Institut für Parallele und Verteilte Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

Evaluierung verschiedener Technologien zur Speicherung und Verknüpfung von strukturierten und unstrukturierten Daten

Paul Dieterich

Studiengang:	Softwaretechnik
Prüfer/in:	Prof. Dr.-Ing. habil. Bernhard Mitschang
Betreuer/in:	M.Sc. Christian Weber, M.Sc. Cornelia Kiefer
Beginn am:	1. Dez 2016
Beendet am:	1. Juni 2017
CR-Nummer:	H.3.2 K.6.3

Kurzfassung

In der heutigen Zeit ist Wissen aufgrund des Wandels zu einer Wissensgesellschaft ein wichtiges Wirtschaftsgut. In Unternehmen liegen bis zu 90% der Daten in unstrukturierter Form vor. Ein derzeit großes ungelöstes Problem ist der effektive Einsatz von unstrukturierten Daten für Entscheidungsfragen. Deshalb ist das Verknüpfen von unstrukturierten und strukturierten Daten notwendig. In dieser Arbeit wurden verschiedene Open-Source-Softwarelösungen, die in der Lage sind diese Daten zu speichern, zu durchsuchen und zu verknüpfen, untersucht. Für die Untersuchung wurden Kriterien definiert, um die entsprechenden Softwarelösungen zu bewerten. Mit der dadurch entstandenen Tabelle ist es möglich, die bestehenden Lösungen zu vergleichen. Die Selektion einer passenden Lösung zur Adressierung spezifischer Anforderungen wurde anhand eines exemplarischen Anwendungsfalls demonstriert. Die Verknüpfung von strukturierten und unstrukturierten Daten wurde mit der gewählten Lösung vorgenommen und dokumentiert.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Unternehmen stehen unter dem ständigen Druck besser, schneller und effizienter zu sein um wettbewerbsfähig zu bleiben. Deshalb ist es wichtig, dass Unternehmen in der Lage sind Wissen zur Verfügung zu stellen, so dass dieses gewinnbringend eingesetzt werden kann, denn Wissenskapital ist für Firmen ein wichtiges Wirtschaftsgut. Eine beliebte Anschauung des Aufbaus von Wissen gibt die Wissenspyramide. Laut Hildebrand [Hil11] besteht diese Pyramide aus vier Ebenen, die aufeinander aufbauen. Die unterste Ebene sind Zeichen, die mithilfe einer Syntax zu Daten werden. Daten werden durch Hinzufügen von Semantik zu Informationen und Informationen werden mithilfe der Vernetzung von Kontext und Erfahrungen zu Wissen. Laut van den Hoven [van01] sind in Unternehmen nur 10% der Daten strukturiert, die restlichen 90% sind unstrukturiert. Momentan werden zur Entscheidungsunterstützung allerdings nur strukturierte Datenquellen eingesetzt. Strukturierte Daten liegen typischerweise als Datenbanken vor, während unstrukturierte Daten zum Beispiel aus Texdokumenten oder PDF-Dateien bestehen. Daher benötigen Unternehmen Technologien, die strukturierte und unstrukturierte Daten verknüpfen, speichern und durchsuchbar machen. Dadurch können aus den Daten mehr Informationen gewonnen werden und diese effizienter eingesetzt werden. Durch das Verknüpfen der Daten können beispielsweise Produktionsarbeiter auf dem Shopfloor bei ihrer Arbeit unterstützt werden, denn dann es ist möglich, relevante Informationen wie technische Dokumente oder Videos zu Arbeitsgängen schneller abzurufen. Van den Hoven [van01] beschreibt, dass derzeit ein großer Teil des Wissens in Unternehmen nicht zentral verfügbar und nur für wenige Abteilungen zugänglich ist. Im Rahmen der Industrie 4.0 ist das Verknüpfen von Informationen rund um den Produktlebenszyklus von besonderem Interesse, wie der Arbeitskreis Industrie 4.0 [Arb13] und Yongkui Liu [Yon17] beschreiben. Die Integration bietet Servicetechnikern die Möglichkeit bei der Fehlersuche schneller auf relevante Dokumente zuzugreifen. Dadurch kann die Effizienz von Produktionsanlagen gesteigert werden und diese können schneller auf neue Bedingungen angepasst werden. Außerdem können Zusammenhänge erkannt werden, um besser auf Ausfälle oder besondere Kundenwünsche reagieren zu können. Um diese Vision umzusetzen, ist es notwendig immense Mengen von Daten in strukturierter wie unstrukturierter Form zu verarbeiten, zu speichern und in Relation zu setzen. In dieser Arbeit werden momentan auf dem Markt existierende Open-Source-Technologien verglichen, die die Verknüpfung von unstrukturierten mit strukturierten Daten ermöglichen. Es ist dabei von besonderer Bedeutung zu evaluieren, wie interoperabel die vorgestellten unstrukturierten Datenspeicher mit einer relationalen Datenbank verknüpft werden können. Um die vorgestellten Technologien zu bewerten, werden Kriterien erarbeitet, die es ermöglichen eine Auswahl für einen individuellen Anwendungszweck zu treffen.

Aus diesen Bedingungen lassen sich folgende zentrale Fragen ableiten:

- Welche Lösungen existieren, um strukturierte und unstrukturierte Daten zu speichern?
- Welche Möglichkeiten bieten die Systeme zur Speicherung von Metadaten?
- Welche Open-Source-Technologien eignen sich für welchen Anwendungsfall?

Vorgehensweise

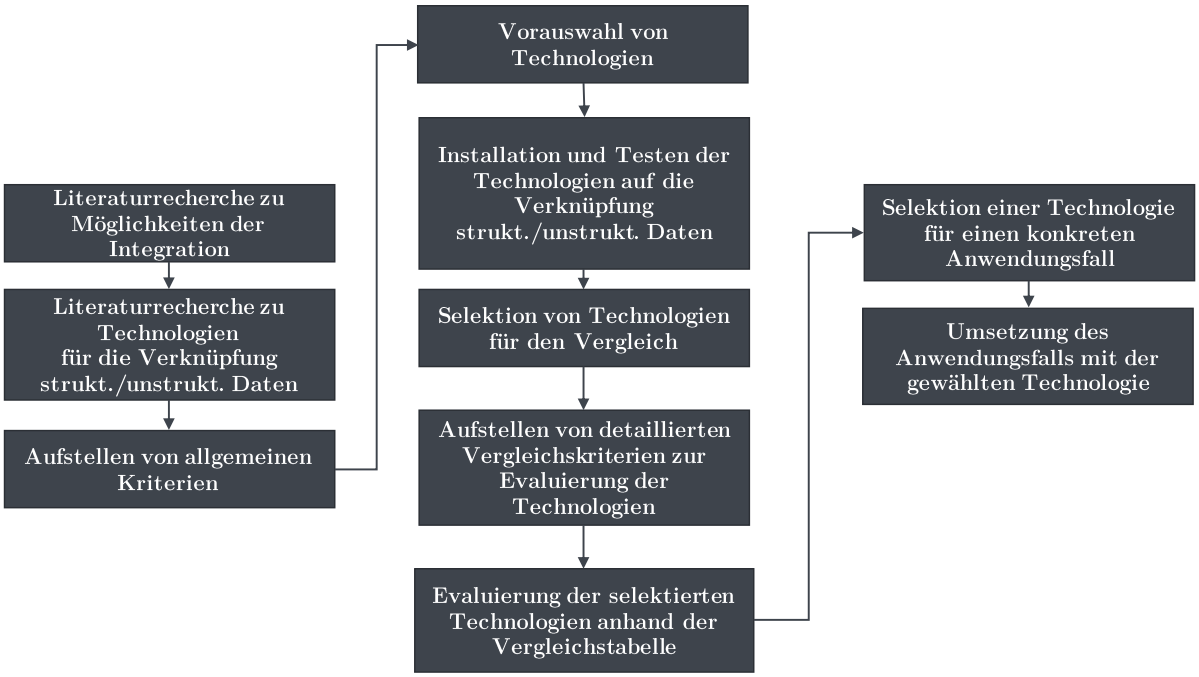

Abbildung 1.1: Schematische Darstellung der Vorgehensweise

Die Arbeit ist wie folgt gegliedert: Zuerst wird eine Literaturrecherche zu den Möglichkeiten der Integration für strukturierte und unstrukturierte Daten durchgeführt. Anschließend werden Technologien zur Verknüpfung von strukturierten mit unstrukturierten Daten untersucht. Danach werden allgemeine Kriterien für die Wahl von Technologien aufgestellt und eine Vorauswahl getroffen. Die Technologien der Vorauswahl werden installiert und getestet und anschließend die Technologien ausgewählt, welche die allgemeinen Kriterien erfüllen. Für die ausgewählten Technologien werden detailliertere Vergleichskriterien erarbeitet. Dann wird durch diese und die ausgewählten Technologien eine Vergleichstabelle aufgestellt. Anhand der Tabelle wird eine Technologie für einen konkreten Anwendungsfall selektiert. Der Anwendungsfall wird mit der gewählten Technologie umgesetzt.

1.1 Struktur der Arbeit

Die Aufbau der Arbeit gestaltet sich wie folgt:

Kapitel 2 – Theoretische Grundlagen: In diesem Kapitel werden die theoretischen Grundlagen für die Arbeit definiert.

Kapitel 3 – Verwandte Arbeiten: Beinhaltet die Literaturrecherche zu verwandten Arbeiten sowie Konzepte und Architekturen zur Verknüpfung von Daten.

Kapitel 4 – Anforderungen: Dieses Kapitel beschreibt die Anforderungen, die an diese Arbeit gestellt werden.

Kapitel 5 – Auswahl von Lösungen: Enthält Kriterien und Technologien für eine Vorauswahl der Softwarelösungen.

Kapitel 6 – Festlegung von Kriterien für den weiteren Vergleich: Hier werden die detaillierteren Kriterien für den weiteren Vergleich definiert.

Kapitel 7 – Bewertung der Technologien: In diesem Kapitel werden die ausgewählten Technologien durch die zuvor definierten Kriterien bewertet.

Kapitel 8 – Anwendungsfall: Dieses Kapitel beinhaltet die Umsetzung des Anwendungsfalls anhand der gewählten Technologie.

Kapitel 9 – Zusammenfassung und Ausblick: Das letzte Kapitel ist eine Zusammenfassung der Arbeit und gibt Anknüpfungspunkte für weitere Forschungsarbeiten.

2 Theoretische Grundlagen

In diesem Kapitel wird auf die für die Arbeit relevanten theoretischen Grundlagen eingegangen. Für ein einheitliches Verständnis der nachfolgend benutzten Begriffe werden im folgenden Abschnitt oft genutzte Terminologien erklärt.

2.1 Arten von Daten

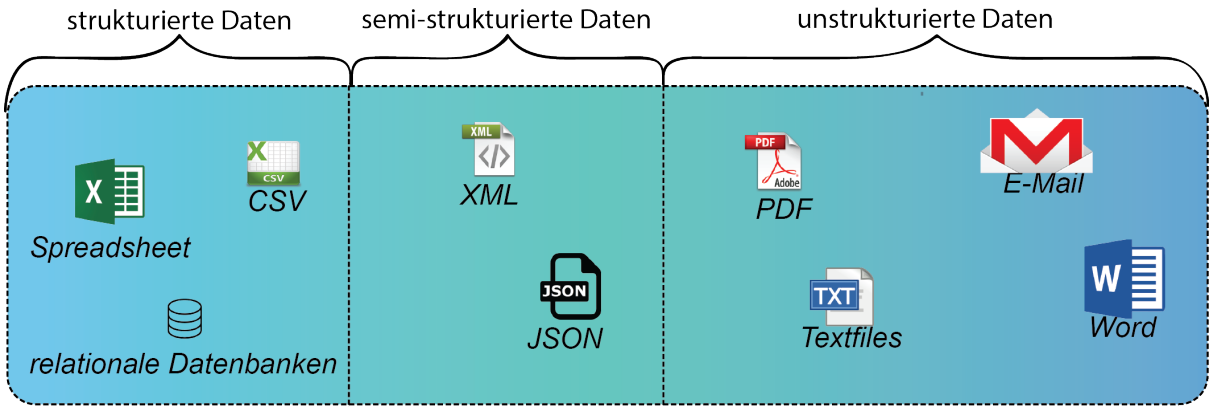

Abbildung 2.1: Unterteilung verschiedener Arten von Daten nach Russom [Rus07].

Man unterscheidet verschiedene Arten von Daten: unstrukturierte, semi-strukturierte und strukturierte Daten. Strukturierte Daten besitzen eine gemeinsame Struktur und unterliegen einem Datenmodell, deshalb können sie effizient durch relationale Datenbanken verwaltet werden. Semi-strukturierte Daten besitzen nach Buneman [Bun97] keine allgemeingültige Struktur, sondern enthalten zusätzliche Strukturinformationen. Sie unterliegen aber keinem Datenmodell. Unstrukturierte Daten beinhalten wie Blumberg et al. [BA03] beschreiben keine Strukturinformationen und unterliegen keinem Datenmodell. Oft liegen unstrukturierte Daten in natürlicher Sprache vor. Die Abbildung 2.1 zeigt Beispiele für verschiedene Datenarten. Strukturierte Daten können zum Beispiel in Tabellen oder relationalen Datenbanken gespeichert werden. Semi-strukturierte Daten sind häufig als JSON- oder XML-Dokumente organisiert. Unstrukturierte Daten bestehen meist aus Fließtext in natürlicher Sprache und sind beispielsweise als PDF-Dateien oder Textdokumente gespeichert. In den kommenden Jahren werden laut Russom [Rus07] voraussichtlich immer mehr unstrukturierte und semi-strukturierte Daten generiert, während weniger strukturierte Daten erzeugt werden.

2.2 Systemtypen

Zur Verwaltung von Daten gibt es verschiedene Systeme. Im folgenden Abschnitt werden für diese Arbeit relevante Systemtypen zur Verwaltung von strukturierten und unstrukturierten Daten vorgestellt.

2.2.1 Verwaltung strukturierter Daten

Zur Verwaltung von strukturierten Daten gibt es verschiedene Technologien. Im folgenden Abschnitt werden für diese Arbeit relevante Technologien zur Verwaltung von strukturierten Daten beschrieben.

Relationale Datenbanken

Zur Speicherung von strukturierten Daten werden klassischerweise relationale Datenbanken eingesetzt. Dabei sind die Daten mithilfe von Tabellen und Verknüpfungen organisiert wie Seipel [Sei13] beschreibt. Durch Relationen zwischen den Tabellen können Operationen mithilfe von relationaler Algebra durchgeführt werden. Häufig wird zum Analysieren und Manipulieren der gespeicherten Datensätze die auf relationaler Algebra basierende Sprache **S**tructured **Q**uery **L**anguage (SQL) eingesetzt [Sei13]. Ein Beispiel dieser Technologie stellt MySQL dar.

Data Warehouses

Rahm [Rah14] beschreibt Data Warehouses als Systeme, die Daten aus verschiedenen heterogenen Datenquellen (z.B relationale Datenbanken) für Analysezwecke bündeln und zentral zur Verfügung stellen. Im Rahmen von Data Warehouses wird oft **O**n**l**ine **A**nalytical **P**rocessing (OLAP) diskutiert. Durch OLAP kann auf die Daten sehr flexibel zugegriffen werden, um diese dynamisch zu untersuchen [Rah14]. Indem aus den Datenquellen Informationen und Wissen abgeleitet werden, können Unternehmen bei Entscheidungen unterstützt werden. Durch Data Warehouses ist es möglich, alle strukturierten Unternehmensdaten an einer zentralen Stelle auszuwerten, wodurch weniger Daten „vergessen“ werden. W. H. Inmon [Inm02] definiert Data Warehouses als themenorientierte (subject-oriented), integrierte (integrated), nicht flüchtige (non-volatile), Langzeit- (time-variant) Sammlung an Daten. Themenorientiert bedeutet, dass das System nicht nur für eine Aufgabe genutzt werden sollte, sondern für quellübergreifende Auswertungen aus verschiedenen Perspektiven genutzt werden kann. Integriert bedeutet, dass die Datenbasis aus mehreren Datenquellen zusammengesetzt ist. Nicht flüchtig sagt aus, dass die Daten persistent gespeichert und typischerweise nicht geändert werden. Langzeitsammlung beschreibt die Speicherung der Daten über einen längeren Zeitraum, wodurch ein zeitlicher Vergleich der Daten möglich wird [Rah14].

2.2.2 Verwaltung unstrukturierter Daten

Zur Verwaltung unstrukturierter Daten existieren verschiedene Softwarelösungen. In diesem Kapitel werden für die Arbeit relevante Systemtypen zur Verwaltung von unstrukturierten Daten konzeptionell beschrieben.

Content-Management-Systeme

Riggert [Rig09] definiert Content-Management wie folgt:

Definition 2.2.1 (Content-Management nach Riggert [Rig09])
Content Management beschreibt die Planung, Verwaltung, Steuerung und Koordination aller Aktivitäten, die auf den Content und dessen Präsentation in Unternehmen abstellen.

Content-Management-Systeme (CMS) ermöglichen das Erstellen, Verwalten, Freigeben, Veröffentlichen und Archivieren von Inhalten. Dabei kann das Content-Management mit einem Lebenszyklus verglichen werden. Dieser beginnt mit der Erstellung des Contents, gefolgt von der Qualitätssicherung und Veröffentlichung. Der Lebenszyklus endet entweder mit der Archivierung oder Vernichtung des Contents [Rig09]. CMS bieten verschiedene Möglichkeiten auf Content zuzugreifen, wie beispielsweise Handys oder Websites. Content-Management-Systeme können in statische und dynamische CMS unterteilt werden. Statische CMS können Inhalte schneller anzeigen, sind jedoch weniger aktuell, während dynamische CMS-Inhalte im Moment des Zugriffs zusammenstellen und dadurch langsamer aber aktueller sind.

Web-Content-Management-Systeme

Eine besondere Form von Content-Management-Systemen stellen **W**eb-**C**ontent-**M**anagement-**S**ysteme (WCMS) dar. WCMS dienen zur Content-Verwaltung auf Web-Sites und Portalen. Der Kerngedanke von WCM-Systemen ist die Trennung von Layout und Inhalt. Dadurch kann das Design einheitlich auf dem gesamten Webauftritt gestaltet werden, während der Inhalt unabhängig für andere Zwecke eingesetzt werden kann. Die Verwendung eines WCMS erfordert meist keine Programmierkenntnisse, wodurch die Rolle von Redakteur, Designer und Administrator von unterschiedlichen Personen übernommen werden kann. Deshalb unterscheiden sich WCMS bei der Veröffentlichung von Content von CM-Systemen [Rig09].

Dokument-Management-Systeme

Ein Dokument-Management-System (DMS) ermöglicht die Planung, Verwaltung und Koordination der Aktivitäten auf Dokumente [Rig09]. Sie stellen eine Spezialisierung von Content-Management-Systemen dar. Je nach DMS können Dokumente aus verschiedenen Quellen angebunden werden, zum Beispiel einem Scanner, Netzwerklaufwerken, etc. Die in ein solches

System abgelegten Dokumente sind meist schwach bis unstrukturierte Daten, welche mithilfe von Datenbankservern und Dateiservern organisiert werden. An die abgelegten Dokumente können typischerweise beliebige Metadaten angebunden werden. Viele DMS bieten die Möglichkeit, die angebundenen Dokumente zu durchsuchen. Dokument-Management-Systeme werden unter anderem für kaufmännische Dokumente, technische Zeichnungen, Bibliotheken oder Behörden eingesetzt. Mithilfe von Workflow-Management-Systemen können Arbeitsabläufe unterstützt und organisiert werden. Allerdings bieten die Systeme keine Möglichkeit, strukturierte Daten aus relationalen Datenbanken anzubinden.

Enterprise-Content-Management-Systeme

Ein **E**nterprise-**C**ontent-**M**anagement-**S**ystem (ECMS) umfasst typischerweise: ein DMS zum Verwalten von Dokumenten, ein Portal, ein Workflow-Management sowie eine Möglichkeit, gemeinsam Daten zu bearbeiten. ECM-Systeme sollen als ganzheitlicher Datenspeicher vor allem für Unternehmen dienen. Riggert [Rig09] definiert ECMS folgendermaßen:

Definition 2.2.2 (ECMS nach Riggert [Rig09])
ECM besteht aus Technologien, Werkzeugen und Methoden, um Inhalte (Content) unternehmensweit zu erfassen, zu verwalten, zu speichern, zu schützen und zu verteilen.

Die meisten ECM-Systeme bieten allerdings keine Funktion, strukturierte Daten anzubinden. Wie ein DMS beinhalten auch ECM-Systeme die Möglichkeit, die gespeicherten Dokumente zu durchsuchen. Mithilfe der Kollaborationsfunktion wird es ermöglicht, Arbeitsgruppen zu erstellen und gemeinsam Dokumente zu bearbeiten/erstellen. Durch die Portalsoftware bietet ein ECMS eine plattformunabhängige, webbasierte Benutzeroberfläche. Die Komponenten eines ECMS können nach Riggert [Rig09] in folgende Kategorien unterteilt werden: Erfassung, Verwaltung, Ausgabe und langfristige Speicherung. Die Unterteilung der Komponenten leitet sich aus dem Content-Management-Lebenszyklus ab.

Ontologiebasierte Systeme

Gruber [Gru93] definiert Ontologien wie folgt:

Definition 2.2.3 (Ontologie nach Gruber [Gru93])
An ontology is an explicit specification of a conceptualization. The term is borrowed from philosophy, where an ontology is a systematic account of Existence. For knowledge-based systems, what „exists“ is exactly that which can be represented.

Somit ist eine Ontologie eine eindeutige Spezifikation einer Konzeption von Entitäten und Beziehungen innerhalb einer Domäne. Das **R**esource **D**escription **F**ramework (RDF) ist eine Spezifikation, konzipiert von dem World Wide Web Consortium (W3C), in dem Daten als Subjekt-Prädikat-Objekt-Tripel gespeichert werden, wobei das Prädikat die Verbindung

zwischen Subjekt und Objekt darstellt. Eine Erweiterung des RDF stellt die **W**eb **O**ntologie **L**anguage (OWL) dar, eine Beschreibungssprache für Ontologien, die Entitäten und Beziehungen formal beschreibt, um diese programmatisch auswerten zu können. OWL bietet bereits ein vordefiniertes Vokabular, um den Aufbau von Ontologien zu vereinfachen. OWL wird vor allem im Semantik Web eingesetzt. Das Semantik Web erweitert das World Wide Web, um Daten einfacher auswerten und in Relation stellen zu können. Auf Basis dieser Technologien gibt es verschiedene Tools, welche die Verwaltung und Nutzung von Ontologien ermöglichen.

Hadoop-Ecosystem

Das Hadoop-Ecosystem ist eine aus mehreren Komponenten bestehende Open-Source-Softwareplattform. Die Hauptkomponenten sind das **Ha**d**o**op **F**ile**s**ystem (HDFS) sowie MapReduce, ein Programmiermodell zum verteilten Verarbeiten von Daten. HDFS ist ein nach Google White Papern entwickeltes Dateisystem. Das Hadoop-Ecosystem bietet gegenüber klassischem **E**xtrakt-**T**ransform-**L**oad (ETL) zwei Vorteile: einerseits können unstrukturierte Daten schnell in das HDFS geladen werden, ohne vorher ein Schema zu definieren, und andererseits bietet Hadoop durch MapReduce ein mächtiges Tool, um Roh-Daten parallel zu verarbeiten und beispielsweise in ein Data Warehouse aufzunehmen. Durch die Skalierbarkeit von MapReduce können mit Hadoop ETL-Vorgänge deutlich beschleunigt werden. Klassischerweise werden bei einem ETL basierten System im ersten Schritt aus einer Datenquelle Informationen extrahiert (Extract), anschließend in ein gemeinsames Format transformiert (Transform) und im letzten Schritt in ein Datenbanksystem für eine darauffolgende Analyse geladen (Loading) [Int13]. Als Alternative zum ETL-Verfahren wurde das ELT-Verfahren entwickelt, welches die aufwändigen Transformationen innerhalb der Datenbank ausführt. Ein Hybrid beider Verfahren ist der ETLT-Ansatz, dieser ermöglicht es, Transformationen innerhalb der Datenbank durchzuführen. Bei Operationen, welche nicht von der Datenbank unterstützt werden, bietet es die Möglichkeit, diese vor dem Loading-Schritt durchzuführen. Allerdings ist keines dieser Verfahren einfach oder schnell, besonders wenn es sich um große unstrukturierte Datenmengen handelt [LHC16]. ETLT-Prozesse können beschleunigt werden, indem die Aufnahme, Transformation und Integration von unstrukturierten Daten in das Data Warehouse an Hadoop ausgelagert werden [Int13].

Big-Data-Lösungen

In den letzten Jahren haben sich klassische Datenbankmanagement-Lösungen als unzureichend erwiesen, weshalb viele Unternehmen stattdessen NoSQL-Datenbanken einsetzen [Pok13]. Deshalb vergleichen Gölzer et al. [GCA15] verschiedene NoSQL-Datenbanken. Diese sind: Apache Cassandra - ein Column-Store, MongoDB - eine dokumentenorientierte Datenbank sowie Amazon SimpleDB - ein Key-Value-Store. Ein Column-Store speichert Inhalte spaltenweise als Tabelle. Eine dokumentenorientierte Datenbank unterliegt keinem festen Schema, wodurch auch unstrukturierte Daten abgelegt werden können. Key-Value-Stores speichern

Daten als Key-Value-Paare, vergleichbar mit der Map-Datenstruktur. Unterschiedliche Systeme sind für verschiedene Ansprüche ausgelegt. Apache Cassandra, MongoDB und SimpleDB ermöglichen Real-Time-Queries, können jedoch Daten nicht durch Batch-Processing verteilt verarbeiten. Zum schnellen Verarbeiten von Daten wird die Lambda-Architektur diskutiert. Diese Architektur besteht aus einem Batch-, Speed- und Serving-Layer. Der Batch-Layer dient zur Speicherung und Vorberechnung der Daten. Der Speed-Layer dient zur schnellen Verarbeitung und zur Anfrage von aktuellen Daten und im Serving-Layer werden die Ergebnisse des Batch- und Speed-Layers kombiniert. Dadurch können große Datenmengen verarbeitet und mit geringer Latenz zur Verfügung gestellt werden. Die einzelnen Layer sind durch verschiedene Big-Data-Lösungen realisierbar [KMM+15]. Neben den bereits genannten Systemen und Typen existieren Data Grids. Dies sind verteilte Systeme, die das Verarbeiten und Speichern von Daten ermöglichen. Der Ansatz für Daten Grids basiert auf Gridcomputing. Gridcomputing ermöglicht es, verteilte Systeme zu nutzen, um Probleme schneller berechnen zu können, dabei müssen die Knoten jedoch nicht innerhalb eines lokalen Netzwerks sein, sondern können auf der gesamten Welt verteilt sein. Das Data Grid ist ein Konzept, um das Gridcomputing um effizientere Mechanismen zur Verwaltung von großen Datensätzen zu erweitern [MDKW16].

2.3 Lizenzen

Es gibt verschiedene Möglichkeiten, Software zu lizenzieren. Bei proprietärer Software werden dem Nutzer nur wenig Nutzungsrechte eingeräumt und es ist in der Regel verboten, die Software zu vervielfältigen, im Gegensatz zu freier Software, bei der dem Nutzer das Recht zur Vervielfältigung eingeräumt wird. Dies ist jedoch meist mit besonderen Pflichten und Einschränkungen verbunden. Es gibt im Wesentlichen zwei verschiedene Typen von Open-Source-Lizenzen, zum einen Copyleft-Lizenzen, zum anderen non-Copyleft oder auch tolerante Lizenzen. Je nach Lizenz kann sich die Ausprägung der obligatorischen Pflichten unterscheiden. Alle Open-Source-Lizenzen enthalten die Einschränkungen, dass keine Haftung und Garantie übernommen werden. Manche Lizenzen verbieten außerdem das Nutzen der Warenzeichen.

2.3.1 Copyleft-Lizenzen

Copyleft-Lizenzen erzwingen, dass Software, die unter einer derartigen Lizenz veröffentlicht wurde und von einem Entwickler modifiziert wurde, ebenfalls unter derselben Lizenz veröffentlicht wird. Die häufigste Copyleft-Lizenz ist die GNU **G**eneral **P**ublic **L**icense (GPL). Durch die GPL wird gewährleistet, dass Open-Source-Software unter GPL immer unter GPL veröffentlicht werden muss und nicht in proprietäre Software umgewandelt werden kann. Die GPL ist inzwischen in verschiedenen Versionen verfügbar. Es ist erlaubt Software, die unter einer älteren GPL veröffentlicht wurde, zu modifizieren und unter einer neueren Version der GPL zu veröffentlichen. Neben der Copyleft-Pflicht enthält die GPL je nach Variante weitere Pflichten. Diese Pflichten können sein:

- Der Quellcode muss bei der Verteilung der Software verfügbar gemacht werden.
- Bei einer Verbindung mit der Software über ein Netzwerk hat der Verbindungspartner das Recht, eine Kopie des Codes zu erhalten.

Neben der GPL gibt es weitere Copyleft-Lizenzen wie die **M**ozilla **P**ublic **L**icense (MPL) oder die **L**esser **G**eneral **P**ublic **L**icense (LGPL). Diese Lizenz erfordert nicht, dass Software, die durch das Netzwerk oder Ähnlichem angebunden wird, ebenfalls unter GPL veröffentlicht sein muss. Diese Lizenz ermöglicht es, Teile der Software, die nicht unter GPL oder sogar unter proprietären Lizenzen veröffentlicht sind, auszulagern. Dabei muss es jedoch für den Nutzer möglich sein, die Teile der Software, welche nicht unter GPL veröffentlicht wurden, zu trennen.

2.3.2 Tolerante Lizenzen

Tolerante Lizenzen erzwingen nicht, dass Änderungen am Code ebenfalls unter derselben Lizenz veröffentlicht werden. Dadurch können Entwickler freie Software, die unter einer toleranten Lizenz veröffentlicht wurde, um einen proprietären Code erweitern. Beispiele für diese Art von Lizenz sind die BSD-Lizenzen, die **A**pache **P**ublic **L**icense (APL) und die MIT License. Jedoch enthalten auch non-Copyleft-Lizenzen oft gewisse Pflichten wie zum Beispiel:

- Eine Kopie der Lizenz und Copyright-Informationen müssen der Software beigefügt werden.
- Entwickler, die den Code ändern, müssen dies kenntlich machen.

2.3.3 Übersicht

	GPL	MPL	APL	MITL
	Rechte			
Kommerziell	✓	✓	✓	✓
Vervielfältigung	✓	✓	✓	✓
Veränderungen	✓	✓	✓	✓
Patentierbar	✓	✓	✓	
Private Nutzung	✓	✓	✓	✓
	Pflichten			
Quellcode veröffentlichen	✓	✓		
Kopie von Lizenz und Copyright	✓	✓		✓
Copyleft	✓	✓		
Änderungen kenntlich machen	✓			
	Einschränkungen			
Haftbarkeit	✓	✓	✓	✓
Garantie	✓	✓	✓	✓
Trademark			✓	

Tabelle 2.1: Vergleich von wichtigen Open-Source-Lizenzen

Die Tabelle 2.1 stellt eine Übersicht von wichtigen Open-Source-Lizenzen dar. Diese Lizenzen werden anhand ihrer Rechte, Pflichten und Einschränkungen tabellarisch verglichen. In der ersten Spalte sind die Kriterien aufgeführt und in den nachfolgenden Spalten die Lizenzen. Die Reihen beinhalten die Rechte, Pflichten und Einschränkungen der jeweiligen Lizenz.

3 Verwandte Arbeiten

Innerhalb der Industrie 4.0 ist für die vertikale Integration die Verknüpfung von strukturierten und unstrukturierten Daten notwendig, da die im Produktlebenszyklus anfallenden Daten analysiert werden müssen, um dadurch mögliche Verbesserungen für die Produktion abzuleiten. Unter vertikaler Integration versteht man die Integration von IT-Systemen über verschiedene Hierarchieebenen wie dem Shop- und dem Officefloor zu einer durchgängigen Lösung [Arb13]. In diesem Kapitel werden relevante Arbeiten vorgestellt, welche bereits verschiedene Technologien zum Speichern oder Verknüpfen von strukturierten und unstrukturierten Daten evaluiert haben oder neue Konzepte sowie Implementierungen vorstellen.

Unter Big Data versteht man nach Katal et al. [KWG13] große Datenmengen im Peta- oder in Zukunft sogar im Zettabyte-Bereich. Diese Daten bestehen nicht nur aus strukturierten, sondern auch aus semi-strukturierten sowie gänzlich unstrukturierten Daten. Es ist für klassische Data-Warehouse-Lösungen eine große Hürde diese Daten zu bereinigen, zu verknüpfen, zu transformieren und die hohe Geschwindigkeit an eingehenden Daten zu verarbeiten. Zur Aufdeckung von Trends ist es von besonderer Bedeutung, Korrelationen innerhalb der gespeicherten Daten aufzudecken, denn dadurch können Businessstrategien abgeleitet werden.

Ein Überblick und Vergleich zwischen existierenden Big-Data-Technologien und Anforderungen werden von Yang et al. [YHL+16] präsentiert. Yang et al. vergleichen nicht nur Technologien zur Speicherung und Analyse von Big Data, sondern auch Kommunikationstechnologien und Infrastruktur-Methoden, die im Rahmen von Big Data eingesetzt werden. Der Vergleich erfolgt anhand von festen Eigenschaften, die erfüllt werden oder nicht. Diese Eigenschaften sind: Speicher, Transfer, Management, Verarbeitung, Analyse, Visualisierung, Integration, Architektur, Sicherheit, Qualität, Kosten und Energieeffizienz. Dieser Vergleich stellt eine Basis zum Verständnis für die Herausforderungen, die es rund um Big Data gibt, dar und differenziert verschiedene Terminologien und Technologien voneinander.

Gölzer et al. [GCA15] vergleichen existierende Big-Data-Softwarelösungen, wobei die Technologien anhand ihrer technischen Eigenschaften verglichen werden. Sie vergleichen das Hadoop-Filesystem mit MapReduce, Apache Cassandra, MongoDB und Amazon SimpleDB anhand:

- des eingesetzten Datenmodells
- der Möglichkeit, verteilte Berechnungen durchzuführen
- des Supports von Real-Time-Queries

- des Supports für schnelle, zufällige Zugriffe auf die Daten
- der horizontalen Skalierbarkeit durch das Hinzufügen von mehr Rechner-Knoten
- der Stärken der Technologien
- der genutzten Architekturbasis des Systems
- der angestrebten Eigenschaften des CAP-Theorems

Das CAP-Theorem besagt nach Gilbert et al. [GL02], dass ein verteiltes System nicht Konsistenz, Verfügbarkeit und Ausfalltoleranz gleichzeitig bieten kann, sondern nur maximal zwei Anforderungen erfüllt werden können.

Apache Cassandra, MongoDB und Amazon SimpleDB bieten Real-Time-Queries und schnellen Zugriff auf zufällige Daten, haben allerdings gegenüber Hadoop (HDFS) in Verbindung mit MapReduce keine Möglichkeit, Daten verteilt zu verarbeiten. Jede der vorgestellten Technologien skaliert horizontal durch das Hinzufügen von weiteren Rechnerknoten. Die Stärken von Hadoop liegen im Verarbeiten der Daten. Apache Cassandra ermöglicht besonders effizientes Schreiben der Daten, während MongoDB besonders schnell Daten abrufen kann. Amazon SimpleDB bietet vollständiges Indexing der Daten.

Durch diesen Vergleich ist es einfacher, die richtige Big-Data-Lösung für einen bestimmten Anwendungsfall auszuwählen. Es wird jedoch nicht geklärt, welche Möglichkeiten die Technologien bieten, um strukturierte Daten mit unstrukturierten Daten zu verknüpfen.

Klassische Ansätze zum Verknüpfen von unstrukturierten mit strukturierten Daten innerhalb eines Data Warehouses basieren gemäß Baars und Ferrucci et al. [BK08; FL04] darauf, Metadaten aus Content-Objekten wie den Autor, das Datum des Erstellens, die Länge des Contents und das behandelte Produkt als Klassifikation für unstrukturierte Daten einzusetzen. Content-Objekte sind unstrukturierte Daten, zum Beispiel im PDF-Format. Die extrahierten Metadaten werden im Data Warehouse mit der ID des Content-Objekts als Fakt verknüpft. Diese Ansätze haben den Nachteil, dass Analyseanforderungen im Vorfeld bekannt sein müssen, um die Data-Warehouse-Schemas diesbezüglich anzupassen. Dadurch ist die Integration neuer Typen von Content-Objekten erschwert und kann die Anpassung bereits existierender Analysetechnologien erfordern.

Neuere Ansätze basieren auf der Integration von unstrukturierten Daten mithilfe des Resource Description Framework im Data Warehouse. Dafür werden Daten als Tripel abgelegt. Ein Tripel besteht aus Subjekt, Prädikat und Objekt, wobei das Prädikat die Verknüpfung zwischen Subjekt und Objekt darstellt. Abelló et al. [ADE+13] erweitern dieses Konzept, so dass beliebige Textdaten integriert werden können. Hausenblas et al. [HGHC12] diskutieren Technologien, welche es ermöglichen, die von Gölzer et al. verglichenen Datenbanken zu nutzen, um sie als RDF-Tripel-Stores für semantische Web-Anwendungen zu nutzen. Neben linkbasierten Architekturen existieren auch föderierte Systeme auf Basis ontologiebasierter Datenintegration. Das Aletheia-Forschungsprojekt [KBF+10; KD10] ist ein System, das die Möglichkeit

bietet, strukturierte und unstrukturierte Daten mithilfe von ontologiebasierten Methoden zu verknüpfen.

Gröger [Grö15] stellt einen Entwurf sowie eine prototypische Implementierung zur Verknüpfung von unstrukturierten und strukturierten Daten mit einem linkbasierten Modell vor. Im Gegensatz zu klassischen RDF-basierten Verfahren bietet es den Vorteil, dass beliebige Content-Objekte ohne aufwendige Schema-Matching-Verfahren, wie von Peukert et al. [PER12] beschrieben, integriert werden können. Die von Gröger vorgestellte Architektur zur Speicherung von strukturierten Daten basiert auf einem relational umgesetzten Data Warehouse, welches sich am Common-Warehouse-Metamodel-Standard, nach Poole et al. [PCTM03], orientiert. Für die Speicherung unstrukturierter Daten nutzt die Architektur ein Content-Management-System basierend auf dem CMIS-Standard. Der CMIS-Standard bietet eine universelle Schnittstelle zum Arbeiten mit Content-Management-Systemen [OAS12]. Beziehungen zwischen den Datenspeichern werden mit Links realisiert, welche zusätzliche Informationen über den Typ der Beziehung enthalten. Links werden in einem separaten Datenhaltungssystem, dem Link-Store, gespeichert. Gröger formuliert vier Anforderungen an seine Link-Store-Architektur: Die Architektur soll einerseits flexibel sein, d.h. neue Content-Objekte können ohne aufwändige Anpassungen des Schemas hinzugefügt werden. Links sollen die Verknüpfung mit zusätzlichen Informationen anreichern. Sie müssen einheitlich genutzt werden und nicht nur Verknüpfungen zwischen strukturierten und unstrukturierten Daten abbilden können, sondern auch bereits existierende Verknüpfungen innerhalb des Data Warehouses. Das bestehende Warehouse-Schema soll nicht verändert werden, so dass aufwändige ETL-Anpassungen ausbleiben und Analyseanwendungen nicht angepasst werden müssen.

Der daraus resultierende Lösungsansatz stellt Links als gerichtete, binäre, typisierte und attributierte Verknüpfung dar. Daraus entsteht eine graphenorientierte Sichtweise, wobei Daten-Objekte als Knoten und Links als Kanten betrachtet werden. Für die prototypische Implementierung wird von Gröger eine IBM DB2-Datenbank als Data Warehouse eingesetzt. Als Content-Management-System nutzt er Alfresco. Der Link-Store wird mithilfe einer Neo4j-Graphendatenbank realisiert. Die Anfrageschnittstelle wurde in Java implementiert, wobei auf das Data Warehouse mithilfe von SQL und auf das CMS durch Anfragen auf Basis des CMIS-Standards zugegriffen wird. Der Link-Store wird über die Neo4j-Anfragesprache Cypher angesprochen. Außer dem von Gröger vorgestellten Modell zur linkbasierten Verknüpfung existiert der von Thor et al. [TR11] vorgestellte iFuice- und CloudFuice-Ansatz, welcher Datenquellen mithilfe von Mapping-Skripten vereinigt.

Neben den vorgestellten, bereits diskutierten Technologien zur Verknüpfung von Daten existieren auch Architekturen sowie Prototypen für Firmenumgebungen, welche es zukünftig ermöglichen könnten, Daten, die während des Produktlebenszyklusses anfallen, besser zu verknüpfen und zu analysieren [Sch16]. Nach Jun et al. [JKX07] besteht ein Produktlebenszyklus aus drei Hauptphasen: **B**egin- (BOL), **M**iddle- (MOL), und **E**nd-**O**f-**L**ife (EOL). Wobei Begin-Of-Life den Entwicklungsprozess sowie das Herstellen des Produkts darstellt, Middle-Of-Life die Wartung und die Verteilung des Produkts und End-Of-Life die Entsorgung bzw. Zerlegung des Produkts. Im Laufe des gesamten Lebenszyklusses werden viele Produktinformationen

wie technische Dokumente, strukturierte Daten und unstrukturierte Daten erzeugt und verarbeitet . Der Ansatz des **c**losed-**l**oop-**P**roduct-**L**ifecycle-**M**anagements (cl-PLM) sieht nach Ameri et al. [AD05] vor, Daten, die beispielsweise während der MOL- und EOL-Phase erzeugt werden, zu strukturieren und im Begin-Of-Life des Produkts zu nutzen. Allerdings beinhaltet dieser Ansatz keine Analyse der Daten und unstrukturierte Daten können nicht genutzt werden. Von Kassner et al. [KGMW15] werden ApPLaUDING, eine Erweiterung der cl-PLM-Architektur, vorgestellt. Die Architektur ist in drei Ebenen unterteilt. Die Integrations-Ebene besteht aus einem ganzheitlichen Wissensspeicher, in dem die Daten nach dem Extrahieren durch Extrakt-Transform-Load-Funktionen gespeichert werden. Gröger et al. [GSM14] stellen einen ganzheitlichen Wissensspeicher, das **M**anufacturing **K**nowledge **R**epository (MKR) vor, der verschiedene Typen strukturierter und unstrukturierter Daten speichert. Die Daten werden mit Prozesskomponenten in Verbindung gesetzt. Die Analyse-Ebene beinhaltet einerseits klassische Analysemethoden für strukturierte Daten wie OLAP und domänenspezifische Textanalyse-Werkzeuge zum Anreichern der Daten. Andererseits beinhaltet diese Ebene ebenfalls komplexe, situationsabhängige spezifische Analysemethoden [LMA09]. Die dritte Ebene beinhaltet das **U**ser **I**nterface (UI) zum Präsentieren der Daten. Ein Konzept für eine weitere Architektur wird von Gröger et al. [GKH+16] vorgestellt - das **S**tuttgart **I**T **A**rchitecture for **M**anufacturing (SITAM). SITAM sieht vor, die Daten durch eine **S**ervice **O**riented **A**rchitecture (SOA) in das System einzubinden. SOA ist ein Softwaredesign, bei dem einzelne Funktionalitäten in Services ausgelagert werden und mit Kommunikationsprotokollen angebunden werden [LL09]. Die verschiedenen Produktlebenszyklen, Prozesse, IT-Systeme und Web-Interfaces werden über einen Produktlebenszyklus-Management-Bus eingebunden. Innerhalb der analytischen Middleware werden die Daten in einem MKR gespeichert und können durch zusätzliche Informationen mithilfe von Data-Mining und Machine Learning angereichert werden. Durch eine Mobile-Middleware können Arbeiter auf Dienste der Fabrik zugreifen. Es ist von essentieller Bedeutung für eine agile Produktion, dass auf die Resultate der Analyse schnell reagiert werden kann. Innerhalb der SITAM-Architektur können Dienste anhand von Nutzerrollen, Informationsbedarf und Zugriffsrechten angefragt werden.

Die vorgestellten Architekturen liefern wichtige Bausteine für die Realisierung zukünftiger Lösungen zur Verknüpfung und Speicherung von unstrukturierten und strukturierten Daten rund um den Produktlebenszyklus.

4 Anforderungen

In diesem Kapitel werden die Anforderungen an diese Arbeit aufgeführt. Sie sind unterteilt in fachliche, methodische und technische Anforderungen.

4.1 Fachliche Anforderungen

In diesem Unterkapitel werden die fachlichen Anforderungen an den in dieser Arbeit durchgeführten Vergleich von aktuellen Technologien zur Verknüpfung von strukturierten und unstrukturierten Daten aufgeführt.

4.1.1 Betrachtung von Open-Source-Technologien

Da die Softwarelösungen zum Verknüpfen von strukturierten und unstrukturierten Daten in Industrieumgebungen eingesetzt werden sollen, ist es notwendig, dass diese die Software erweitern und modifizieren können. Um das zu gewährleisten, ist es eine fachliche Anforderung an die untersuchten Softwarelösungen, dass diese unter Open-Source-Lizenzen veröffentlicht sind.

4.1.2 Definition von Bewertungskriterien

Für den Vergleich der verschiedenen Softwaresysteme ist es notwendig, Kriterien zu definieren. Die Kriterien sollen auf Basis von verwandten Arbeiten und den Erkenntnissen aus der Literaturrecherche aufgestellt werden. Neben der Verknüpfung von strukturierten und unstrukturierten Daten als Hauptkriterium ist auch die Vereinigungsmenge von Merkmalen der verschiedenen Softwaresysteme Basis für den Kriterienkatalog. Diese ergeben sich implizit aus der Evaluation der Softwarelösungen.

4.1.3 Bewertung der Softwarelösungen

Die Softwarelösungen sollen mit den zuvor aufgestellten Kriterien bewertet werden. Als Darstellungsform ist eine Tabelle zu wählen, die die Bewertung der Softwarelösungen übersichtlich darstellt.

4.1.4 Anwendungsfall: Selektion, Installation, exemplarische Demonstration

Für den in Abschnitt 4.4 definierten Anwendungsfall soll auf Basis der Vergleichstabelle die Selektion einer passenden Softwarelösung erfolgen. Anschließend soll diese installiert und die Verknüpfung von strukturierten und unstrukturierten Daten demonstriert werden.

4.1.5 Beispieldaten für den Anwendungsfall

Für den Anwendungsfall sollen Dummy-PDF-Daten in Verbindung mit Tupeln aus einer relationalen Datenbank verwendet werden, um die Verknüpfung von strukturierten und unstrukturierten Daten zu demonstrieren.

4.2 Methodische Anforderungen

Die initialen Kriterien sollen mittels einer Literraturrecherche erarbeitet werden. Zur Definition weiterer Kriterien soll durch die Evaluierung der Softwarelösungen explorativ vorgegangen werden.

4.3 Technische Anforderungen

Für diese Arbeit ist es notwendig, Zugänge zu Arbeitsräumen und Arbeitsrechnern zu beschaffen. Gleichzeitig wird Zugriff auf den GSaME-Cluster benötigt, um innerhalb einer virtuellen Maschine die entsprechenden Softwarelösungen zu installieren und zu evaluieren.

4.4 Anwendungsfall

Der Automobilzulieferer „xy“ besitzt eine Fabrik, in der verschiedene Fertigungslinien getestet werden. Später werden diese an anderen Standorten produktiv eingesetzt. Momentan treten verschiedene Probleme innerhalb der Fertigungslinien während der Produktion auf. Diese werden von Produktionsarbeitern anhand eines Formulars in Papierform händisch erfasst und falls vorhanden direkt auch die Lösung beschrieben. Um das Wissen standortübergreifend allen Produktionsarbeitern, die Lösungen für bereits dokumentierte Fehler suchen, zur Verfügung zu stellen, wird ein IT-System benötigt. Die Formulare mit den Problembeschreibungen und -lösungen wurden daher bereits digitalisiert und als PDF-Dokumente in einem Dateisystem abgelegt. Informationen zu den Fertigungslinien befinden sich jedoch im **M**anufacturing **E**xecution **S**ystem (MES) und weitere Daten zu den Fertigungsaufträgen im

Enterprise-**R**esource-**P**lanning System (ERP). Um nun die Fehlerdaten und Lösungen zu den verschiedenen Maschinen und Fertigungsaufträgen zuordnen zu können, müssen die Daten verknüpft werden. Die MES- und ERP-Daten sind strukturiert, liegen in relationaler Form (Tabellen) vor und können durch die Anfragesprache SQL integriert werden. Die Fehlerberichte sind jedoch unstrukturiert und können nicht per SQL mit den Daten aus ERP- und MES-System integriert werden. Das Unternehmen steht nun vor der Aufgabe, eine passende Softwarelösung zu finden, die es ermöglicht, die Fehlerberichte mit den relationalen Daten zu verknüpfen, um diese später über ein werkübergreifendes IT-System allen Produktionsarbeitern zur Verfügung zu stellen. Das Ziel ist der Aufbau einer holistischen Wissensdatenbank für die Fertigung.

5 Auswahl von Lösungen

In diesem Kapitel werden Technologien zum Verwalten von strukturierten und unstrukturierten Daten beleuchtet. Für das Auswählen geeigneter Lösungen werden Kriterien definiert, welche die zu betrachtenden Systeme erfüllen müssen.

5.1 Kriterien für die Wahl der zu betrachtenden Technologien

Bei der Wahl relevanter Technologien gibt es verschiedene Voraussetzungen. Zum einen müssen die Lösungen unter Open-Source-Lizenzen veröffentlicht worden sein, zum anderen ist es wichtig, dass die Softwarelösungen die Geheimhaltung der gespeicherten Daten gewährleisten sowie die Möglichkeit bieten, unstrukturierte sowie strukturierte Datenspeicher anzubinden um die Daten zu verknüpfen.

5.1.1 Anbindung von Technologien zur Speicherung von strukturierten Daten

Die Technologien müssen es ermöglichen, strukturierte Daten anzubinden. Dies kann entweder durch das Anbinden weiterer Software durch das Bereitstellen von Schnittstellen geschehen oder durch einen in das System integrierten Speicher.

5.1.2 Anbindung von Technologien zur Speicherung von unstrukturierten Daten

Neben strukturierten Daten muss es außerdem möglich sein, unstrukturierte Daten zu nutzen. Diese können analog zu strukturierten Daten entweder intern oder extern durch eine Schnittstelle an das System angebunden werden.

5.1.3 Verfügbarkeit als Open-Source-Software

Die genutzten Technologien müssen unter einer Open-Source-Lizenz, wie in Kapitel 2 beschrieben, veröffentlicht worden sein.

5.2 Verwaltung strukturierter und unstrukturierter Daten

Es gibt verschiedene Lösungen zur Verwaltung strukturierter und unstrukturierter Daten. Im folgenden Abschnitt werden verschiedene Tools zur Verwaltung strukturierter und unstrukturierter Daten vorgestellt und auf ihre Eigenschaften untersucht. Die ausgewählten Tools stellen eine Vorauswahl dar, welche in Abschnitt 5.2.11 verfeinert wird. Die Technologien der Vorauswahl werden anhand ihrer Funktionalität, der verfügbaren Schnittstellen, deren Datenmodell und der Suchfunktion beschrieben.

5.2.1 Aletheia

Aletheia ist ein Forschungsprojekt, das im Verbund mit der AAB AG [AAB11] durchgeführt wurde. Das Ziel des Projekts war es, strukturierte Daten und unstrukturierte Daten zu Wissen zu verknüpfen, so dass Arbeiter bei ihren Tätigkeiten unterstützt werden. Dafür werden Daten aus verschiedenen strukturierten und unstrukturierten Quellen an das System angebunden. Zum einen wird das Web mithilfe eines Crawlers nach relevanten Informationen durchsucht, zum anderen werden Datenbanken, betriebliche Bibliotheken und Applikationsinformation an das System angebunden. Die Daten werden anhand einer Serviceontologie in Zusammenhang gesetzt, um so bei einer Suche relevante Informationen zu finden. Da die Software nicht öffentlich zur Verfügung steht, können die nachfolgenden Kriterien nur anhand der Dokumentation spezifiziert werden.

Funktionalität

Aletheia bietet die Funktion, unstrukturierte Daten mit strukturierten Daten zu verknüpfen. Anhand der Verknüpfung lassen sich anschließend intelligente Suchanfragen absetzen, um relevante Informationen zu erhalten. Zusätzlich wurde ein RFID-Tag-System zur eindeutigen Geräteidentifikation entwickelt. Dadurch können Arbeiter mit maschinenspezifischen Informationen versorgt werden und es erleichtert die Kommunikation mit dem Aletheia-Hauptsystem [AAB11].

Schnittstellen

Das Aletheia-System wurde mit verschiedenen Datenquellen verbunden. Es wurden Firmen-Datenbanken mit strukturierten Daten eingebunden sowie File Server und lokale Netzwerk-Shares mit unstrukturierten Daten. Zusätzlich wurde eine Web-Crawling-Komponente entwickelt, um Daten aus dem Internet zu extrahieren [AAB11].

Datenmodell

Die Daten wurden in Aletheia innerhalb einer Ontologie organisiert, dabei kamen OntoStudio[1] 2.3.3 und OntoBroker[1] 5.3.3 zum Einsatz [AAB11].

Integrierte Suche

Die in Aletheia eingesetzte facettierte Suche ist Teil des im Rahmen des Projekts entwickelten Frontends. Bei einer facettierten Suche werden die Suchtreffer mithilfe verschiedener Facetten (z.B Metadaten) eingegrenzt [AAB11].

5.2.2 Alfresco

Alfresco[2] ist ein ECMS, das von Alfresco Software Limited entwickelt wurde. Alfresco wird laut Dokumentation [Alf17] in drei Versionen vertrieben: Alfresco Community Edition, Alfresco One und Alfresco in the Cloud. Die **C**ommunity **E**dition (CE) von Alfresco ist Open Source und wird unter der LGPLv3-Lizenz vertrieben. Alfresco One ist eine kostenpflichtige, an Unternehmen gerichtete, Alternative zur Community-Version und bietet zusätzliche Features wie stärkere Skalierung, einfachere Administration, größerer Add-On-Umfang und 24-Stunden-Support. Die Cloud Variante von Alfresco richtet sich an Unternehmen, die Alfresco One als **S**oftware-**a**s-**a**-**S**ervice (SaaS) einsetzen möchten. SaaS ist ein Servicemodell, bei dem ein IT-Dienstleister Software zentral zur Verfügung stellt und als Dienstleistung anbietet [GMKM04]. In der Community- und der One-Version kann Alfresco vollständig auf privaten Servern betrieben werden. Im Rahmen dieser Arbeit wird die Community-Version betrachtet, da nur diese als Open Source veröffentlicht wird. Deshalb wird aus Gründer der besseren Lesbarkeit in dieser Arbeit nur der Begriff Alfresco verwendet. Alfresco wird über eine Weboberfläche (Alfresco Share) genutzt. Darin lassen sich verschiedene Nutzer und Gruppen definieren, wodurch es ermöglicht wird, den Zugriff auf verschiedene Daten einzuschränken. Alfresco ist auf die Verwaltung von unstrukturierten Daten wie Bilder, E-Mails etc. ausgerichtet. Jedoch können mit dem MetaDBConnector Plugin relationale Datenbanken angebunden

[1] http://www.semafora-systems.com/
[2] https://www.alfresco.com/

werden, dieses Plugin ist unter der APL Open Source. Über das Record-Management Plugin kann beispielsweise das Anbinden von Metadaten an Dokumente vereinfacht werden [Alf17].

Funktionalität

- Dokumenten-Management
- Prozess-Management
- Enterprise-Kollaboration
- Mobiles Content-Management
- Plugins

Alfresco beinhaltet unterschiedliche Funktionen: Ein Dokument-Management-System zum Verwalten und Durchsuchen von Dokumenten. Die Möglichkeit, Dokumente bestimmten Nutzern oder Gruppen zur Verfügung zu stellen. Die Option, Regeln zum automatischen Verarbeiten von Daten zu definieren, um zum Beispiel automatisch auf die Dokumente ein Business Process Model anzuwenden. Durch den Enterprise-Kollaborationsaspekt der Software ist es für Teams einfacher zusammenzuarbeiten. Teammitglieder haben die Möglichkeit mithilfe von Kommentaren Feedback zu geben und können Daten auf sozialen Netzwerken teilen [Alf17].

Schnittstellen

Alfresco verfügt über ein Plugin-Interface, über das Erweiterungen eingebunden werden können. Es gibt eine Plugin-Bibliothek[3], über die verschiedene Plugins bezogen werden können. Dadurch können weitere Features und Schnittstellen hinzugefügt werden. Relationale Datenbanken können in Alfresco durch ein Plugin angebunden werden, nativ steht diese Funktionalität nicht bereit. Zusätzlich bietet Alfresco eine CMIS- sowie eine WebDav-Schnittstelle. WebDav (**Web**-based **D**istributed **A**uthoring and **V**ersioning) ist ein Standard, der es ermöglicht, Daten über das HTTP-Protokoll zu verwalten und zu übertragen [Alf17].

Integrierte Suche

Alfresco beinhaltet eine Suche, die in der Lage ist, Dokumenteninhalte zu durchsuchen basierend auf Apache Solr [Alf17]. Solr ist eine Open-Source-Suchplattform auf Basis von Apache Lucene[4].

[3] https://addons.alfresco.com/
[4] http://lucene.apache.org/solr/

Datenmodell

Alfresco nutzt zur Verwaltung der Dokumente eine Datenbank. Bei der Installation ist es möglich, eine MySQL-, Postgres-, Oracle-, DB2-Datenbank oder einen Microsoft SQL Server zu nutzen. In dieser Datenbank werden Alfresco spezifische Informationen hinterlegt. Die eigentlichen Daten werden in Alfresco innerhalb eines File-Content-Stores abgelegt. Der File-Content-Store speichert die Dokumente in einer Ordnerstruktur, abhängig vom Erstellungsdatum. Alternativ zum File-Content-Store können Daten in einen Amazon Simple-Storage-Service (S3), Caching-Content-Store, Aggregating-Content-Store, Encrypted-Content-Store oder einen Centra-Content-Store ausgelagert werden. Neben diesen Speichermöglichkeiten bietet Alfresco zusätzlich die Möglichkeit, über Plugins weitere Stores anzubinden [Alf17].

5.2.3 OpenKM

OpenKM[5] ist ein DMS, das von Open Document Management System S.L. entwickelt wurde. Wie Alfresco wird OpenKM gemäß der Dokumentation [Ope17] in einer Community-, Cloud- und professionellen Variante vertrieben. Auch hier ist nur die Community-Version interessant, da diese als einzige unter GPLv2 quelloffen ist. OpenKM bietet Funktionen um unstrukturierte Daten mit Metadaten zu ergänzen, verfügt über eine Workflow-Engine, ein Record-Management-System und lässt sich durch Plugins erweitern. Zusätzlich bietet OpenKM Features wie etwa einen Scanner Client, um Dokumente zu digitalisieren und in das DMS aufzunehmen sowie einen Barcodereader, um Barcodes aus Dokumenten zu extrahieren und zu identifizieren. Allerdings ist es nicht möglich, nativ strukturierte Daten z.B aus einer Datenbank an das System anzubinden [Ope17].

Funktionalität

- Dokumenten-Management
- Automatisierte Tasks
- Record-Management
- Workflow-Engine
- Module

OpenKM bietet die Möglichkeit, Dokumente zu verwalten und automatisch zu verarbeiten. Zusätzlich verfügt OpenKM über eine Workflow-Engine, mit der Dokumente einfacher und effizienter verarbeitet werden können. Metadaten können durch das Record-Management

[5] https://www.openkm.com/

verwaltet werden und mit Modulen kann die Software um weitere Funktionalitäten erweitert werden [Ope17].

Schnittstellen

OpenKM kann mithilfe von Modulen erweitert werden, um weitere Schnittstellen zur Verfügung zu stellen. Mithilfe des „OpenCMIS-Connectors“ ist es möglich, OpenKM um den CMIS-Standard zu erweitern [Ope17].

Integrierte Suche

In OpenKM steht eine integrierte Suchfunktion zur Verfügung, die nach Dokumenten und Verzeichnissen sowie Inhalten und Metadaten suchen kann. Die Basis dieser Suche bildet Apache Lucene [Ope17].

Datenmodell

Zum Verwalten der Daten nutzt OpenKM eine relationale Datenbank wie MySQL oder PostgresSQL. Die Dokumente werden im Filesystem des **O**perating **S**ystems (OS) abgelegt [Ope17].

5.2.4 CDH

Cloudera **D**istribution Including Apache **H**adoop (CDH) [6] ist eine auf dem Hadoop-Ecosystem basierende Lösung zum Speichern und Analysieren von strukturierten und unstrukturierten Daten, die von Cloudera entwickelt wurde. CDH ist ein Softwarepaket, das Open-Source-Software unter der Apache Software License 2.0 enthält. Neben CDH bietet Cloudera, wie in der Produktdokumentation [Clo17a] beschrieben, eine proprietäre Software Cloudera Enterprise, welche CDH um den Cloudera Manager erweitert. Dadurch wird ein einfacheres Verwalten, Konfigurieren und Updaten der eingesetzten Apache Softwaretools ermöglicht [Clo17a].

Funktionalität

CDH bietet die Möglichkeit, Daten durch verschiedene Softwarelösungen im Hadoop-Filesystem zu verwalten, zu speichern und zu durchsuchen [Apa17; Clo17a; Clo17b]:

- Mithilfe von Apache Flume können Log- und Event-Daten direkt an Hadoop gestreamt werden.

[6] https://www.cloudera.com/products/open-source/apache-hadoop/key-cdh-components.html

- Apache Kafka ermöglicht das Streamen und Verarbeiten von Daten in HDFS.
- Apache HBase ist ein skalierbarer, verteilter NoSQL-Store, bei dem Daten in Tabellen gespeichert sind.
- Apache Accumulo ist ein in HDFS integrierter Key-Value-Store.
- HUE ist ein UI, um die verschiedenen Softwaresysteme zu bedienen.
- Apache Pig ist eine Programmierumgebung, um in Hadoop gespeicherte Daten zu verarbeiten.
- Apache Impala bietet die Möglichkeit, SQL-Queries an die Daten in HDFS oder HBase zu stellen.
- Apache Hive ermöglicht es, SQL-ähnliche Queries auf die im HDFS gespeicherte Daten abzusetzen.
- Apache Spark bietet ein Framework zum Batch- und Stream-Processing von Daten.
- Cloudera Search bietet eine Volltextsuche für gespeicherte Daten.
- Apache Sqoop bietet die Möglichkeit, relationale Datenbanken in das Hadoop-Filesystem zu integrieren.
- Apache Sentry bietet ein rollenbasiertes Rechtesystem für die einzelnen Apache Tools.
- Apache Zookeeper erleichtert die Koordination von Aufgaben zwischen Cluster-Nodes.
- Apache Oozie bietet ein System, um Hadoop-Jobs auf den verschiedenen Nodes zu überwachen.

Schnittstellen

Durch die verschiedenen Softwaretools bietet CDH mehrere Schnittstellen. Mithilfe von Apache Flume und Apache Kafka können Log- und Eventdaten in das System integriert werden, durch Sqoop können relationale Datenbanken an das System angebunden werden und durch Apache Pig, Hive, Spark und MapReduce können Daten im Hadoop-Filesystem ausgewertet werden. Apache Spark bietet Stream- und Batch-Processing [Apa17].

Integrierte Suche

CDH enthält Cloudera Search, eine auf Apache Solr basierende Suchmaschine [Clo17a].

Datenmodell

CDH nutzt als Basis das Hadoop-Filesystem. Dadurch können sehr große Datenmengen verwaltet werden. Da verschiedene Knoten an das Hadoop-System angebunden werden können, skaliert das System horizontal. Es ist möglich, Daten im HDFS durch HBase in Tabellen zu organisieren. Zusätzlich können mithilfe des MapReduce-Programmiermodells die Daten verteilt verarbeitet werden [Clo17a].

5.2.5 Brown Dog

Brown Dog[7] ist ein vom National Center for Supercomputing Applications [Nat17] an der amerikanischen University of Illinois entwickeltes System zum Aufbereiten von unstrukurierten Daten, um diese in ein lesbares Format zu bringen. Brown Dog ist in der Lage Daten zu analysieren, Metadaten zu extrahieren und Daten in verwertbare Formate zu transformieren. Momentan befindet sich Brown Dog im „Friendly User Mode“[7], dies bedeutet, dass nur manuell bestätigte Accounts Zugriff zur Software erhalten. Dadurch ist es nicht möglich auf die Features und Eigenschaften der Software näher einzugehen .

5.2.6 Globus Plattform

Die Globus Plattform[8] ist eine Cloud-Lösung zum dezentralen Speichern und Teilen von Daten, die von der University of Chicago [Uni17] entwickelt wurde. Globus stellt dabei eine API zum sicheren Einloggen und Transferieren von Daten zur Verfügung, wobei Globus den zentralen Service bereitstellt, der Daten aus verschiedenen Endpunkten und Anwendungen verknüpft. Dadurch soll gemeinsames, internationales Forschen und Arbeiten erleichtert werden. Die gespeicherten Daten können mit Metadaten angereichert und nach diesen durchsucht werden. Mithilfe des Globus Connect Tools oder dem Globus Python-SDK können weitere Endpunkte an das System angebunden werden. Globus Connect und Globus Python-SDK sind Open Source unter der Apache Public License 2.0 und der Globus Toolkit Public License. Die Globus Plattform basiert auf dem Globus Toolkit, ein Toolkit für Gridcomputing. Das Toolkit bietet Software zum verteilten Speichern und Verarbeiten von Daten und ist ebenfalls Open Source unter der Apache License 2.0 und der Globus Toolkit Public License. Da die Globus Plattform die proprietäre Globus API nutzt, ist es nicht möglich ein vollständig autonomes System zu entwerfen.

[7] http://browndog.ncsa.illinois.edu/
[8] https://www.globus.org/platform

Funktionalität

Globus bietet die Möglichkeit, Daten aus dem System jederzeit an jedem Ort abzurufen. Dafür muss lediglich eine Verbindung zur Globus Cloud bestehen. Mithilfe dieses Systems lassen sich Teile der Analyse automatisieren. Globus Connect bietet eine Schnittstelle zum Verbinden mit dem Globus Cluster. Mithilfe von Globus Server lassen sich weitere Knoten zum Globus System hinzufügen [Uni17].

Schnittstellen

Globus bietet eine REST-API, um den Fortschritt von Übertragungen zu überwachen, neue Dateien zu übermitteln und Knoten zu durchsuchen [Uni17]. Eine REST-API nutzt statuslose HTTP-Anfragen, um mit einem anderen System zu kommunizieren [LC11].

Integrierte Suche

Globus hat ein integrierte Suche. Die gespeicherten Dokumente können anhand von automatisch erzeugten Metadaten durchsucht werden [Uni17].

Datenmodell

Zum Speichern der Daten nutzt die Globus Plattform das Globus Toolkit, ein Framework zum Speichern und Verarbeiten von Daten im Computer-Grid [Uni17].

5.2.7 Open Semantic Framework

Das **O**pen **S**emantic **F**ramework (OSF)[9] ist ein Softwarestack, der von Structured Dynamics LLC [Str17] entwickelt wurde. Ein Softwarestack ist eine Zusammenstellung von Softwaresystemen, die gemeinsam eine vollständige Plattform bilden [Upw17]. Der Stack vereint verschiedene Technologien, um Daten semantisch zu organisieren. OSF ist unter der Apache Public License 2.0 Open Source. Das Open Semantic Framework bietet die Möglichkeit, Daten aus un-, semi- und strukturierten Quellen zu bündeln und zu durchsuchen. Der Zugriff auf die Daten kann durch ein Rechtemanagement verwaltet werden. Eine Weboberfläche von OSF ist als Plugin in dem WCMS Drupal[10] 7 integriert .

[9] http://opensemanticframework.org/
[10] https://www.drupal.org/

Funktionalität

- Semantische Suche
- Veröffentlichen und Verwalten der Daten
- Verteiltes Speichern von Daten

OSF bietet die Möglichkeit, verschiedene Endpunkte zum Speichern von Daten anzubinden. Die Daten können in Ontologien strukturiert und mithilfe von Drupal veröffentlicht werden. Zudem können die Daten durch eine semantische Suche auf Basis von Apache Solr durchsucht werden [Str17].

Schnittstellen

Daten können über eine REST-API abgerufen werden. OSF bietet ein Plugin für das Drupal WCMS, über das auf das System zugegriffen werden kann [Str17].

Integrierte Suche

Das Open Semantic Framework bietet eine integrierte Suche, die mithilfe von Apache Solr umgesetzt wurde [Str17].

Datenmodell

Die Daten innerhalb von OSF werden als RDF-Tripels in einem Virtuoso Server[11] gespeichert [Str17].

5.2.8 Elastic Stack

Der Elastic Stack[12] ist ein Softwarestack, der von Elasticsearch [Ela17] entwickelt wurde und unter der Apache License 2.0 Open Source ist. Er beinhaltetet Logstash und die Beats-Plattform, um verschiedene Datenquellen anzubinden, wobei der Fokus auf Log-Files liegt. Die Dateien werden in ein geeignetes Format transformiert und mit Elasticsearch durchsuchbar gemacht. Die Daten können aus verschiedenen Quellen wie z.B. Log-Dateien, Netzwerkdaten, System-Metriken stammen und mithilfe der Beats-Plattform angebunden werden. Kibana stellt eine Weboberfläche zur Konfiguration und Verwaltung des Stacks bereit. Innerhalb von Kibana

[11] https://virtuoso.openlinksw.com/
[12] https://www.elastic.co/products

können verschiedene Visualisierungen der gefilterten Daten erzeugt werden, um diese zu präsentieren.

Funktionalität

- Anbindung verschiedener Datenstream-Quellen
- On-the-fly Transformation
- Visualisierung und Analyse der Daten

Der Elastic Stack bietet die Möglichkeit, verschiedene Datenstream-Quellen anzubinden und diese on-the-fly zu transformieren. Die Daten können innerhalb von Kibana gefiltert und visualisiert werden [Ela17].

Schnittstellen

Der Elastic Stack ermöglicht es, verschiedene Datenquellen mithilfe von Plugins anzubinden. Diese können entweder selbst entwickelt werden oder aus einer Bibliothek bezogen werden. Ebenso können verschiedene Plugins zur Anbindung von Datenspeichern genutzt werden [Ela17].

Integrierte Suche

Der Elastic Stack beinhaltet Elasticsearch. Elasticsearch ist eine Suche über verschiedene Datenspeicher. Die Suchanfragen werden in Form von json-Statements gestellt [Ela17].

Datenmodell

Der Elastic Stack bietet Plugins, um Daten auf verschiedenen Datenspeichern abzulegen, dazu zählen Hadoop und relationale Datenbanken [Ela17].

5.2.9 Apache Marmotta

Apache Marmotta[13] ist eine quelloffene Link-Dataplattform der Apache Software Foundation [Apa17], die unter der Apache License 2.0 veröffentlicht ist. Marmotta ermöglicht das Aufbauen von Ontologien auf Basis eines Tripel-Stores.

[13] http://marmotta.apache.org/

Funktionalität

- Anbindung an den Java EE Stack
- Linked-Data-Plattform Querying
- **S**PARQL **P**rotocol **a**nd **R**DF **Q**uery **L**anguage (SPARQL) Querying
- LDPath Querying

Marmotta ist eine Plattform für Linked Data, welche vor allem im Rahmen des Java EE Stacks eingesetzt werden kann. Sie bietet SPARQL Querying und experimentellen Support für die Linked-Data-Plattform 1.0 Spezifikation. Auf unstrukturierte Daten kann innerhalb der Ontologie verwiesen werden. Strukturierte Daten müssen in RDF transformiert werden.

Schnittstellen

Apache Marmotta verfügt über eine SPARQL Query-Schnittstelle zum Absetzen von Queries. Außerdem können die Daten mithilfe von LDPath und **L**inked-**D**ata-**P**lattform 1.0 (LDP) angesprochen werden. LDP ermöglicht es, mithilfe von HTTP-Anfragen, Ressourcen aus dem Web zu lesen und zu schreiben, dadurch können Webinhalte an Marmotta angebunden und als solche zur Verfügung gestellt werden. LDPath ist eine Query-Sprache ähnlich zu SPARQL um Daten anzufragen, allerdings sind die Anfragen nicht SQL-ähnlich, sondern mit Java-Programmen vergleichbar [Apa17].

Integrierte Suche

Marmotta bietet abgesehen von den Query-Schnittstellen keine eigene Suchfunktion [Apa17].

Datenmodell

Marmotta bietet die Möglichkeit, verschiedene Tripel-Stores einzusetzen. Für das Backend wird standardmäßig ein KiWi-Tripel-Store genutzt. Dieser speichert die Daten letztendlich innerhalb einer relationalen Datenbank. Es können jedoch auch andere Stores eingesetzt werden wie Sesame Native, bei dem die Daten im Dateisystem des OS abgelegt werden oder einen experimentellen Big-Data-Store für große Datensätze [Apa17].

5.2.10 Übersicht

Die Tabelle 5.1 stellt eine Übersicht der vorgestellten Technologien dar. Dabei werden die Systeme nach folgenden Kriterien bewertet:

- Typ

  Der Typ beschreibt nach welchen Architektur-Grundsätzen die Software entworfen wurde.

- Sicherheit

  Die Sicherheit der Software beschreibt, ob die Daten in einer externen Cloud oder intern gespeichert werden und ob ein Mechanismus zum Schutz der Daten existiert.

- Lizenz

  Die Lizenz beschreibt, ob die Software Open Source ist und unter welchen Bedingungen.

- Verknüpfung strukturierter und unstrukturierter Daten

  Dieses Kriterium beschreibt, ob die Technologie in der Lage ist, strukturierte und unstrukturierte Daten zu verknüpfen.

	Aletheia	OpenKM	Alfresco	CDH	Brown Dog	Globus	O-S-F	Elastic	Marmotta
Typ	ontologie-basiert	DMS	ECMS	Hadoop	Metadata Analyse	Gridstore	Linked Data	Transformation, Visualisierung	Ontologie basiert
Sicherheit	unbekannt	lokal, Rechte-system	lokal, Rechte-system	lokal, Rechte-system	unbekannt	remote, Rechte-system	lokal, Rechte-system	lokal, Rechtesystem	lokal, Rechte-system
Lizenz	unbekannt	GPLv2	LGPLv3 +ASLv2	ASLv2	unbekannt	proprietär ASLv2 GTPL	GPLv2 ASLv2	ASLv2	ASLv2
Verknüpfung strukt./ unstrukt Daten	-	✗	✓	✓	-	✗	✓	✗	✓

Tabelle 5.1: Vorauswahl der Systeme

5.2.11 Evaluation

Da das Aletheia-System nicht öffentlich verfügbar und somit nicht Open Source ist, kann es nicht weiter analysiert werden. Auch Brown Dog ist momentan nicht öffentlich verfügbar und scheidet deshalb ebenfalls aus einer weiteren Analyse aus.

Neben Alfresco und OpenKM existieren weitere ECM- und DM-Systeme wie LogicalDoc, BitFarm-Archive etc. OpenKM und Alfresco werden repräsentativ für ECM- und DM-Systeme betrachtet, da die anderen Systeme keinen Mehrwert bezüglich der Verknüpfung von unstrukturierten Daten mit strukturierten Daten bieten. Auch das Dokument-Management-System OpenKM unterscheidet sich von Alfresco nur wenig. OpenKM ist stärker an die Verwaltung und Analyse von Dokumenten angepasst, während in Alfresco der Kollaborationsaspekt zwischen Arbeitern stärker im Mittelpunkt steht. Da OpenKM keine Möglichkeit bietet, strukturierte Daten an das System anzubinden, ist es nicht mehr Teil der weiteren Auswertung. Stattdessen wird Alfresco näher betrachtet.

CDH ist eine sinnvolle Kombination existierender Tools des Hadoop-Ecosystems. Diese Technologie ist unter der Apache-Lizenz Open Source und somit auch im industriellen Kontext gut einsetzbar. Durch die verschiedenen Hadoop-Tools ist es möglich, Verknüpfungen zwischen den Daten zu erstellen und zu verwalten. Daher wird CDH im nächsten Kapitel genauer betrachtet.

Globus bietet ein System zum Durchsuchen und Verwalten von Daten-Endpunkten, wobei die Accountverwaltung zentral durch Globus erfolgt. Dies beeinträchtigt die Sicherheit der Daten. Da Globus keine strukturierten Daten direkt anbinden kann und nicht vollständig lokal einsetzbar ist, scheidet es aus der weiteren Analyse aus.

Elastic Stack ist stark auf Log- und Event-Files ausgerichtet und unterstützt andere Daten nur bedingt. Da es keine Möglichkeit besitzt, strukturierte und unstrukturierte Daten zu verknüpfen, ist es nicht Teil der weiteren Evaluierung.

Da OSF während der Installation und Evaluierung viele Probleme verursacht hat und nicht einsatzfähig ist, scheidet es ebenfalls aus der weiteren Auswertung aus.

Neben Apache Marmotta existieren weitere semantische Web-Ansätze wie Callimachus und CubicWeb. Diese bieten allerdings keine Vorteile für die Verknüpfung von unstrukturierten mit strukturierten Daten, weshalb Marmotta stellvertretend für Linked-Data-Systeme betrachtet wird.

Für die detaillierte Analyse verbleiben nach der Grobselektion Apache Marmotta, CDH und Alfresco CE.

6 Festlegung von Kriterien für den weiteren Vergleich

In diesem Kapitel werden Kriterien zur Analyse der ausgewählten Softwaresysteme definiert. Diese Kriterien sind unterteilt in funktionale und nicht funktionale Anforderungen.

6.1 Funktionale Kriterien

Funktionale Kriterien beschreiben, was ein System leistet. Die Qualität dieser Kriterien lässt sich objektiv beschreiben. Im nachfolgenden Abschnitt werden die funktionalen Kriterien zur Bewertung der Softwaresysteme aufgeführt.

6.1.1 Verknüpfungsmethodik

Mit diesem Kriterium werden die Möglichkeiten zur Verknüpfung der Daten zusammengefasst, dies ist das Kernkriterium dieser Arbeit. Hierbei wird darauf eingegangen, welche Möglichkeiten zur Verknüpfung von unstrukturierten Daten mit strukturierten Daten zur Verfügung stehen. Bei der Verknüpfung von Daten ist es wichtig mithilfe der Beziehungen zwischen den Daten, weitere Informationen zu erschließen, um so Daten semantisch anzureichern.

6.1.2 Datenmodell

Das Datenmodell beschreibt, wie die Daten in der untersuchten Technologie gespeichert werden. Dieses Kriterium stammt aus dem Vergleich von Big-Data-Lösungen von Gölzer, Cato und Amberg [GCA15].

6.1.3 Verwaltung von Metadaten

Dieses Kriterium beschreibt, wie die untersuchten Technologien Metadaten an die gespeicherten Daten anbinden und welche Möglichkeiten zur Verwaltung von Metadaten zur Verfügung stehen. Dies ist ein wichtiges Kriterium, da Metadaten gerade bei der Verknüpfung von Daten wichtig sind, da diese zusätzliche Informationen über die Verknüpfung beziehungsweise die Daten geben und für Auswertungen sehr wichtig sein können.

6.1.4 Unterstützung von Analysen

Um Entscheidungsfragen lösen zu können, ist es wichtig, dass die Daten analysiert werden können. Insbesondere ist es von Bedeutung, die Verknüpfungen in die Analyse miteinzubeziehen. Dieses Kriterium beschreibt, welche Möglichkeiten zur Analyse der Daten in den Technologien zur Verfügung stehen.

6.1.5 Integrierte Suche

Um die gespeicherten Daten nutzbar zu machen, ist es wichtig, dass diese durchsucht werden können. Integrierte Suche beschreibt, ob das System eine Möglichkeit zum Durchsuchen der verknüpften Daten anbietet und welche zusätzlichen Funktionen ggf. zur Verfügung stehen. Dieses Kriterium wird durch den in Abschnitt 4.4 definierten Anwendungsfall implizit gefordert, da für die Produktionsarbeiter die Suche nach Daten wichtig ist.

6.1.6 Schnittstellen

Die Schnittstellen, welche die Software bietet, sind wichtig, da durch diese das System erweitert werden kann. Dadurch können entweder weitere Daten angebunden werden oder besser auf die bestehenden Daten zugegriffen werden. Das Kriterium Schnittstellen beschreibt die verschiedenen Möglichkeiten zur Kommunikation mit dem System.

6.1.7 Skalierbarkeit

Die Skalierbarkeit ist ein wichtiges Kriterium, vor allem, wenn die Software in großem Umfang eingesetzt werden soll. Dabei beschreibt Skalierbarkeit, ob die untersuchte Lösung horizontal oder vertikal skaliert. Ein vertikal skalierendes System beschreibt Software, welche auf einem einzelnen Knoten ausgeführt wird und durch das Hinzufügen von Ressourcen typischerweise in Form von Arbeitsspeichern oder weiteren CPU-Kernen zusätzliche Performanz gewinnt. Horizontal skalierende Systeme beschreiben Software, die durch Hinzufügen von Knoten eine

höhere Performanz erzielt. Dieses Kriterium wurde ebenfalls in der Arbeit von Gölzer et al. [GCA15] in ihrem Vergleich zu Big-Data-Lösungen verwendet.

6.2 Nicht funktionale Kriterien

Darunter versteht man Kriterien, die nicht objektiv messbar sind und subjektive Eindrücke während des Umgangs mit dem System beschreiben.

6.2.1 Lizenz

Eine Anforderung an diese Arbeit ist es, dass Open-Source-Lizenzen verwendet werden, deshalb gilt es, die Software durch ihre Lizenzen zu unterscheiden. Dieses Kriterium beschreibt die eingesetzten Lizenzen. Dabei werden die Systeme nach Copyleft- und toleranten Lizenzen wie in Kapitel 2 kategorisiert.

6.2.2 User-Experience

Die User-Experience beschreibt die Erfahrungen der Nutzer bei der Interaktion mit der Software. Bei der untersuchten Software ist die Benutzbarkeit und die Einfachheit besonders interessant. User-Experience ist ein Überbegriff und beinhaltet mehrere Kriterien und wird nach ISO Norm 9241 - 210 wie folgt definiert:

Definition 6.2.1 (ISO Norm 9241 - 210)
A person's perceptions and responses that result from the use and/or anticipated use of a product, system or service.

Somit beinhaltet die User-Experience alle Erfahrungen, die der Nutzer während der Interaktion mit der Software hat. Der Anwendungsfall in Abschnitt 4.4 fordert, dass das System von Mitarbeitern in der Produktion verwendet werden muss, deshalb spielt User-Experience eine wichtige Rolle.

Benutzbarkeit

Benutzbarkeit beschreibt, wie effizient Ziele mit dem System durch einen Anwender erreicht werden. Der Begriff Benutzbarkeit ist ähnlich zur Benutzerfreundlichkeit. Für die Auswertung ist der Aspekt Benutzbarkeit interessant, denn bei hoher Benutzbarkeit ist das System besser in einer Produktionsumgebung einsetzbar.

Einfachheit

Neben der Benutzbarkeit ist auch die Einfachheit, die beschreibt, wie gut sich die Software konfigurieren, installieren und verwalten lässt, wichtig. Denn dadurch wird die Wartung und der Betrieb der Software vereinfacht.

6.2.3 Stärken

Die meisten Systeme sind für einen bestimmten Anwendungsfall konzipiert und haben deshalb oft Stärken gegenüber anderen Systemen, die für andere Anwendungsszenarios entwickelt worden sind. Das Kriterium „Stärken“ beschreibt, in welchem Aspekt die Systeme besonders viele oder gute Features bieten. Dieses Kriterium basiert auf der Arbeit von Gölzer et al. [GCA15].

6.2.4 Sicherheit

Das Sicherheitskriterium beschreibt, wie stark das Rechtesystem der Systeme ist und ob die Daten ausreichend vor dem Zugriff durch unbefugte Dritte geschützt werden können. Gerade im industriellen Umfeld ist Sicherheit ein wichtiges Kriterium, denn bei unzureichender Sicherheit könnten wichtige Daten kompromittiert werden.

6.2.5 Flexibilität

Flexibilität ist ein wichtiges Kriterium, denn die Anforderungen an ein System können sich verändern. Die Flexibilität beschreibt, wie aufwändig es ist, weitere Komponenten oder Datenspeicher an das System anzubinden. Die Flexibilität hängt direkt mit den Schnittstellen der Software zusammen. Durch viele Schnittstellen ist es einfacher weitere Komponenten an das System anzubinden.

6.2.6 Kollaboration

Der Kollaborations-Aspekt beschreibt, wie gut Nutzer innerhalb der Software miteinander interagieren und zusammenarbeiten können. Vor allem in Unternehmensumgebungen ist es wichtig, dass Personen gemeinsam an Daten arbeiten können. Verschiedene Softwarelösungen bieten unterschiedlich ausgeprägte Tools zur Kollaboration, weshalb dieses Kriterium sinnvoll ist.

7 Bewertung der Technologien

Die in Kapitel 5 vorselektierten Technologien sind potentiell für die Verknüpfung von strukturierten und unstrukturierten Daten geeignet, besitzen jedoch verschiedene Merkmale. Diese werden nun mit den in Kapitel 6 erarbeiteten Kriterien bewertet.

7.1 Vergleichstabelle

	Alfresco CE	CDH	Apache Marmotta
	Funktionale Anforderungen		
Verknüpfungsmethodik	linkbasiert	programmatische Verknüpfung	ontologiebasiert
Datenmodell	Filesystem, DB	Filesystem	RDF
Verwaltung von Metadaten	Record-Management	Apache HBase	RDF
Unterstützung von Analysen	AM-Plugin	Apache Pig, Spark, Hive	SPARQL
Integrierte Suche	Apache Solr	Apache Solr	-
Schnittstellen	AM-Plugin, CIMS	Apache Impala, Hive, Sqoop, Kafka, Flume	SPARQL, LDPath
Skalierbarkeit	vertikal	horizontal	horizontal
	Nicht funktionale Anforderungen		
Lizenz	LGPLv3, ALSv2	ASLv2	ASLv2
Benutzbarkeit	hoch	mittel	niedrig
Einfachheit	mittel	niedrig	hoch
Stärken	Dokumenten-Management	Batch-Processing	Linked-Data
Sicherheit	lokal mit Rechtesystem	lokal mit Rechtesystem	lokal mit Rechtesystem
Flexibilität	hoch	mittel	niedrig
Kollaboration	Gruppen, Arbeitskreise	-	-

Tabelle 7.1: Weiterer Vergleich der Systeme

Tabelle 7.1 stellt die in Kapitel 5 ausgewählten Technologien sowie die in Kapitel 6 festgelegten Kriterien und deren Bewertung anhand der einzelnen Technologien dar. Dabei sind in der ersten Spalte die Kriterien aufgeführt und in den nachfolgenden Spalten die Softwarelösungen. Die Reihen beinhalten die Bewertung der Kriterien anhand der Systeme, wobei die Kriterien

in zwei Kategorien: „funktionale Anforderungen“ und „nicht funktionale Anforderungen“ unterteilt sind.

7.2 Alfresco Community Edition

Alfresco ist ein quelloffenes ECMS, das es ermöglicht, unstrukturierte Daten wie zum Beispiel: PDF-Dateien, E-Mails etc. zu speichern und zu verwalten. Darüber hinaus verfügt Alfresco über Kollaborationsfunktionen, ein Workflow-Management und kann mithilfe von Plugins erweitert werden.

7.2.1 Funktionale Kriterien

Im folgenden Abschnitt wird Alfresco anhand der in Kapitel 6 festgelegten funktionalen Kriterien bewertet.

Verknüpfungsmethodik

Die Verknüpfung der Daten in Alfresco erfolgt über ein Plugin, welches über den Add-On-Marktplatz von Alfresco bezogen werden kann. Das MetaDBConnector Plugin ermöglicht es, externe relationale Datenbanken anzubinden, um Dokumente mit Informationen aus diesen strukturierten Quellen anzureichern. Die Verknüpfung erfolgt ähnlich zu dem von Gröger [Grö15] vorgestellten Konzept. Dabei werden Verweise auf die strukturierten Informationen direkt an die Dokumente angehängt. Das Plugin stellt nach der Installation die Funktion bereit, über XML-Dokumente und JDBC-Bibliotheken relationale Datenbanken anzubinden. **J**ava **D**ata**B**ase **C**onnectivity (JDBC) ist eine standardisierte Datenbankschnittstelle, die mithilfe verschiedener Bibliotheken den Zugriff auf Datenbanken diverser Hersteller in Java bereitstellt. Jedes XML-Dokument symbolisiert den View auf eine Tabelle, wobei die Daten bei Bedarf vorher mit einer SQL-Query gefiltert werden können. In Alfresco können die strukturierten Daten händisch an die Dokumente angebunden werden. Diese Verknüpfungsmethodik bietet nur die Möglichkeit, Daten händisch zu verknüpfen. Eine automatische Verknüpfung, mithilfe von Textminingmethoden oder Ähnlichem, ist nicht möglich.

7.2.2 Datenmodell

Das Datenmodell von Alfresco besteht aus mehreren Komponenten. Die gespeicherten Dokumente sind im Filesystem des Betriebssystems abgelegt, die an die Dokumente geknüpften Metadaten werden in einer an Alfresco angebundenen relationalen Datenbank gespeichert. Die strukturierten Daten sind ebenfalls in einer relationalen Datenbank abgelegt. Dadurch

besteht das Datenmodell zum einen aus dem Filesystem des Betriebssystems und zum anderen aus relationalen Datenbanken.

Verwaltung von Metadaten

Alfresco bietet die Möglichkeit, Metadaten an die gespeicherten Dokumente anzubinden. Es gibt bereits eine Auswahl an Feldern für Metadaten, diese können programmatisch erweitert werden. Um zusätzliche Felder für Metdadaten zu erstellen, muss in der Community-Version von Alfresco ein XML-Modell erzeugt werden. Zusätzlich besteht die Möglichkeit, ein Record-Management Plugin in Alfresco zu installieren, dieses erweitert und vereinfacht die Anbindung von Metadaten. Da es allerdings das Standard-Konzept für Metadaten ersetzt, verhindert es die Anbindung von strukturierten Daten an Dokumente, die über das Record-Management Plugin verwaltet werden.

Unterstützung von Analysen

Alfresco bietet grundsätzlich keine Analysemöglichkeiten. Es ist jedoch möglich, Analyse Plugins über den Add-On-Marktplatz zu beziehen oder eigene Erweiterungen für Alfresco zu entwickeln.

Integrierte Suche

Die integrierte Suche von Alfresco basiert auf Apache Solr. Die integrierte Suche bietet Volltextsuche in Dokumenten und ist in Alfresco vorkonfiguriert, so dass diese direkt nach der Installation bereit steht.

Schnittstellen

Alfresco bietet eine CIMS-Schnittstelle sowie eine WebDav-Schnittstelle siehe Kapitel 5.

Skalierbarkeit

In der Community Edition lässt sich Alfresco nur auf einem Knoten installieren, deshalb skaliert das System nur vertikal. Es ist allerdings möglich, weitere Datenquellen, die auf anderen Knoten liegen, anzubinden. Auch die Datenbank, welche die strukturierte Daten beinhaltet, muss nicht auf demselben Knoten zur Verfügung stehen, sondern kann über einen weiteren Knoten bereitgestellt werden.

7.2.3 Nicht funktionale Kriterien

Im folgenden Abschnitt wird Alfresco anhand der in Kapitel 6 festgelegten, nicht funktionalen Kriterien bewertet.

Lizenz

Alfresco ist unter der LGPL Open Source, das bedeutet, dass Änderungen an der Software veröffentlicht werden müssen. Allerdings müssen durch die LGPL Plugins nicht unbedingt unter der GPL veröffentlicht werden siehe Kapitel 2. Dadurch ist es möglich, proprietäre Plugins für Alfresco zu entwickeln, was in der Industrie wünschenswert ist. Das MDBC Plugin zum Anbinden von relationalen Datenbanken ist unter der APL Open Source.

User-Experience

Alfresco lässt sich durch die Weboberfläche sehr einfach benutzen. Es ist intuitiv verständlich, wie man in der Weboberfläche navigiert. Dadurch erzielt das System eine hohe Benutzbarkeit. Allerdings ist das Konfigurieren der Software nicht immer direkt ersichtlich, vor allem beim Installieren von Plugins. Durch die Dokumentation kann dies jedoch mit angemessenem Zeitaufwand bewerkstelligt werden. Deshalb ist die Einfachheit der Softwarelösung mit „mittel“ bewertet.

Stärken

Die Stärken der Software liegen im Dokumenten-Management, da Alfresco dahingehend ausgerichtet ist. Alfresco bietet die Möglichkeit Sites zu erstellen, über die Daten verwaltet werden können. Innerhalb der Sites können Arbeitsgruppen gebildet werden, in denen autorisierte User miteinander an einem Projekt arbeiten können. Zusätzlich zu den Sites bietet Alfresco ein Workflow-Management sowie Tools für das Zuweisen von Aufgaben.

Sicherheit

Alfresco besitzt ein Rechtesystem, mit dem der Zugriff auf bestimmte Dokumente oder Ordner eingeschränkt werden kann. Die Software ist lokal installierbar und benötigt keine Internetverbindung, dadurch ist sie unabhängig einsetzbar.

Flexibilität

Durch das Plugin-System ist Alfresco sehr flexibel, es lässt sich dynamisch auf neue Anforderungen anpassen. Durch die bereits existierende Plugin-Bibliothek sind bereits viele Funktionen verfügbar.

Kollaboration

Alfresco unterstützt die Kollaboration von Nutzern, denn es können Nutzergruppen erstellt werden und es existieren Funktionen wie Sites und Google Docs, um das gemeinsame Arbeiten in Dokumenten zu vereinfachen.

7.3 CDH

CDH ist ein Open-Source-Softwarestack. Er basiert auf dem Hadoop-Ecosystem und beinhaltet Apache Hadoop (Core), Apache Accumulo, Apache Flume, Apache HBase, Apache Hive, HUE, Apache Impala, Apache Kafka, Apache Pig, Apache Sentry, Cloudera Search, Apache Spark und Apache Sqoop. Dadurch wird es möglich, vielfältige Daten anzubinden und auszuwerten. Im Rahmen dieser Arbeit wird, stellvertretend für CDH, das vollständige Softwarepaket Cloudera Enterprise bewertet, dieses beinhaltet den Cloudera Manager, der das Konfigurieren der Software vereinfacht. Der Cloudera Manager ist unter proprietären Lizenzen veröffentlicht. Dieser Manager erspart Zeit bei der Konfiguration des Softwarestacks, bringt jedoch keinen funktionalen Vorteil bei der Datenintegration oder Verarbeitung.

7.3.1 Funktionale Kriterien

Im folgenden Abschnitt wird CDH anhand der in Kapitel 6 festgelegten funktionalen Kriterien bewertet.

Verknüpfungsmethodik

CDH bietet eine programmatische Verknüpfungsmethodik, um strukturierte und unstrukturierte Daten zu integrieren. Dafür muss ein Skript innerhalb von Apache Pig erstellt werden, das unstrukturierte Daten analysiert und mit Daten aus Apache Sqoop verknüpft. Alternativ zu Apache Pig kann zum Aufbau von Verknüpfungen auch Apache Spark eingesetzt werden. Die Information über die Verknüpfung muss selbst verwaltet werden, wobei Apache HBase eine gute Möglichkeit darstellt.

7.3.2 Datenmodell

Das Datenmodell von CDH basiert auf dem Hadoop-Filesystem. Durch die Integration der verschiedenen Hadoop-Tools können weitere Datenquellen wie relationale Datenbanken durch Apache Sqoop an das System angebunden werden. Außerdem ist es möglich, innerhalb von CDH weitere Datenspeicher einzurichten. Apache Accumulo ist ein verteilter Key-Value-Store und Apache HBase ist ein Column-Store. Beide Systeme sind nach dem Vorbild von Google Bigtable [CDG+06] entworfen und in das HDFS eingebunden.

Verwaltung von Metadaten

Zur Verwaltung von Metadaten bietet CDH Apache HBase. HBase ist eine Datenbanklösung für das Hadoop-Filesystem, wobei der HBase-Store beispielsweise über Impala oder Hive extern angesteuert werden kann. Die Metadaten können mittels programmatischen Analysen durch Apache Pig oder Spark automatisch extrahiert werden.

Unterstützung von Analysen

Mit Apache Pig ist es möglich, Daten durch Skripts zu analysieren. Diese Skripts bieten die Möglichkeit, die verschiedenen Softwaretools innerhalb von CDH anzusteuern. Dabei führt jedes Skript eine spezifische Analyse durch, die ins MapReduce-Programmierkonzept übersetzt wird und auf den einzelnen Knoten verteilt wird. Neben Apache Pig können Analysen auch durch Apache Spark oder Apache Hive durgeführt werden. Apache Hive bietet eine SQL-ähnliche Schnittstelle für Anfragen und Apache Spark ermöglicht das Erstellen von Spark-Programmen.

Integrierte Suche

Die integrierte Suche von CDH wird von Apache Solr bereitgestellt, diese kann über die Hue-Weboberfläche genutzt werden. Dadurch können spezifische Seiten aufgebaut werden, die Visualisierungen der Suchanfragen bereitstellen. Bevor Solr jedoch genutzt werden kann, muss Solr konfiguriert werden und ein Suchindex aufgebaut sein.

Schnittstellen

CDH bietet viele unterschiedliche Schnittstellen. Apache Sqoop ermöglicht es, externe relationale Datenbanken an CDH anzubinden. Mithilfe von Apache Flume können Logdateien gesammelt und effizient verwaltet werden. Apache Kafka bietet das Anbinden und Verarbeiten von Datenströmen aus verschiedenen Streamingquellen. Mithilfe von Apache Hive können SQL-ähnliche Anfragen an die im HDFS gespeicherten Daten gestellt werden, während mit Apache Impala direkt SQL-Anfragen abgesetzt werden können.

Skalierbarkeit

Das Konzept des Hadoop-Ecosystems basiert auf der verteilten Speicherung und Verarbeitung von Daten, dadurch sind die einzelnen eingesetzten Apache Tools auf horizontale Skalierung ausgerichtet und somit ist auch CDH horizontal skalierbar.

7.3.3 Nicht funktionale Kriterien

Im folgenden Abschnitt wird CDH anhand der in Kapitel 6 festgelegten, nicht funktionalen Kriterien bewertet.

Lizenz

Die eingesetzten Apache Tools sind unter der Apache Software License veröffentlicht und somit Open Source siehe Kapitel 2. Auch die Cloudera Search ist unter der Apache Software License veröffentlicht, dadurch ist CDH vollständig unter der APL Open Source.

User-Experience

Durch Hue wird der Einsatz des Hadoop-Ecosystems stark vereinfacht. Allerdings wird dennoch Hintergrundwissen über Hadoop benötigt, um CDH sinnvoll einsetzen zu können, weshalb die Benutzbarkeit mit „mittel“ bewertet wird. Ohne den Cloudera Manager ist die Konfiguration der einzelnen Softwaresysteme aufwändiger und im Vergleich zur Cloudera Enterprise Version deutlich erschwert. Dies beeinflusst die Einfachheit der Software stark und wird deshalb mit „niedrig“ bewertet.

Stärken

Die Stärken von CDH liegen in der Verarbeitung und Verwaltung von Big Data, das bedeutet, dass große Datenmengen sicher gespeichert und gleichzeitig effizient verarbeitet werden können. Dies liegt in der Natur des Hadoop-Ecosystems, da dieses konzeptionell für Big Data entworfen wurde. Durch das MapReduce-Programmiermodell und Apache Spark liegen die Stärken von CDH im Batch-Processing. Batch-Processing ist ein Konzept, bei dem große Datensätze in kleinere zerlegt werden, um diese parallel zu verarbeiten [GCA15].

Sicherheit

Innerhalb des Hadoop-Filesystems ist es möglich, die einzelnen Daten so zu konfigurieren, dass nur bestimmte Nutzer darauf Zugriff haben. Zusätzlich bietet Apache Sentry die Möglichkeit, ein fein unterteiltes Rollen- und Rechtesystem aufzubauen, um den Zugriff auf die im Hadoop-Cluster gespeicherten Daten zu restringieren. Außerdem ist es möglich, das Hadoop-Cluster lokal zu installieren, ohne dass eine Internet-Verbindung benötigt wird.

Flexibilität

Dadurch, dass die Software verschiedene feste Softwaresysteme enthält, ist die Flexibilität von CDH auf die Flexibilität der einzelnen Systeme begrenzt. Es wird vermutlich in Zukunft möglich sein, weitere Apache Tools anzubinden, jedoch könnte es nötig werden, die bereits bestehenden Systeme zu aktualisieren, was ohne den Einsatz der Cloudera Manager Software jedoch mit erheblichem Mehraufwand verbunden ist. Dadurch ist die Flexibilität nur „mittel“ gegeben.

Kollaboration

In CDH selbst ist es für verschiedene Nutzergruppen nur indirekt möglich, gemeinsam an Dokumenten zu arbeiten. Denn CDH bietet ohne eine Erweiterung keine Möglichkeit an, Arbeitsgruppen zu erstellen.

7.4 Apache Marmotta

Apache Marmotta ist eine Linked-Data-Plattform. Das Ziel von Marmotta ist es, einfach benutzbar zu sein und gleichzeitig erweiterbar. Marmotta entstand aus dem **L**inked **M**edia **F**ramework (LMF), das verschiedene Apache Tools wie Stanbol, ein System zur Erweiterung von Content-Managment-Systemen um semantische Informationen, und Apache Solr beinhaltete. Die Kernkomponenten zur Speicherung und Verwaltung von Linked-Data innerhalb des LMF wurden genutzt, um Apache Marmotta zu entwickeln.

7.4.1 Funktionale Kriterien

Im folgenden Abschnitt wird Apache Marmotta anhand der in Kapitel 6 festgelegten funktionalen Kriterien bewertet.

Verknüpfungsmethodik

Das Verknüpfen innerhalb von Marmotta geschieht über das Integrieren der Daten in eine gemeinsame Ontologie. Dabei werden die gespeicherten Informationen stetig um neue Aspekte erweitert. Um jedoch strukturierte Daten aus beispielsweise einer relationalen Datenbank anzubinden, ist es notwendig, diese mithilfe eines eigens entwickelten Scripts oder Programms programmatisch in die Ontologie zu integrieren. Alternativ können strukturierte Daten aus Tabellen mithilfe von OpenRefine[1] in RDF transformiert werden und in Marmotta importiert werden. Auf unstrukturierte Daten kann innerhalb der Ontologie verwiesen werden.

7.4.2 Datenmodell

Das Datenmodell von Marmotta besteht aus Eclipse RDF4J, welches aus OpenRDF Sesame entstanden ist, und einem KiWi-Tripel-Store. Die Daten selbst liegen innerhalb einer relationalen Datenbank, die von KiWi verwaltet wird. Eclipse RDF4J bietet eine standardisierte Schnittstelle, dadurch können verschiedene Backends angebunden werden. Neben KiWi (das Standard-Backend von Marmotta) ist es möglich, Backends wie Sesame Native, welches die Daten im Filesystem des Betriebssystems lagert, oder das momentan experimentelle Big-Data-Backend für sehr große Datensätze anzubinden.

[1] http://openrefine.org/

Verwaltung von Metadaten

Ähnlich zur Integration der Daten geschieht die Verwaltung von Metadaten durch Integrieren dieser Metadaten in die Ontologie.

Unterstützung von Analysen

Marmotta bietet abgesehen von SPARQL keine Analysemöglichkeiten. Mithilfe von SPARQL ist es möglich, Queries auf die in Marmotta gespeicherten Daten abzusetzen, welche die Daten filtern. SPARQL ist eine Abfragesprache für RDF-Daten, die sich syntaktisch an SQL orientiert [CDES05].

Integrierte Suche

Marmotta enthält keine integrierte Suche, lediglich die Möglichkeit, Queries auf die Daten der Ontologie abzusetzen. Unstrukturierte Daten können nicht durchsucht werden.

Schnittstellen

Apache Marmotta verfügt über eine SPARQL Querying-Schnittstelle zum Absetzen von Queries. Zudem können die Daten mithilfe von LDPath, einer Abfragesprache für die Linked-Data-Plattform, abgerufen werden. LDP ermöglicht das Aufbauen einer Linked-Data-Cloud, wobei die Daten verteilt gespeichert werden.

Skalierbarkeit

Apache Marmotta skaliert horizontal, da der KiWi-Tripel-Store, verteilt auf verschiedene Systeme, eingerichtet werden kann. Außerdem können über das LDP-Protokoll weitere Datenquellen angebunden werden.

7.4.3 Nicht funktionale Kriterien

Im folgenden Abschnitt wird Apache Marmotta anhand der in Kapitel 6 festgelegten, nicht funktionalen Kriterien bewertet.

Lizenz

Marmotta ist vollständig unter der Apache-Software-Lizenz veröffentlicht und somit Open Source siehe Kapitel 2.

7.4.4 User-Experience

Die Einrichtung und Installation von Apache Marmotta verlief ohne Probleme, weshalb die Einfachheit der Software mit „hoch“ bewertet wird. Marmotta bietet eine Weboberfläche, um die Software zu konfigurieren, diese ist nicht immer besonders intuitiv zu bedienen und bietet nur wenig Funktionsumfang. Marmotta ist eine Plattform, auf die weitere Software aufgebaut werden soll. Im Idealfall soll ein eigenes UI mithilfe von Java EE entwickelt werden. Dadurch ist die Benutzbarkeit eingeschränkt und deshalb mit „niedrig“ bewertet.

7.4.5 Stärken

Die Stärken von Marmotta sind das Anlegen und Verwalten von Ontologien. Marmotta ist daher besonders gut als Linked-Data-Plattform geeignet.

7.4.6 Sicherheit

Apache Marmotta kann lokal installiert werden und benötigt keine Internetverbindung. Verschiedene Benutzer können mithilfe eines simplen Systems in Nutzer und Rollen eingeteilt werden, um den Zugriff zu restringieren.

7.4.7 Flexibilität

Die Flexibiltät von Marmotta wird mit „niedrig“ bewertet, da die Software nur wenige Schnittstellen bietet und eigene Programme entwickelt werden müssen, die neue Datenquellen anbinden.

7.4.8 Kollaboration

Apache Marmotta bietet keine Möglichkeit, gemeinsam an Projekten zu arbeiten.

8 Anwendungsfall

In diesem Kapitel wird ein Anwendungsfall beschrieben, bei dem die Verknüpfung von un- und strukturierten Daten benötigt wird. Aus diesem Anwendungsfall ergeben sich die Anforderungen an die Softwarelösung. Anhand dieser Anforderung und der in Kapitel 7 aufgestellten Vergleichstabelle wird die geeignetste Lösung ausgewählt.

8.1 Anforderungen

Es lassen sich aus dem Anwendungsfall verschiedene Anforderungen formulieren. Zum einen benötigt der Automobilzulieferer ein System, das in der Lage ist, unstrukturierte mit strukturierten Daten zu verknüpfen. Dabei soll das System in der Lage sein, die strukturierten Daten aus dem MES- und ERP-System durch SQL anzubinden sowie die unstrukturierten in das System zu integrieren. Die verknüpften Daten sollen die Arbeit der Produktionsarbeiter unterstützen. Deshalb wird ein System benötigt, das es den Produktionsarbeitern erlaubt, auf die verknüpften Daten zuzugreifen. Darüber hinaus sollen die Daten werksübergreifend verfügbar gemacht werden.

Zusammenfassend werden folgende Features benötigt:

- Anbindung der strukturierten Daten durch SQL
- Anbindung von unstrukturierten PDF-Dateien
- Verknüpfung von unstrukturierten PDF-Dateien mit strukturierten MES- und ERP-Daten
- Gewährung des Zugriffs auf die verknüpften Daten für die Produktionsarbeiter
- Werksübergreifende Bereitstellung der verknüpften Daten

8.2 Auswahl einer geeigneten Lösung anhand der Vergleichstabelle

Anhand der in Kapitel 7 vorgestellten Systeme soll die am besten zum Anwendungsfall passende Lösung ausgewählt werden. Alle drei Systeme bieten die Möglichkeit, unstrukturierte und strukturierte Daten zu integrieren sowie zu verknüpfen. Innerhalb von CDH werden die

Daten programmatisch verknüpft, während bei den beiden anderen Systemen die Daten händisch verknüpft werden. Jedoch ist bei den anderen beiden Systemen eine programmatische Verknüpfung innerhalb der Ontologie bzw. der Aufbau von Links mithilfe eines entwickelten Tools/Plugins (z.B auf Basis von Textminingmethoden) nicht ausgeschlossen.

Jedes der drei Systeme bietet die Verwaltung von Metadaten an, wobei Alfresco die zusätzlichen Informationen an die Dokumente mithilfe des Record-Managements anbindet und CDH die Möglichkeit bietet, Metadaten in einer HBase-Datenbank zu speichern. Bei Marmotta können Metadaten direkt in die Ontologie aufgenommen werden. Dadurch bieten alle drei Lösungen ein gutes System zur Verwaltung von Metadaten an, wobei Apache Marmotta die Metadaten am besten integriert.

Nur CDH bietet ein integriertes Analysewerkzeug, wobei dafür die Entwicklung eines Analyseskripts notwendig ist, dies ist ein Vorteil gegenüber den anderen Systemen. Auch für Marmotta und Alfresco wäre es möglich, Plugins bzw. Skripte zu entwickeln, die die Daten analysieren. Vor allem Marmotta bietet viel Potenzial bezüglich der Analyse der Daten, denn durch die Strukturierung der Daten und Metadaten innerhalb einer Ontologie ist es möglich, logische Schlussfolgerungen aus den Daten abzuleiten.

Alfresco und CDH bieten eine Suchfunktion auf Basis von Apache Solr. Bei CDH ist es notwendig, die Suche für gespeicherte Daten zu konfigurieren, während bei Alfresco die Suche automatisch konfiguriert wird. Marmotta enthält keine integrierte Suche, sondern lediglich die Möglichkeit, SPARQL Queries abzusetzen. Die Suche ist im Anwendungsfall nicht konkret spezifiziert, jedoch ein sinnvoller Bestandteil des für die Produktionsarbeiter zugänglichen Teils der Software.

Die drei Systeme bieten unterschiedliche Schnittstellen, Marmotta bietet die SPARQL- sowie eine experimentelle LDP-Schnittstelle. CDH stellt durch die verschiedenen Apache Tools eine Vielzahl von Schnittstellen zur Verfügung: Apache Accumulo zum sicheren Anbinden von performanzintensiven Big-Data-Anwendungen, Apache Flume zum Sammeln von Log-Daten sowie Apache Sqoop zum Anbinden relationaler Datenbanken. Diese sind allerdings für diesen Anwendungsfall nur teilweise sinnvoll. Alfresco bietet die CIMS-Schnittstelle sowie den MetaDBConnector zum Anbinden einer relationalen Datenbank. Marmotta bietet keine SQL-Schnittstelle zum direkten Anbinden einer relationalen Datenbank, was im Anwendungsfall explizit gefordert wurde.

Marmotta und CDH skalieren horizontal, während Alfresco nur vertikal skaliert, wobei der Server für die relationale Datenbank auf einem separaten System eingerichtet werden kann. Im Anwendungsfall wird horizontale Skalierbarkeit nicht direkt gefordert, allerdings ist dies in einer Unternehmensumgebung durchaus wünschenswert. Alfresco ist im Gegensatz zu den anderen Systemen nicht unter der APL, sondern der LGPL lizenziert, einer Copyleft-Lizenz siehe Kapitel 2. Obwohl Alfresco selbst mit einer industrieunfreundlichen Lizenz veröffentlicht ist, können Plugins unter anderen Lizenzen als die GPL veröffentlicht werden.

Im Punkt User-Experience ist Alfresco die beste Software. Sie ist einfach zu installieren, abgesehen von den Plugins und intuitiv benutzbar. Im Gegensatz zu CDH, das ohne den

Cloudera Manager sehr aufwändig zu installieren ist und nur durch Personen, die bereits Wissen über das Hadoop-Ecosystem haben. Auch die Benutzerfreundlichkeit von CDH ist gering, da es nicht intuitiv nutzbar ist. Dies steht im Widerspruch zu den Anforderungen, da Produktionsarbeiter in der Lage sein sollten, die Software zu nutzen. Auch Marmotta ist nicht besonders intuitiv nutzbar, jedoch sehr gut einrichtbar.

Die Stärken von CDH liegen im Batch-Processing, während Marmotta besonders für Linked-Data ausgerichtet ist und Alfresco für das Dokumenten-Management. Dadurch eignen sich gerade Marmotta und Alfresco für den beschriebenen Anwendungsfall. Alfresco ist das flexibelste und leicht gewichtigste System mit vielen Plugins und Erweiterungsmöglichkeiten.

Kollaboration ist für diesen Anwendungsfall nicht zwingend notwendig.

Aus den genannten Gründen ergibt sich, dass CDH sich in diesem Anwendungsfall nur bedingt einsetzen lässt, da es für Produktionsarbeiter schwierig ist, die Software zu nutzen. Zudem beinhaltet es viele unnötige Funktionen und ist schwierig zu verwalten und zu warten. Marmotta bietet ebenfalls keine Möglichkeit, den Mitarbeitern in der Produktion die Daten einfach zur Verfügung zu stellen und hat darüber hinaus keinen direkte Anbindungsmöglichkeit für relationale Datenbanken. Es bietet sich jedoch besonders gut für die weitere Analyse der Daten an. Marmotta eignet sich besonders, wenn eine spezifisches UI entwickelt wird. Alfresco bietet die Möglichkeit, die verknüpften Daten den Produktionsarbeitern zur Verfügung zu stellen und ist am flexibelsten. Alfresco bietet am meisten Funktionen, ist allerdings nicht horizontal skalierbar und unter LGPL veröffentlicht. Dadurch fällt die Entscheidung für diesen Anwendungsfall zwischen Alfresco und Marmotta. Da Marmotta nicht die geforderte SQL-Anbindung bietet, fällt die Wahl auf Alfresco.

8.3 Exemplarische Umsetzung

In diesem Abschnitt wird eine exemplarische Umsetzung des Anwendungsfalls anhand von Alfresco durchgeführt. Dafür wurde die Alfresco Community Edition in der Version 5.0.d eingesetzt. Die Daten für diesen Anwendungsfall bestehen aus Dummy PDF-Dateien und der Sakila Datenbank[1]. Die Sakila Datenbank ist eine Beispieldatenbank für das MySQL Datenbanksystem und wird stellvertretend für die MES- und ERP-Systeme eingesetzt, während die unstrukturierten PDF-Dateien von Dummy Dateien ersetzt werden. Da Alfresco nicht direkt die Anbindung von strukturierten Datenbanken unterstützt, muss es um diese Funktion über ein Plugin erweitert werden. Ein Plugin, das dies ermöglicht, ist das **M**eta**DBC**onnector Plugin (MDBC)[2], welches unter der Apache Software License veröffentlicht ist. Zum Installieren des Plugins ist es notwendig, die folgenden Schritte auszuführen:

1. Kompilieren des AMP-Modules mithilfe von *mvn clean package*
2. Kompilieren des Share Plugins mithilfe von *ant*
3. Installieren des AMP-Moduls durch

   alfresco-mmt.jar install amps/mdbc-1.0-SNAPSHOT.amp tomcat/webapps/alfresco.war
4. Installieren des Share Plugins durch Kopieren der JAR-Datei nach

   $ALF_ROOT/tomcat/webapps/share/WEB-INF/lib
5. Neustart von Alfresco und Tomcat
6. Konfigurieren der relationalen Datenbank in

   $ALF_ROOT/tomcat/webapps/alfresco/WEB-INF/classes/alfresco/module/mdbc/

Nachdem das MDBC Plugin installiert wurde, steht unter dem Admin-Tools Reiter ein neues Element „MetaDBConnectors“ mit dem Unterelement „MetaDBConnectorsConfig List“ zur Verfügung. Darin sind die definierten Connectoren aufgelistet. Ein Connector stellt ein View auf eine Tabelle der Datenbank dar, dabei kann der View jedoch vorher gefiltert sein, so dass nur relevante Informationen zur Verfügung stehen. Die Filterung geschieht durch eine where-Klausel, welche innerhalb der Konfiguration XML definiert wird. Die Auflistung besteht aus drei Spalten, dem Titel des Connectors, dessen Beschreibung und dem Aspekt, der für diesen Connector angelegt wurde. Diese Aspekte sind für die Verknüpfung von strukturierten Daten mit Dokumenten in Alfresco wichtig. Metadaten werden in Alfresco über Aspekte angebunden wie in Kapitel 7 beschrieben. Jedes Dokument in Alfresco hat Aspekte, wenn beispielsweise eine Audiodatei in das Alfresco Repository geladen wird, kann dies mit dem „Audio-Aspekt“ durch die „Aspekte verwalten“ Option erweitert werden. Dadurch befinden sich anschließend

[1] https://dev.mysql.com/doc/sakila/en/
[2] https://addons.alfresco.com/addons/alfresco-metadbconnector-component

unter den Dokument-Eigenschaften neue Felder wie Artist, Titel, Jahr etc. Das MDBC Plugin funktioniert nach demselben Prinzip. Bevor an Dokumente strukturierte Daten angeknüpft werden können, muss zuerst der jeweilige Connector-Aspekt an das Dokument geknüpft werden siehe Abbildung 8.1.

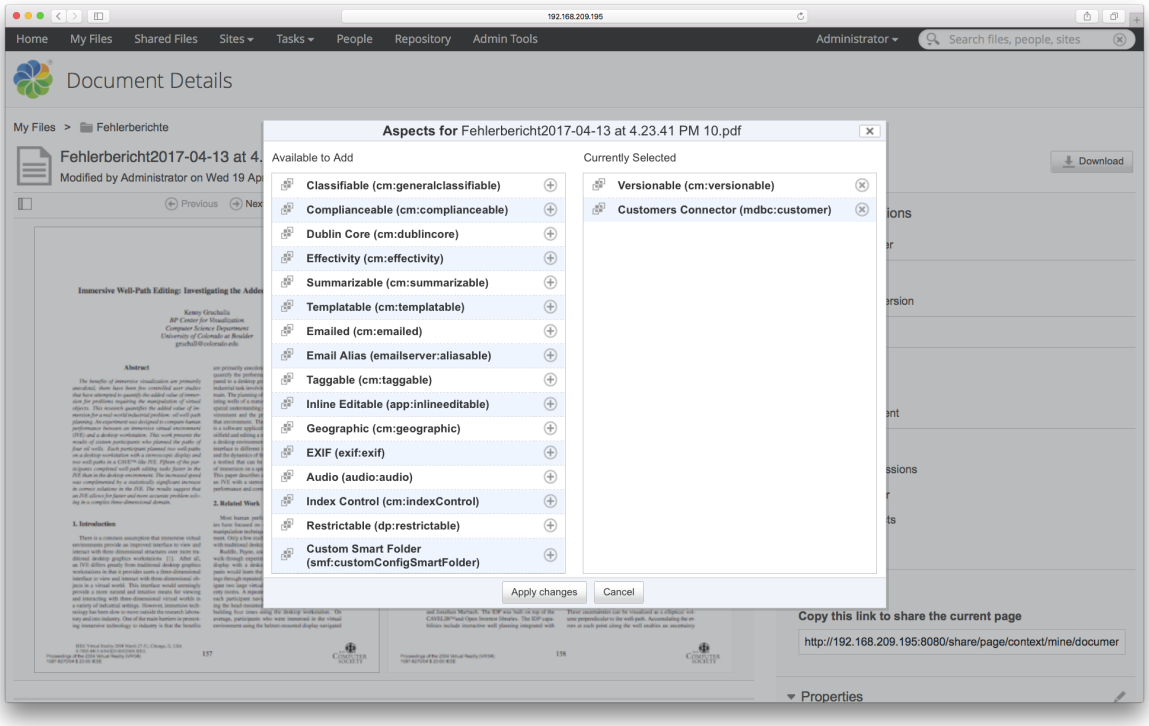

Abbildung 8.1: Erweitern eines Dokuments um den Aspekt „Customers Connector“

Das Hinzufügen von Aspekten zu Dokumenten kann in Alfresco automatisiert werden, dafür müssen Regeln erstellt werden. Regeln können an Ordner innerhalb des Repository geknüpft werden. Eine Regel kann beispielsweise, sobald neue Dokumente in den Ordner geladen werden, ausgeführt werden. Dabei können verschiedene Funktionen von Regeln automatisch ausgelöst werden, unter anderem das Hinzufügen von Aspekten. Dadurch kann zum Beispiel ein Ordner definiert werden, der automatisch neuen Dokumenten den Connector-Aspekt hinzufügt. Nachdem der Connector-Aspekt zu einem Dokument hinzugefügt wurde, ist es möglich unter den Eigenschaften des Dokuments, strukturierte Daten an dieses anzuknüpfen. Dafür muss zunächst unter dem neu hinzugefügten Abschnitt auf den Select-Button geklickt werden. Anschließend erscheint ein neues Fenster siehe Abbildung 8.2.

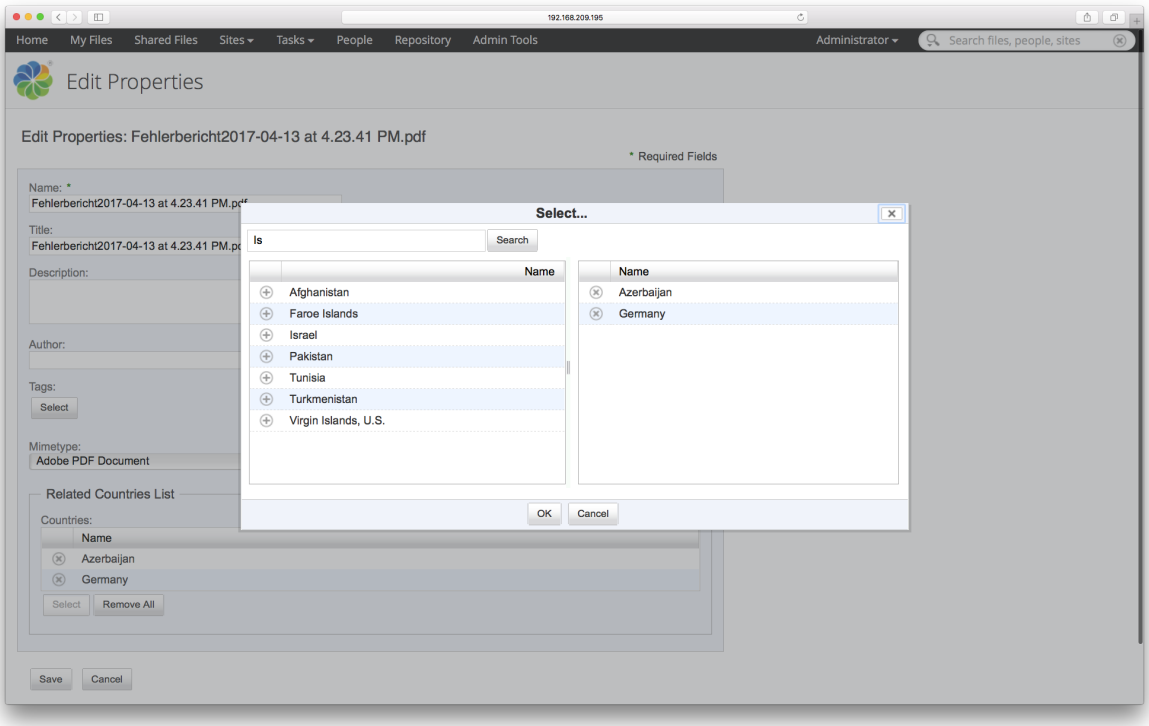

Abbildung 8.2: Selektionsfenster für strukturierte Daten

Dieses Fenster ermöglicht die Filterung des Views auf die Tabelle mithilfe des Suchfeldes. Diese Filterung geschieht anhand der zuvor in der XML-Datei festgelegten where-Klausel, wobei das Suchfeld die Wildcard „%{0}%“ ersetzt. Der linke Abschnitt des Fensters zeigt das Ergebnis der Suche, der rechte die ausgewählten Elemente. Um Elemente auszuwählen, muss auf das „+“ geklickt werden. Nachdem die gewünschten Daten ausgewählt wurden, kann das Fenster durch den OK-Button bestätigt werden. Anschließend kann die Änderung mithilfe des Save-Buttons gespeichert werden. Die neu hinzugefügten Metadaten werden innerhalb von Alfresco in einer relationalen Datenbank abgespeichert.

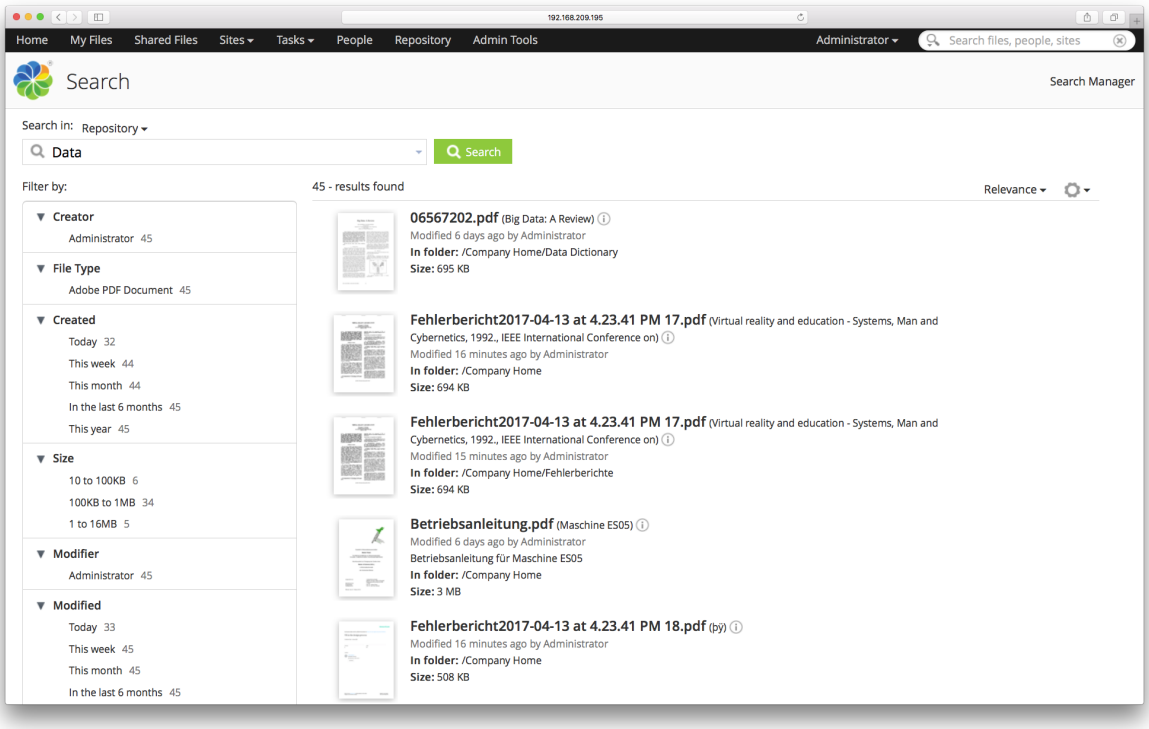

Abbildung 8.3: Volltextsuche mit dem Suchbegriff „Data“

Die gespeicherten Dokumente können in Alfresco mithilfe der Volltextsuche, welche auf Apache Solr basiert, durchsucht werden. Abbildung 8.3 zeigt eine Beispielsuche mit dem Suchbegriff „Data“. Auf der Abbildung ist gut zu erkennen, dass die Suche nicht nur nach Dateinamen erfolgt, sondern den Inhalt der PDF-Dateien in die Suche miteinbezieht.

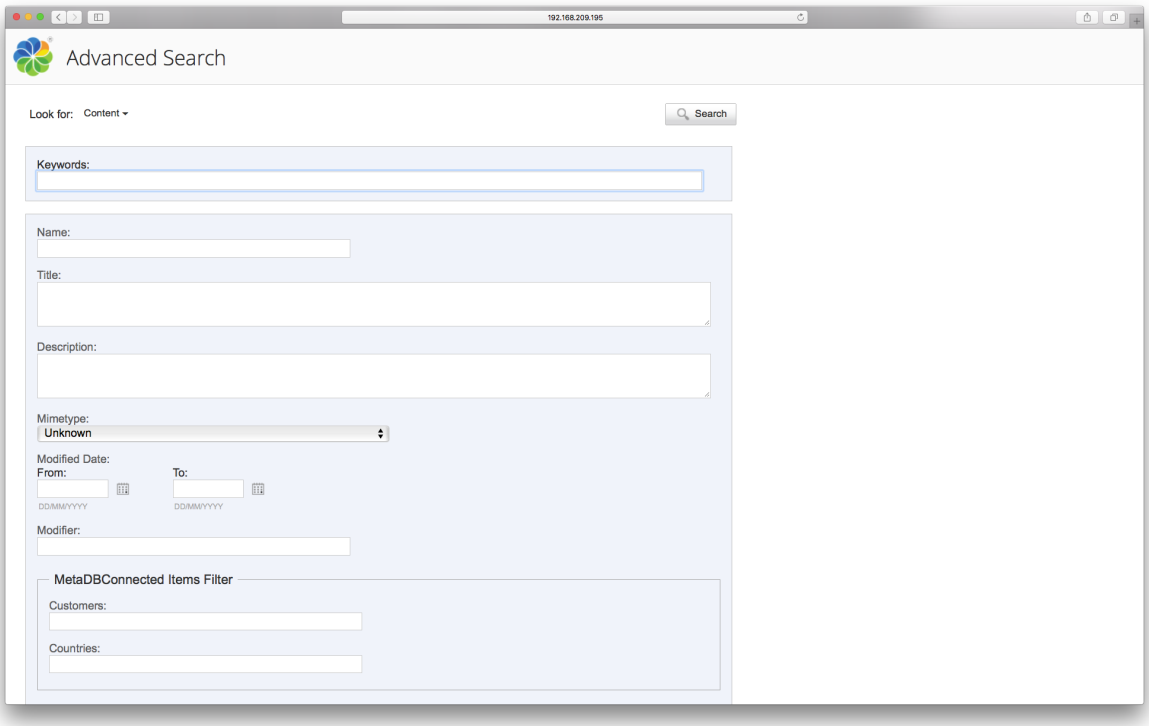

Abbildung 8.4: Das erweiterte Such-Formular mit MDBC Plugin

In Abbildung 8.4 ist die erweiterte Suche von Alfresco sowie das Suchformular des MDBC Plugin sichtbar. Dabei wird das zusätzliche Menü des MDBC Plugins nicht korrekt dargestellt, was auf Fehler im Plugin zurückzuführen ist. Eigentlich sollte hier ein Menü wie in Abbildung 8.2, zum Hinzufügen von strukturierten Daten angezeigt werden.

8.3.1 Lösungsstrategien beim Einrichten von Alfresco

Bei der Einrichtung und Nutzung von Alfresco und dem MDBC Plugin sind verschiedene Probleme aufgetreten. Einerseits gab es initial Probleme, das Plugin zu nutzen, da nicht alle Funktionen wie spezifiziert funktionierten. Deshalb musste, um das Plugin mit Alfresco zu nutzen, zuerst ein Fehler innerhalb der Software behoben werden. Der Fehler führte dazu, dass keine Daten aus der angebundenen relationalen Datenbank an Dokumente in Alfresco verknüpft werden konnten. Um das Problem zu lösen, mussten zwei Dateien verändert werden:

1. Bei der *alfresco/tomcat/webapps/alfresco/WEB-INF/classes/alfresco/module/model/mdbc-model.xml* XML-Datei muss *<default><xml></xml></default>* unter *<type>d:text</type>* für jeden Connector-Aspekt eingefügt werden.

2. Bei der *alfresco/tomcat/webapps/alfresco/WEB-INF/classes/alfresco/extension/tempates/webscript/mdbc/aspect.get.js*
JS-Datei muss *aspectName="http://venzia.es/model/mdbc/1.0"+aspectName.split(":")[1];* nach *aspectName = args.aspectName;* eingefügt werden.

Neben diesem Fehler besteht zudem das Problem, dass nicht korrekt nach Daten aus der relationalen Datenbank gesucht werden kann. Dieses Problem ist einerseits bedingt durch das MDBC Plugin, andererseits durch die Suchfunktion der Community-Version von Alfresco. Das MDBC Plugin sollte eigentlich laut Hersteller ein zusätzliches Menü zum Hinzufügen von strukturierten Daten aus der Datenbank bereitstellen, dies wird allerdings nicht korrekt angezeigt. Es sollte jedoch trotzdem möglich sein, die Content-Objekte anhand von strukturierten Daten zu durchsuchen, indem der Suchbegriff direkt in das Suchfeld eingetragen wird. Das erweiterte Suchformular von Alfresco scheint jedoch nicht immer korrekt nach Aspekten zu suchen. Das Problem konnte trotz verschiedener Alfresco Versionen und mehreren Lösungsansätzen[3] nicht gelöst werden. Dieses Problem könnte in zukünftigen Versionen von Alfresco behoben sein. Hierbei ist jedoch zu beachten, dass die Suchfunktion für die Community Version gegenüber der One Version von Alfresco reduziert wurde.

[3] https://community.alfresco.com/thread/201242-share-aspects-and-advanced-search

9 Zusammenfassung und Ausblick

In dieser Arbeit wurden verschiedene Technologien zur Verknüpfung von strukturierten und unstrukturierten Daten verglichen. Zuerst wurden im Rahmen der Literaturrecherche bestehende Möglichkeiten zur Integration der Daten untersucht. Zur Integration von strukturierten Daten wurden relationale Datenbanken sowie Data Warehouses beleuchtet. Für unstrukturierte Daten wurden CMS, WCMS, DMS, ECMS, ontologiebasierte Systeme, das Hadoop-Ecosystem sowie weitere Big-Data-Lösungen vorgestellt. Anschließend wurden Technologien zur Verknüpfung von strukturierten und unstrukturierten Daten gesammelt. Diese Recherche ergab folgende Technologien: Aletheia, Alfresco CE, OpenKM, Cloudera CDH, Brown Dog, Globus Plattform, Open Semantic Framework, Elastic Stack und Apache Marmotta. Diese Technologien wurden anhand ihrer Lizenz, ihrer Sicherheit und der Möglichkeit zur Verknüpfung von strukturierten und unstrukturierten Daten bewertet, indem sie installiert und getestet wurden. Diese Kriterien gehen aus den Anforderungen an diese Arbeit hervor, denn einerseits müssen die Systeme in der Lage sein, Verknüpfungen zwischen Daten herzustellen. Andererseits ist die Geheimhaltung der Daten sowie die Möglichkeit, die Systeme zu verändern von essentieller Bedeutung. Dadurch wurde die Vorauswahl auf Cloudera CDH, Alfresco CE und Apache Marmotta reduziert. Für diese Softwarelösungen wurden detailliertere Bewertungskriterien aufgestellt, um diese miteinander vergleichen zu können. Die Bewertungskriterien wurden in funktionale und nicht funktionale Kriterien unterteilt. Die funktionalen Kriterien beinhalten: die Verknüpfungsmethodik, das eingesetzte Datenmodell, das System zur Verwaltung von Metadaten, die integrierte Suche, die verfügbaren Schnittstellen sowie die Skalierbarkeit. Die nicht funktionalen Kriterien beinhalten: die Lizenz, die Benutzbarkeit, die Einfachheit, die Stärken, die Sicherheit, die Flexibilität sowie Kollaborationsmöglichkeiten der Software. Diese Kriterien stammen einerseits aus der Literaturrecherche sowie den Anforderungen an diese Arbeit und andererseits aus den gesammelten Erfahrungen beim Untersuchen der relevanten Softwarelösungen. Die nach der Vorauswahl übriggeblieben Softwarelösungen wurden mithilfe einer Tabelle anhand der Bewertungskriterien verglichen. Anschließend wurde durch den vorher definierten Anwendungsfall ein System selektiert, das die Anforderungen am besten erfüllt. Bei dieser Entscheidung fiel die Wahl auf Alfresco CE. Denn diese Technologie bietet ein gutes User Interface zur Verfügbarmachung der verknüpften Daten, die Möglichkeit relationale Datenbanken anzubinden und ist in der Lage PDF-Dateien zu verwalten. Mithilfe von Testdaten wurde zum Schluss der Anwendungsfall simuliert, indem Daten aus einer relationalen Datenbank mit in Alfresco gespeicherten PDF-Dateien verknüpft wurden.

Ausblick

Für weitere Arbeiten wäre es interessant, im Hinblick auf die entstandenen Schwierigkeiten beim Umsetzen des Anwendungsfalls mit Alfresco, diesen mit einer der beiden anderen Technologien zu realisieren. Diese Lösungen sind weniger leichtgewichtig, bieten allerdings die Möglichkeit, größere Datenmengen zu verwalten und diese besser zu analysieren. Für die Umsetzung des Anwendungsfalls mit Cloudera oder mit Apache Marmotta wäre zudem mehr Eigenentwicklungsaufwand nötig. Für Apache Marmotta wäre es notwendig, eine Schnittstelle für relationale Datenbanken sowie ein Frontend für die Produktionsarbeiter zu entwickeln. Für die Verknüpfung der Daten innerhalb der Ontologie wäre außerdem ein System notwendig, das diese Aufgabe händisch oder programmatisch erfüllt. Da die Daten in Marmotta durch Tripel-Stores organisiert sind, wäre es möglich logische Schlussfolgerungen aus den Daten abzuleiten, wodurch interessante Analyse- und Suchmöglichkeiten entstehen. Cloudera würde ebenfalls die Entwicklung eines Frontends sowie Skripts zur automatischen Verknüpfung der strukturierten und unstrukturierten Daten erfordern. Dies hätte jedoch den Vorteil, dass selbst sehr große Datenmengen automatisch verarbeitet werden könnten und dadurch eine händische Verknüpfung entfällt. Durch die Entwicklung eines spezifischen Frontends wären diese Lösungen besser an den Anwendungsfall angepasst und würden zudem eine horizontal skalierende Architektur bieten. Darüber hinaus wäre es interessant, diese möglichen Lösungsansätze mit der in dieser Arbeit vorgestellten Umsetzung durch Alfresco zu vergleichen. Dabei sind die Herausforderungen bei der Durchführung sowie die Möglichkeiten bei der Umsetzung mit einer der anderen Technologien von besonderem Interesse. Durch einen derartigen Vergleich wäre es möglich abzuschätzen, ob der erhebliche Mehraufwand bei der Umsetzung mit Marmotta oder Cloudera in der Praxis lohnenswert ist.

Literaturverzeichnis

[AAB11] AAB AG - Forschungszentrum Ladenburg. *Schlussbericht im Verbundprojekt Aletheia: Semantische Förderation umfassender Produktinformationen*. 2011.

[AD05] F. Ameri, D. Dutta. *Product Lifecycle Management: Closing the Knowledge Loops*. Bd. 2. 2005.

[ADE+13] A. Abelló, J. Darmont, L. Etcheverry, M. Golfarelli, J.-N. Mazón, F. Naumann, T. Pedersen, S. B. Rizzi, J. Trujillo, P. Vassiliadis, G. Vossen. *Fusion Cubes*. Bd. 9. 2013.

[Alf17] Alfresco Software. *Aflresco Documentation: Alfresco Community Edition*. 2017. URL: http://docs.alfresco.com/.

[Apa17] Apache Software Foundation. *Apache Impala (incubating)*. 2017. URL: https://impala.incubator.apache.org/.

[Arb13] Arbeitskreis Industrie 4.0. *Umsetzungsempfehlungen für das Zukunftsprojekt Industrie 4.0: Deutschlands Zukunft als Produktionsstandort sichern*. 2013.

[BA03] R. Blumberg, S. Atre. *The Problem with Unstructured Data*. 2003.

[BK08] H. Baars, H.-G. Kemper. *Management Support with Structured and Unstructured Data—An Integrated Business Intelligence Framework*. Bd. 25. 2008.

[Bun97] P. Buneman. *Semistructured Data*. 1997.

[CDES05] E. I. Chong, S. Das, G. Eadon, J. Srinivasan. *An Efficient SQL-based RDF Querying Scheme*. Oracle, 2005.

[CDG+06] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, R. E. Gruber. *Bigtable: A Distributed Storage System for Structured Data*. Google, Inc., 2006.

[Clo17a] Cloudera. *Cloudera Product Documentation*. 2017. URL: https://www.cloudera.com/documentation.html.

[Clo17b] Cloudera. *HUE*. 2017. URL: http://gethue.com/.

[Ela17] Elasticsearch BV. *The Open Source Elastic Stack*. 2017. URL: https://www.elastic.co/products.

[FL04] D. Ferrucci, A. Lally. *UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment*. 2004.

[GCA15] P. Gölzer, P. Cato, M. Amberg. *Data Processing Requirements of Industry 4.0 - Use Cases for Big Data Applications.* AIS Electronic Library, 2015. ISBN: 9783000502842.

[GKH+16] C. Gröger, L. Kassner, E. Hoos, J. Königsberger, C. Kiefer, S. Silcher, B. Mitschang. *The Data-Driven Factory: Leveraging Big Industrial Data for Agile, Learning and Human-Centric Manufacturing.* 2016.

[GL02] S. Gilbert, N. Lynch. *Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web.* 2002.

[GMKM04] N. Gold, A. Mohan, C. Knight, M. Munro. *Understanding service-oriented software.* Bd. 21. 2004.

[Grö15] C. Gröger. *Advanced Manufacturing Analytics: Datengetriebene Optimierung von Fertigungprozessen.* JOSEF EUL Verlag, 2015. ISBN: 978-3-8441-0420-2.

[Gru93] T. R. Gruber. *A translation approach to portable ontology specifications.* Bd. 5. 1993.

[GSM14] C. Gröger, H. Schwarz, B. Mitschang. *The Manufacturing Knowledge Repository.* 2014.

[HGHC12] M. Hausenblas, R. Grossman, A. Harth, P. Cudre-Mauroux. *Large-Scale linked data processing: Cloud computing to the rescue?* 2012.

[Hil11] K. Hildebrand. *Daten- und Informationsqualität.* Wiesbaden: Springer Fachmedien, 2011. ISBN: 9783834814531.

[Inm02] W. H. Inmon. *Building the Data Warehouse.* 2002.

[Int13] Intel. *Extract, Transform, and Load Big Data with Apache Hadoop.* 2013.

[JKX07] H.-B. Jun, D. Kiritsis, P. Xirouchakis. *Research issues on closed-loop PLM.* Bd. 58. 2007.

[KBF+10] S. Kunz, F. Brecht, B. Fabian, M. Aleksy, M. Wauer. *Aletheia–Improving Industrial Service Lifecycle Management by Semantic Data Federations.* 2010.

[KD10] A. Konzag, F. Dau. *Bereitstellung unstrukturierter Daten und Verknüpfung mit strukturierten Daten in der Konzeptentwicklung von Automobilen.* 2010.

[KGMW15] L. Kassner, C. Gröger, B. Mitschang, E. Westkämper. *Product Life Cycle Analytics – Next Generation Data Analytics on Structured and Unstructured Data.* Bd. 33. 2015.

[KMM+15] M. Kiran, P. Murphy, I. Monga, J. Dugan, S. S. Baveja. *Lambda architecture for cost-effective batch and speed big data processing.* 2015.

[KWG13] A. Katal, M. Wazid, R. H. Goudar. *Big Data: Issues, Challenges, Tools and Good Practices: 8 - 10 Aug. 2013, Noida, India.* Piscataway, NJ: IEEE, 2013. ISBN: 9781479901906.

[LC11] L. Li, W. Chou. *Design and Describe REST API without Violating REST: A Petri Net Based Approach.* 2011.

[LHC16] H.-K. Lin, J. A. Harding, C.-I. Chen. *A Hyperconnected Manufacturing Collaboration System Using the Semantic Web and Hadoop Ecosystem System*. Bd. 52. 2016.

[LL09] K. B. Laskey, K. Laskey. *Service oriented architecture*. Bd. 1. 2009.

[LMA09] A. Lang, M. Mera Ortiz, S. Abraham. *Enhancing Business Intelligence with unstructured data*. 2009.

[MDKW16] B. S. P. Mishra, S. Dehuri, E. Kim, G.-N. Wang, Hrsg. *Techniques and Environments for Big Data Analysis: Parallel, Cloud, and Grid Computing*. 1st ed. 2016. Bd. 17. Studies in Big Data. Cham: Springer International Publishing, 2016. ISBN: 9783319275185.

[Nat17] National center for supercomputing applications of university of illinois. *Brown Dog*. 2017. URL: http://browndog.ncsa.illinois.edu/.

[OAS12] OASIS Open. *Content Management Interoperability Services*. 2012.

[Ope17] Open Document Management System S.L. *Open KM Documentation*. 2017. URL: https://www.openkm.com/en.html.

[PCTM03] J. Poole, D. Chang, D. Tolbert, D. Mellor. *Common Warehouse Metamodel Developer's Guide*. 2003.

[PER12] E. Peukert, J. Eberius, E. Rahm. *A Self-Configuring Schema Matching System*. 2012.

[Pok13] J. Pokorny. *NoSQL databases: A step to database scalability in web environment*. Bd. 9. 2013.

[Rah14] E. Rahm. *Data Warehouses - Einführung: Sommersemester*. Leipzig, 2014.

[Rig09] W. Riggert. *ECM - Enterprise Content Management: Konzepte und Techniken rund um Dokumente*. 1. Aufl. Wiesbaden: Vieweg+Teubner Verlag / GWV Fachverlage GmbH Wiesbaden, 2009. ISBN: 9783834808417.

[Rus07] P. Russom. *BI Search and text analystics: New Additions to the BI Technology Stack*. 2007.

[Sch16] A.-W. Scheer. *Industrie 4.0: Von der Vision zur Implementierung*. 2016.

[Sei13] D. Seipel. *Vorlesung Datenbanken: im Wintersemester*. 2013.

[Str17] Structured Dynamics. *The Open Semantic Framework*. 2017. URL: http://wiki.opensemanticframework.org/index.php/Main_Page.

[TR11] A. Thor, E. Rahm. *CloudFuice: A Flexible Cloud-Based Data Integration System*. Bd. 6757. 2011.

[Uni17] University of Chicago. *Globus Platform-as-a-Service*. 2017. URL: https://www.globus.org/platform.

[Upw17] Upwork Global. *Choosing the Right Software Stack*. 2017. URL: https://www.upwork.com/hiring/development/choosing-the-right-software-stack-for-your-website/.

[van01] J. van den Hoven. *Information Resource Management: Foundation for Knowledge Management*. Bd. 18. 2001.

[YHL+16] C. Yang, Q. Huang, Z. Li, K. Liu, F. Hu. *Big Data and cloud computing: Innovation opportunities and challenges*. Bd. 10. 2016.

[Yon17] X. X. Yongkui Liu. *Industry 4.0 and Cloud Manufacturing: A Comparative Analysis*. 2017.

Alle URLs wurden zuletzt am 10. 05. 2017 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift