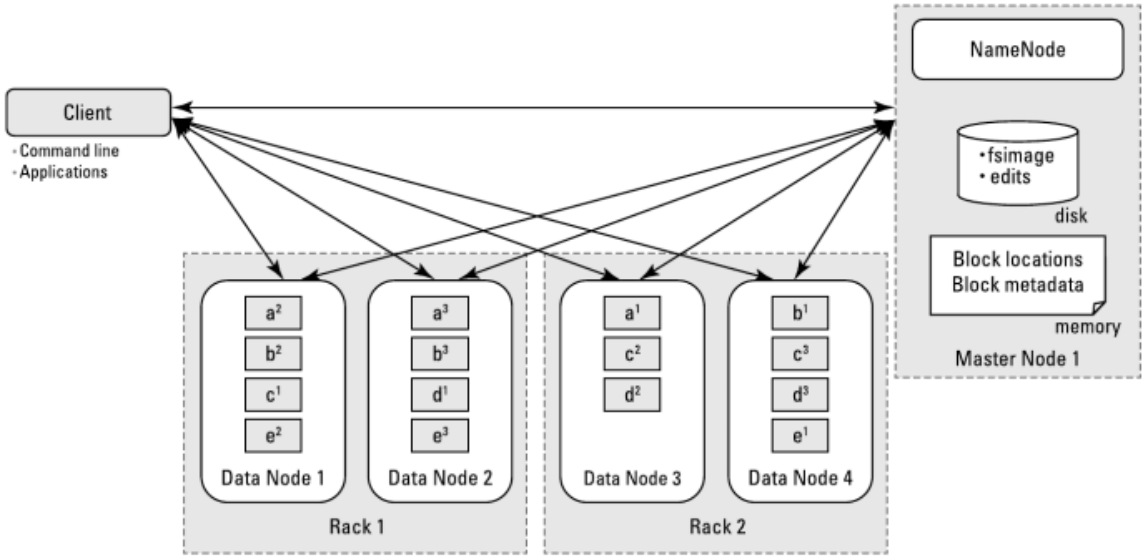

Abbildung 4: Interaktion zwischen den HDFS Komponenten [vgl.: Literatur: [01]]

Im Gegensatz zum Name Node beinhalten die Data Nodes nur die Daten. In Abbildung 3 ist die Interaktion zwischen dem Name Node und dem Data Node sowie dem Client dargestellt.

MapReduce nutzt die Datenlokalität, welche den Kerngedanken der Verknüpfung des Cluster-Dateisystems darstellt. Die Verarbeitung einer großen Datenmenge ist nur dann sinnvoll, wenn diese Daten dem verarbeitenden System zugrunde liegen Die Bestandteile von MapReduce layer, wie in Abbildung 2, sind: Task Tracker und Job Tracker.

[30] Vgl.: **Literatur: [01]** & Vgl.: **Literatur: [26]**

Task Tracker: Diese gelangen parallel zur Verarbeitung, dabei kommt es zur Verteilung auf allen denjenigen Servern, auf denen die Aufarbeitung von Daten ablaufen soll, und führen hier die Map- und Reduce-Tasks aus.

Job Tracker: Er koordiniert Jobs und ist zuständig für die Aufteilung von Jobs in Tasks bzw. „Splits“, enthält Informationen über die Speicherorte der zu aufarbeitenden Daten und überwacht periodisch den Lebenszyklus eines jeden Tasks.[31]

Hadoop 1.x verbindet HDFS mit dem parallelen Programmier-Modell MapReduce, das Batch-orientiert ist und somit den Einsatz für interaktive und iterative Analysen sehr einschränkt. Mit der Hadoop 1.x Version ist es nicht möglich, die Technologie in Echtzeit anzuwenden. Bei Hadoop 1.x ist es nur möglich, MapReduce auf dem Cluster ausführen zu lassen. MapReduce besteht aus den Phasen Map, Reduce, Shuffel und Sort, wobei Shuffle und Sort in einer Phase agieren. Die Anzahl der aufgeteilten Map-Reduce Splits ist gleich der Datenblöcke, die der Client auf diesen anwendet. Infolgedessen finden die Berechnungen parallel statt. In der Map Phase werden aus diesen Splits durch die Mapper Funktion „Listen“.

In dem folgenden Beispiel erfolgt die Übergabe von zwei Dokumenten an den Mapper, die als Schlüssel-Wert-Paar (Schlüssel: Zeilennummer, Wert: Textzeile) dienen:

(“name_1“, “Ich␣schreibe␣meine␣Bachelorarbeit“)
(“name_2“, “Ich␣bin␣auf␣der␣Arbeit␣und␣schreibe␣meine␣Bachelorarbeit“)

Die Map-Funktion ruft sich bei jedem übergebenen Dokument einmal auf, dabei kommt es zur Iteration über die Textzeile und bei jeder Iteration wird ein Schlüssel-Wert-Paar als Ausgabe erzeugt. Hierbei ist einerseits der Schlüssel das Wort und andererseits der Wert die Anzahl. Bei dem vorliegenden Beispiel ist das hier die Ziffer „1“.

(“name_1“, “Ich␣schreibe␣meine␣Bachelorarbeit“) = { (“Ich“, “1“), (“schreibe“, “1“), (“meine“, “1“), (“Bachelorarbeit“, “1“) }

(“name_2“, “Ich␣bin␣auf␣der␣Arbeit␣und␣schreibe␣meine␣Bachelorarbeit“) = { (“Ich“, “1“), (“bin“, “1“), (“auf“, “1“), (“der“, “1“), (“Arbeit“, “1“), (“und“, “1“), (“schreibe“, “1“), (“meine“, “1“), (“Bachelorarbeit“, “1“) }

[31] Vgl.: **Literatur: [01]** & Vgl.: **Literatur: [26]**

Es findet die Übergabe dieser an den Zwischenspeicher statt (siehe Tabelle 1):

Map-Phase:

Map 1	Key	Value	Map 2	Key	Value
	“Ich“	“1“		“Ich“	“1“
	“schreibe“	“1“		“bin“	“1“
	“meine“	“1“		“auf“	“1“
	“Bachelorarbeit“	“1“		“der“	“1“
				“Arbeit“	“1“
				“und“	“1“
				“schreibe“	“1“
				“meine“	“1“
				“Bachelorarbeit“	“1“

Tabelle 1: Mapper Phase - Darstellung der Key/Value Lists

Anschließend werden die Werte aus der Tabelle 1 auf demselben Data Node, als Output File abgelegt und für die Shuffle- sowie Sort-Phase zur Verfügung gestellt (siehe Tabelle 2).

Shuffel- und Sort-Phase:

Key	Value		Key	Value
“Arbeit“	“1“	→	“Arbeit“	“1“
“auf“	“1“	→	“auf“	“1“
“Bachelorarbeit“	“1“	→	“Bachelorarbeit“	“1“
“Bachelorarbeit“	“1“			“1“
“bin“	“1“	→	“bin“	“1“
“der“	“1“	→	“der“	“1“
“Ich“	“1“	→	“Ich“	“1“
“Ich“	“1“			“1“
“meine“	“1“	→	“meine“	“1“
“meine“	“1“			“1“
“schreibe“	“1“	→	“schreibe“	“1“
“schreibe“	“1“			“1“
“und“	“1“	→	“und“	“1“

Tabelle 2: Shuffle- und Sort-Phase - Darstellung Key/Value Lists

Nachfolgend gelangen die Lists aus der Tabelle 2 zur Ablage auf dem jeweiligen Data Node, auf dem es zur Ausführung des dazugehörigen Reducers kommt. Die dritte und letzte Phase addiert die Reducer Funktion die Anzahl der Value Werte eines Keys und es erfolgt das Erstellen der Zahl als Value mit dem zugehörigen Key in einer neuen Liste (siehe Tabelle 3):

Reduce-Phase:

In der Reduce-Phase kommt es zum Aufruf jedes Schlüssels und der dazugehörigen Liste von „Einsen“, die sie addiert.

Key	Value		Key	Value
“Arbeit“	“1“	→	“Arbeit“	“1“
“auf“	“1“	→	“auf“	“1“
“Bachelorarbeit“	“1“	→	“Bachelorarbeit“	“2“
	“1“			
“bin“	“1“	→	“bin“	“1“
“der“	“1“	→	“der“	“1“
“Ich“	“1“	→	“Ich“	“2“
	“1“			
“meine“	“1“	→	“meine“	“2“
	“1“			
“schreibe“	“1“	→	“schreibe“	“2“
	“1“			
“und“	“1“	→	“und“	“1“

Tabelle 3: Reducer Phase - Darstellung Key/Value Lists

Die in der Tabelle 3 ausgewerteten Daten gelangen zur Ablage auf einem Speicherort im HDFS; dieser ist zuvor festzulegen. Die beiden Funktionen map() und reduce() sind beliebig nach der jeweiligen Aufgabenstellung zu programmieren.

YARN

Wie bereits in Abschnitt HDFS und MapReduce erwähnt, ist MapReduce Batch-orientiert; seit Hadoop 2.x und der Entwicklung von YARN ist es möglich, diverse Datenanalysen Applikationen auszuführen. YARN steht für „*Yet Another Resource Negotiator*“, im Groben ist damit YARN als Ressourcenvermittler gemeint. Die Bestandteile von YARN sind:

Ressourcen Manager (RM): Dies ist die höchste Autorität, dieser ist zuständig für alle Ressourcen im Cluster und unterstützt bei der Verwaltung der Anwendungen auf dem YARN-System, ebenso findet eine Kommunikation zwischen dem RM und den NM sowie AM statt.

Node Manager (NM): Dieser nimmt Anweisungen vom RM entgangen und verwaltet die Ressourcen auf einem einzelnen Knoten.

Application Master (AM): Dieser startet die Container, auf denen die Berechnungen erfolgen.

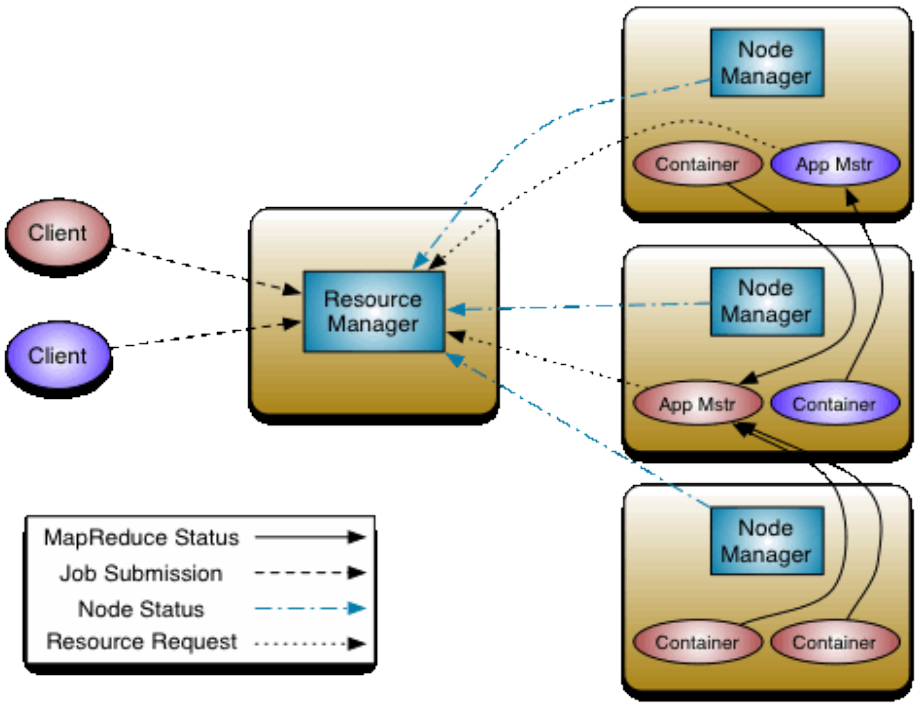

Abbildung 5: YARN – Architektur [vgl. Apache.org]

Zum Erlangen eines besseren Eindrucks über YARN folgt das Beschreiben der in Abbildung 5 dargestellten YARN-Architektur und der Ausführung einer Applikation mittels MapReduce in nachstehenden Schritten.

[1. Der Client, ein Java Code oder auch die Linux Eingabe Konsole sendet einen Request an den Resource Manager.

2. Der Resource Manager fordert den Node Manager auf, einen Application Master für diese Applikation zu erstellen, hier MapReduce. Der Node Manager erstellt einen Container für diesen und startet ihn.

3. Dieser neue Application Master initialisiert sich selbst durch die Registrierung am Resource Manager.

4. Der Application Master fordert von dem Name Node die Namen, Speicherorte und Informationen über die Datenblöcke an, um dementsprechend die Anzahl der Map- und Reduce Tasks berechnen zu können.

5. Der Application Master fordert mittels eines Requests von dem Recource Manager die notwendigen Ressourcen an und sendet in einem Intervall eine Signal-Message an den Ressource Manager über die aktuellen Ressourcen und Veränderungen, z. B., wenn es zum Killen eines Requests kommt.

6. Anschließend akzeptiert der Resource Manager den Request, dieser gelangt zur Ablage in einer Queue, in der alle zur Abarbeitung anstehenden eingegangenen Requests liegen.

7. Sobald die Data Nodes die Ressourcen über den Resource Manager erhalten, stellt der Resource Manager dem Application Master die Container an den spezifischen Data Nodes zur Verfügung.

8. Die Container der jeweiligen Node Manager werden dem Application Master zugewiesen und senden an ihn einen Container Launch Context (CLC). Der CLC umfasst alles, was der Application Task zur Inbetriebnahme benötigt: Umgebungsvariablen, Authentifizierung Token, lokale Ressourcen. Die lokalen Ressourcen sind zur Ermittlung der Laufzeit vonnöten, z. B. zusätzliche Daten oder auch die Applikationslogik in JARs und eine Befehlsfolge, um den aktuellen Prozess zu starten. Anschließend erstellt der Node Manager für die angeforderten Container einen Prozess und führt diesen aus.

9. Das Ausführen der Applikation geschieht während des Container Prozesses. Der Application Master überwacht dessen Prozesse. Falls bei einem Container oder Node ein Fehler auftritt, startet der Task auf einem alternativen verfügbaren Node den Prozess neu, aber im Falle des mehrfachen Ausfalls desselben Tasks (dieser Ausfallwert ist abhängig vom User) benachrichtigt der Application Master den Client, damit dieser darauf reagieren kann.

10. Ebenso kann der Resource Manager einen Kill-Befehl an den Node Manager senden, die Priorität im Scheduling oder auch die Applikation selbst kann den Container-Prozess beenden.

11. In den Containern werden die aufgeteilten Map- und Reducetasks, wie im „HDFS und MapReduce“ dargestellt, ausgeführt. Bei einem Abschluss eines Maptasks kommt es zur Zwischenspeicherung dieser, es schließen sich die Verarbeitung durch den Reducetask und die Übergabe an den Application Master an.

12. Wenn alle Tasks erfolgreich abgeschlossen haben, sendet der Application Master die Ergebnisse an den Client, informiert den Resource Manager über die erfolgreiche Ausführung, meldet sich bei ihm ab und beendet den Prozess.][32]

Unterschied von Hadoop 1.x zu Hadoop 2.x

Die Unterschiede der Hadoop Versionen bilden die Tabellen 4 und 5 ab. Da Hadoop 2.x Bestandteil dieser Arbeit ist, folgt die Erläuterung der Funktionsweise von Hadoop 2.x mittels eines Beispiels:

Deamons

HDFS Deamons sind in Hadoop 1.x und 2.x gleich. Der größte Unterschied ist, dass Hadoop 2.x YARN verwendet anstelle von MapReduce (siehe Tabelle 4).

Daemons	Hadoop 1	Hadoop 2
HDFS (Keine Veränderungen)	Name Node (master) [einer pro Cluster] Secondary Name Node Data Node (worker bzw. slave) [viele pro Cluster, einer pro Node]	Name Node (master) [einer pro Cluster] Secondary Name Node Data Node (worker bzw. slave) [viele pro Cluster, einer pro Node]
Processing	MapReduce v1 Job Tracker (Master) [einer pro Cluster] Task Tracker (worker bzw. slave) [viele pro Cluster, einer pro Node]	YARN (MR v2) Resource Manager [einer pro Cluster] Node Manager [viele pro Cluster, einer pro Node] Application Master [viele pro Master]

Tabelle 4: Verschiedenheit der Daemons von Hadoop 1.x zu Hadoop 2.x

Die erste Hadoop Version ist eine einfache Batch-Prozess-Engine, auf der ausschließlich MapReduce läuft. Aufgrund von Hadoop 2.x und YARN erfuhr das simple Batchverfahren

[32] Vgl.: **Literatur: [01]**

seine Ersetzung durch eine Plattform, die eine Anwendung verschiedener Verfahren ermöglicht. Zur Verdeutlichung der Unterschiede zwischen Hadoop 1.x und Hadoop 2.x erfahren die beiden Versionen in Tabelle 5 einen Vergleich und in Abbildung 6 und 7 findet dessen Veranschaulichung statt. Die Schicht Distributed Storage ist für die Datenablage, Ressource Management für die Verwaltung der Prozesse, Processing Framework für die Verwendung der Frameworks und deren Abarbeitung und Application Programming Interface für die Programmierung und Nutzung der APIs zuständig. (siehe Tabelle 5):

Schichten	Hadoop 1.x	Hadoop 2.x
Distributed Storage	Das HDFS ist für die Lagerung der Daten, für Zwischenergebnisse und End-Ergebnisse zuständig.	Es hat sich nichts verändert.
Ressource Management	Diese Schicht überwacht die Daten in Bezug auf CPU-Zyklus, RAM, Netzwerk Bandbreite und ebenso Informationen über die Berechnungen im Data Node. Dies sind die Aufgaben des JobTrackers, welches aus dem Name Node läuft.	Diese Schicht setzt einen Resource Manager ein, der auf dem Name Node läuft, der ausschließlich für die unterschiedliche Verwaltung der Applikationen sorgt mittels eines Schedulers und im Gegensatz zu Hadoop 1.x keine Informationen über die Data Note enthält.
Processing Framework	Die im „HDFS und Map Reduce“ dargestellten drei Phasen übernimmt der TaskTracker, der auf den Data Nodes diese Verfahren abarbeitet.	Da YARN für mehrere Verfahren entwickelt wurde und nicht nur für das Batchverfahren, übernimmt diese Schicht den Prozess.
Application Programming Interface (API)	Die Schicht ist zuständig für die von dem User übernommene Programmierung, Hadoop 1.x bietet wie in Abbildung 7 zur Verfügung gestellte Frameworks, weitere APIs siehe Abbildung 4.	Um z. B. in der API Schicht das HBase verwenden zu können, wurde das Processing Framework Hoya entwickelt; vergleiche Abbildung 8. Weitere APIs siehe Abbildung 4

Tabelle 5: Unterschiede der einzelnen Schichten von Hadoop 1.x zu Hadoop 2.x

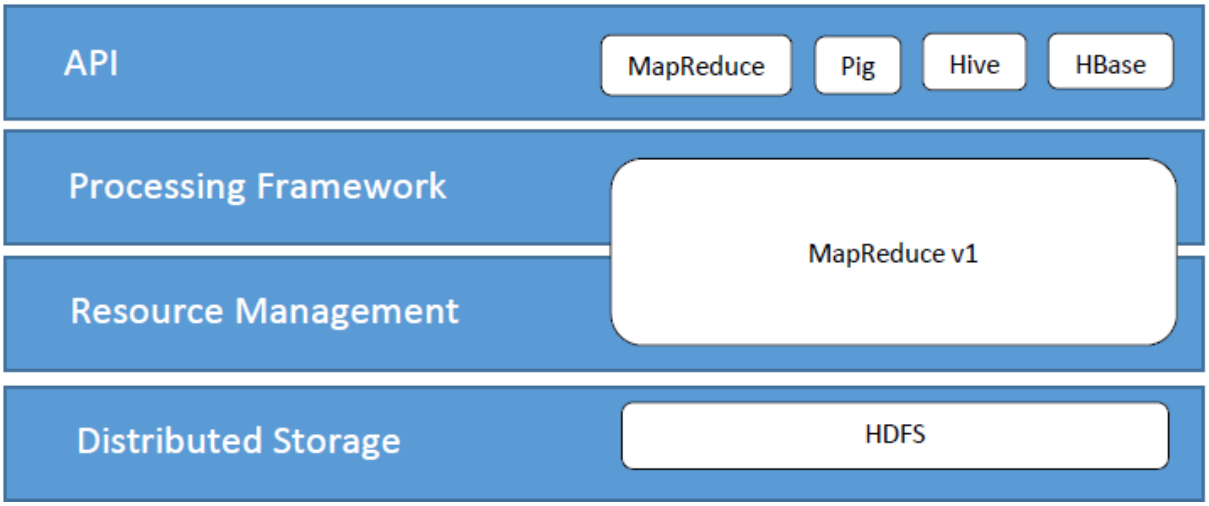

Abbildung 6: Übersicht von Hadoop 1.x

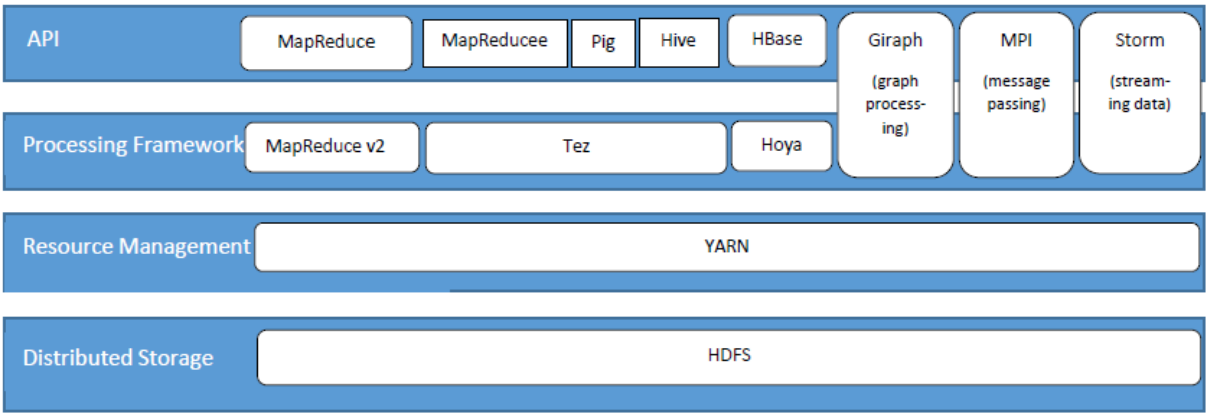

Abbildung 7: Übersicht von Hadoop 2.x

Apache Spark

Die University of California in Berkeley hat das Open-Source-Projekt „Spark“ im Jahr 2009 entwickelt und 2010 vorgestellt. In 2013 fand Spark Aufnahme in die Apache Software Foundation. Anfang 2014, mit der Veröffentlichung der Version 1.0, erhielt Spark den Status eines Top-Level-Projektes der Open-Source-Organisation. Die Spark Commmunity wächst kontinuierlich, denn Firmen wie Amazon, Yahoo etc. haben Spark in ihre Systeme integriert und tragen zur Weiterentwicklung des Frameworks bei. Wie bereits in Abschnitt HDFS und MapReduce erwähnt, stellt MapReduce eine Ineffizienz für interaktive Abfragen und iterative Berechnungen auf Basis von Hadoop HDFS dar. Hinsichtlich der Ineffizienz von MapReduce entwickelten die Entwickler der University of California das Framework Spark. Spark verarbeitet Berechnungen und Abfragen in Bezug auf das HDFS im Vergleich zu MapReduce in einer kurzen Latenzzeit. Vor der Publikation der ersten Forschungsarbeit war Spark für bestimmte Analyseanwendungen 10 bis 20 Mal schneller als MapReduce. Bei der

Veröffentlichung von Version 1.0 erreicht Apache Spark mit dem In-Memory-Konzept eine 100 Mal schnellere und auf den Speicher-Platten eine 10 Mal schnellere Leistungsfähigkeit gegenüber MapReduce. Infolge der Kostensenkung der Arbeitsspeicher gerät In-Memory für mehrere Unternehmen zunehmend interessanter. Das In-Memory-Konzept ist im „Big Data“ Umfeld zu einem wichtigen Schlagwort geworden, dabei kommt es direkt im Hauptspeicher zum Vorhalten von hohen Datenvolumina, die dort mit großer Geschwindigkeit zur Verarbeitung gelangen. Die Programmiersprachen Scala, Phyton und Java beinhalten die Spark API zu ihrer Verfügung. Diese Arbeit verwendet ausschließlich die Spark Java API.

Kernkonzept von Spark sind die RDDs (Resilient Distributed Datasets). Ein RDD besteht aus einer „Collection“ von Elementen. Zwei Möglichkeiten, die Spark anbietet, ein RDD zu initialisieren, sind: Parallelisieren einer vorhandenen „Collection“ und Verweis auf eine Datenmenge auf einem gemeinsamen Dateisystem wie HBase, HDFS etc. Abbildung 8 enthält die zwei Arten von Operationen, die für eine Analyse der Daten Anwendung finden: zum einen Transformationen (Tranformations), die einen neuen Datensatz aus einem vorhandenen erstellen und diesen iterativ ausführen, und zum anderen Aktionen (Actions), die einen Wert (Value) nach der Ausführung einer Berechnung auf diese Datenmenge zurückgeben.

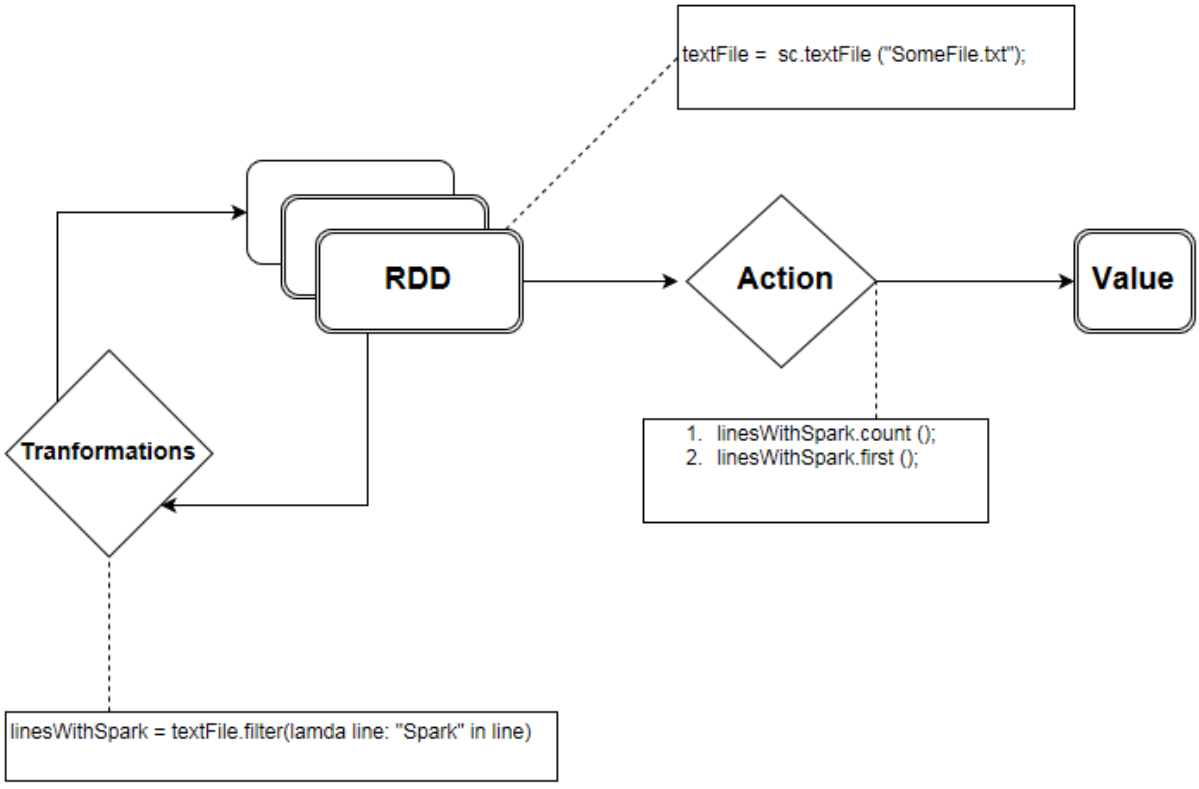

Abbildung 8: Funktionsweise von Action und Transformations

Bei dem In-Memory-Konzept werden die Daten vom RDD beim Starten des Spark Context in den Cache gelegt, bei einem erneuten Zugriff auf dem gleichen RDD werden die Daten vom Cache schneller gelesen und minimiert die Zugriffs- bzw. Lese-zeit. Die Transformation von Spark stellt bei MapReduce die Map dar und Action den Reduce. Im Vergleich zu MapReduce arbeitet Spark nach dem Prinzip der Lazy Evaluation, d. h. Spark wird erst bei einer Action ausgeführt. Das Lazy Evaluation Konzept ist sowohl hilfreich für die Testumgebung als auch für die Nutzbarkeit von Spark RDDs, denn damit muss der Spark User keine Mapper und Reducer Jobs optimieren.[33]

Hadoop Ökosystem

Im Umfeld von Hadoop 2 kam es zur Entwicklung vieler Apache Top Level Projekte, die eine Datenanalyse ohne vorhandene Programmierkenntnisse in Java ermöglichen. Pig, Hive, HBase Storm, Spark und Solr sind Client- sowie Server-seitige Projekte, diese gelangen zur Ausführung auf YARN, deshalb findet keine Installation auf dem Hadoop Cluster statt. Hingegen sind die clientseitigen Tools sowie Frameworks Ambari und ZooKeeper zu den Verwaltungstools einzuordnen. Diese ermöglichen eine einfachere Verwaltung der Ressourcen im Hadoop-Cluster.

[Pig: Pig ist eine Highlevel Sprache, die es ermöglicht, komplexe MapReduce-Jobs zu schreiben. Pig besteht aus einer abstrakten Skriptsprache (*Pig Latin*), welche es erlaubt, auf einer abstrakteren Ebene den Datenfluss eines MapReduce-Jobs zu beschreiben. Darauf basierend erzeugt der Pig-Compiler bereits optimierte MapReduce-Jobs, welche dann wieder in gewohnter Form mittels MapReduce zur Ausführung gelangen, sodass der Benutzer nicht alle Details selbst implementieren muss.

Hive: Hive ist ebenfalls eine Abstraktionsschicht. Hive bringt eine SQL-ähnliche Sprache (HiveQL) mit, diese ermöglicht Aufgaben wie Aggregationen, Analysen und weitere Abfragen auf in HDFS gespeicherte Datenmengen durchzuführen. Am Ende wird jedoch auch HiveQL wieder in MapReduce-Jobs transformiert und so ausgeführt.

HBase: Hbase ist eine Datenbank, die auf dem BigTable-Ansatz basiert, d. h., sie speichert die Daten spaltenorientiert ab. Dabei setzt HBase auf HDFS und bietet so eine fehlertolerante Speicherung von Daten an, diese werden mit schnellem Zugriff auf die verteilten großen Datenmengen kombiniert. Außerdem erweitert HBase das Hadoop-System durch das Transaktionsmanagement um benutzerdefinierte Updates sowie Insert- und Delete-Operationen.

---

[33] Vgl.: **Literatur: [29]**

Storm: Storm ist ein verteiltes Echtzeit-Rechensystem für die schnelle Verarbeitung großer Datenströme. Storm verfügt über zuverlässige Echtzeit-Datenverarbeitungsfähigkeiten, dadurch kommt es zum Erschaffen neuer Geschäftsmöglichkeiten mit Low-Latency-Dashboards, Sicherheitswarnungen. Außerdem ermöglicht Storm seine Ausführung mit weiteren Anwendungen wie MapReduce auf dem Hadoop Cluster.

Solr: Das Tool Solr ist für die Index basierende Suche von Daten in HDFS zuständig. Solr bietet leistungsstarke Volltextsuche und eine fast Echtzeit-Indizierung an. Mit Solr ist eine schnelle Suche möglich, egal ob ein Nutzer nach Tabellen, Texten, Geo-Locations oder Sensordaten in Hadoop sucht.

Ambari: Das Managementtool Ambari ist für die Vereinfachung der Speicherverwaltung sowie für die Überwachung des Hadoop-Clusters zuständig, da dieser durchaus aus Tausenden Knoten bestehen kann. Ambari umfasst eine Reihe von einfach bedienbaren Tools und APIs.

ZooKeeper: Das Managementtool ZooKeeper ist für die Koordinierung der verteilten Prozesse im Hadoop-Cluster zuständig. Verteilte Anwendungen können kann ZooKeeper zur Speicherung, Verteilung und Aktualisierung wichtiger Konfigurationsinformationen verwenden.][34]

Big Data Technologien

Wie bereits in Kapitel 2.1 unter den „3 Vs“ erwähnt, sind die klassischen Transaktionalen IT-Systeme, z. B. ERP, in der Regel nur in der Lage, strukturierte Daten zu speichern und zu verarbeiten. Auch dies fand in Kapitel 2.1 Erwähnung, bei der Betrachtung von Big Data Technologien ist in der Regel immer das Framework Hadoop beteiligt. Zum einen ist der Vorteil von Big Data Technologien gegenüber dem Transnationalen IT-System, dass es möglich ist, nahezu alle Datenformate in einem System zu analysieren, zum anderen sind die Kosten niedriger und die Skalierung ist in kleineren Schritten möglich. Big Data Technologien enthalten aber auch Nachteile; es ist für die Last- und Datenverteilung zu sorgen und es ergibt sich eine höhere Fehlerrate durch einfachere Hardware. Transaktionale Sicherheit ist nicht gegeben. Bei Verwendung von Big Data Technologien ist zu überlegen, ob die Vielzahl an Daten zu dem jeweiligen Anwendungsfall einen Mehrwert (Value) erzielt. Falls ja, muss überprüft werden, ob die Möglichkeit besteht, die drei Herausforderungen: Anzahl sowie Größe der Daten (Volume), Vielfalt der Daten (Variety) und die schnelle Analyse (Velocity) mithilfe von Big Data Technologien zu lösen.

---

[34] Vgl.: **Literatur: [01],** *Vgl.:* **Literatur: [26]**

2.2 Data Mining

Data Mining ist in vielen Bereichen eines Unternehmens einsetzbar wie Marketing, Controlling und Forschung. Dabei werden die Erkenntnisse aus den vorliegenden Informationen gewonnen. Prof. Dr. Richard Lackes von der Technischen Universität Dortmund erklärt Data Mining in einem Satz:

„Unter Data Mining versteht man die Anwendung von Methoden und Algorithmen zur möglichst automatischen Extraktion empirischer Zusammenhänge zwischen Planungsobjekten, deren Daten in einer hierfür aufgebauten Datenbasis bereitgestellt werden.“[35]

Data Mining umfasst ein breites Spektrum, daher erfolgt lediglich nur eine kurze Beschreibung der zwei Anwendungsklassen wie Cluster-Analyse und Web Mining. Grund hierfür ist, dass die Beschreibung weiterer Anwendungsklassen den Rahmen dieser vorliegenden Arbeit sprengen werden. Als Letztes folgt eine nähere Erläuterung von Text Mining.

Das Ziel der Cluster-Analyse ist es, eine große Datenmenge in Teilmengen, sogenannte Cluster, zu zerlegen. Versicherungen können mittels dieser Analyse Kunden in Abhängigkeiten von Alter und Arztbesuchen in Gruppen unterteilen und aufgrund dessen z. B. eine spezifische Versicherung anbieten.

Web Mining ist ein spezielles Verfahren, dabei kommt es im Internet auf den einzelnen Webseiten nicht nur nach der Suche nach Texten, sondern auch nach Strukturen. Hinsichtlich einer gezielten Werbung findet eine Analyse des Kundenverhaltens statt.

Bei Text Mining geht es darum, Informationen und Zusammenhänge zwischen Textbausteinen und Texten einzubeziehen. Ein Beispiel hierfür ist ein Vergleichsportal für Reisen, welches erkennt, wenn ein Kunde unzufrieden mit einem Hotel war. Dabei erkennt die Software bei diesem Satz: „Zum Glück bin ich jetzt zu Hause“ die Unzufriedenheit des Kunden mit dem Hotel. Zur Analyse des Textes gibt es ein breites Spektrum an Technologien (siehe Kapitel 2.1 Big Data), die solch eine Analyse ermöglichen. Es folgen eine Veranschaulichung des Text Mining Prozess in Abbildung 9 und eine nachfolgende Beschreibung[36].

---

[35] Vgl.: **Quelle: [16]**
[36] Vgl.: **Quelle: [07]**

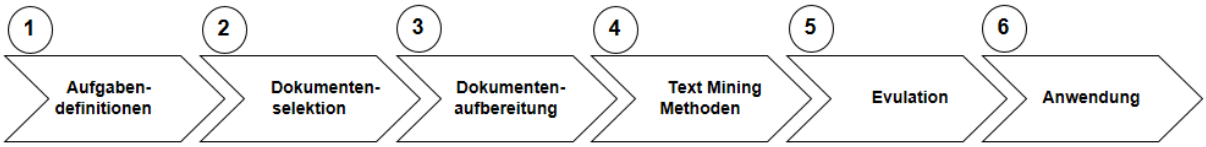

Abbildung 9: Text Mining Prozess[37]

[1. *Aufgabendefinitionen*: Zunächst ist es wichtig, einen Anwendungsfall ausführlich zu definieren und die Text Mining Ziele abzuleiten.

2. *Dokumentenselektion*: Anhand der vorab definierten Ziele sind die Datenquellen auszuwählen. Dabei können verschiedene Dokumente von Bedeutung sein wie z. B. E-Mail, Berichte, Artikel etc.

3. *Dokumentenaufbereitung*: Anhand der ausgewählten Daten sind Terme aus den Texten zu filtern. Zu den gefilterten Termen können irrelevante Wörter (Stop words) gehören, aber es gilt, auch Schlüsselwörter zu erkennen, damit keine Informationen verloren gehen. In dieser Phase sind auch die Daten für die nachfolgende Phase vorzubereiten, z. B. aus CSV Tabellen Spalten zu filtern, die nicht von Bedeutung sind.

4. *Text Mining Methoden*: Diese Phase dient der Durchführung der aufbereiteten Dokumente, Analysen; dabei gibt es viele Methoden und Tools wie Spark, welches in Kapitel 2.1 ausführlich Erklärung fand.

5. *Interpretation und Evaluation*: Hier vollzieht sich die Auswertung der Ergebnisse der vorherigen Analyse.

6. *Anwendung*: Anwendung findet dies in Bereichen, die für die Analyse bzw. Auswertung vieler textueller Dokumente bestimmt sind. Anwendungsgebiete können sein: Portale, Intranet, Internet, Call Center, Plagiatserkennung, Banken etc. z. B. ist es in einem Call Center möglich, ein sich wiederholendes Problem schneller zu identifizieren und zu beheben.][38]

In der Abbildung 9 und Phase 3 der Datenaufbereitung gibt es eine Vielzahl an anzuwendenden Methoden, nachfolgend kommt es zur Vorstellung des Einfachen Levenshtein-Abstands, des Jaro-Winkler-Abstands und des Damerau Levenshtein-Abstands. Es existieren zwei weitere Methoden, Part-Of-Speech (POS) und N-Gramm, die vor den oben genann-

---

[37] Vgl.: **Quelle: [07]**
[38] Vgl.: **Quelle: [07]**

ten Methoden zur Anwendung gelangen können; diese Methoden sprengen jedoch den Rahmen dieser Arbeit, wobei N-Gramm mit Apache Lucene die Schnelligkeit von N-Gramm erhöhen könnte.

Levenshtein-Abstand:

1966 entwickelte Vladimir Joseph Levenshtein diese Methode zur Fehlererkennung und -korrektur von per Funk übertragenen Codes.[39] Dies ist nachweislich eine mächtige Methode. Mittels dieser Methode lassen sich die Abstände zweier Objekte messen. Es sollten Strings sein und sie sollten sich nur über das Alphabet repräsentieren lassen. Der Algorithmus verfügt über Einfüge-, Lösch- und Ersetz-Operationen, für jede dieser Operationen entstehen Kosten in Höhe des Wertes Eins „1“. Kommt es dabei zu einem Vergleich von zwei Strings, wobei das WORT1 in WORT2 überführt, entstehen dabei wie folgt kurz zu erläuternde Operationen.

Berechnung:

Delete(D);
Löschen von WORT2(i), Kosten: 1
Insert(I);
Einfügen des Zeichens WORT1(i) in WORT2, Kosten: 1
Match(M);
Übereinstimmung von WORT1(i) und WORT2(i), Kosten: 0
Austausch(A);
Sind die Zeichen WORT1(i) und WORT2(i) unterschiedlich, so wird das Zeichen durch WORT2(i) in WORT1(i) ersetzt, Kosten: 1 (Je nachdem wie der Algorithmus implementiert ist, kann ein Kostenfaktor von Zwei „2“ entstehen)

Die Laufzeit beträgt dabei O(m*n), wobei m die Anzahl der zu vergleichenden Wörter sind.[40]

Anhand eines Beispiels erfolgt eine kurze Erläuterung des Algorithmus. Es findet eine Überführung von WORT1 in das WORT2 statt (siehe Tabelle 6) und eine von WORT0 in WORT2 (siehe Tabelle 7).

Beispiel: WORT0: Maus, WORT1 : Häuser, WORT2: Haus

---

[39] Vgl.: **Literatur: [15]**
[40] Vgl.: **Quelle: [01]**

WORT1	H	ä	u	s	e	R	
WORT2	H	a	u	S			
Operation	M	A	M	M	D	D	
Kosten	0	1	0	0	1	1	

Tabelle 6: Beispiel des Levenshtein-Abstandes zwischen Haus und Häuser

WORT1	M	a	u	s			
WORT2	H	a	u	S			
Operation	A	M	M	M			
Kosten	1	0	0	0			

Tabelle 7: Beispiel des Levenshtein-Abstandes zwischen Haus und Maus

In der Tabelle 6 und Tabelle 7 und der Zeile Kosten werden nun alle Operationen aufaddiert; für die Tabelle 6 ergibt sich: 0+1+0+0+1+1 = 3. Für die Tabelle 7 ergibt sich: 1+0+0+0 = 1. Der Abstand aus der Tabelle 7 ist kleiner als der aus der Tabelle 6, da aber das Wort Haus auch existiert, ist dort der Abstand 0. Die Vorgehensweise in Bezug auf die Werte ist anwendungsspezifisch und sollte der jeweiligen Problemstellung als Informationen dienen.

Jaro und Jaro-Winkler Abstand:

Jaro und Jaro-Winkler Algorithmus messen den Abstand zwischen zwei Zeichenketten, dabei liegen die Werte im Bereich [0,1]. Bei einem Wert von 1 bzw. 0,99 sind die Wörter annähernd gleich, bei 0 sind die Wörter unterschiedlich. Im Vergleich zum Jaro-Winkler Abstand ist der Jaro Abstand ausschließlich ein gutes Verfahren für Strings mit geringen Abweichungen und hat eine Laufzeit von O(m+n).[41] Anhand eines Beispiels findet eine nähere Erläuterung des Algorithmus statt.

String1: Moritz; String2: Moritz Hacht

Zunächst erfolgt die Bestimmung aller in String1 und String2 existierenden Zeichen; dabei kommt es zur Bestimmung von Abstand d, der nicht die Länge der Hälfte der kleinsten Zeichenkette überschreiten darf. D. h.:

[41] Vgl.: **Literatur: [22]** S. 95

$d = |x(i) - y(j)| \leq \left\lfloor \frac{\min(|x|,|y|)}{2} * \frac{1}{1} \right\rfloor \; mit \; x = String1, y = String2.$

Nach der Identifizierung des Abstands d = 4 und durch die Iteration über die Zeichenketten aller vorkommenden, in x und y ermittelten gemeinsamen Zeichen erfolgt eine erneute Iteration über die Zeichenketten x und y, dabei wird ein Zähler um 1 inkrementiert, wenn das Zeichen x(i) nicht gleich y(i) ist. Die Anzahl der Transposition t ist gleich der Hälfte des inkrementierten Zählers. Stellen die gemeinsamen Zeichen die Menge m dar, ergibt sich für die Ähnlichkeitsgleichung:

$$Jaro(x,y) = \frac{1}{3} \times \left(\frac{|m|}{|x|} + \frac{|m|}{|y|} + \frac{|m| - t}{|m|}\right)$$ [42]

Mit m ={M,o,r,i,t,z} somit ist m = 6, t = 3, x = 6, y = 12;

Um den Jaro Algorithmus auch für Namensfelder zu verbessen, kommt es zur Ermittlung des Jaro-Winkler Abstands. Dieser geht von einem Präfix p und einem Korrekturfaktor f aus, daraus ergibt sich die folgende Formel:

$$JaroWinkler(x,y) = Jaro(x,y) + |p| \times f \times \big(1 - Jaro(x,y)\big)$$ [43]

Das gemeinsame Präfix ist p = „Moritz“ = 6 und die Konstante Korrekturfaktor 0.1. Da es bei der Menge m keine Zeichenvertauschung gibt, ist t =0, daraus ergibt sich JaroWinkler(x,y)= Jaro(x,y) + 6 * 0.1* (1 – Jarao(x,y).

Damerau Levenshtein-Abstand:

Der Damerau Levenshtein-Abstand ist in der Theorie eine verbesserte Form des Levenshtein-Abstands, dabei berücksichtigt der Algorithmus die Vertauschung zweier benachbarter Zeichen und zählt diese als Kostenfaktor = 1.[44]

2.3 Verwendete Tools

Dieser Abschnitt geht nicht detailliert auf die Tools ein, da der Hauptfokus dieser Arbeit nicht auf die Tools liegt. Für die Realisierung des Web-Interfaces und für die Vereinfachung des Build-Prozesses werden ein Build – Tool und ein Servlet Container verwendet. Für diese

[42] Vgl.: **Literatur: [14]** S. 414- 420
[43] Vgl.: **Literatur: [27]**
[44] Vgl.: **Literatur: [06]** S. 171-176, **Literatur: [22]**

Ausarbeitung kommt es zur kurzen Auseinandersetzung mit dem Servlet Container für das Web-Interface und mit dem Build-Tool für die Build-Prozesse. Zunächst erfolgt die Behandlung des Build-Tools, dann das des Servlet Containers. Ist mehr als ein Projekt vorhanden, z. B. Projekt A und Projekt B, ist es sinnvoll, ein Build-Tool zu verwenden. Das folgende Beispiel erläutert den Grund. In Projekt A sind dies der Controller und die View, in Projekt B das Model. Es wird das Projekt B dem Projekt A hinzugefügt, damit in Projekt A der Controller die Model Klassen des Projekts B aufrufen kann. Ändert sich an dem Model eine Klasse, Methode oder kommt es zur Testerweiterung, dann ist das Projekt B neu zu kompilieren und dem Projekt A hinzuzufügen. Bei einer umfangreicheren Software mit mehreren miteinander agierenden Projekten gestaltet sich dies zunehmend aufwendig. Dieses und ein weiterer Vorteil, die Vereinfachung durch das Einbinden von Frameworks in die Projekte, sind ausreichend, ein Build-Tool für diese Ausarbeitung zu verwenden. Es gibt mehrere Build-Tools auf dem Markt, die bekanntesten sind Ant, Gradle und Maven. Diese Ausarbeitung hat Maven verwendet, da Maven im Vergleich zu Ant zur Laufzeit die Abhängigkeiten zu den anderen Frameworks erstellt. Gradle ist im Vergleich zu Maven und Ant komplexer, da die Abhängigkeiten nicht wie bei Ant und Maven in Xml zu schreiben sind, sondern in Java, das macht Gradle komplexer. Als Abhängigkeit ist bei Maven das Anbinden der Bibliotheken von Frameworks zur Laufzeit gemeint. Das sind die Gründe für die Verwendung von Maven. Für weitere Vorteile siehe Verweis auf den Link.[45] Bei Maven ist eine bestimmte Projektstruktur einzuhalten, damit sich beim Kompilieren keine Fehler ergeben (siehe Abbildung 10).

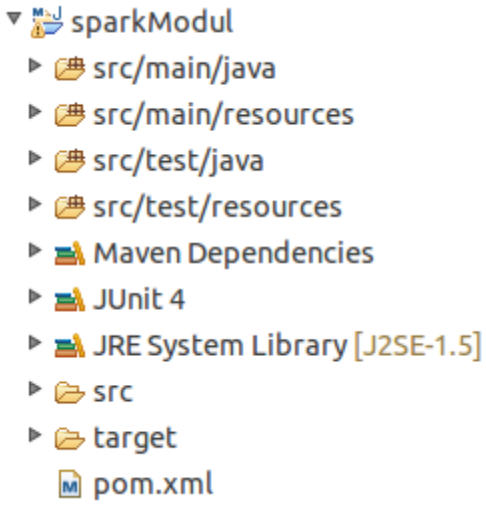

Abbildung 10: Projektstruktur bei Maven

[45] Vergleich von Ant, Maven und Gradle: http://blog.seitenbau.com/gradle-was-kommt-nach-maven/

Abbildung 11 verdeutlicht das Herunterladen der Abhängigkeiten von Maven, bevor es den Code kompiliert. Das ist am „Downloading“ in Abbildung 11 zu erkennen. „Downloading“ ist die URL[46] für den Download der Java Bibliotheken. Für die Installation und die Konfiguration von Maven wurde das Buch von Bachmann, K. U. (2010) Maven 2 – Studentenausgabe vewendet. Für ein näheres Einlesen ist dieses Buch zu empfehlen.

```
[INFO] --- tomcat7-maven-plugin:2.3-SNAPSHOT:run (default-cli) @ \
Downloading: https://repository.apache.org/content/repositories/sr
Downloaded: https://repository.apache.org/content/repositories/sna
Downloading: https://repository.apache.org/content/repositories/sr
Downloaded: https://repository.apache.org/content/repositories/sna
[INFO] Running war on http://localhost:8080/ViewProject01
[INFO] Using existing Tomcat server configuration at /home/kumar/v
[INFO] create webapp with contextPath: /ViewProject01
```

Abbildung 11: Java Code und Tomcat kompilieren mit Maven

Für den Servlet Container gibt es eine Reihe von Produkten auf dem Markt. In der Abbildung 12 sind diese abgebildet; dabei steht LP für Liberty profile, TC für Tomcat, GF für Glassfish, Jetty und JB für JBoss.[47] Für diese Ausarbeitung fiel die Entscheidung für Tomcat. Tomcat nimmt, wie in Abbildung 12 zu sehen, den zweiten Platz ein und ist bei der Installation und Konfiguration einfacher. Die anderen Servlet Container finden hier keine Berücksichtigung.

	LP	TC	GF	JETTY	JB
Download and Installation	4	5	3.5	5	4
Tooling support	4	4.5	4.5	3.5	4.5
Server Configuration	5	4	3	4	4
Real performance metrics	3.5	5	4.5	4	4.5
Features & Open Standards compliance	4	3	5	3	5
Documentation & community	3.5	5	4	3	4.5
Administration & Management/UI	3	2.5	5	2	4.5
Cost $$$/Licensing	4	5	5	5	4
Total	**31**	**34**	**34.5**	**29.5**	**35**

Abbildung 12: Tabelle zur Auswahl eines Servlet Containers (Vgl.: Quelle: [23])

[46] Wird im beim Verweis 55 kurz erklärt.
[47] Vgl.: **Quelle: [23]**

Für die Einrichtung von Tomcat erfuhr der folgende Link unter dem Verweis Verwendung[48]. Tomcat wurde nur für das System konfiguriert, ein weiterer Einsatz erfolgte damit nicht. Tomcat stellt einen Container zur Verfügung. Mit Container ist eine Umgebung gemeint, die den Java-Code auf dem Web Server ausführt. Der Servlet Container ist in Java geschrieben, der Java Servlets (JS) und Java Server Pages (JSP) übersetzt und ausführt. Die JS stellen in dem ViewProject01 die Controller und JSP die View dar. In Abbildung 14 ist dargestellt, wie der Verlauf eines Servlets im Zusammenhang mit dem Tomcat verläuft; es erfolgt nachstehend eine Beschreibung. Bei Tomcat ist unbedingt eine bestimmte Struktur einzuhalten (siehe Abbildung 13). Dieser sucht in der Struktur nach der web.xml Datei, denn in dieser sind die Controller den jeweiligen Benutzer Oberflächen zugeordnet.

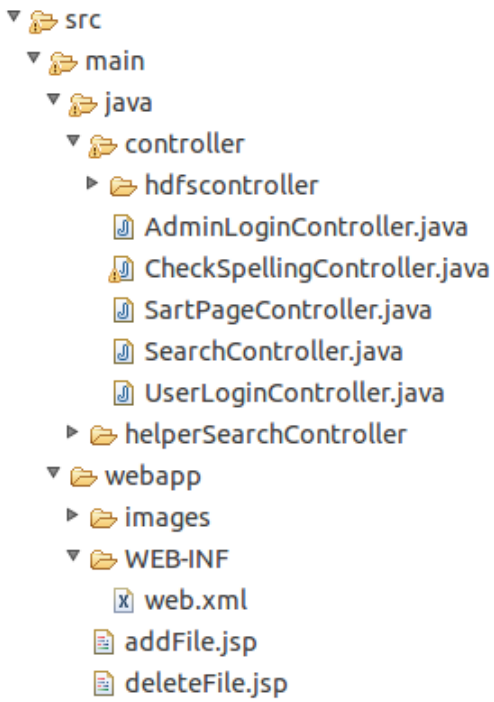

Abbildung 13: Projektstruktur bei Tomcat

Zunächst kommt es kurz zur Erläuterung der Begriffe http Protocol und URL. Als erstes geht es um das Betrachten des http Protokolls, dann der URL. Das http Protokoll ist die: *„Abkürzung für HyperText Transfer Protocol; im Internet zur Übertragung von für Dokumente verwendetes Protokoll. Unter Verwendung dieses Protokolls dekodiert der Browser die in HTML-Dokumenten enthaltenen Auszeichnungsanweisungen (Tags) und stellt diese dann*

[48] Zum Vertiefen des Wissens zu Tomcat Link:
https://codedecoder.wordpress.com/category/ubuntu/

dar“[49]. URL ist die Abkürzung von „*Uniform Resource Locator; eindeutige Identifikation bzw. Adresse eines HTML-Dokuments im Internet. Die URL setzt sich aus der Domäne und der Angabe des Ortes des Dokuments auf dem Server zusammen.*“[50]

Nachfolgend wird anhand der Abbildung 14 der Verlauf für das Webinterface von der Eingabe durch den Service Desk Mitarbeiter bis hin zur Anzeige der Antwort beschrieben.

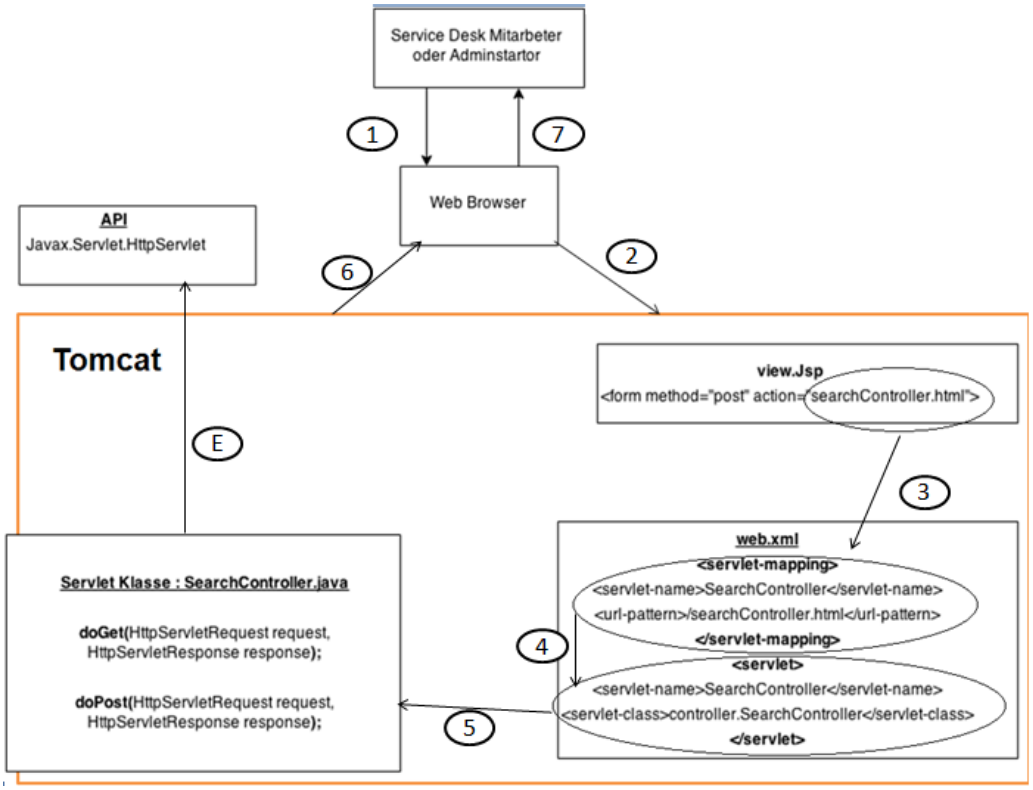

Abbildung 14: Ablauf eines Java Servlets mit Tomcat

1. Der Service Desk Mitarbeiter befindet sich auf der URL „*http://localhost:8080/ViewProject01/searchController.html*“[51] im Browser und kann z. B. in einem Textfeld einen Text eingeben und anschließend auf den Button „Suchen“ klicken. Durch das Klicken auf den Button „Suchen“ kommt es zum Auslösen eines Events.

2. Es wird die URL „*http://localhost:8080/ViewProject01/searchController.html*“ an den Tomcat gesendet mittels des in Punkt 1 ausgelösten Events.

3. Es wird der letzte Teil vom URL, hier „*searchController.html*“, verglichen in dem Ordner Web-INF mit dem URL-Pattern in der Datei web.xml.

---

[49] Vgl.: **Quelle: [14]**
[50] Vgl.: **Quelle: [15]**
[51] http://[host]:8080/[anwendung]/servlet/[Servlet Name].

4. Anschließend wird unter dem Ordner Web-INF in der Datei web.xml geschaut, zu welcher Servlet Klasse der letzte Teil vom URL „*searchController.html*“ gehört.

5. Die Servlet Klasse in diesem Fall ist der SearchController.java. Es erfolgt das Auslösen desselben, die Parameter werden von der Benutzer Oberfläche über den Controller an den Model weitergeleitet. Das Model bearbeitet die Werte und gibt ein Ergebnis an den SearchController.java zurück.

6. Anschließend wird durch den SearchController eine neue View mit den Ergebnissen erstellt und dem Browser mit einer neuen URL übergeben.

7. Der Service Desk Mitarbeiter kann sich das Ergebnis ansehen.[52]

Dies war die Beschreibung für die Servlet und den Tomcat. Tiefergehende Informationen sind den zahlreichen Publikationen in verschiedenen Medien zu entnehmen.

3 Datenanalyse

Dieses Kapitel geht zunächst auf die Daten ein und analysiert diese. Anschließend folgt das Aufzeigen von Methoden und Ideen, mit denen es möglich ist, eine Analyse der aufbereiteten Daten auszuführen. Im Weiteren ist eine einheitliche Sprache notwendig, deshalb steht zunächst die Beleuchtung der wichtigen Begriffe an. Zur Erläuterung gelangen zunächst die Begriffe Call ID, Incident ID, Call, Ticket und Ticketsystem[53]. Es schließt sich ein kurzes Beispiel an.

1. Mit der Call ID ist es möglich, den kompletten Verlauf des Problems zu verfolgen. Die Call ID hat die Form CALLXXXXX, wobei die X Zahlen darstellen.

2. Anhand der Incident ID sind lediglich die Teilprobleme oder versuchten Methoden, die ein Service Desk Mitarbeiter vorgenommen hat, einsehbar. Die Incident ID hat die Form IMXXXXX, wobei die X Zahlen darstellen.

3. Das Ticketsystem ist das System, das die Aufnahmen der Probleme und Daten des Kunden enthält.

Nun zum Beispiel: Gibt ein Kunde ein Problem beim Mitarbeiter des Service Desk auf, heißt dieser Call. Ruft der Kunde wegen des gleichen Problems an und gibt mehr Informationen

---

[52] Vgl.: **Literatur: [16]**
[53] Vgl.: (ITIL Foundation) **Literatur: [28]**

über dasselbe Problem, kommt es zur Aufnahme dieser Informationen, die Incident heißen, und zur Zuordnung dem jeweiligen Call gegenüber. Ein Ticketsystem nimmt die Daten und Probleme von Kunden auf.

3.1 Anforderungen

Dieser Abschnitt erläutert, welche Anforderungen für die Datenanalyse, die Systemarchitektur und die Software für diese Arbeit erfüllt sein müssen, um eine Analyse der Service Desk Daten auszuführen. Das Ziel ist es, dem Service Desk Mitarbeiter eine Lösung für sein Problem anzuzeigen. Dabei gibt er über ein Suchfeld auf dem Web-Interface einen Text ein. In diesem Text steht das Problem des Kunden.

Anforderungen an die Daten

Zunächst bestehen Anforderungen an die Service Desk Daten. Es ist wichtig, aus den Service Desk Daten eine Struktur zu erkennen, das heißt, es sollten Beschreibungen des Problems und die Lösung ersichtlich sein, falls eine vorhanden ist. Des Weiteren gilt es, Zusammenhänge festzustellen. Ist z. B. nicht ersichtlich, zu welchem Problem welche Lösung gehört, dann ist eine Zuordnung nicht nachzuvollziehen. Umso mehr Informationen vorhanden sind über das Problem des Kunden, desto besser sind die Möglichkeiten, dem Service Desk Mitarbeiter eine hilfreiche Lösung anzuzeigen. Des Weiteren sind folgende Kriterien für die zu analysierenden Daten von Bedeutung:

1. Es soll die Möglichkeit geben, anhand einer Information den kompletten Stand und Verlauf des Problems zu sehen;

2. es sollen Lösungen vorhanden sein, falls es welche gibt;

3. es soll ersichtlich sein, ob es zur Behebung des Problems des Kunden gekommen ist, denn nur dann können Lösungen vorhanden sein;

4. wann wurde das Problem behoben bzw. das Problem aufgenommen,

Anforderungen an die Systemarchitektur

Im Weiteren finden Anforderungen an die Systemarchitektur statt. Für diese Arbeit kam lediglich eine Hardware zum Einsatz. Diese agiert als Master und Slave. Es gilt, umso leis-

tungsfähiger die CPU[54] der einzelnen Hardware ist, desto besser ist die Berechnung für die einzelnen Slaves. Die in Kapitel 4 vorgestellte Systemarchitektur beschränkt sich nicht nur auf einen Master und Slave, sondern ist auf einem Master und n Slaves anzuwenden, wobei n für die Anzahl der Slaves steht. Ebenso gilt es, dem System eine realistische Zeit vorzugeben, an die es sich zu halten hat. Von der Eingabe über das Web-Interface bis zur Anzeige der Lösung ist für den Prototypen der Zeitrahmen von drei Minuten einzuhalten. Dieser Zeitrahmen ist groß. Aber diese Zeit ist abhängig von der Auswahl des Algorithmus (siehe Kapitel 2.2).

Anforderungen an die Software

Die Software sollte lose gekoppelt sein, das heißt, der Austausch einer Komponente der Software sollte keine großen Änderungen erfordern. Des Weiteren sollte die Software übersichtlich und gut dokumentiert sein. Auch ist ein Konzept für eine Web-Anwendung zu erstellen; dabei sind die Entwurfsmuster[55] zu verwenden. Auch gilt es, für einen Testfall ein Beispielkonzept zu erstellen. Des Weiteren sind die folgenden Anforderungen einzuhalten. Zu jedem nachfolgenden Punkt werden die Anforderungen kurz begründet.

1. Der Service Desk Mitarbeiter darf im Suchfeld nur 120 Zeichen eingeben, damit dieser nur die wichtigen Begriffe eingibt.

2. Dem Service Desk Mitarbeiter ist die Möglichkeit zu geben, seine Suche weiter einzugrenzen. Falls der Service Desk Mitarbeiter weiß zu welcher Kategorie das Problem gehört kann dieser die Suche einzugrenzen.

3. Bei einem Rechtschreibfehler soll das Anzeigen der Korrektur erfolgen. Bei einer Suche mit Rechtschreibfehler ist es naheliegend das keine Lösung angezeigt wird, deshalb sollte auf diese Fehler aufmerksam gemacht werden.

4. Das Ergebnis der Suche soll der Mitarbeiter des Service Desks in einem Überblick angezeigt erhalten. Des Weiteren ist anzuzeigen, ob das folgende Ergebnis Lösungen beinhaltet oder nicht. Durch diese Anforderung kann der Service Desk Mitarbeiter auf dem ersten Blick erkennen ob es eine Lösung zu dem Call ID vorhanden ist.

---

[54] *Central Processing Unit (CPU);* „Kern" eines Computers (vgl. http://wirtschaftslexikon.gabler.de/Definition/zentraleinheit.html?referenceKeywordName=CPU)

[55] Entwurfsmuster sind ein Konzept aus der Software-Architektur, in der immer wieder gleichartige Aufgabenstellungen (wenn auch oft mit unterschiedlichen Programmiersprachen) gelöst werden müssen. Entwurfsmuster beschreiben auf hoher Abstraktionsebene jeweils einen Ansatz für eine solche Lösung.(Vgl. http://www.itwissen.info/definition/lexikon/Entwurfsmuster-design-pattern.html)

3.2 Service Desk Daten Analyse

Der Service Desk ist ein wesentlicher Prozess in einer Organisation (Unternehmen). Dieser ist der erste Anlaufpunkt für den Endanwender für sein Problem. So ist es in der Information Technology Infrastructrue Library, kurz ITIL[56], beschrieben. ITIL ist ein Framework, das Methoden, Ansätze und Betriebsprozesse für eine optimierte, serviceorientierte und qualitätsgesicherte IT-Service-Struktur anbietet. ITIL wurde nicht nur von einer, sondern von mehreren Organisationen (Unternehmen) und aufgrund ihrer Erfahrungswerte weiterentwickelt.[57] Die vorliegende Arbeit verwendet Service Desk Daten von Audi. Hinsichtlich der Analyse der Daten kommt es zunächst zur Fragestellung, „woher werden die Daten bezogen?“. Dabei gibt es zum einen das Ticketsystem und zum anderen die Reporting Engine. Zunächst geht es um die Behandlung des Ticketsystems, anschließend um die Reporting Engine. Das Ticketsystem ist die erste Anlaufstelle für den Kunden; dort kommt es zur Aufnahme der vom Kunden angegebenen Störungen. Es gibt eine Reihe von Herstellern für Ticketsysteme wie z. B. HP Service Center Peregrine, BMC Remedy ITSM, ServiceNow, Omnitracker und viele andere. In dieser Ausarbeitung wird das Ticketsystem HP Service Center Peregrine verwendet. Abbildung 15 stellt den Verlauf einer Kundenanfrage anhand eines UML Diagrammes für das Ticketsystem dar. Am Startpunkt nimmt der Service Desk Mitarbeiter die Störung auf, dabei erstellt das Ticketsystem automatisch eine Call ID. Es kommt bei jeder aufgenommenen Störung zur Vergabe einer eindeutigen Call ID, die Incident ID ist ebenfalls eine eindeutige, dem Call zugeordnete ID. Eine Call ID kann mit mehreren Incident ID verlinkt werden. Im Anschluss an die Beschreibung von Tabelle 10 erfolgt ein näheres Eingehen auf die Call- und Incident IDs. Im nächsten Schritt, „Kategorisierung“, erfolgt die Zuordnung der Störung einer Kategorie; diese ist einzutragen. Jede Kategorie hat eine Unterkategorie, die „Category“ und „Sub Category“ heißen. Nur der Administrator ist befugt, die Kategorien zu löschen bzw. eine neue hinzuzufügen; dies ist nicht üblich und findet strengste Überwachung. Bei der „Priorisierung der Störung“ erhält der Mitarbeiter eine Aufforderung, im Freitextfeld die Problemstellung zu beschreiben. Ist die Störung bzw. das Problem bekannt, erfolgt eine Suche danach; existiert eine Lösung dazu, kommt es zu ihrer unverzüglichen Behebung. Ist aber das Problem nicht bekannt oder existiert dafür keine Lösung, findet eine Weiterleitung der Störung zum „2nd Level“ Support statt. Falls der „2nd Level Support“ die Störung auch nicht beheben kann, nehmen die Experten vom „3rd Level Support“ die Störung an und schließen den Auftrag, sobald die Störung behoben ist. Im Anschluss an die Behebung der Störung ist im letzten Schritt die Lösung unbedingt gut zu dokumentieren.

---

[56] Vgl.: **Literatur: [28]** S.3
[57] Vgl.: **Literatur: [28]** S.3

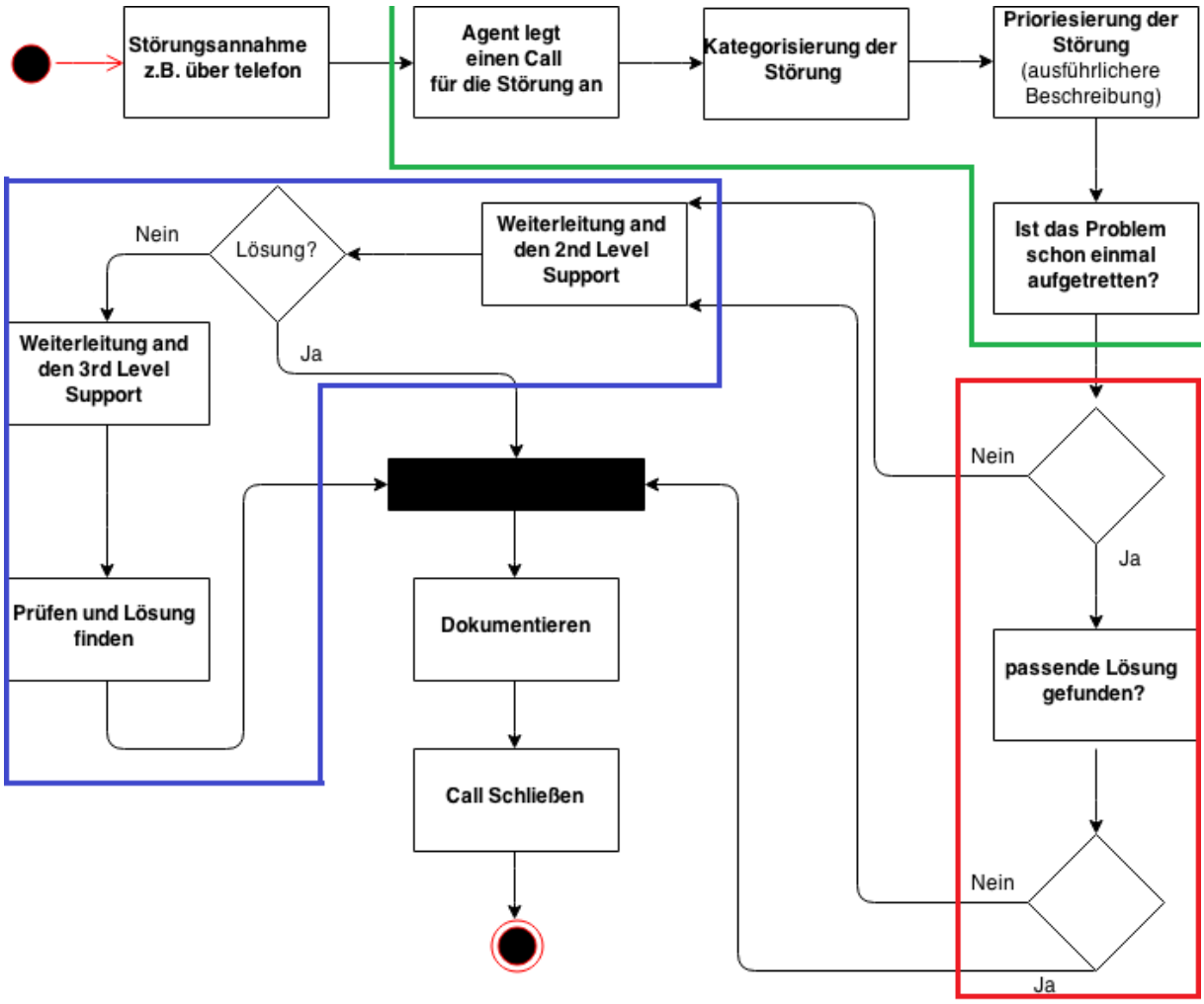

Abbildung 15: Verlauf zur Behebung des Kundenproblems

Wie bereits in der Beschreibung der Abbildung 15 erwähnt, kommt es zur Zuordnung einer neuen Incident ID an den jeweiligen Call. Ein Beispiel: Der Kunde A gibt seine Störung auf, der Mitarbeiter nimmt diese auf und erzeugt dadurch eine neue Call ID. Im Anschluss an das Einpflegen der Daten und des Problems des Kunden erfolgt mittels des Abspeicherns dieser Call ID das Erzeugen einer Incident ID und ein Link zu dieser Call ID. Es vollzieht sich das Hinterlegen der Call ID mit dem Status „Offen“. Meldet sich der Kunde A zu einem späteren Zeitpunkt mit der gleichen Störung, wodurch es zum Gewinnen neuer Informationen kommt, geschieht das Erstellen einer neuen Incident ID mit neuen Informationen und zu dieser Call ID, die offen ist, eine Verlinkung. Im Vergleich zum Ticketsystem ist es mit der Reporting Engine möglich, die Daten aus dem Ticketsystem grafisch oder tabellarisch darzustellen. Ein Beispiel ist ein Auszug der Incident IDs und eine Erstellung des Datums in tabellarischer Form. Ein Auszug ist eine exportierte Datei aus dem Ticketsystem oder der Reporting

Engine in Form von Tabellen oder Grafiken. Die Auszüge aus der Reporting Engine heißen auch Reports. Es erfolgt das Speichern dieser Reports, die immer wieder zur Verfügung stehen. Auch ist es möglich, die Reports automatisiert generieren zu lassen und zum Beispiel dem Management diese einmal in der Woche zukommen zu lassen. Aus diesen Daten könnte z. B. das Management die Anzahl der an einem Tag erstellten Incidents feststellen, um dadurch neue Erkenntnisse zu gewinnen. Diese neu gewonnenen Erkenntnisse dienen der Prozessoptimierung.

Der vorherige Abschnitt erläuterte die Quelle der Daten für die Analyse des Service Desks. Nachfolgend kommt es zum Analysieren der Daten nach dem Schema des Text Mining Prozesses der ersten drei Phasen (siehe Kapitel 2.2, Abbildung 9), wobei die erste Phase der Zielsetzung in Kapitel 1.2 getätigt wurde. Die zweite Phase beschreibt die Selektion der Daten, die von Nutzen sind für die Analyse und die dritte Phase sucht in den selektierten Daten nach Zusammenhängen und dient der Datenaufbereitung.

Phase 1: Selektieren der Daten

Es folgt eine grobe Vorstellung der Daten. Diese Daten liegen in Form von Excel Dateien vor, die in den Abbildungen 11, 12, 13, 14 und 15 abgebildet sind. Diese Dateien enthalten Spaltenüberschriften, die im Anhang unter den Punkt 8.1 tabellarisch beschrieben sind und für das Verständnis wichtig sind. Diese Tabelle beschreibt sämtliche Felder im Ticketsystem und das Reporting Engine. Die Abbildungen 11, 12, 13, 14, 15 werden von oben links beginnend nach unten rechts endend gelesen. Anschließend folgt eine Begründung, welche Daten von Nutzen sind.

1. HPSC-Export Files

Es fand ein Exportieren der in der Abbildung 16 dargestellten Excel File aus dem HPSC Peregrine Ticketsystem statt. Es sind alle Spalten außer der Call ID und der Aktivitäten Spalte enthalten. Diese Datei enthält viele Informationen über das Kundenproblem, u. a. die in der Tabelle im Anhang unter dem Punkt 8.1 erklärte Spalte „Status“. Diese Spalte enthält nur den Status „Closed“, d. h., der Call ist geschlossen.

	A	B	C	D
1	Incident ID	Open Time	Update Time	Reopen Time
2	IM2629727	31.08.2012 00:00	07.09.2012 05:12	
3	IM2629728	31.08.2012 00:02	07.09.2012 05:12	

	E	F	G	H
1	Unsuspend Time	Close Time	Status	Incident Status
2		07.09.2012 05:12	closed	Closed
3		07.09.2012 05:12	closed	Closed

	I	J	K	L
1	Cause Code	Initial Cause Code	Category	Sub Category
2	8037	8037	SUP services	application
3	2007	2007	desktop services	login

	M	N	O	P
1	Problem Type	Product Type	Type	Brief Description
2		PPM-Center		PPM-Center: Antrag Kennwortrücksetzung
3	reset	password		MyNet: Kennwortrücksetzung

	Q	R	S	T
1	Severity Code	Open Group	Assignment	Opened By
2	4	SD	SD SOLVED	Ulf Hasenclever
3	4	SD	SD SOLVED	Ulf Hasenclever

	U	V	W	X
1	Last Updated By	Reported by first name	Reported by last name	Reported by ID
2	Problem Process	Max Mustermann	Max Mustermann	1900C2A11EA16B11
3	Problem Process	Max Mustermann	Max Mustermann	784367

	Y	Z	AA	AB
1	User Priority	Department	Call Origin	Closed By Group
2	4		Phone	SD SOLVED
3	4		Phone	SD SOLVED

	AC	AD	AE	AF
1	Res Analyst Code	Closed By	audimg.tslmcode	eVIP
2		Max Mustermann	false	false
3		Max Mustermann	false	false

	AG	AH	AI	AJ
1	Update Action	Kalenderwoche	Monat	Ticketalter
2	0	35	8	843
3	0	35	8	843

	AK	AL	AM
1	letzte Aktualisierung	SKM verändert	Monat
2	836	Nein	8
3	836	Nein	8

Abbildung 16: HPSC - Export Excel File

2. Incident IDs Excel File

Die in der Abbildung 17 abgebildete Excel File wird aus der Reporting Engine exportiert. Diese Excel File weicht von der Struktur des Ticketsystems ab und enthält nur Spalten aller geschlossenen Incidents. Sie beinhaltet keine Lösungen zu den Störungen, auch wenn ein Mitarbeiter diese in ein Ticketsystem eingepflegt hat.

6	Incident ID	Title	Initial Cause Code	Cause Code	Status	Category
7	IM2251361	[Notebook]: [Keine Netzwerkverbindung in Györ]	4222	4222	Closed	network services
8	IM2782553	TS * IM21008079 bitte an AMS EHANI Support VW (Herr Toth Balint) weiterleiten	4222	6891	Closed	Others

6	Subcategory	Product Type	Open Time	Close Time	Open Group	Close Group
7	LAN	other	27.01.2012 13:49:37	26.04.2013 15:32:52	SD AUDI	SD SLM AUDI
8	Customer	Complaint	28.11.2012 13:40:27	30.04.2013 15:10:45	SD SLM AUDI	SD USER AUDI

Abbildung 17: Incident – Export aus der Reporting Engine

3. Call ID zur Incident ID File

Die Abbildung 18 enthält alle geschlossenen Calls IDs. Es kam zum Schließen dieses Incidents im Mai und des Calls im Juni, sodass die Call IDs nicht immer in dem jeweiligen Monat zu finden sind. Dennoch zeigt diese Datei, welche Call ID zu welcher Incident ID gehört.

4	Interaction ID	INC Incident ID	Open Group	Open Time
5	CALL5117501	IM2782553	SD S	03.12.2012 09:47:29
6	CALL5219361	IM2834859	SD S	16.01.2013 12:23:21

Abbildung 18: Calls zur Incident Export File aus dem Ticketsystem

4. Aktivitäten zu Incidents

Der Export der Excel Tabelle in Abbildung 19 geschieht aus der Reporting Engine und beinhaltet nicht alle Spalten. Die Struktur weicht vom Ticketsystem ab und enthält die Aktivitäten der Incidents, die unter 2. Incidents Excel File und Abbildung 17 dargestellt sind.

4	INC Activity Activity Number	INC Activity Incident ID	INC Activity AssignmentGroupFrom	INC Activity AssignmentGroupTo	INC Activity StatusChangeFrom
5	001A18342589	IM2251361			
6	001A18342590	IM2251361			

4	INC Activity StatusChangeTo	INC Activity Type	INC Activity Date/Time
5		First Assignment	27.01.2012 13:49:38
6		Open	27.01.2012 13:49:38

Abbildung 19: Aktivitäten zu den Incidents

5. Call IDs – Export File

Die in Abbildung 20 abgebildete Excel Tabelle ist eine weitere exportierte Datei aus dem Ticketsystem. Diese Datei umfasst nicht alle Spalten des Ticketsystems. Dennoch sind die aufgenommenen Daten im Freitextfeld „Description“ ausführlicher als die in Abbildung 16 unter „Brief Description“ beschriebenen.

	A	B	C	D	E
1	opened.by	open.time	open.group	incident.id	cause.code
2	Vorname Nachname	06.05.2013 05:52	AHM ZLT	CALL5554740	
3	Vorname Nachname	06.05.2013 00:43	SD	CALL5554741	4251

	F	G	H	I	J
1	Status	contact.name	Description	assignment	problem.type
2	Closed	2534	"G1 F11-12 között és K1	"VOITH FM G1", , "VOITH	configuration
3	Closed	247257	"Mynet: Fingerprint funkti	"SD BACKDESK ",	configuration

	K	L	M	N	O
1	product.type	Category	subcategory	worked.time	Handle Time
2	400_imap	SUP services	B2E Portalservice		06:33:09
3	400_imap	SUP services	B2E Portalservice		00:13:09

	P	Q	R	S
1	Type	Solved by Level 1	gl.number	closed.by
2		false	IM3030556	linker
3	computer	false	IM3030515	linker

Abbildung 20: Call IDs Export File aus dem Ticketsystem

Die Abbildungen 16-20 vermitteln einen Überblick über die Struktur der Daten. Es folgt die Erläuterung der wichtigsten Kriterien. Anschließend unterbleibt bei der Datenanalyse die Berücksichtigung aller Dateien, die nicht mindestens zwei dieser Kriterien erfüllen. Enthält eine Datei die Call ID, ist diese automatisch von Bedeutung, da die Call ID alle Informationen ersichtlich macht. Die Kriterien, die eine Datei enthalten sollte, sind:

1. Call ID

Anhand der Call ID lassen sich alle Informationen über ein Problem des Kunden über den kompletten Zeitraum entnehmen.

2. Incident ID

Anhand der Incident ID kommt es zur Verdeutlichung der Informationen über die einzelnen Aktivitäten[58], die der Service Mitarbeiter vorgenommen hat.

3. Beschreibung des Problems

Die Datei sollte eine „Description“ beinhalten, auf der eine Suche ausgeführt wird.

4. Status des Auftrages

Anhand des Status ist es möglich, zu ersehen, ob ein Call geschlossen ist, denn nur bei einem geschlossenen Call besteht die Möglichkeit einer Lösung.

[58] Eine Aktivität ist, wenn ein Service Desk Mitarbeiter versucht hat das Problem zu beheben.

Eine Datei ist zunächst von Nutzen, wenn diese mindestens zwei dieser Kriterien erfüllt. Die Datei unter 1. HPSC – Export File beinhaltet fast alle in einem Ticketsystem hinterlegten Informationen; daher kommt diese Datei in die engere Wahl (siehe Abbildung 16) und erfüllt das Kriterium. Die Datei unter 2. Incident ID – Excel File enthält keinerlei neue Informationen in einer Gegenüberstellung zur HPSC – Export File, auch wenn diese das Kriterium erfüllt, daher sind dieser Datei keine neuen Informationen zu entnehmen. Deshalb erfolgt keine weitere Berücksichtigung dieser Datei unter 2. (siehe Abbildung 17). In der Datei unter 3.Call Id zur Incident ID ist die Spalte Call ID von Bedeutung, da diese Spalte nicht in den vorherigen Dateien vorkommt; außerdem kommt es zur Herstellung einer Verbindung zwischen der Call ID und der Incident ID. Aus der Datei unter 4. Aktivitäten zu den Incidents gestalten sich alle Spalten neu. Eine Gewinnung neuer Erkenntnisse aus diesen Spalten findet allerdings nicht statt, da diese nur den aktuellen Stand eines Incidents darstellen, z. B., dass dieser in Bearbeitung ist, außerdem erfüllt diese Datei nicht das Kriterium (siehe Abbildung 18). Die Daten aus der Abbildung 20 unter 5. Calls IDs – Export File beinhalten viele Informationen, dabei fällt ein wesentlicher Unterschied in der Spalte Description im Vergleich zur HSPSC – Export File unter Abbildung 16 in der „Brief Description“ auf. Als Unterschiedsmerkmal enthält die Export IDs – File in der Briefdescription mehr Text; dadurch sind mehr Informationen über ein Kundenproblem zu entnehmen. Das ist ein wichtiger Grund für die nähere Betrachtung dieser Datei. Die ausgewählten Dateien sind die folgenden unter 1.HPSC – Export File (siehe Abbildung 16), 3.Call IDs zu Incident IDs (siehe Abbildung 18) und 5.Call IDs – Export File (siehe Abbildung 20).

Phase 3: Suchen eines Zusammenhangs in den ausgewählten Daten und Aufbereiten der Dateien

Nach der Ermittlung der wesentlichen Daten ist es von Bedeutung, einen Zusammenhang unter diesen Daten zu finden. Dabei ist es naheliegend, den Zusammenhang für die Call und Incident ID zu finden.

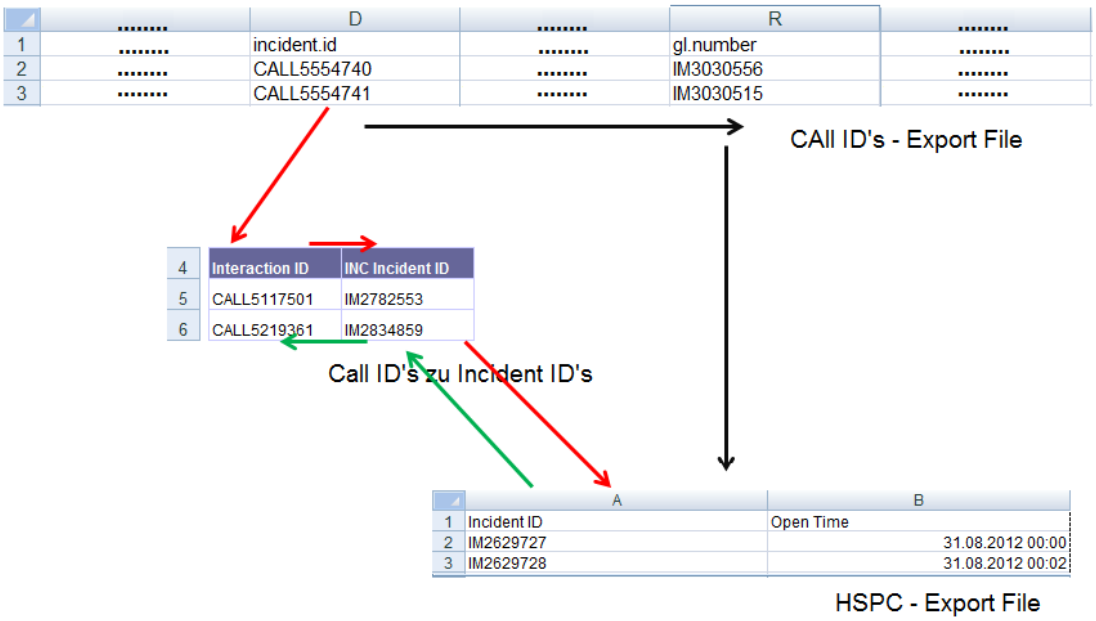

Abbildung 21: Zusammenhang der wichtigen Service Desk Daten

In der Abbildung 21 ist zum einen die Datei Call IDs – Export File mit dem Call ID sowie Incident ID abgebildet und zum anderen die HPSC – Export File mit der Incident ID. Des Weiteren ist die Datei Call IDs zu Incident IDs in abgespeckter Form zu sehen. Die Datei HPSC – Export File beinhaltet die Spalten für die Beschreibung „Brief Description“ und Lösung „Update Action“. Deshalb erfolgt lediglich die Suche der Call ID nach der zugehörigen jeweiligen Incident ID. Die grünen Pfeile markieren den Verlauf. Die Datei Call IDs – Export File beinhaltet die Spalte für die Beschreibung, aber nicht die Lösung. Deshalb kommt es im ersten Fall in der Call ID – Export File zu der jeweiligen Call ID nach der Suche nach der Incident ID. Ist eine Incident ID vorhanden, kommt es in der HPSC – Export File zur Suche nach der Lösung zu dieser ID (siehe schwarzen Pfeil). Ist die Incident ID nicht in der Call IDs – Export File vorhanden, dann geschieht im zweiten Fall, in der Datei Call Ids, zur Incident IDs zu der jeweiligen Call ID die Suche nach der dazugehörigen Incident ID und anschließend nach der Lösung in der HPSC – Export File (siehe roter Pfeil). Nach der Entdeckung eines Zusammenhangs ist es von Bedeutung, unwesentliche Informationen aus den Dateien zu filtern. Es werden alle drei ausgewählten Dateien HPSC – Export File, Call IDs – Export File und Call IDs zu Incident ID’s im Einzelnen wie aufgezählt betrachtet und dabei erfolgt die Begründung, welche Spalten für wichtig zu halten sind. Als erstes sind die folgenden Daten in den Spalten der Datei HPSC – Export File von Bedeutung: Incident ID, Open Time, Status, Category, Sub Category, Brief Description, Update Action. Anhand des Incidents sind die

Aktivitäten ersichtlich. Die Spalte Open Time kann von Bedeutung sein, wenn es um das Erstellen einer Sortierung der Lösung nach der Zeit geht.

Der Status teilt dem Mitarbeiter des Service Desks mit, ob der Call geschlossen ist und es deshalb eine Lösung zu dem Problem Lösung geben kann. Die Spalten Category und Sub Category eignen sich für das Einschränken einer Suche. Brief Description ist eine wichtige Spalte, weil darin die Suche mit dem eingegebenen Text des Mitarbeiters erfolgt. Die Spalte Update Action ist sehr wichtig, da diese Spalte dem Mitarbeiter das Anzeigen der Lösung ermöglicht. Als zweites sind die wichtigen Daten in der Datei Call IDs – Export File die gleichen Spalten wie auch in der Datei HPSC – Export File. Bei der dritten Datei sind die Call ID und Incident ID wichtig, da die beiden IDs das Herstellen einer Verknüpfung ermöglichen. Diese gelang nur für die Verknüpfung der HPSC –Export File und Call IDs Datei zur Anwendung. Für ein besseres Verständnis dieser Arbeit heißen diese drei Dateien aufbereitete Dateien.

3.3 Ideen zu Suchoptimierungen

Nach der Aufbereitung der Dateien aus 3.2 schließt sich die Entwicklung folgender Ideen für eine Suche in den aufbereiteten Daten an. Zunächst kommt es zur Bearbeitung der Suchmaske, wobei dies eine Analyse und Verarbeitung des eingegebenen Textes beinhaltet. Als nächstes dient der verarbeitete Text der Suche in Bezug auf die aufbereiteten Dateien. Der Service Desk Mitarbeiter gibt einen längeren Text ein; dabei sind nicht alle Begriffe von Bedeutung. Z. B. erschließt sich aus manchen Artikeln nicht sofort, um was es geht, daher schließt sich zunächst ein Herausfiltern aller nicht relevanten Begriffe an; der Abschnitt „Stopp Wörter (Stop Words)“ erläutert dies näher. Des Weiteren kann dem Service Desk Mitarbeiter bei der Eingabe des Textes in der Suchmaske ein Rechtschreibfehler unterlaufen. Erhält der Mitarbeiter keinen Hinweis auf den Fehler, zieht sich dieser über den ganzen Prozess hinweg und bei einer Suche nach dem Text würde dann kein Ergebnis erscheinen. Zur Vermeidung kam es zur Entwicklung der Idee der Rechtschreibkorrektur. Zunächst wird auf die Stopp Wörter eingegangen, auf die Rechtschreibkorrektur, auf die Ermittlung der Bedeutung eines Begriffes und anschließend auf die Suche nach den aufbereiteten Daten.

Stopp Wörter (Stopwords)

Im ersten Schritt erfolgt das Herausfiltern sämtlicher Stopp Wörter, Namen und Sonderzeichen aus dem Text. Stopp Wörter bzw. Stopwords sind z. B. unbestimmte Artikel (ein, eine, einer, einem, eines), für diese Arbeit kommt es zur Herausfilterung aller Wortarten außer Nomen (Windows, Linux, Hardware). Nomen verfügen über wichtige Informationen wie Telefon und Cisco. Z. B. sind Artikeln keine Informationen zu entnehmen. Außerdem

gestaltet sich die Suche schwierig, wenn neben Nomen z. B. Artikel vorkommen, denn die meisten Sätze beinhalten einen Artikel. Damit eine Eingrenzung auf die aufbereiteten Dateien möglich ist, gelangen nur Nomen zur Anwendung. Es kommt zum Schreiben dieser Stopp Wörter in eine Datei (siehe in der beiliegende DVD im Verzeichnis: 2.Software/ModelProject/src/main/recources/GermanDicParts/Stopwoerter) und zu einem Vergleich mit dem jeweiligen Wort aus dem eingegebenen Text und letztlich zum Herausfiltern. Das Gleiche gilt für Personennamen, Satz- sowie Sonderzeichen.

Rechtschreibkorrektur

Für die Rechtschreibkorrektur erfolgte zum einen die Verwendung des Levenshtein – Abstandes und zum anderen des Jaro-Winkler – Abstandes Algorithmus. In der Theorie ist der Damerau Levenshtein Algorithmus effizienter als der Levenshtein – Abstand siehe Kapitel 2.1, dennoch war in der Praxis kein Unterschied festzustellen (siehe Abbildung 22). In der Abbildung 22 ist ebenso die Korrektur von „Windows“, „Linux“ und „Computer“ zu sehen; dabei benötigt der Damerau Levenshtein eine länger Berechnungszeit als der Levenshtein Abstand, weil bei dem Levenshtein Abstand eine zusätzliche Funktion mit integriert ist. Bei einem Abstand größer als n wird die Rechenoperation bei dem aktuellen Wort abgebrochen und mit dem neuen Wort ausgeführt, dabei ist n größer gleich 1.

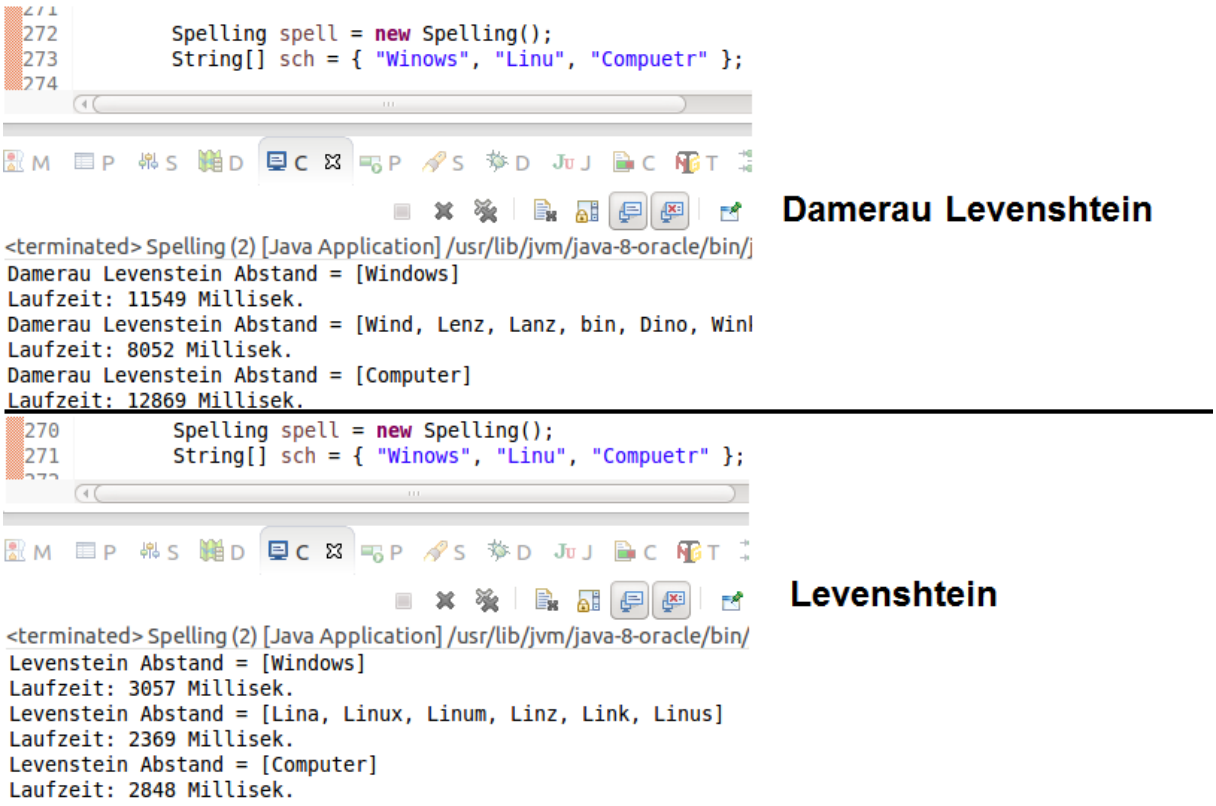

Abbildung 22: Levenshtein und Damerau Levenshtein

Der Jaro-Winkler Abstand Algorithmus grenzt bei einer engeren Auswahl an Begriffen die ähnlichsten ein und optimiert somit die Rechtschreibungsidee. Für die Rechtschreibkorrektur kam es zur Erstellung von vier Dateien mit 1.657481 Millionen deutschen Begriffen (siehe in der beiliegende DVD im Verzeichnis: 2.Software/ModelProject/src/main/recources/GermanDicParts/CompleteGermanDic). Anschließend werden mittels des Levenshtein Abstands[59] Algorithmus die Rechtschreibung überprüft und die ähnlichen Wörter eingegrenzt. Anschließend erfolgt die Verwendung des Jaro Winkler Algorithmus die erneute Eingrenzung der bereits eingegrenzten Begriffe. Dieses Prinzip erhöht die Wahrscheinlichkeit des Vorschlagens des richtigen Wortes dem Mitarbeiter gegenüber. Bei dieser Rechtschreibkorrektur geht es zunächst darum, die aus dem Suchtext über den Service Desk Mitarbeiter eingegebenen Stopp Wörter entsprechend dem Prinzip aus dem Abschnitt „Stopp Wörter“ zu verarbeiten, ansonsten gestaltet sich die Berechnungszeit länger als bei effektiven Algorithmen (siehe Kapitel 2.2).

[59] Dieser wurde in Kapitel 2.1 erklärt

Ermittlung der Bedeutung eines Begriffes und Durchführung der Suche nach den aufbereiteten Daten-

Ein weiteres Problem bei den Service Desk Daten sind neue Begriffe oder Begriffe, die es nicht zu berücksichtigen gilt. Beispielsweise ist Windows-2010 neu auf dem Markt und somit weiß ein System nicht, wie es mit den neuen Begriffen umgehen soll. Nicht mehr verwandte Begriffe wie z. B. Windows-XP wartet Microsoft nicht mehr und Kunden nutzen dies höchstwahrscheinlich nicht mehr. Somit ist der neue Begriff „Windows-2010“ zu berücksichtigen und nicht der verwandte Begriff „Windows-XP“. Um diesem Problem entgegenzuwirken, kam es zur Entwicklung eines nachfolgend beschriebenen Konzepts. Zunächst erfolgt die Untersuchung der aufgearbeiteten Datei (siehe Kapitel 3.1), dabei wird unter der Spalte „update.Action“ geschaut, in welcher Zeile es eine Lösung gibt, z. B. Spalte „update.Action“, Zeile „120“. Die Spalte „Brief Description“ Zeile „120“ nimmt die Beschreibung ein und alle Wörter in einem erstellten vordefinierten Wörterbuch gelangen zur Herausfilterung, somit bleiben unbekannte Begriffe enthalten. Dies geschieht mit allen Zeilen unter der Spalte „Brief Description“, wozu es auch eine Lösung in der „update.Action“ gibt. Es ist auch möglich, alle Zeilen in der Spalte „Brief Description“ zu behandeln, aber der Zeitaufwand, den die Software benötigt, ist im Vergleich zum Nutzen der Datei zu hoch, deshalb unterbleibt dies. Im Fall der Ermittlung der Begriffe ist nachzusehen, wie oft ein Begriff in der HPSC-Export File in all den Zeilen in der Spalte „Brief Description“, wozu es eine Lösung gibt, vorkommt. Es kommt zum Schreiben des Begriffes mit seiner Anzahl in eine neue Datei und zum Speichern auf Hadoop in einen Ordner z. B. „WordCount“ und zur Verarbeitung mit MapReduce. zum besseren Verständnis wird diese Datei „ImportantWordsCount“ genannt. Immer beim Hinzufügen einer neuen HPSC-Export File auf Hadoop erfolgt das Erstellen dieser neuen Datei „ImportantWordsCount“ und das Speichern in dem Ordner „WordCount“ in Hadoop. Somit erhält dieser Ordner immer wieder neue Daten. Die Idee in Bezug auf das Erstellen der Datei „ImportantWordsCount“ findet Erläuterung anhand eines Beispiels. Der Mitarbeiter A gibt den Text „Ich habe ein Problem mit Windows, dieses spinnt schon seit Tagen und Excel funktioniert da nicht und Word auch nicht“ ein. Wie bereits beschrieben, kommt es zum Entfernen von Stopp Wörtern. Somit bleiben die Begriffe „Windows“, „Excel“, „Word“. Jetzt steht ein Vergleich mit der Datei „Important WordsCount“ an, welches der Wörter am häufigsten vorkommt; es folgt ein Sortieren dieser in einer Liste nach der Häufigkeit. Ist ein Wort in der Datei „ImportantWordsCount“ vorhanden, steht dieses an der ersten Stelle, die anderen beiden Wörter behalten ihre Reihenfolge und folgen dem ersten Wort. Die Reihenfolge ist ein wichtiges Kriterium für die Suche, dies ist anhand eines Beispiels zu erklären. Menge A stellt die Datei in Abbildung 23 dar, in der die Suche stattfindet und in der alle Begriffe enthalten sind. Die Suche in den Dateien erfolgt

mittels des Frameworks Spark. Bei einer rein zufälligen Begriffsauswahl und nach einem Suchen zuerst nach „Excel“ und einem Speichern in den RDD[60] bliebe nur Zeile 2 in der Abbildung 23 übrig. Anschließend erfolgt die Suche nach „Word“ auf demselben RDD. Da „Word “ nicht in der Zeile steht und das Ergebnis dann leer sein würde, wird das letzte Ergebnis, in dem noch etwas steht, angezeigt. Es findet aber eine systematische Suche statt, welches Wort am häufigsten vorkommt. In diesem Fall ist das Wort „Windows“, welches genau dreimal vorkommt, nämlich in der zweiten Zeile, vierten Zeile und als letztes in der fünften Zeile. Durch eine weitere Eingrenzung durch das zweithäufigste Wort „Word“ kommt es zum Anzeigen der wichtigsten Zeile mit der Lösung vier. Dieses Konzept für die Suche dürfte nicht immer funktionieren, da zuerst die Suche nach „Windows“ abläuft und alle Zeilen, die „Windows“ beinhalten, verfügen über keine Lösungen. Somit geht das Konzept nicht immer auf. Es folgt die Beschreibung einer weiteren Überlegung. Es schließt sich die Suche nach der Häufigkeit des Begriffes an. Für jede Suche kommt es zur Erstellung eines eigenen RDDs[61]. Nach der Suche nach allen Begriffen der Reihenfolge geschieht das Zusammenfassen aller RDDs zu einer Menge mit einer sich anschließenden Suche nach der Lösung. Existiert keine Lösung dazu, findet die jeweilige Zeile keine Berücksichtigung. Diese Methode deckt den größten Anteil einer Menge, die die Daten enthält, ab.

	A	B
1	**Beschreibung**	**Lösung**
2	Problem mit Windows und Excel	keine Lösung
3	Problem mit Office	Lösung
4	Window und Word Funktioniert nicht	Lösung
5	Problem mir Windows	keine Lösung

Abbildung 23: Beispielhafte Datei für die Suche

Diese Stelle greift auf, welche Dateien für die Analyse zur Verfügung stehen. Die aufbereiteten Dateien sind: HPSC – Export File, Call IDs – Export File und Call ID und Incident ID. Hinzu kommt die Datei „ImportantWordsCount“. Anhand dieser Dateien und des aufbereiteten Such-Textes des Service Desk Mitarbeiters findet eine Suche in diesen vier Dateien statt. Zur Anzeige einer optimalen Lösung sind mehrere zu berücksichtigende Schritte vonnöten.

1. Die Suche erfolgt unter der Spalte „Brief Description“ in der aufbereiteten HPSC – Export File Datei. Bei einer erfolgreichen Suche des Begriffes wird die aus der Datei Call IDs zu Incident IDs nach dem jeweiligen Call ID gesucht, die zu dieser Incident ID, die in der Datei

[60] Vergleiche Kapitel 2.2 Abschnitt Apache Spark
[61] Siehe kapitel 2.2 Abschnitt Apache Spark

HPSC – Export File ist, gehört anschließend erhält der Mitarbeiter die Call ID, Incident ID und die Lösung falls vorhanden angezeigt.

2. Eine nicht erfolgreiche Suche unter 1. kann vorkommen, da in der Spalte „Brief Description“ nur kurz und bündig das Problem des Kunden steht, woraus sich nicht viele Informationen ableiten lassen. Dann schließt sich eine Suche in der aufbereiteten Datei Call IDs – Export File an in der Spalte „Description“. Ist die Suche erfolgreich, vollzieht sich die Suchen nach der Incident ID. Falls keine Incident ID in der gleichen Zeile wie „Description“ und Call ID vorhanden ist, schließt sich eine Suche in der Datei Call IDs zu Incident IDs zu der jeweiligen Incident ID an, die zu dieser Call ID gehört. Anhand dieser Incident ID geschieht die Suche in der Datei HPSC – Export File gesucht und der Mitarbeiter erhält die Anzeige der Call ID, Incident ID und der Lösung falls vorhanden.

3. Ist weder in der aufbereiteten Datei HPSC – Export File noch in der Call IDs – Export File eine Suche erfolgreich, bekommt der Mitarbeiter nichts angezeigt.

Das Kapitel Design unter „5.3 Funktionen“ erhält eine nähere Erläuterung des eben beschriebenen Verlaufs. Damit der Mitarbeiter eine schnelle Lösung findet, bekommt der Mitarbeiter nur die zehn aktuellen, geschlossenen Calls angezeigt. Beim Anzeigen sämtlicher möglicher Lösungen gibt es im schlimmsten Fall mehr als 50 Ergebnisse. Des Weiteren ist der Mitarbeiter dazu geneigt, alle Lösungen anzuschauen. Deshalb kommt es lediglich zu einer Anzeige der zehn aktuellsten Lösungsmöglichkeiten. Das Datum, wann es zum Schließen des Calls gekommen ist, bestimmt die Aktualität. Der Mitarbeiter erhält Informationen über die Lösung, die Call und Incident ID und welcher Mitarbeiter diesen Call behandelt hat. Falls keine Lösung vorhanden ist, kann sich der Mitarbeiter an den Kollegen wenden, der das gleiche oder ein ähnliches Problem schon einmal behandelt hat.

Eingrenzung durch Kategorien

Des Weiteren ist es möglich, die Suche mittels einer Eingrenzung an Kategorien zu verfeinern. Dabei kommt es zu einer näheren Betrachtung der Dateien HPSC – Export File und Call IDs – Export File. Die Spalten „Category“ und „Sub Category“ sind festgelegte, vom Administrator des Ticketsystems erstellte Kategorien. Dabei sind die Kategorien branchen- und unternehmensspezifisch. Anhand dieser Kategorien erhält der Mitarbeiter die Möglichkeit, die Suche einzugrenzen. Ein Beispiel ist, wenn im Zusammenhang mit Kategorie A und B der Begriff Windows auftritt, könnte anhand einer Eingrenzung über die Kategorien die Definition des Zusammenhangs vonstattengehen. Der Mitarbeiter muss bei der Verwendung von „Category“ und „Sub Category“ folgendes berücksichtigen:

1. Category: nicht verwendet, Sub Category: nicht verwendet
Es ist keine Auswahl zu treffen.

2. Category: verwendet, Sub Category: verwendet.
Es ist die Verwendung beider Kategorien möglich.

3. Category: verwendet, Sub Category: nicht verwendet.
Möglich ist nur die Verwendung von Category; möchte der Mitarbeiter eine noch genauere Eingrenzung haben, kann er auch Sub Category verwenden.

4. Category: nicht verwendet, Sub Category: verwendet.
Hier darf der Mitarbeiter nichts unternehmen, weil die Sub Categories in verschiedenen Categories vorkommen und dadurch keine Gewährleistung einer Eingrenzung besteht. Es gibt die Sub Category C unter der Category A und B. Erfolgt eine Eingrenzung durch Sub Category C, ist das Ergebnis das gleiche, als wenn ohne die Eingrenzung von Sub Category die Suche nach einer Lösung geschieht.

An dieser Stelle ist zu begründen, wieso zum Ersten MapReduce für die Erstellung der Datei „ImportantWordsCount“ Spark für die Suche auf die aufbereitete Datei und Hadoop bzw. das Hadoop Distributed Filesystem Verwendung gefunden hat. MapReduce ist ein Batchverfahren, es ist nicht geeignet, große Datenmengen zu analysieren, aber für die Aufbereitung der Dateien ist MapReduce sinnvoll, da es einmal am Tag zum Anstoßen des MapReduce Jobs kommt oder dementsprechend beim Hinzufügen neuer Daten. Spark ist Im Vergleich zu MapReduce ein effizineteres Framework, da zum einen das In-Memory Konzept verwendet wird und zum anderen ist es nicht nötig aufgrund des lazy Evulation Prinzips die Mapper und Reduce Funktionen effizent zu programmieren (siehe Kapitel 2.2). Des Weiteren gelangt Hadoop synonym zu Big Data zur Anwendung. Einerseits ist die Standard Hardware auf der Hadoop läuft kostengünstiger und andererseits in kleineneren schritten erweiterbar. Der Vorteil von Hadoop im Vergleich zu den klassischen Transaktionalen IT-Systeme, z. B. ERP ist, dass diese IT-Systeme in der Regel nur in der Lage sind strukturierte Daten zu verarbeiten und Hadoop alle Daten verabeiten kann.

4 Systemarchitektur

Wie bereits in Kapitel 3.1 erwähnt, verwendet diese Arbeit eine Hardware, Hadoop als Single Node. Dennoch ist es möglich, das hier aufgebaute System mit weiteren Slave Nodes zu verknüpfen.

Systemarchitektur

Für diese Arbeit ist die Systemarchitektur für einen Cluster aus „n“ Slaves und einen Master ausgelegt. Die Verknüpfung mit weiteren Slave Nodes ist möglich, wobei n die Anzahl der Slaves darstellt. Kapitel 2.2 hat die Architekturen von Hadoop, MapReduce und Spark bereits behandelt, deshalb ist ihre Behandlung hier zu vernachlässigen. Abbildung 24 stellt die Systemarchitektur für diese Arbeit dar. Der Client stellt den Mitarbeiter des Service Desks bzw. den Administrator dar mit dem Agieren über das Web-Interface. Das Feld Hadoop in Abbildung 24 stellt die Standard Hardware dar, auf der Hadoop läuft. Zunächst erfolgt die Beschreibung der Systemarchitektur aus der Sicht des Administrators und anschließend aus der Sicht des Service Desk Mitarbeiters im Hinblick auf die Suche. Der Administrator kann neue Dateien auf dem Hadoop hinzufügen, löschen, umbenennen und den Ordner-Pfad ändern. Dabei gelangt das HDFS zur Anwendung (siehe Kapitel 2). Beim Hinzufügen neuer Daten auf Hadoop kommt es zum Anstoßen eines MapReduce Jobs, d. h., es geschieht mittels HDFS das Hinzufügen der Datei HPSC – Export File und es erfolgt das Speichern auf Hadoop in einem Verzeichnis (Abbildung 24. Pfeil vom Client in Richtung HDFS über das HDFS Framework). Mittels MapReduce vollzieht sich anschließend das Lesen und Verarbeiten aus dem Verzeichnis (Abbildung 24: Pfeil von HDFS in Richtung des MapReduce Frameworks). Es kommt zu einem Speichern der verarbeiteten Datei in Hadoop in einem anderen Verzeichnis; dies geschieht sowohl mit der Datei Call IDs Export – File, Call IDs zu Incident IDs File und der Datei „ImportantWordsCount“ (siehe Kapitel 3.2) (Abbildung 24: Pfeil von MapReduce Framework in Richtung HDFS). Es erfolgt das Anstoßen des MapReduce Jobs nur beim Hinzufügen einer neuen Datei, beim Löschen und Umbenennen von Dateien unterbleibt dies. Anschließend erhält der Administrator eine Bestätigung, ob dies erfolgreich war oder nicht (Abbildung 24: Pfeil von HDFS Framework in Richtung Client). Gibt der Mitarbeiter vom Service Desk eine Suche auf, wird mittels Spark auf die aufbereiteten Daten in Hadoop zugegriffen und darauf eine Suche durchgeführt (siehe Kapitel 3.2) (Abbildung 24: Pfeil von Client Richtung HDFS über Spark Framework). Das Ergebnis bekommt der Client angezeigt (Abbildung 24: Pfeil von Spark Framework in Richtung Client).

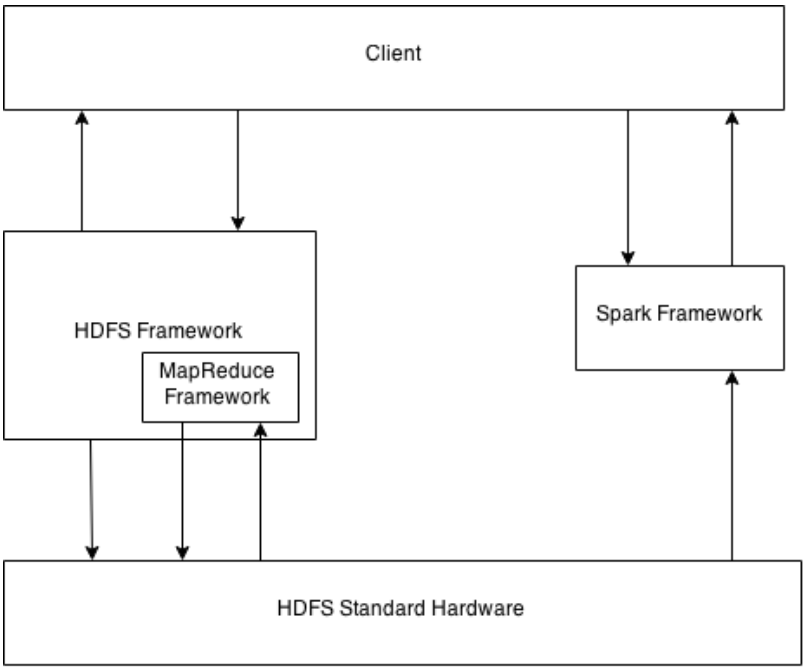

Abbildung 24: Systemarchitektur für diese Arbeit

5 Design

Dieses Kapitel geht zunächst auf die Umsetzung der Software ein, es schließen sich die Tests und die dafür verwandten Tools an. Das aktuelle Kapitel setzt Grundkenntnisse der Softwareentwicklung voraus.

5.1 Umsetzung

Für die Umsetzung erfolgt zunächst die Behandlung des Model View Controller Entwurfsmusters[62], es schließen sich das Erstellen und Erklären eines modifizierten Model View Controller Entwurfsmusters an. Weiter folgt die Betrachtung der Projekt Struktur mit der Vorstellung der einzelnen Projekte. Das Unterkapitel Klassen (Funktionen) stellt zwei Klassen vor und erläutert den Verlauf dieser anhand von Sequenzdiagrammen. Des Weiteren vollzieht sich eine Charakterisierung des Aufbaus der Web-Interfaces.

[62] Vgl.: **Literatur: [12]** S. 3

5.1.1 Modifizierte Model View Controller Architektur

Die „Gang of Four“ [63] hat die Model View Controller Architektur entwickelt, die dazu dient, eine lose Kopplung zwischen der Logik des Algorithmus und der View des Web-Interface zu trennen, worauf der Service Desk Mitarbeiter seine Suche eingeben und der Administrator die Daten bearbeiten kann. Gerade bei Web Anwendungen ist es besser, diese zu trennen, denn so brauchen sich Entwickler über den Aufbau keine Gedanken zu machen. Auch ermöglicht dies das Wechseln einzelner Komponenten, welches ein großer Vorteil ist. Denn beim Auftreten von neuen Technologien oder bei einer nicht weiter fortgesetzten Wartung eines Frameworks sind nicht alle Klassen zu bearbeiten, sondern es ist lediglich eine Klasse zu ersetzen. Der View stellt in diesem Fall das Web-Interface dar. Dieser beinhaltet nur Komponenten zu Anzeige und Bestätigungen einer Eingabe wie „Buttons“[64]. Auf der einen Seite nimmt die View über Getter-Methoden nur Werte entgegen vom Model und auf der anderen Seite erfolgt das Starten einer Statusabfrage über das Model, ob sich die Werte geändert haben oder nicht. Der Controller nimmt die Bestätigung z. B. über einen „Button“ entgegen und ruft die jeweilige Logik im Model auf und leitet gegebenenfalls vom Nutzer über die View eingegebenen Werte an das Model weiter. Der Controller ist auch für die Änderung des Web-Interface zuständig. Möchte der Nutzer auf eine andere Seite gehen, klickt er z. B. auf einen „Button“. Der zuständige Controller erhält eine Mitteilung darüber und steuert die jeweilige, dann angezeigte View an. Das Model enthält die Logik, diese ist getrennt von Model und Controller. Das Model bearbeitet Dinge wie auf die Datenbank schreiben oder lesen. Der Controller kontrolliert somit den Datenfluss, der sich leicht erweitern lässt.[65]

Es ist möglich, aus dem eben beschriebenen Model View Controller Konzepte für das eigene System anzupassen. Das Model View Controller, welches bekannt ist von „Gang of Four“, ist in Abbildung 24 dargestellt. Die Abbildung lässt sich anhand eines Beispiels aus dem Bankensektor gut erklären. Bei der Bank gilt es, die Aktien immer auf dem aktuellsten Stand anzuzeigen. Zunächst meldet sich der Kunde an, das Sender und Verifizieren der Kundendaten geschieht über den Controller zum Model. Hat der Kunde Zugriff, sieht dieser die View. Die View in der Abbildung 25 stellt die die angezeigten Aktienkurse auf der Webseite dar. Die View fragt bei dem Model nach dem Status, ob neue Aktienkurse vorhanden sind. Falls neue Aktienkurse beim Model eingegangen sind, holt sich die View die neuen Daten über das Model. Sind neue Daten beim Model eingetroffen, aktualisiert sich die View. In dieser

---

[63] Vgl.: **Literatur: [12]**
[64] Ein Button ist ein Feld auf dem Web-Interface, durch Klicken auf dem Button wird ein Event ausgelöst.
[65] Vgl.: **Literatur: [19]** S. 107 u. 108

Arbeit erfolgt eine Anpassung des Model View Controller Konzeptes. Denn der Service Desk Mitarbeiter gibt über das Web Interface den Text ein und erwartet ein Ergebnis, dessen Anzeige er erhält, falls eins existiert. Der Unterschied zum eigentlichen Model View Controller ist, dass in diesem Fall erst dann ein Ergebnis erscheinen soll, wenn der Service Desk Mitarbeiter eine Anfrage startet. Es folgt für diesen Fall in Abbildung 26 die Darstellung eines modifizierten MVC Konzeptes mit einer nachfolgenden Beschreibung.

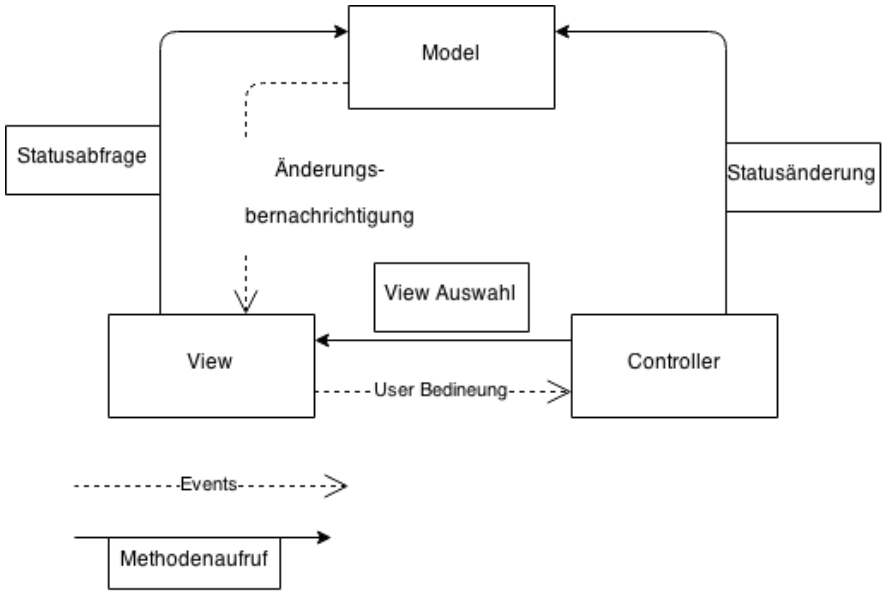

Abbildung 25: Model View Controller von "Gang of Four"

Die View stellt die Oberfläche für den Service Desk Mitarbeiter und Administrator dar. Beim Bedienen des Web-Interfaces, z. B. über einen „Button“, kommt es zum Auslösen eines Events mit einem Aufruf des jeweiligen Controllers. Der jeweilige Controller ruft dabei eine bestimmte Klasse und die Methode auf, die das Model beinhaltet. Das Model verarbeitet seine Aufgabe auf dem Hadoop Distributed File System und gibt ein Ergebnis zurück; dieses Ergebnis erhält der jeweilige Controller zurückgesandt und dieser ruft die jeweilige, anzuzeigende View auf.

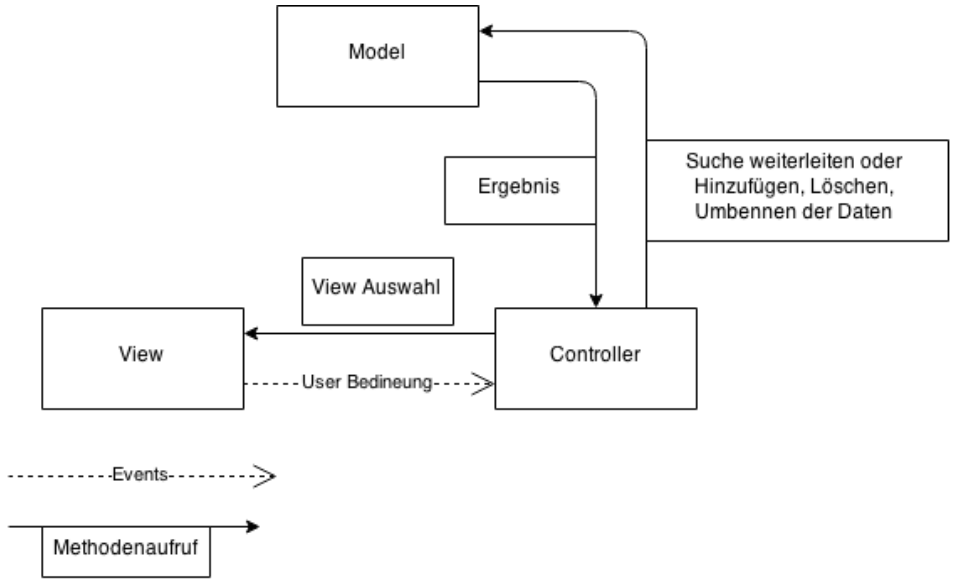

Abbildung 26: Modifiziertes Model View Controller Konzept

5.1.2 Projekt Strukturen

Dieser Abschnitt geht mittels der Abbildungen auf der DVD im Verzeichnis: 1.1 Bachelorarbeit Anhang/Projekt und Klassen UML.pdf näher auf die Struktur ein. Zunächst geht es um das Beschreiben der vier Projekte ModelProject, SparkModul, SearchModul und ViewProject01 und dazu um das Nennen der Packages. Anschließend kommt es anhand des Beispiels von der Eingabe des Service Desk Mitarbeiters über das Web-Interface bis zur Anzeige der Lösung zur Nennung der zu durchlaufenden Packages, Klassen und Projekte.

ModelProject

Das ModelProject beinhaltet die Algorithmen und greift mittels Spark auf Hadoop zu, um die Daten zu lesen. Auch erfolgt das Schreiben der Daten mithilfe von HDFS in Hadoop, mittels MapReduce schließen sich das Bearbeiten und das Ablegen in Hadoop an. In diesem Projekt befinden sich die Packages „algorithmmodel“, „dic“, „googlesearch“, „hdfsmodel“, „loginmodel“, „mapreduce“, „properties“ und „propertiesDictionary“ auf der DVD im Verzeichnis: 1.1 Bachelorarbeit Anhang/Projekt und Klassen UML.pdf in dieser Abbildung unter ModelProject.

SparkModul

In diesem Projekt sind Spark und HDFS implementiert. Für die Implementierung von Spark wurde das Singleton Pattern[66] verwendet. Das Singleton Pattern erzeugt nur ein Objekt dieser Klasse, deren Verwendung immer wieder möglich ist. Dies kam deshalb zur Anwendung, um im Ersten Zugriff auf diese Klasse ein Spark Object zu erstellen, das dann immer wieder zu verwenden ist. Für die Implementierung von HDFS gelangte das Factory Pattern[67] zur Anwendung; damit ist es möglich, weitere Funktion hinzuzufügen. Zurzeit sind die Funktionen Löschen, Einfügen und Umbenennen implementiert. Spark und HDFS sind beide in einem Projekt erstellt, weil Spark auf Hadoop zugreift; dabei müssen die Versionen von Spark und HDFS bzw. Hadoop aufeinander abgestimmt sein (siehe dazu die Website von Spark)[68]. Für diese Arbeit gelangten Spark 1.0.2 und die dazugehörige Hadoop Version 2.2.0 zur Anwendung. (siehe auf der DVD im Verzeichnis: 1.1 Bachelorarbeit Anhang/Projekt und Klassen UML.pdf in dieser Abbildung unter SparkModul).

SearchModul

Dieses Projekt implementiert die Google Suche, die über die Google API zu verwenden ist. Dabei kommt es zum Weiterleiten des verarbeiteten Suchtextes an Google, der Service Desk Mitarbeiter erhält das Angebot einer Hilfe. (siehe auf der DVD im Verzeichnis: 1.1 Bachelorarbeit Anhang/Projekt und Klassen UML.pdf in dieser Abbildung unter SearchModul).

ViewProject01

Dieses Projekt enthält die Views für die Anzeige. Ebenso sind hier die durch ein Event über die View aufgerufenen Controller enthalten. Die Controller rufen dann eine Klasse im Model-Project auf.(siehe auf der DVD im Verzeichnis: 1.1 Bachelorarbeit Anhang/Projekt und Klassen UML.pdf in dieser Abbildung unter ViewProject01).

Jedes dieser vier Projekte gehört zu einem Bereich des in Kapitel 5.1.1 erläuterten modifizierten Model View Controllers. ModelProject, SparkModul und SearchModel gehören dem Model an. In dem ViewProject01 sind das Model und der Controller enthalten.

---

[66] Vgl.: **Literatur: [12]**
[67] Vgl.: **Literatur: [12]**
[68] Vgl.: https://spark.apache.org/downloads.html

Jetzt erfolgt das Behandeln des zu Beginn dieses Unterkapitels beschriebenen Beispiels. Über das ViewProject01 vollzieht sich die Anzeige auf dem Suchfeld der Benutzeroberfläche, in diesem geschieht die Eingabe des Suchtextes, anschließend erfolgt das Klicken auf Enter. In Viewproject01 kommt es im Package zum Aufruf der Klasse „Checkspelling.java“; es findet ein Herausfiltern sämtlicher Stopp Wörter und irrelevanter Wörter statt; es schließt sich die Anwendung der Rechtschreibkorrektur an. Es öffnet sich eine neue Website, auf der die „Category“ und „Sub Category“ vorhanden sind und die Website zeigt auch diese Rechtschreibkorrektur an. Beim Klicken auf Enter tritt erneut ein Event auf und es schließt sich ein Aufruf der Controller Klasse „SearchController.java“ im Package „controller“ an. Über diese Klasse kommt es zum Aufruf der Klasse „HdfsModel.java“ im Projekt ModelProject unter dem Projekt „algorithmmodel“ und der Klasse „MyGooglesearch.java“ im Package „googlesearchmachine“. Es vollzieht sich hier eine Beschränkung auf die Klasse „HdfsModel.java“. Anschließend erfolgt ein Aufruf von Spark, um die Daten von Hadoop auszulesen und der Klasse „WorkOnHdfsData.java“ zu übergeben, dabei erhält die Klasse die Übergabe der „Sub Category“, „Category“ und des Suchtextes, der Service Desk Mitarbeiter bekommt im Anschluss die Lösungen angezeigt.

5.1.3 Klassen (Funktionen)

Dieses Unterkapitel geht näher auf die Klassen und Methoden ein. Die wichtigsten Klassen liegen im ModelProjekt, denn in diesem sind die Logik enthalten bzw. die Algorithmen. Das Kapitel 5.1.2 erläuterte die Packages des Projekts „ModelProjekts“. Die wichtigsten Packages sind „algorithmmodel“ und „dic“. Im Package „algorithmmodel“ sind die Algorithmen implementiert und in diesem findet die Suche nach den verarbeiteten Dateien aus Kapitel 3.2 statt. Das Package „dic“ enthält die Implementierung für die Filterungen und die Rechtschreibkorrektur. Das Package „algorithmmodel“ beinhaltet die Klassen „FilterSentence.java“, „HdfsModel.java“, „ImportancyOfWords.java“, „UsefullyResults“ und „WorkOnHdfsData“. Das Package „dic“ enthält „AllName“, „Filter“, „MyGenericQuciksort“, „Spelling“ und „StopWords“. Es folgt eine kurze Erläuterung der Klassen „allgorithmmodel“ und „dic“, anschließend erfolgen eine Bezugnahme auf die zwei Klassen „WorkOnHdfsData“ und „Spelling“ und eine Definition anhand von Sequenzdiagrammen. Es kam zur Auswahl der beiden Klassen, weil mittels der Klasse „WorkOnHdfsData“ die Suche auf die verarbeiteten Daten aus Kapitel 3.2 geschieht; dies ist ein bedeutender Bestandteil dieser Arbeit. Die Klasse „Spelling“ nimmt die Rechtschreibkorrektur des verarbeiteten Suchtextes des Service Desk Mitarbeiters vor. Es kam zum Einsatz der Rechtschreibkorrektur nach dem Prinzip wie in Kapitel 3.3 unter „Rechtschreibkorrektur“. Die Klassen aus dem Package „algorithmmodel“

rufen die Klassen aus dem Package „dic“ auf, daher findet keine weitere Erläuterung der erwähnten Klasse aus dem Package „dic“ statt.

1. Klasse FilterSentence.java ruft die Klasse Filter.java aus dem „dic“ auf. Diese filtert alle nicht relevanten und Stopp Wörter heraus (siehe Kapitel 3.3 unter Stopp Wörter). Die Filterung findet anhand des vom Service Desk Mitarbeiter eingegebenen Suchtextes statt.

2. Die Klasse „HdfsModel.java“ dient zum Aufruf der Klasse „WorkOnHdfsData“.

3. Die Klasse „ImportancyOfWords.java“ prüft, wie häufig ein Begriff in der Datei „ImportancyOfWordsCount“ vorkommt und ermittelt anhand der Durchführung nach dem erläuterten Prinzip aus Kapitel 3.3 unter „Wichtigkeit eines Begriffes ermitteln und Suche auf die aufbereiteten Daten“.

4. Die Klasse „UsefullyResults“ nimmt die Ergebnisse der Klasse „WorkOnHdfsData“ entgegen und filtert aus diesen Ergebnissen die wichtigsten Daten heraus. Die wichtigen Daten sind in den Anforderungen in Kapitel 3.1 erläutert.

5. Die Klasse „WorkOnHdfsData“ findet in diesem Kapitel unter dem Abschnitt „Sequenzdiagramm der Klasse WorkOnHdfsData“ nähere Erläuterung.

Das Package „dic“ enthält die Klassen:

1. „MyGenericQuicksort.java“ enthält den Sortieralgorithmus „Quicksort“[69].

2. „Spelling.java“ enthält in diesem Kapitel unter dem Abschnitt „Sequenzdiagramm der Klasse Spelling“ eine detaillierte Erläuterung.

Außerdem kommt es zur Erläuterung der Klassen „WorkOnHdfsData“ und „Spelling“ anhand von Sequenzdiagrammen; dabei stellt die Abbildungen auf der DVD im Verzeichnis: 1.1 Bachelorarbeit Anhang/ Sequenzdiagramm WorkOnHDFSData.pdf „WorkOnHdfsData“ und auf der DVD im Verzeichnis: 1.1 Bachelorarbeit Anhang/Sequenzdiagramm Spelling.pdf „Spelling“ den Verlauf in Sequenzdiagrammen dar. Zunächst erfolgt die Behandlung des Sequenzdiagramms aus der Abbildung auf der DVD im Verzeichnis: 1.1 Bachelorarbeit Anhang/Sequenzdiagramm WorkOnHDFSData.pdf von „WorkOnHdfsData“, anschließend die von „Spelling“.

[69] Quicksort ist ein Sortieralgorithmus siehe Vorlesung Softwareentwicklung Prof. Dr. Henning Dierks HAW Hamburg Department TI

Sequenzdiagramm der Klasse „WorkOnHdfsData“

Als Erstes sind über den Konstruktor die aufbereiteten Dateien Call IDs – Export File, HPSC – Export File, Call ID zu Incident ID, „ImportantWordsCount“, aufbereiteter Suchtext, „Category“ und „Sub Category“ als Parameter an „WorkOnHdfsData“ zu übergeben. Anschließend ist die Methode getResult() vom Entwickler aufzurufen. Zunächst kommt es in der Methode getResult() zum Aufruf der Klasse „ImportancyOfWords“. Diese Klasse enthält zum einen die Übergabe des verarbeiteten Suchtextes und danach die der Datei „ImportantWordsCount“. Für den Rückgabewert dieser Klasse gibt es drei Fälle:

1. Lassen sich alle Begriffe im Suchtext finden, geschieht eine Sortierung und Rückgabe entsprechend der Bedeutung der Begriffe. Die Sortierung und Rückgabe eines Begriffes ist auch möglich beim Finden desselben in einem anderen, Beispiel: Suchbegriff Haus. In „ImportantWordsCount“ sind die Begriffe Haus und Haushalt vorhanden. Es vollzieht sich eine Rückgabe der beiden Begriffe, da im Haushalt der Begriff Haus enthalten ist.

2. Kein Begriff aus dem Suchtext war in der Datei „ImportancyOfWordsCount“ zu finden; es erfolgt die Rückgabe einer Leerliste ohne Begriffe.

3. Einige Begriffe waren zu finden und einige nicht. Gefundene Begriffe erhielten eine Sortierung in der Liste sortiert, hinsichtlich der nicht gefundenen Begriffen unterblieb das Hinzufügen in die Liste.

Anschließend kommt es zum Aufruf der Methode getValidList(), die in „WorkOnHdfsData“ implementiert ist. Diese Methode prüft zunächst, welcher der drei gerade beschriebenen Fälle auftritt. Beim ersten Fall geschieht eine unveränderte Rückgabe der Liste. In Fall zwei erfolgt eine unsortierte Rückgabe der Begriffe aus dem Suchtext. Tritt Fall drei ein, ergibt sich eine unsortierte Hinzufügung der nicht in der Datei „ImportancyOfWordsCount“ vorhandenen Begriffe aus dem Suchtext am Ende der sortierten Begriffe.

Anschließend kommt es zur Suche in den verarbeiteten Dateien Call IDs – Export File, HPSC – Export File und Call IDs zu Incident IDs. Dabei ist nach dem Prinzip aus Kapitel 3.3 unter dem Abschnitt „Wichtigkeit eines Begriffes ermitteln und Suche auf die aufbereiteten Daten durchführen“ vorzugehen. Der Rückgabewert dieser Methode ist eine Liste in folgender Form [Incident ID, Call ID, Date, Status des Calls, Lösung]. Auch kommt es zur Erläuterung des Rückgabewertes in Kapitel 3.3 unter demselben Abschnitt.

Sequenzdiagramm der Klasse „Spelling“

In der Klasse „Spelling“ vollzieht sich zum einen die Übergabe der von Spark gelesenen Datei und zum anderen die des Suchtextes des Service Desk Mitarbeiters. Die gelesene Datei

von Spark umfasst 1.657.485 Millionen deutsche Begriffe. Vor der Übergabe des Suchtextes vom Service Desk Mitarbeiter an die Klasse „Spelling“ kommt es zur Verarbeitung der Stopp Wörter, Sonderzeichen, Satzzeichen und irrelevanten Begriffe nach dem Prinzip aus Kapitel 3.3 unter „Stopp Wörter“ und anschließend zur Übergabe an die Klasse „Spelling“. Mittels der Methode getMapOfCheckedSpelling(String sentence) geschieht die Korrektur der Begriffe aus dem verarbeiteten Suchtext auf Rechtschreibung hin; dabei erhält diese Methode die Übergabe des verarbeiteten Suchtextes. Des Weiteren erfolgt das Vorgehen dieser Methode nach dem Prinzip aus Kapitel 3.3 unter dem Abschnitt „Rechtschreibkorrektur“. Eine Erklärung schließt sich an dieser Stelle in zwei Phasen an.

Phase 1. Der Levenshtein-Abstand wählt aus den 1. 657. 485 Millionen Begriffen alle die Begriffe aus, deren Abstand nach Levenshtein kleiner gleich zwei ist.

Phase 2. Der JaroWinkler-Abstand engt mittels der ausgewählten Begriffe aus Phase 1 die Auswahl ein, dabei wird der kleinste Abstand zu eins genommen. Haben mehrere Begriffe den kleinsten Abstand, kommt es zur Rückgabe aller dieser Begriffe.

Hat ein Begriff in der Phase 1 den Abstand eins, so geschieht dessen Rückgabe und weitere Berechnungen unterbleiben. Die Methode gibt den zu korrigierenden Begriff im Zusammenhang mit den korrigierten Begriffen zurück z. B. den zu korrigierenden Begriff Hau -> die korrigierten Begriffe Haus, Maus.

5.2 Verwendete Tools für das Testen und Funktionale Tests

Für das Testen von Java Code gibt es zwei bekannte Frameworks für Unit[70] Tests, einerseits Junit und andererseits TestNg. TestNG ist eine Erweiterung von Junit. In dieser Arbeit kommt es zur Verwendung der aktuelleren TestNG. Hinzu gibt es mehrere Testarten wie „Integrations testing“, „Last test“, „White Box Testing“, „Black Box testing“, „Komponenten Testing“ etc., um die komplette Software auf Herz und Nieren zu testen. Das TestNG Framework ist dazu gedacht, Unit Tests zu erstellen. Zu dem Unit Test gehört u. a. „White Box Testing“, „Black Box testing“, „Komponenten Testing“. Dieser Abschnitt geht auf „White Box testing“, „Black Box testing“ und „Komponententests“ ein. Zunächst erfolgt eine Bezugnahme auf „White Box Testing“, dann auf „Black Box Testing“ und anschließend auf „Komponenten Testing“. Beim „White Box Test“ kennt der Tester den Code der Klasse und versucht eine 100-prozentige Abdeckung zu erreichen. Im Vergleich zum „White Box Testing“ kennt der Tester beim „Black Box Testing “ den Code nicht und testet hier die zu erfüllenden

[70] Sind Tests für einzelne Klassen siehe Vorlesung Softwareentwicklung Prof. Dr. Henning Dierks HAW Hamburg Department TI

Anforderungen. Ein Beispiel ist: Die Anforderung besagt, bei einer Eingabe X in die zu übergebende Methode in der Klasse wird die Ausgabe Y z. B. auf der Konsole erwartet. Bei einem „Komponenten Test“ vollzieht sich dieser klassenübergreifend. Mehrere Klassen sind mittels eines Tests abzudecken. Dabei muss sich der Tester an die Anforderungen halten wie beim „Black Box testing“. Ein Beispiel verdeutlicht den „Komponenten Test“. Gegeben sei die Klasse A mit der Methode A1 und Klasse B mit der Methode B1. Die Methode A1 erwartet einen Wert X und als Rückgabeparameter gibt diese Y zurück. Die Methode B1 erwartet einen Wert Y von der Methode A1 und gibt als Rückgabeparameter den Wert Z zurück. Der Tester programmiert diesen Verlauf und übergibt der Methode A1 den Wert Y und schaut, ob die Methode B1 den Wert Z auch zurückgibt. Es ist empfehlenswert, die Reihenfolge der Testarten einzuhalten. Geschieht z. B. der Komponenten Test vor dem Black Box und dem White Box Test und es tritt ein Fehler auf, dann ist für den Tester sehr schwer nachzuvollziehen, woher der Fehler stammt. Kommt es andernfalls zunächst zur Durchführung des White Box Tests, lässt sich der Fehler schneller identifizieren. Ein weiteres eingesetztes Hilfsmittel zum Erreichen einer 100-prozentigen Testabdeckung ist das Tool Eclemma. Dies wird vom Eclipse Market Place heruntergeladen und in Eclipse eingebunden. Die Rede war mehrmals von einer 100-prozentigen Test Abdeckung. Eine Testabdeckung besteht, wenn es zum Aufrufen der Testmethode der jeweiligen Methoden kommt und ein Durchlauf der Zeile in der Methode vonstattengeht. Von einer 100-prozentigen Testabdeckung ist erst die Rede, wenn ein Durchlauf aller Zeilen der zu testenden Methode erfolgt. Das Tool Eclemma unterstützt den Tester bei einer 100-prozentigen Testabdeckung, indem es zur Anzeige der jeweiligen Test Abdeckung in Eclemma kommt, es erfolgt auch die Anzeige der in der zu testenden Methode durchlaufenen Zeilen. In der Abbildung 27 ist ein Ausdruck abgebildet, wie eine Testabdeckung aussehen kann. Der Punkt 1 in der Abbildung 27 gibt die abgedeckte Prozentzahl an. Die Prozentzahl ist abhängig von der Sichtweise. Kommt es in diesem Beispiel zum Schauen auf die Klasse “MyFenericQuciksort.java“, beträgt die Abdeckung 100 Prozent. Geschieht die Sichtweise aus der Package Sicht, vollzieht sich das Schauen in Bezug auf die Abdeckung über alle Klassen. Punkt 2 gibt die Anzahl der abgedeckten Zeilen an. Punkt 3 zeigt an, wie viele Zeilen keine Abdeckung erhalten haben, und Punkt 4, wie viele Zeilen diese Klasse bzw. dieses Package enthält. Das Doppelklicken auf den Namen der Klasse öffnet diese und es ist zu sehen, was alles abgedeckt worden ist. Ist die Zeile rot in der Klasse, kam es nicht zu ihrer Abdeckung, ist die Zeile grün, geschah eine Abdeckung zu 100 Prozent; ist die Zeile gelb, existiert keine komplette Abdeckung. Das heißt, erfolgt bei einer If(); Abfrage nicht die Abdeckung aller Bedingungen, erfolgt das Anzeigen dieser in Gelb und der Tester weiß, es fand nur zu einem Teil der Bedingung eine Abdeckung statt.

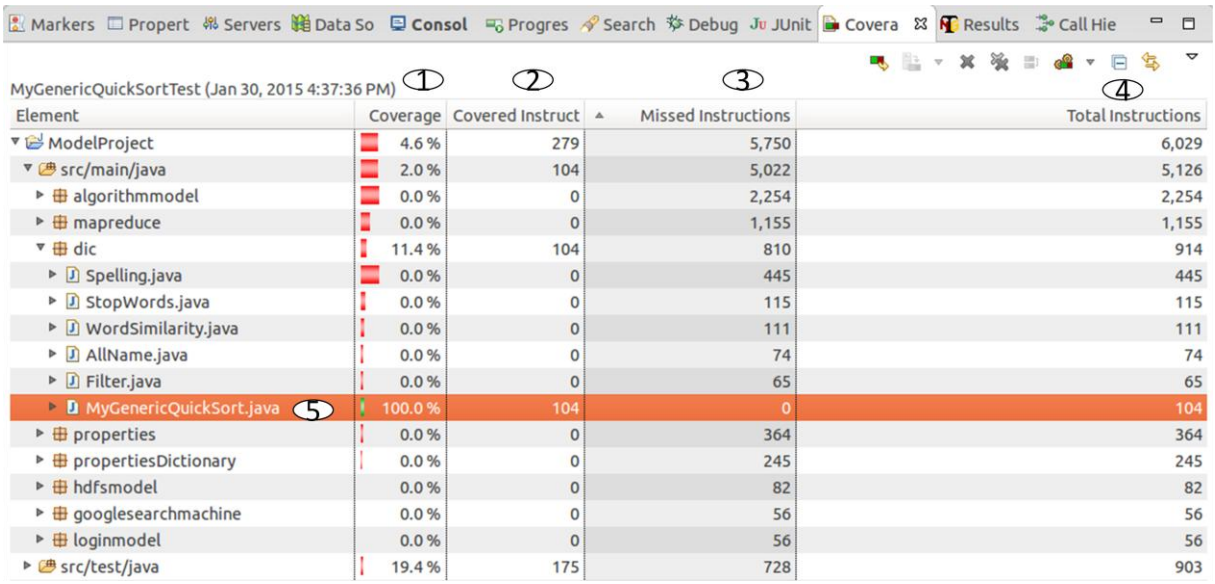

Abbildung 27: Beispiel einer Testabdeckung mit Eclemma

Funktionale Tests

Funktionale Tests sind Tests, die das Einhalten der Anforderungen überprüfen. Eine Anforderung an die Software hier zu nennen wäre, dass die Methode „WorkOnHdfsData“, die im „ModelProjekt“ unter dem „Package“ „algorithmmodel“ liegt, die Dateien HPSC – Export File (siehe Kapitel 3.2), Call IDs – Export File (siehe Kapitel 3.2), die Datei Call IDs zu Incident IDs (siehe Kapitel 3.2) und die erstellte Datei „ImportantWordsCount“ (siehe Kapitel 3.3) in aufbereiteter Form vom Spark gelesen und an WorkONHdfsData übergeben wird. Hinzu kommen die Parameter „Category“, „Sub Category“ und der analysierte und ausgefilterte Suchtext des Service Desk Mitarbeiters, die an WorkOnHDfsData übergeben werden. Infolgedessen sind Lösungsvorschläge in Form von „Incident ID, Call ID, Date, Status, update.Action“ zurückzugeben. Um das Erfüllen dieser Anforderung zu prüfen, sind Tests zu schreiben. Kommt es nicht zur Erfüllung dieser Anforderung, weil ein Entwickler die Klasse WorkOnHdfsData ändert, empfiehlt es sich, ihn infolge des fehlgeschlagen Tests aufmerksam zu machen; somit sollte der Entwickler die Anforderungen und den Test anpassen. Für den Aufbau eines Tests für die eben beschriebene Anforderung sind folgende Überlegungen und Schritte durchzuführen.

1. Eine Person, die über kein Wissen über Programmierung, aber über Fachwissen verfügt, sollte in der Lage sein, diesen Test mit verschiedenen Testfällen durchführen zu können. Daher erfolgt die Übergabe der Attribute, die die Klasse WorkOnHdfsData.java benötigt, über eine Datei. Deshalb geschieht das Ablegen der Dateien HPSC – Export File (siehe Kapitel

3.2), Call IDs – Export File (siehe Kapitel 3.2), der Datei Call IDs zu Incident IDs (siehe Kapitel 3.2) und der erstellten Datei „ImportantWordsCount“ (siehe Kapitel 3.3) in abgekürzter und aufbereiteter Form im Verzeichnis src/main/resources durch das ModelProjekt. Ebenso kommt es für die Parameter „Category“, „Sub Category“ und den ausgefilterten Suchtext zur Erstellung einer Datei mit dem Ablegen im gleichen Verzeichnis im ModelProjekt wie die anderen Dateien. Somit kann der Tester die Attribute über die Dateien ändern und verschiedene Testfälle durchführen.

2. Es gilt, eine Datei zu erstellen, die die zu erwartenden Lösungen beinhaltet. Diese Datei dient zum Vergleich der Ergebnisse, die die Klasse WorkOnHdfsData zurückgibt.

3. Als Nächstes folgt das Erstellen einer Testklasse mithilfe von mehreren Methoden. Die ersten Methoden greifen auf die im Verzeichnis src/main/resources im ModelProjekt abgelegten Dateien mittels Spark zu. Anschließend vollzieht sich das Erstellen der Testmethode, diese wird mit @Test annotiert, in dieser Methode werden die Ergebnisse verglichen.

Dies sind die vorzunehmenden Maßnahmen, um einer Person, die über keine Programmierkenntnisse verfügt, das Testen zu ermöglichen.

6 Anwendung

Dieses Kapitel beschreibt, wie der Nutzer die entwickelte Anwendung anwenden kann. Grafische Abbildungen verdeutlichen die Möglichkeiten des Anwenders. Es gibt zwei Anwendungsarten für diese Anwendung, zum einen gibt es den nach einer Lösung suchenden Service Desk Mitarbeiter, hier Client genannt, und zum anderen den Administrator, der die Daten pflegt, hier Admin genannt. Zunächst geschieht das Beschreiben der Anwendung aus Sicht des Service Desk Mitarbeiter, anschließend aus Sicht des Administrators. Bei einer Auswahl über die Benutzeroberfläche erfolgt die Bestätigung durch „Submit Query“.

Client

Die Abbildung 28 ist der Startpunkt der Anwendung, an dieser Stelle erhält der Anwender die Aufforderung, sich für die Rolle als Client oder Admin auszuwählen. Dabei muss der Anwender nach der Auswahl auf den Button „Submit Query“ klicken.

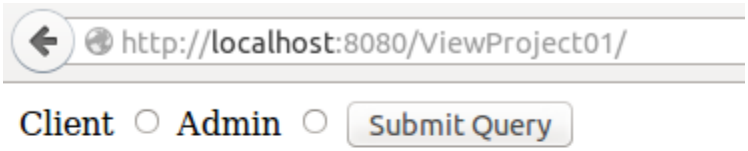

Abbildung 28: Client und Admin Auswahl Oberfläche

Durch das Klicken auf „Sub Query“ in Abbildung 28 erscheint ein neues Fenster (siehe Abbildung 29). Diese Abbildung fordert den Client auf, sich anzumelden und anschließend auf „submit Query“ zu klicken. War die Anmeldung erfolgreich, kommt es zur Weiterleitung des Client auf das nächste Fenster, war die Anmeldung nicht erfolgreich, erscheint eine Fehlermeldung (siehe Abbildung 30).

Abbildung 29: Login für Client

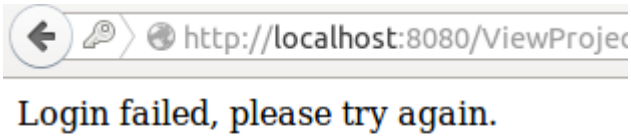

Abbildung 30: Fehlermeldung bei falschem Login

Bei einer erfolgreichen Anmeldung gibt es für den Client eine Weiterleitung zum nächsten Fenster (siehe Abbildung 31). Im Feld Freitext erhält dieser die Aufforderung, seine Suche aufzugeben. Diese Suche ist mit dem Button „Submit Query“ zu bestätigen. Hierdurch erfolgt

zum einen die Filterung des Suchtextes, beschrieben in Kapitel 3.3 unter Stopp Wörter, und zum anderen der Rechtschreibkorrektur, beschrieben in Kapitel 3.3 unter Rechtschreibkorrektur.

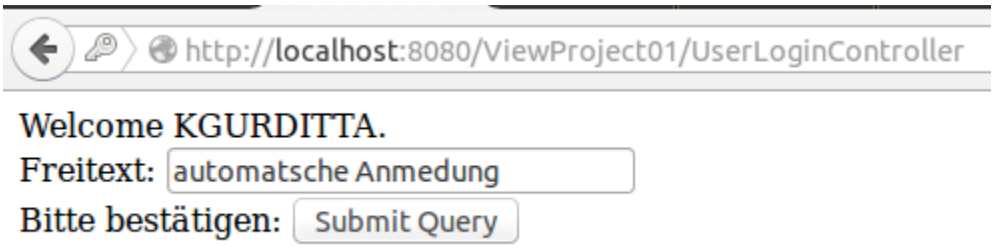

Abbildung 31: Sucheingabe vom Client

In der Abbildung 32 sind Links unter Punkt 1 die „Categories“ und „Sub Categories“, erläutert in Kapitel 3.3 unter Eingrenzung durch Kategorien. In Punkt 2 ist die Rechtschreibkorrektur zu sehen. Die Korrektur muss der Client selbst vornehmen, dabei kommt es zum Anzeigen aller relevanten Begriffe unter Punkt 3; diese kann der Client bearbeiten. Anschließend mit dem Button „Submit Query“ geschieht das Bestätigen der Suche. Durch die Bestätigung erfolgt die Suche mittels Spark (siehe Kapitel 3.3 unter Wichtigkeit eines Begriffes ermitteln und Suche auf die aufbereiteten Daten durchführen) auf die aufbereiteten Dateien aus Kapitel 3.2.

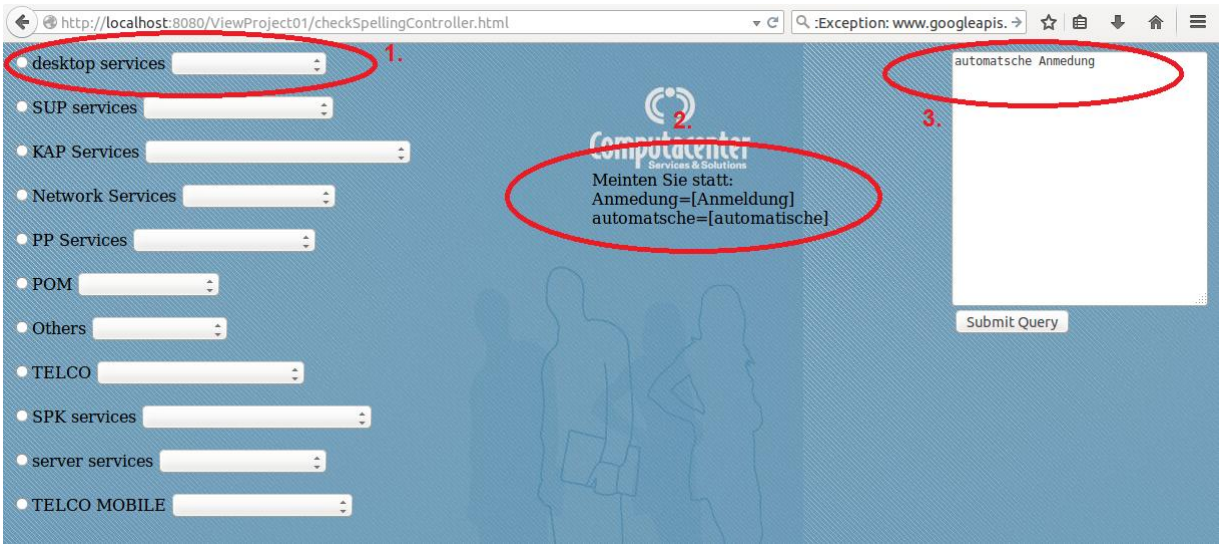

Abbildung 32: Rechtschreibkorrektur und Category sowie Sub Category Auswahl

Nach der Suche auf die aufbereiteten Dateien erfolgt die Darstellung des Ergebnisses wie in Abbildung 33. Links in Punkt 1 erhält der Client die dynamische Anzeige der Call- und

Incident-ID auf dem Button. Ist ein Datum näher an dem Tag dran, an dem der Client die Suche aufgibt, steht diese Suche zu dem Datum an erster Stelle. Unter Punkt 2 sieht der Client auf einen Blick, ob es eine Lösung zum Ergebnis gibt oder nicht. „Lösung vorhanden“ besagt, es ist eine Lösung vorhanden, „keine Lösung vorhanden, aber Mitarbeiter“ daraus kann der Client erkennen, es existiert keine Lösung, aber der Mitarbeiter, der dieses Ticket geschlossen hat, ist nach dem Klicken ersichtlich. Des Weiteren gibt es zwei Arten von „Keine Lösung vorhanden“; einerseits gibt es die Möglichkeit, dass nichts gefunden wird, dann erscheint auf dem Button „match not found, match not found“, andererseits gestaltete sich eine erfolgreiche Suche der Call- und Incident-ID, aber keine Lösung, dann stehen auf dem Button die jeweiligen IDs. Es gibt ebenfalls die durch den Button „Suche starten“ erfolgte Google Suche, die die URLs zu den Seiten vorschlägt, bei denen eine Lösung vorhanden sein könnte (siehe Abbildung 34).

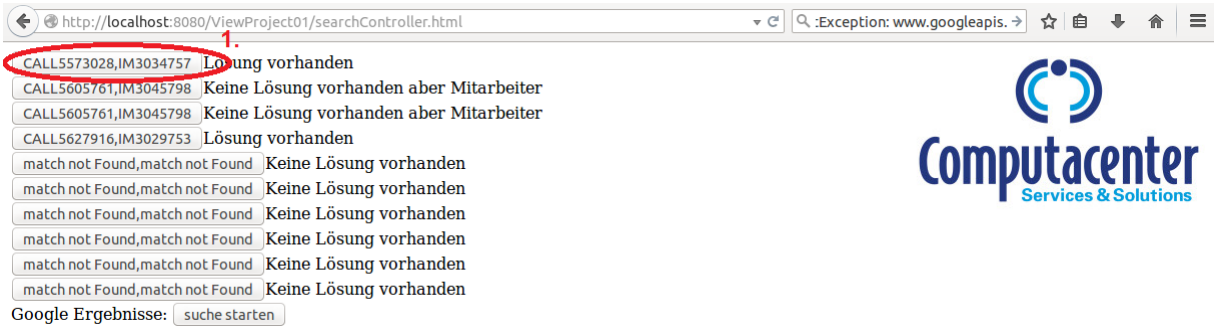

Abbildung 33: Lösungen zu der Suche vom Client

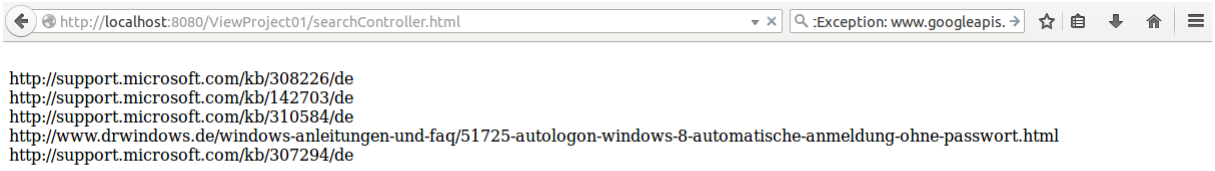

Abbildung 34: Google suche URLs

Beim Klicken auf den Button mit Call- und Incident-ID, erscheint eine Lösung, falls vorhanden, siehe Abbildung 35. Punkt 1 gibt an, an welchem Datum welcher Mitarbeiter diesen Eintrag gemacht hat. Unter Punkt 2 ist der Eintrag des Mitarbeiters zu sehen.

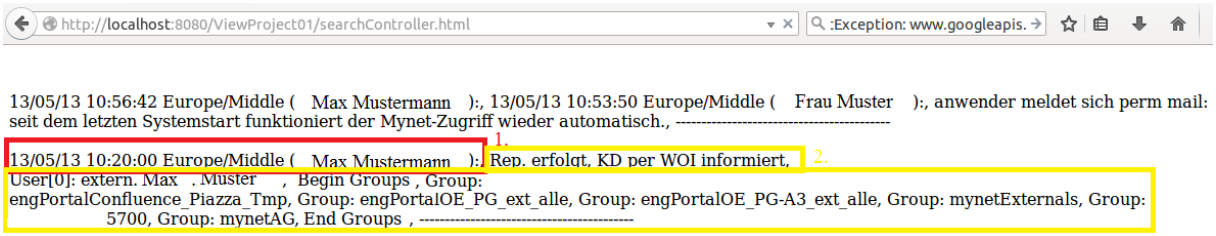

Abbildung 35: Lösungsinhalt

Dies war das Vorgehen für den Client, es folgt nachstehend das Beschreiben des Vorgehens des Admin.

Admin

Zunächst muss der Admin sich anmelden. Das Vorgehen gleicht dem des Client. Im Anschluss zum Login kann der Admin zwischen „Delete File“, Löschen von Dateien, „Add File“, Hinzufügen von Dateien und „Rename File“, Umbenennen von Dateien, auswählen und mit „Submit Query“ bestätigen (siehe Abbildung 36).

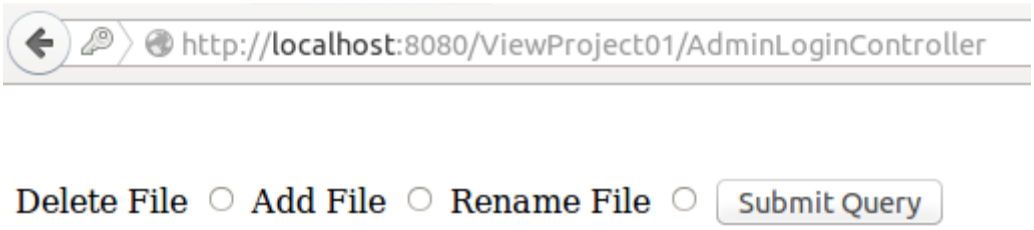

Abbildung 36: Bearbeitungsart der Daten auf Hadoop

Beim „Delete File“ kann der Admin das Verzeichnis der Datei in Hadoop unter Punkt 1 eingeben und somit das Verzeichnis oder die Datei löschen (siehe Abbildung 37).

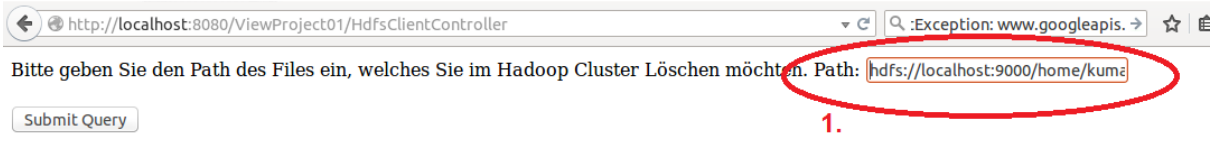

Abbildung 37: Löschen von Daten und Verzeichnisse auf Hadoop

Beim „Add File“ erhält der Admin das Verzeichnis, in dem die Datei abgelegt werden soll, in Abbildung 38 unter Punkt 1 dargestellt. Unter Punkt 2 muss der Admin unter 2.1 das Verzeichnis der Datei angeben, die auf seinem Rechner liegt und unter 2.2 gibt er das Verzeichnis von Hadoop an, wohin er dies hinzufügen möchte; dabei muss er die Verzeichnisse unter Punkt 1 berücksichtigen. Durch Klicken auf „Submit Query“ erfolgt das Schreiben auf Hadoop. Bei Hinzufügen der Dateien kommt es zur automatischen Bearbeitung dieser mittels MapReduce und zur Ablage auf Hadoop. Die bearbeiteten Dateien sehen dann wie in Kapitel 3.2 beschrieben aus.

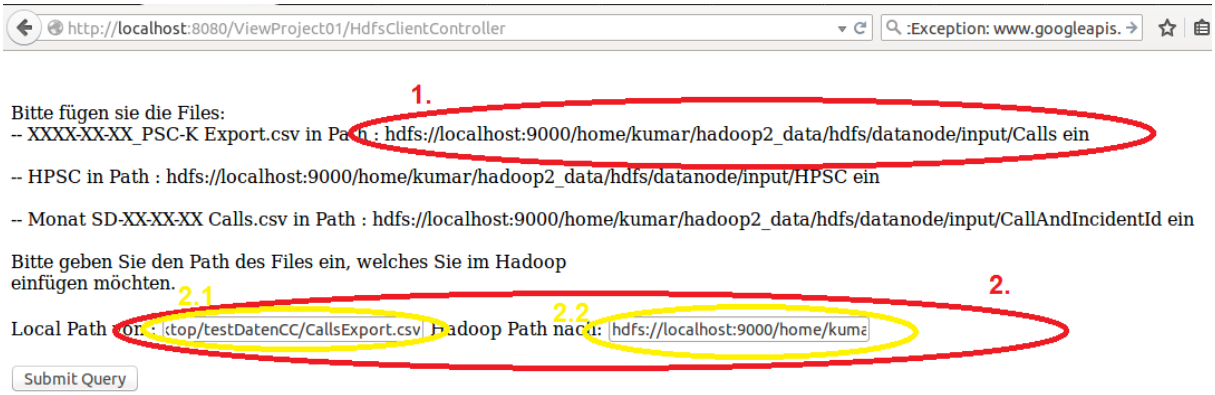

Abbildung 38: Hinzufügen von Dateien auf Hadoop

Bei „Rename File“ kann der Admin die Datei umbenennen; dabei gibt er das Verzeichnis mit dem Dateinamen unter Punkt 1 an, welches existiert, und unter Punkt 2 gibt er das Verzeichnis und den Dateinamen an, in den er die Datei aus Punkt 1 umbenennen möchte (siehe Abbildung 39).

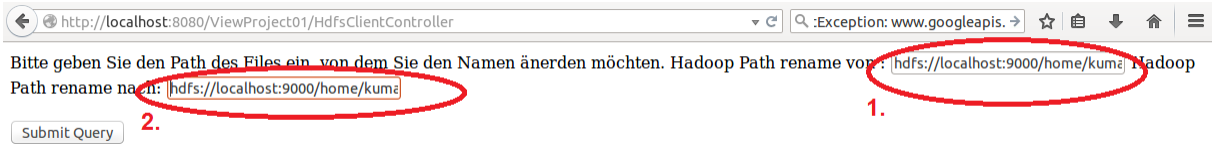

Abbildung 39: Umbenennen von Dateien und Verzeichnisse auf Hadoop

7 Fazit und Ausblick

Zum Abschluss dieser Arbeit folgt ein Fazit, das auf die Ziele eingeht und auch kurz die Herangehensweise beschreibt. Hinzu kommt meine persönliche Stellungnahme, welche Erkenntnisse durch die Bearbeitung der Bachelorarbeit gewonnen werden konnte und was sie mir gebracht hat. Im Anschluss daran beleuchtet der Ausblick eine kritische Auseinanderset-

zung des entworfenen Systems sowie eine Empfehlung für die Verbesserung bzw. Optimierung des Systems. Des Weiteren geht es im Ausblick um Möglichkeiten bzw. Ideen in Bezug auf das Verknüpfen dieses Systems mit einem Ticketsystem.

7.1.1 Fazit

Das Ziel der Arbeit war das Realisieren einer beispielhaften Umsetzung einer Analyse von Service Desk Daten unter Verwendung von Big Data Technologien. Für die Realisierung erfolgte erst einmal das Einlesen in die Big Data Technologien, um ein Grundgerüst aufzubauen. Anschließend ging es um das Vorstellen der Anforderungen, anhand dieser Anforderung kam es zur Analyse der Service Desk Daten. Dabei wurden die Daten einerseits aus dem Ticketsystem sowie dem Reporting Engine vorgestellt und andererseits nach der Durchführung der Analyse als auch der Aufbereitung der Daten entsprechend den Anforderungen präsentiert. Anschließend war die Konzentration auf den eingegebenen Suchtext – normalerweise vom Service Desk Mitarbeiter eingegeben – möglich. Die Herausforderung lag darauf, aus dem Suchtext des Service Desk Mitarbeiters wichtige Informationen herauszufiltern. Dafür kam es zur Herausfilterung unwichtiger Informationen wie Stopp Wörter, Sonderzeichen und Wörter, aus denen kein Nutzen zu ziehen war. In Bezug auf diese Realisierung fand ein selbst erstelltes Wörterbuch im Einsatz. Des Weiteren standen für die Rechtschreibkorrektur Algorithmen bereit, nicht die effizientesten, für den Zeitrahmen aber ganz brauchbar. Das Problem liegt beim Herausfiltern der wichtigen Informationen. Denn es nimmt viel Rechenzeit in Anspruch, aus dem eingegebenen Suchtext die Wörter zu filtern und die Rechtschreibung zu korrigieren. Damit aus den wichtigsten Begriffen aus dem Suchtext die Identifizierung des wichtigsten Begriffes möglich ist, muss die Datei dynamisch sein. Diese Datei findet ihre Abspeicherung in einem Verzeichnis und es erfolgt eine dynamische Erweiterung des Verzeichnisses mithilfe von neuen Dateien. Anschließend geschah auf den Service Desk Daten die Suche nach Lösungen; dafür erfolgte der Einsatz von Spark. Dabei kam es zum Ersatz von MapReduce durch Spark für die Suche. Des Weiteren erfolgte die Darstellung der Systemarchitektur. Das Model View Controller Konzept fand in modifizierter Form Anwendung für die Software. Dabei wurde auch auf Losekopplung geachtet, sodass es einfach möglich ist, den implementierten Algorithmus durch einen neuen zu ersetzen. Des Weiteren dürfen die Tests nicht unerwähnt bleiben; dafür diente der Aufbau eines kleinen Konzepts, um ein kleines Beispiel zu zeigen. Somit kam es zur Entwicklung eines Systems, welches dem Service Desk Mitarbeiter weiterhelfen kann und durch den Entwickler leicht weiterzuentwickeln ist.

Hinsichtlich des persönlichen Fazits erhoffte ich mir vor der Bachelorarbeit, Erfahrungen im Big Data Technologiebereich zu sammeln, da ich großes Interesse hatte, den Umgang mit solchen Daten zu erlernen. Ebenfalls wollte ich meine Programmierkenntnisse erweitern. Außerdem interessierten mich der Aufbau eines Software Projekts und die zu verwendenden Tools und Frameworks.

Die erhofften Ziele wurden erreicht. Ich habe im Rahmen der Bachelorarbeit Erfahrungen mit Big Data Technologien gesammelt, ich bin auch viel in Kontakt mit der Realität gekommen, wie der Stand der Technik ist und wie die Wirtschaft auf große Datenmengen reagiert. Meine Programmierkenntnisse und Kenntnisse über den Aufbau eines Software Projekts konnte ich durch die Umsetzung der Prototypen deutlich verbessern. Die Schwierigkeit lag bei der Konfiguration und Installation von Tomcat und Maven auf Eclipse. Dies erforderte sehr viel Zeit, da ich innerhalb des Studiums keine Konfrontation mit Build-Tools und Servern hatte. Durch den Aufbau des Systems habe ich vieles gelernt über die Tools, mit denen es möglich ist, die Software auch ohne großen Aufwand weiterzuentwickeln. Auch fiel es mir schwer, in einer kurzen Zeit einen effizienten Algorithmus zu verstehen. Dabei habe ich viele Frameworks verwendet und immer wieder ausprobiert, es hat einiges geklappt, aber die befindliche Mathematik hinter der semantischen Analyse bereitete Schwierigkeiten, die in kurzer Zeit nachzuvollziehen.

7.1.2 Ausblick

Bei der anschließenden Betrachtung der Analyse der Service Desk Daten und der Prototypen wird ersichtlich, dass es viele Möglichkeiten gibt, den Prototypen zu erweitern.

An dieser Stelle liegt der Fokus auf der Betrachtung der Schwachstellen dieses Systems und welche Möglichkeiten es gibt, das System zu optimieren, wozu die Zeit für das Erstellen dieser Bachelorarbeit nicht gereicht hat. Anhand der Abbildung 40 werden zu jedem einzelnen Abschnitt ein Optimierungsvorschlag und die Schwachstelle identifiziert. Jeder Abschnitt wird durch eine Zahl repräsentiert.

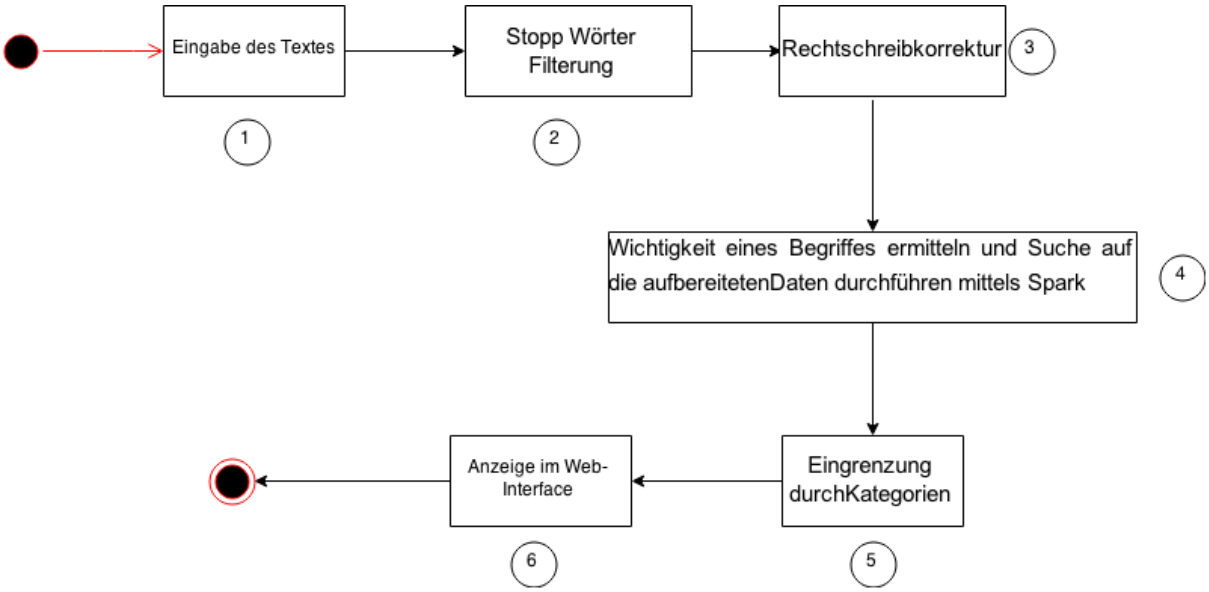

Abbildung 40: Ablauf der Suche

In Punkt 1 ist das Web Interface dargestellt, worüber der Service Desk Mitarbeiter den Suchtext eingeben kann. In Schritt 2 für die Filterung der nicht relevanten Begriffe, Sonder- und Satzzeichen ist anzuraten, einen effizienteren Algorithmus zu verwenden, der anstatt der Nutzung eines Wörterbuches die Durchführung eines Vergleiches unternimmt. Ein interessanter Algorithmus ist der Part-Of-Speech[71], dabei erfolgt die Gruppierung von Wörtern und Satzzeichen zu einer Wortart. Dieser Algorithmus ist sehr aufwendig und fand keine weitere Behandlung, aber hinsichtlich Weiterentwicklung ist dieser eine Option. Eine weitere Möglichkeit besteht darin, einen N-Gramm[72] Algorithmus im Zusammenhang mit dem Framework Lucene zu nutzen. Zunächst gelangte das Framework Lucene zur Anwendung. Es ist möglich, ein Wörterbuch zu nutzen, wie das in dieser Arbeit existierende und es könnte mittels Lucene in ein indexbasiertes Wörterbuch umgewandelt werden. Z. B. ist das Wort „kennen“ im Wörterbuch, es kommt in dem von Lucene[73] erstellten Wörterbuch zur Abspeicherung; die Indexe sind dabei „ke,nn,en“ welche mit dem N-Gramm erstellt werden können. Gehen wir In diesem Fall mit N gleich 3 aus 3-Gramm. Bei der Suche nach dem Wort „kennen“ geschieht das Zerteilen von „kennen“ mittels 3-Gramm in „ke,nn,en“. Es folgt die Betrachtung der Indexe. Zunächst kommt es zum Anzeigen sämtlicher Begriffe mit dem Index „ke“, anschließend zum Schauen, in welchem Index „ke“ und „nn“ vorkommt; somit erfolgt eine Einschränkung der Begriffsauswahl und schließlich wird in den Indexen nach „ke“, „nn“ und „en“ gesucht, wobei eine weitere Einschränkung der Begriffsauswahl vonstattengeht. In der letzten einge-

[71] Vgl.: **Literatur: [30]**
[72] Vgl.: **Literatur: [30]**
[73] Vgl.: **Literatur: [31]**

schränkten Begriffsauswahl geht es um das Suchen und Finden nach dem Wort „kennen“. Im nächsten Schritt unter Punkt 3 geschieht die Verwendung der Rechtschreibkorrektur; dafür ist der Algorithmus relativ gut und es wird auch in kürzerer Zeit berechnet, aber dies gilt nur bei einer geringen Anzahl an Begriffen. Steigt die Anzahl der Begriffe, dann nimmt die Berechnungszeit proportional ebenfalls zu. Die Idee aus Schritt 2 ist auch für die Rechtschreibkorrektur anzuwenden, um eine schnellere Berechnung durchzuführen. Bei den Punkten 5 und 6 sind die Ideen optimal. Außer unter Abschnitt 4 und in Punkt „Wichtigkeit eines Begriffes ermitteln“ ist es möglich, noch Optimierungen mittels der Weiterentwicklung der Klasse „ImpotrancyOfWords.java“ vorzunehmen.

Des Weiteren schwebt mir eine andere Vorgehensweise vor. Die Verwendung eines Ticketsystems live zu nutzen kann einen besseren Einblick gewähren. Es bestand allerdings nicht die Möglichkeit, ein Ticketsystem live zu sehen; dadurch war kein genauer Eindruck zu gewinnen.

Eine weitere Idee für die Zukunft besteht im Falle der Möglichkeit der Verknüpfung meines Systems mit verschiedenen Ticketsystemen. Gemeint ist damit ein Anbinden der Daten verschiedener Ticketsysteme an die Hardware bzw. die Übertragung sämtlicher Daten auf Hadoop; dann bestünde die Möglichkeit, verschiedene Systeme anzuzapfen. Somit ergäben sich wesentlich mehr Daten mit daraus zu gewinnenden Informationen, bei denen sich die Wahrscheinlichkeit erhöht, dass der Service Desk Mitarbeiter bei vielen Daten auf jeden Fall eine Lösung erhält und somit das Ticket so schnell wie möglich schließt.

8 Anhang

8.1 Tabelle Spaltenbeschreibungen der Service Desk Daten

Spaltenname	**Spalten Bedeutung**
opened.by	User des Systems, der den Call eröffnet hat. Format Vorname Nachname
open.time	Zeitstempel, den das System setzt, bei Eröffnung des Calls
open.group	Zuweisungsgruppe, bei der der Call eröffnet wurde
incident.id	Eindeutige ID des Calls (Call Nummer)
cause.code	Referenz (ID) zum Artikel aus dem verwendeten Knowledge Management bei der Eröffnung des Calls verwendet. Wenn kein Artikel verwendet wurde, ist dieses Feld leer.
Status	Bearbeitungsstatus des Calls
contact.name	Personen ID für den der den Call gemeldet hat. Oftmals der Kunde oder ein Ansprechpartner.
Description	Kurzbeschreibung des Calls
problem.type	Kategorie Ebene 3 der Incident Kategorisierung. Wird im Produktkatalog des Ticketsystems festgelegt.
product.type	Kategorie Ebene 4 der Incident Kategorisierung. Wird im Produktkatalog des Ticketsystems festgelegt.
Category	Kategorie Ebene 1 der Incident Kategorisierung. Wird im Produktkatalog des Ticketsystems festgelegt.
subcategory	Kategorie Ebene 2 der Incident Kategorisierung Wird im Produktkatalog des Ticketsystems festgelegt.
Handle Time	Dauer der Bearbeitung des Calls, bis dieser entwe-

	der geschlossen wird oder aus ihm ein Incident entsteht
Solved by Level 1	Flag (true / false), welches anzeigt ob ein Call direkt geschlossen werden (true) konnte oder ein Incident erstellt werden musste
gl.number	Eindeutige ID des Incidents (Incident Nummer), wenn ein Incident aus dem Call erstellt wurde
update.action	Freitextfeld, indem der User seine Tätgikeiten dokumentieren kann
closed.by	Benutzer der das Ticket geschlossen hat als Klarname (Format Vorname Nachname) oder linker, wenn ein zugehöriger Incident geschlossen wurde
Incident ID	Eindeutige ID des Incidents (Incident Nummer)
Update Time	Zeitstempel der bei der Incident Aktualisierung durch das System gesetzt wird
Reopen Time	Zeitstempel der bei der Incident Wiedereröffnung durch das System gesetzt wird
Unsuspend Time	Zeitstempel der durch das System gesetzt wird, wenn der Incident in einen Wartestatus bei der Bearbeitung gesetzt wird. Beispiel: Kunde erhält eine Lösung und muss diese erstmal testen.
Close Time	Zeitstempel der bei der Incident Schließung durch das System gesetzt wird
Status	Interner Ticketsystem Bearbeitungsstatus des Incidents
Incident Status	Bearbeitungsstatus des Incidents mit der Sicht zum Benutzer
Cause Code	Referenz (ID) zum Artikel aus dem verwendeten Knowledge Management bei der Schließung des Incidents verwendet
Initial Cause Code	Referenz (ID) zum Artikel aus dem verwendeten Knowledge Management bei der Eröffnung des Incidents verwendet. Wenn kein Artikel verwendet wurde, ist dieses Feld leer
Brief Description	Kurzbeschreibung des Incidents

Severity Code	Einstufung der Kritikalität des Incidents
Open Group	Zuweisungsgruppe, bei der der Incident eröffnet wurde
Assignment	Aktuelle Zuweisungsgruppe, bei der sich der Incident zum Zeitpunkt des Exportes befindet.
Opened By	User des Systems, der den Incident eröffnet hat als Klarname. Format Vorname Nachname
Last Updated By	User des Systems, der das Incident als letztes aktualisiert hat als Klarname. Format Vorname Nachname
Reported by first name	Kundenvornamen, für den der Incident eröffnet wurde
Reported by last name	Kundennachname, für den der Incident eröffnet wurde
Reported by ID	Personen ID, für den der Incident eröffnet wurde
User Priority	Einstufung der Kritikalität des Melders
Department	Abteilung dem der Kunde, der das Incident gemeldet ist, zugeordnet ist
Call Origin	Typ, wie der Incident eröffnet wurde. Beispiel: Phone, E-Mail
Closed By Group	Zuweisungsgruppe, bei der der Incident geschlossen wurde
audimg.tslmcode	Flag, wenn der Kunde der das Incident gemeldet ein tVIP ist.
eVIP	Flag, wenn der Kunde der das Incident gemeldet

	ein eVIP ist.
Update Action	Freitextfeld, indem der User seine Tätgikeiten dokumentieren kann
Interaction ID	Eindeutige ID des Calls (Call Nummer)
INC Incident ID	Eindeutige ID des Incidents (Incident Nummer)
INC Activity Activity Number	Eindeutige ID der Aktivität
INC Activity Incident ID	Eindeutige ID des Incidents (Incident Nummer)
INC Activity AssignmentGroupFrom	Ist nur gefüllt, wenn der Activity Typ ein Assignment Change ist. Gibt an, von welcher Gruppe das Incident weitergeroutet wurde.
INC Activity AssignmentGroupTo	Ist nur gefüllt, wenn der Activity Typ ein Assignment Change ist. Gibt an, an welche Gruppe das Incident weitergeroutet wurde.
INC Activity StatusChangeFrom	Ist nur gefüllt, wenn der Activity Typ ein Status Change ist. Gibt an, welchen Bearbeitungsstatus der Incident vor Durchführung der Aktivität hatte.
INC Activity StatusChangeTo	Ist nur gefüllt, wenn der Activity Typ ein Status Change ist. Gibt an, in welchen Bearbeitungsstatus das Incident durch die Aktivität gebracht wird.
INC Activity Type	Gibt die Art der Aktitiät vor. Es gibt verschiedene Typen von Aktivitäten. Status Change Zuweisungsgruppen Änderung Aktualisierung des Incidents Schließen des Incidents

INC Activity Date/Time	Zeitstempel, der bei Erstellung des Aktivitätendatensatzes durch das System gesetzt wird

8.2 Quellenverzeichnis

Quelle: [01] A., J. S. (21.11. 20021). *www.Uni-München.de*. Abgerufen am 2015.01.19 von URL: *http://www.cis.uni-muenchen.de/~micha/praesentationen/rechtschreibkorrektur/Levenshtein.html*

Quelle: [02] *Apache.org*. Abgerufen am 08.12. 2014 von URL: http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html&h=385&w=622&tbnid=1ukNzUgp5aF3rM:&zoom=1&tb

Quelle: [03] Bayer, M. (17.08.2012). *Computerwoche*. Abgerufen am 11.11.2014 von URL: http://www.computerwoche.de/a/big-data-die-datenflut-steigt,2500037

Quelle: [04] Borthakur, D. (2007). *The Hadoop Distributed File System: Architecture and Design*. Abgerufen am 15. 12 2014 von The Apache Software Foundation.: URL: https://svn.eu.apache.org/repos/asf/hadoop/common/tags/release-0.16.3/docs/hdfs_design.pdf

Quelle: [05] Buttong, T. (19. 12 2013). *ETHAS*. Abgerufen am 11.11.2014 von http://www.ethas.net/technologie/2013/12/19/big-data-schafft-neue-realitaeten/

Quelle: [06] Dean, J., & Ghemawat, S. (2010). *ACM*. Abgerufen am 13.12.2014 von URL: http://cacm.acm.org/magazines/2010/1/55744-mapreduce-a-flexible-data-processing-tool/fulltext

Quelle: [07] Dr. Hippner, H., & Dipl.-Kfm. Rentzmann, R. (2006). *Geselschaft für Informatik*. Abgerufen am 18. 01. 2015 von URL: https://www.gi.de/service/informatiklexikon/detailansicht/article/text-mining.html

Quelle: [08] Falkenberg, G., Kisker, D. H., & Urbanski, J. (2014). *BITKOM*. Abgerufen am 12. 12. 2014 von URL: http://www.bitkom.org/files/documents/BITKOM_Leitfaden_Big-Data-Technologien-Wissen_fuer_Entscheider_Febr_2014.pdf

Quelle: [09] FH-Köln. *Wikis FH-Köln*. Abgerufen am 15.12.2014 von URL: http://wikis.gm.fh-koeln.de/wiki_db/Datenbanken/HDFS

Quelle: [10] Grimm, R. (17.11.2009). *Linux Magazin*. Abgerufen am 13. 11. 2014 von URL: http://www.linux-magazin.de/Online-Artikel/Funktionale-Programmierung-3-Das-MapReduce-Framework

Quelle: [11] IBM. (07.03.2012). *IBM: Pressemitteilung*. Abgerufen am 11. 11.2014 von URL: http://www-03.ibm.com/press/de/de/pressrelease/37101.wss

Quelle: [12] *HCTL Technologies*. Abgerufen am 12.12. 2014 von URL: http://www.hcltech.com/sites/default/files/big-data-timeline.jpg

Quelle:[13] Klein, D., Tran-Gia, P., & Hartmann, M. (2013). *Gesellschaft für Informatik e.v.* Abgerufen am 11.11.2014 von URL: http://www.gi.de/nc/service/informatiklexikon/detailansicht/article/big-data.html

Quelle: [14] Kollmann, P. D. *Gabler Wirtschaftslexikon*. Abgerufen am 18.12.2014 von URL: http://wirtschaftslexikon.gabler.de/Definition/http.html

Quelle: [15] Lackes, P. D. *Gabler Wirtschaftslexikon*. Abgerufen am 18.12.2014 von URL: http://wirtschaftslexikon.gabler.de/Definition/url.html
Quelle: [16] Lackes, P. D. *Springer Gabler Verlag (Herausgeber), Gabler Wirtschaftslexikon*. Abgerufen am 18.01.2015 von URL: http://wirtschaftslexikon.gabler.de/Archiv/57691/data-mining-v8.html

Quelle: [17] Landua, R. *CERN und LHC* . Abgerufen am 11.12.2014 von Presentation: Daten und Fakten: URL:http://www.teilchenphysik.de/sites/site_teilchenphysik/content/e26/e73816/e74703/e74713/infoboxContent74715/CERN_Intro_061101.pdf

Quelle: [18] NESSI White Paper. (2012). *Nessi-europe* . Abgerufen am 01.08.2015 von URL: http://www.nessi-europe.com/Files/Private/NESSI_WhitePaper_BigData.pdf

Quelle: [19] Nussbaum, B. (13.01.2014). *CeBIT Blog*. Abgerufen am 11.12.2014 von URL: http://blog.cebit.de/2014/01/13/big-data-auto-wird-zur-datenschleuder/

Quelle: [20] Press, G. (21.12.2013). *Forbes Titel: A Very Short History Of Big Data*. Abgerufen am 21.12.2014 von URL: http://www.forbes.com/sites/gilpress/2013/05/09/a-very-short-history-of-big-data/

Quelle: [21] Prof. Klaus Mainzer, T.-M. *VIERNULL*. Abgerufen am 11.12.2014 von URL: http://www.viernull-magazin.de/big-data-und-die-neue-weltordnung/

Quelle: [22] project-consult. *project-consult*. Abgerufen am 18.01.2015 von URL: http://www.project-consult.de/ecm/in_der_diskussion/spiegel_online_28_zettabyte_informationszuwachs_2012

Quelle: [23] REBELLABS. *mastersoft*. Abgerufen am 14.12.2014 von URL: http://mastersoft.diskstation.me/blog/wp-content/uploads/2013/10/the-great-java-app-server-debate.pdf

Quelle: [24] Seiler, U. (14.08.2013). *Codecentric*. Abgerufen am 14.12.2014 von URL: https://blog.codecentric.de/2013/08/einfuhrung-in-hadoop-die-wichtigsten-komponenten-von-hadoop-teil-3-von-5/

Quelle: [25] Speyer, & Junk, M. (21.05.2014). *BITKOM*. Abgerufen am 11.12.2014 von URL: http://www.dhv-speyer.de/Martini/Lehre/SS%202014_V%20105_Junk_Pr%C3%A4sentation.pdf

Quelle: [26] Urbanski, J., & Weber, D. M. (2012). *BITKOM*. Abgerufen am 11.11.2014 von URL: http://www.bitkom.org/files/documents/BITKOM_LF_big_data_2012_online(1).pdf

Quellen: [27] www.1989history.eu.. *Geschichte & Geschichtliches aus aller Welt*. Abgerufen am 20.12.2014 von URL: http://www.1989history.eu/geschichte-computer.html

Quellen: [28] Zicari, R. V. *gotocon (Voerlsung: Goethe Univerity Frankfurt)*. Abgerufen am 12. 11 2014 von URL: http://gotocon.com/dl/goto-aar-2012/slides/RobertoV.Zicari_BigDataChallengesAndOpportunities.pdf

8.3 Literaturverzeichnis

Literatur: [01] al., D. d. (2014). *Hadoop for Dummies.* new Jersey: John Willey & Sons, Inc.

Literatur: [02] Bachmann, K. U. (2010). *Maven 2 - Studentenausgabe - inkl. Bonusmaterial auf CD: Eine Einführung, aktuell zu 2.0.9 (Open Source Library).* Addison-Wesley Verlag.

Literatur: [03] Bachmann, R., Kemper, G., & Gerzer, T. (11. Februar 2014). *Big Data - Fluch oder Segen? Unternehmen im Spiegel gesellschaftlichen Wandels .* mitp.

Literatur: [04] Baron, P. (2013). *Big Data für IT-Entscheider.* München: Carl Hanser Verlag München.

Literatur: [05] Brücher, C. (2013). *Rethink Big Data .* Deutschland: mitp.

Literatur: [06] Damerau, F. (1964). *Technique for Computer Detection and Correction of Spelling Errors. .* Communication of the ACM.

Literatur: [07] deRoos, D., Zikopoulos, P. C., PhD, R. B., Brown, B., & Coss, R. (2014). *Big Data for Dummies.* New Jersey: John Wiley & Sons, Inc.

Literatur: [08] Dumbill, E. (2012). *Big Data Now.* Sebastopol in United States of America: O'Reilly Media, Inc.

Literatur: [09] Eckert, R. (2014). *Business Model Prototyping: Geschäftsmodellentwicklung im Hyperwettbewerb. Strategische Überlegenheit als Ziel.* DE (Düsseldorf): Springer Gabler.

Literatur: [10] Fortmann, H. R. (2006/2007). *Interaktive Trends 2006/2007 - Jahrbuch Deutscher Multimedia Award.* J&S Dialog Medien GmbH.

Literatur: [11] Freiknecht, J. (2014). *Big Data in der Praxis Lösungen mit Hadoop, HBase und Hive. Daten speichern, aufbereiten,.* Deutschland: Carl Hanser Verlag GmbH & Co. KG.

Literatur: [12] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (2004). *Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software .* Addison-Wesley Verlag.

Literatur: [13] Hohanty, S., Jagadeesh, M., & Srivasta, H. (2013). *Big Data Imperatives.* Apress.

Literatur: [14] Jaro, M. A. (1989). *Advances in Record-Linkage Methodology as Applied to Machting the 1985 Census of Tampa Florida.* Florida. In: Journal of the American Statical Association.

Literatur: [15] Levenshtein, V. J. (1966). *Binary codes capable of correcting deletions, insertions, and reversals.* Soviet Physics Doklady.

Literatur: [16] Moczar, L. (2004). *Tomcat 5: Einsatz in Unternehmensanwendungen mit JSP und Servlets.* Addison-Wesley Verlag.

Literatur: [17] Rossak, I. (17. Januar 2013). *Datenintegration: Integrationsansätze, Beispielszenarien, Problemlösungen, Talend Open Studio .* Carl Hanser Verlag GmbH & Co. KG .

Literatur: [18] Scheller, M., Boden, K.-P., Geenen, A., & Kampermann, J. (1994). *Internet: Werkzeug und dienste.* Berlin Heidelberg: Springer.

Literatur: [19] Strobel, C. (2003). *Web-Technologien: in E-Commerce-Systemen.* Oldenbourg Wissenschaftsverlag.

Literatur: [20] teutberg, H.-J., & Neutsch, C. (1998). *Vom Flügeltelegraphen zum Internet.* Stuttgart: Franz Steiner verlag Stuttgart.

Literatur: [21] Turkington, G. (2013). *Hadoop Beginner´s Guide.* Packt Publishing (22. Februar 2013).

Literatur: [22] Veit, K., Kai-Uwe, S., & Gunter, S. (2012). *Data Warehouse Technologien.* Ilmenau: Verlagsgruppe Hüthig-Jehle-Rehm.

Literatur: [23] Viktor Mayer-Schönberger, V., & Mayer-Schönberger, C. (2013). *Big Data die Revolution die Unser Leben verändert.* München: REDLINE Verlag AG.

Literatur: [24] Wartala, R. (22. Februar 2013). *Hadoop: Zuverlässige, verteilte und skalierbare Big-Data-Anwendungen .* Open Source Press.

Literatur: [25] White, T. (2009). *Hadoop - The Definitive Guide.* O´REILLY.

Literatur: [26] White, T. (2012). *Hadoop: The Definitive Guide.* USA (CA): O'Reilly Media.

Literatur: [27] Winkler, W. E., & Thibaudeau, Y. (1991). *An Application of the Fellegi-Sunter Model of Record Linkagde to the 1990 U.S. Census. Technical report.* US Bureau of the Census.

Literatur: [28] Wischki, C. (2009). *ITIL V2, ITIL V3 und ISO/IEC 20000, Gegenüberstellung und Praxisleitfaden für die Einführung/Umstieg.* HANSER Fachbuch.

Literatur: [29] Holden Karau, e. a. (2014). *Learning Spark .* USA: O'Reilly.

Literatur: [30] Florian Wild, Tobias Lang und et. Al. (2006). Text Mining: Wissensgewinnung aus natürlichsprachigen Dokumenten. Universität Karlsruhe (TH), Institut für Programmstrukturen und Datenorganisation

Literatur: [31] Erik Hatcher und et. Al. (2010). Lucene in Action. USA; Manning.

Versicherung über die Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §16(5) APSO-TI-BM ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

Hamburg, 03. März 2015 _____
Ort, Datum Unterschrift