

Wintersemester 2019/2020

Visualisierungen für IT-Konsolidierungsprojekte

Thesis

zur Erlangung des Grades

Bachelor of Science

im Studiengang WirtschaftsNetze

an der Fakultät Wirtschaftsinformatik

der Hochschule Furtwangen University

vorgelegt von

Katharina Schemel

Referenten:

Prof. Dr. Marianne Andres

Martin Jelen

Eingereicht am

25. Februar 2020

Eidesstattliche Erklärung

Ich Katharina Schemel erklären hiermit an Eides statt, dass ich die vorliegende Bachelorthesis selbständig und ohne unzulässige fremde Hilfe angefertigt habe.

Die verwendeten Quellen sind vollständig zitiert.

Furtwangen, den 25. Februar 2020

Inhaltsverzeichnis

1	Einleitung	1
1.1	Motivation	2
1.2	Ziel der Arbeit	3
2	Grundlagen	4
2.1	Enterprise Architecture Management	4
2.2	Visualisierungskonzepte	6
2.3	Funktionsweise einer Graph-Datenbank	10
2.4	Cypher	12
2.5	Node.js	14
2.6	Webserver	15
2.7	Webtechnologien zur Visualisierung	16
3	Konzeption	17
3.1	Graphenmodell	17
3.2	Visualisierungskonzepte für Graphen	19
3.3	Kombiniertes Visualisierungskonzept	22
4	Implementierung	27
4.1	Infrastrukturaufbau	27
4.2	Entwicklungsumgebung	28
4.3	Beschreibung der implementierten Anwendung	30
4.4	Verwendete Technologien	35
4.5	Inbetriebnahme der erstellten Anwendung	36

4.5.1	Inbetriebnahme der Entwicklungsumgebung.....	38
4.5.2	Inbetriebnahme des finalen Build.....	40
4.6	Ergebnisse.....	40
5	Evaluation	43
6	Zusammenfassung und Ausblick	44
6.1	Zusammenfassung	44
6.2	Ausblick	45
	Glossar	49
	Anhang.....	51

Abbildungsverzeichnis

Abbildung 1: Brücke zwischen Business und IT (Business-IT, 2020)	5
Abbildung 2: Bebauungsplan Grafik.....	7
Abbildung 3: Portfolio-Grafik	8
Abbildung 4: Funktionales Referenzmodell.....	9
Abbildung 5: Zusammenhang Entitäten-Beziehungsmodell, Relationenmodell, Graphenmodell.....	11
Abbildung 6: Ergebnis Relationenabfrage.....	13
Abbildung 7: Ergebnis Relationenabfrage mit Einschränkungen auf Attributwerte	13
Abbildung 8: Client-Server Modell.....	15
Abbildung 9: Graphenmodell.....	18
Abbildung 10: Grad der Vernetzung.....	19
Abbildung 11: Darstellung einer spezifischen Relation	20
Abbildung 12: Ampelsystem.....	21
Abbildung 13: Darstellung in Abhängigkeit zu quantitativen Attributen	21
Abbildung 14: Tech Radar	23
Abbildung 15: Erweitertes Tech Radar Beziehung.....	25
Abbildung 16: Erweitertes Tech Radar Dichte der Vernetzung.....	26
Abbildung 17: Infrastrukturaufbau	28
Abbildung 18: Entwicklungsumgebung	29
Abbildung 19: Ablauf des Programms.....	31
Abbildung 20: Call-Back Methode.....	32
Abbildung 21: Chaining.....	33
Abbildung 22: Zusammenhang Knoten und Kanten.....	34
Abbildung 23: Start Datenbankserver durch PowerShell	37
Abbildung 24: Webinterface Datenbankserver.....	38
Abbildung 25: Start Webservers durch PowerShell	39
Abbildung 26: Visualisierung des gesamten Graphen	40
Abbildung 27: Objekte in Abhängigkeit zu quantitativen Attributen.....	41
Abbildung 28: Erweiterung durch Relation	42

1 Einleitung

In der Realität ist es beispielsweise so, dass innerhalb eines Unternehmens viele verschiedene IT-Systeme verwendet werden, welche schnell schwierig zu überblicken sind, so (Winter et al., 2013). Diese Systeme beinhalten Programme, welche auch gepflegt werden müssen. Dies resultiert beispielsweise daraus, dass diese immer auf dem aktuellen Stand sein sollten, um Sicherheitslücken zu vermeiden. In großen Unternehmen hat man kaum einen Überblick, welche IT-Systeme in den einzelnen Abteilungen verwendet werden. Daher empfiehlt sich laut (Hanschke, 2013), eine Konsolidierung vorzunehmen. Das bedeutet, dass die IT-Systeme verschlankt werden.

Ein Aspekt der Konsolidierung ist die Ineffizienz von IT-Systemen zu eliminieren. Oft werden verschiedene Systeme verwendet, welche im Prinzip, das exakt gleiche tun. Daher wäre es von Vorteil eine einheitliche Struktur festzulegen, um den Bestand der Software zu vereinfachen.

Eine andere Thematik bei der Konsolidierung ist die Entscheidung, welche Komponenten konsolidiert werden sollten. Eine Konsolidierung bietet eine Entscheidungsgrundlage für zu konsolidierende Komponenten. Um eine aussagekräftige Entscheidungsgrundlage bilden zu können, wird eine solide Datengrundlage vorausgesetzt. Die Entscheidung, welche Komponenten letztlich konsolidiert werden, liegt im Ermessen des Unternehmens. Der Anstoß für eine IT-Konsolidierung kann aus aktuellen Anlässen erfolgen (Probleme innerhalb des Unternehmens, Pflege der IT-Landschaft oder Fusionen von mehreren Unternehmen) oder auch aufgrund eines strategischen Nutzens des Unternehmens.

All diese Gründe führen zu der Konsolidierung, um beispielsweise den Geschäftswert bzw. die Betriebskosten des Unternehmens zu optimieren oder zur Identifikation von Geschäftsprozessen. Eine Konsequenz der Identifikation und der Zuordnung von Geschäftsprozessen kann beispielsweise Outsourcing sein. Somit ist deutlich zu erkennen, dass die Konsolidierung ein wichtiger Aspekt in Unternehmen darstellt.

Das Problem bei der Konsolidierung ist jedoch, die einzelnen Komponenten zu identifizieren. Eine Datenbeschaffung ist recht zeitaufwendig und kann fast nie vollständig erfolgen. Hieraus entsteht meist ein Vorgehen auf Basis von Teilbeständen, was letztlich nicht dem Ideal entspricht.

Die Komponente Visualisierung wird sowohl für die Präsentation der Analyseergebnisse wie auch als Werkzeug für die Analyse eingesetzt. Um die Analyse zu verbessern, ist es vorstellbar Graphdatenbanken in diesen Prozess mit einzubinden, da ein Graph schnell einen Überblick über Beziehungen und Zusammenhänge der einzelnen Komponenten vermittelt. Die Inhalte der Visualisierung werden mittels der Graphdatenbank generiert. Daraus kann eine Entscheidungsgrundlage gebildet werden, welche Komponenten entsprechend eliminiert oder verändert werden sollten.

1.1 Motivation

Die Thematik der IT-Konsolidierung ist noch recht jung und sehr aktuell. Da eine IT-Konsolidierung in Unternehmen meist intern vorgenommen wird, hat man diesbezüglich kaum Einblicke. Aus diesem Grund sei hier beispielsweise die IT-Konsolidierung des Bundes näher beleuchtet. Diese hat erst im Jahr 2015 begonnen. Daher ist es auch sehr spannend, diese Thematik weiter voranzubringen. Auslöser für eine IT-Konsolidierung kann z. B. eine vorherige Fusion darstellen. Laut (Bundesministeriums des Innern, für Bau und Heimat, 2020) sind die folgenden Punkte Ziele einer IT-Konsolidierung:

- Informationssicherheit trotz steigender Komplexität zu gewährleisten
- die eigene IT soll zu jederzeit Souverän und beherrscht sein
- flexibel auf innovative technologische Trends reagieren zu können
- ein zukunftsfähiger Betrieb, welcher leistungsfähig und stabil ist
- für das IT-Fachpersonal ein attraktiver Arbeitgeber bleiben

IT-Konsolidierung: Eine einfache Definition

„Konsolidierung bedeutet einfach übersetzt „Zusammenführung“. Im Bereich der IT sollen also Infrastrukturen, Datenbestände und Anwendungen zusammengeführt und idealerweise auch vereinheitlicht werden. Dadurch werden Kosten gespart, Abläufe vereinfacht sowie beschleunigt und die IT-Qualität erhöht sich insgesamt.“ (Klein, 2017)

Diese Definition zeigt, dass die IT-Konsolidierung sehr wichtig ist, da neben einer Kosteneinsparung auch die Qualität der IT erhöht werden kann, was insgesamt eine Bereicherung für die heutige Zeit darstellt, in welcher die Themen Digitalisierung und Industrie 4.0 immer allgegenwärtiger werden. Auch die Zusammenführung von Infrastrukturen, Datenbeständen und Anwendungen erleichtern das tagtägliche Arbeiten sowie die Pflege der Software erheblich. Die Konsolidierung wird im Enterprise Architecture Management (EAM) durchgeführt, da dieses einen konzeptionellen und organisatorischen Rahmen für eine Konsolidierung bietet. Dadurch kann die Unternehmensarchitektur

zielgerichtet aufgebaut und erweitert werden. Zusammenhänge können z. B. mittels Diagramme sehr gut darstellt werden. Besonders bei komplexen Sachverhalten ist dieses Vorgehen deutlich effizienter.

Die Thematik der Visualisierung im Kontext IT-Konsolidierung ist ein wichtiger Bestandteil dieser Arbeit, da eine Präsentation der Analyseergebnisse mittels Excel-Tabellen, wie es häufig von IT-Beratern praktiziert wird, nur einen geringen Mehrwert bietet. Anhand einer visuellen Darstellung dieser Ergebnisse, können Sachverhalte besser dargestellt werden und beispielsweise kann dadurch das Management besser ins Gespräch miteinbezogen werden.

1.2 Ziel der Arbeit

Das Ziel der Arbeit besteht in der „Grundsteinlegung“ auf dem Gebiet der Visualisierung in IT-Konsolidierungsprojekten mit Anbindung an eine Graphdatenbank für die Firma ISB AG. Die Arbeit gliedert sich somit in einen theoretischen wie auch einen praktischen Teil. Der theoretische Teil umfasst die Einarbeitung auf den Themengebieten EAM, Graphdatenbanken und Visualisierung mittels D3 (Data-Driven Documents). Neben der Einarbeitung stellt sich die zentrale Frage, wie Graphdatenbanken und die Visualisierung auf dem Gebiet der IT-Konsolidierungsprojekte zusammen zu bringen sind. Hierfür sollten prototypische Visualisierungskonzepte erarbeitet werden, unter Berücksichtigung von typischen Visualisierungskonzepten aus dem Bereich des EAM.

Nicht jeder Datensatz passt aufgrund seiner spezifischen Eigenschaften zu jedem Visualisierungskonzept. Aus diesem Grund sollten Datenbestände zunächst analysiert werden und dementsprechend eine geeignete Visualisierung aufgrund der Daten gewählt werden. Bezüglich einer Graphdatenbank folgt aus einer derartigen Analyse der Daten in der Regel ein Graphenmodell, welches im Zuge dieser Arbeit für eine fiktive IT-Landschaft prototypisch zu erstellen war. Dieses wiederum bildete die Grundlage für die Erstellung der Visualisierungskonzepte.

Die Grundlagenlegung für den praktischen Teil resultierte wiederum in der Erweiterung des theoretischen Teils. Es ging im Zuge dessen darum ein Konzept, für eine Infrastruktur für den Betrieb des im praktischen Teil entwickelten Prototyps zu generieren. Weiterhin musste ein Konzept für die Entwicklungsumgebung geschaffen und entsprechend umgesetzt werden, um eine nachhaltige und einfache Organisation des zu erstellenden Softwareprojekts zu gewährleisten. Folglich umfasst das Ziel des praktischen Teils die Entwicklung eines Prototyps, welcher diverse Kernfunktionalitäten exemplarisch implementiert. Diese sind unter anderem die Anbindung an eine Graphdatenbank sowie die Visualisierung der angefragten Daten mittels der JavaScript-Bibliothek D3.

2 Grundlagen

Um die Thematik der IT-Konsolidierung einordnen zu können, wird folgend das Enterprise Architecture Management (EAM) näher beschrieben. Zudem werden Visualisierungskonzepte mit typischen EAM Visualisierungen dargestellt. Da die Graphdatenbank ein wichtiger Bestandteil dieser Arbeit darstellt, werden auch diesbezüglich verschiedene Grundlagen aufgezeigt.

2.1 Enterprise Architecture Management

Die folgenden Definitionen von Enterprise Architecture und Enterprise Architecture Management zeigen die konkreten Abgrenzungen voneinander auf. In der Praxis werden diese Begriffe häufig verwendet. Auch wenn der Name fast identisch ist, gibt es dennoch Unterschiede.

Definition Unternehmensarchitektur (EA):

„Eine Unternehmensarchitektur (Enterprise Architecture) schafft eine gesamthafte Sicht auf das Unternehmen. Sie legt die wesentlichen fachlichen und IT-Strukturen fest und verknüpft sie miteinander. Auf dieser Basis lassen sich das Business und die IT und ihre Zusammenhänge beschreiben. Eine gemeinsame Sprachbasis, „eine Brücke“ zwischen Business und IT, wird geschaffen. So kann die strategische Weiterentwicklung von Business und IT aktiv gesteuert werden.“ (Hanschke, 2013)

Definition EAM:

„EAM ist ein systematischer und ganzheitlicher Ansatz für das Verstehen, Kommunizieren, Gestalten und Planen der fachlichen und technischen Strukturen im Unternehmen. Es hilft dabei, die Komplexität der IT-Landschaft zu beherrschen und die IT-Landschaft strategisch und businessorientiert weiterzuentwickeln. EAM ist ein wesentlicher Bestandteil des strategischen IT-Managements und beinhaltet alle Prozesse für die Dokumentation, Analyse, Qualitätssicherung, Planung und Steuerung der Weiterentwicklung der IT-Landschaft und der Geschäftsarchitektur.“ (Hanschke, 2013)

Einfach gesagt, fungiert die Enterprise Architecture als eine Art Vermittlerrolle zwischen der IT und den entsprechenden Fachbereichen eines Unternehmens, so (BITKOM, 2011). Man könnte es auch bildlich als eine Brücke beschreiben, welche diese Bereiche verbindet.



Abbildung 1: Brücke zwischen Business und IT (Business-IT, 2020)

Durch diese Verbindung wird ein Rahmen für den Ausbau der IT-Landschaft zur Verfügung gestellt. Dieser Rahmen umfasst laut (BITKOM, 2011) strategische, konzeptionelle und organisatorische Aspekte. Weiter führt (BITKOM, 2011) aus, dass besonders die Methoden der Umsetzung und die Überprüfung im Hinblick einer Kosten-Nutzen-Betrachtung im Fokus stehen.

Das Enterprise Architecture Management wiederum „umfasst die Aufgaben zur Erstellung, Pflege und Umsetzung einer EA“ (BITKOM, 2011). Daraus folgt, dass das EAM ein strukturierter Ansatz für die Erstellung, Verwaltung und Nutzung der von EA bereitgestellten Modelle ist.

Je größer ein Unternehmen wird, desto bedeutender ist die Rolle des EAM. Durch neue Technologien und Schnittstellen steigt die Komplexität und der IT-Landschaft wird unübersichtlich. Durch Fehlen oder Vernachlässigen des EAM können Redundanzen und inkonsistente Daten auftreten. Diese Problematik ist unter anderem die Grundlage für die IT-Konsolidierung. Aber auch mit EAM kann über einen längeren Zeitraum eine IT-Konsolidierung notwendig werden, beispielsweise in Folge einer Fusion. Durch das Bereitstellen diverser Hilfsmittel durch EAM, kann die Kontrolle über die IT-Landschaft wiederhergestellt werden und sie kann strategisch und businessorientiert weiterentwickelt werden.

EAM wird oft eingesetzt, da es Transparenz schafft. Diese ist maßgeblich für die Bewertung der IST-Situation und zeigt bei der IT-Landschaft schon nach kurzer Zeit, in welchem Bereich sich Kosten einsparen lassen.

Laut (Hanschke, 2013) kann in Problemfällen durch entsprechende Visualisierungen schneller Klarheit verschafft werden, z. B. welche Geschäftsprozesse von einem Ausfall des IT-Systems betroffen sind, wo die entsprechenden Verantwortlichkeiten liegen oder welche Abhängigkeiten zwischen den IT-Systemen bestehen.

Um die Leistungsfähigkeit der IT einordnen zu können, sind in der Praxis Bewertungen notwendig. Diese sind zumeist unscharfe, relative Bewertungen wie z. B. sehr hoch, hoch, mittel und gering. Diese Bewertung ist einfach und verständlich, damit sie von Fachbereichen, Anwendern und Managern angewendet werden kann. Die Fachbereiche und die Fachfremden Personen sollten jeweils eine solche Bewertung vornehmen, um festzustellen inwieweit sich das Eigen- und Fremdbild unterscheidet und anschließend sollte entsprechend vorgegangen werden, um diese beiden Ansichten anzugleichen, führt (Hanschke, 2013) weiter aus.

2.2 Visualisierungskonzepte

Farben und Muster sprechen den Menschen schneller an als eine einfache Tabelle mit Informationen. Daher ist eine Visualisierung von Daten wichtig, um das Interesse schneller auf relevante Aspekte zu lenken. Bei der Darstellung von Inhalten ist es wichtig sich vorab zu überlegen, welcher Sachverhalt näher dargestellt werden soll und entsprechend können hierfür unterschiedliche Visualisierungsarten mit passenden Eigenschaften herangezogen werden.

Grundsätzlich sei an dieser Stelle darauf verwiesen, dass nicht jede Visualisierungsart für die Darstellung eines beliebigen Sachverhalts geeignet ist. Beispielsweise kann ein Ampeldiagramm sehr gut verwendet werden, um einen Fortschritt in einem Prozess oder die Kritikalität einer Eigenschaft darzustellen. Es ist jedoch nicht geeignet komplexere Beziehungen zwischen Personen, Prozessen oder Softwareprodukten darzustellen.

Auch für den Bereich des EAM gibt es mehrere gängige Visualisierungsarten, welche laut (Hanschke, 2013) die Aussagekraft der Informationen unterstreicht. Auch auf diesem Gebiet gestaltet sich der Einsatzbereich solcher Grafikkonzepte analog zum Einsatzbereich der Ampel Grafik. Sie sind jeweils Anhand ihrer Eigenschaften für die Darstellung gewisser Sachverhalte besser geeignet und für andere schlechter. Beispiele für Visualisierungsarten aus dem Bereich des EAM seien gegeben durch folgende Grafikkonzepte:

- Bebauungsplan-Grafik
- Portfolio-Grafik
- Cluster-Grafik

Die drei genannten Grafikkonzepte wurden exemplarisch aus dem Best-Practice-Visualisierungskatalog gewählt. Der Visualisierungskatalog von Hanschke wurde hierfür gewählt, da dieser aussagekräftiger war als der, der TU

München. Die gewählten Konzepte zeigen durch ihre unterschiedlichen Ausprägungen die verschiedenen Möglichkeiten, welche eine Visualisierung im Bereich des EAM leisten kann. Diese drei Grafikkonzepte werden folgend näher erläutert und zur Veranschaulichung nach Vorlage von (Hanschke, 2013, 2016) nachgestellt.

■ Bebauungsplan-Grafik

Diese Grafik stellt die Zusammenhänge zwischen Elementen der Unternehmensarchitektur in Form einer Matrix dar. Somit können u. a. Informationssysteme zu Geschäftsprozessen und Geschäftseinheiten in Beziehung gesetzt werden. Durch die flexible Struktur können viele Fragestellungen beantwortet werden. Eine solche Fragestellung könnte nach (Hanschke, 2013) beispielsweise sein, welche Geschäftseinheit nutzt welche Komponente für welchen Geschäftsprozess. Zudem kann der Einsatz von farblichen Komponenten diese Visualisierung nochmals unterstreichen.

Die Bebauungsplan-Grafik unterteilt sich in drei Unterpunkte. Diese sind die fachliche Bebauungsplan-Grafik, technische Bebauungsplan-Grafik und die „typische“ Bebauungsplan-Grafik. Für das EAM ist besonders die Ausprägung der „typischen“ Bebauungsplan-Grafik interessant, da diese eine verbreitete Form zur IT-Unterstützung des Unternehmens ist. In einem Bebauungsplan der IT-Branche werden laut (BITKOM, 2011) gegenwärtige und zukünftige Infrastrukturen, als auch Anwendungssoftware definiert. Diese sollen die Geschäftsprozesse eines Unternehmens unterstützen.

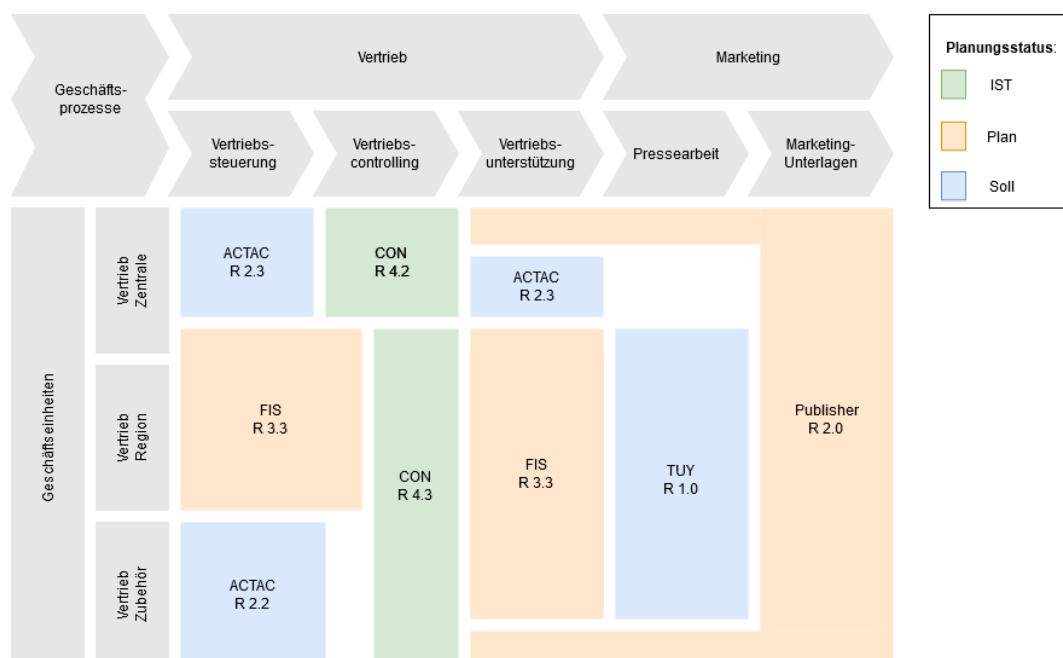


Abbildung 2: Bebauungsplan Grafik

Diese Abbildung zeigt die Geschäftsprozesse auf der X-Achse und die Geschäftseinheiten des Vertriebs auf der Y-Achse. Die einzelnen Bebauungselemente inmitten dieser Grafik geben einen guten Überblick, welche Geschäftseinheiten für welche Geschäftsprozesse zuständig sind.

Diese Art der Grafik ist jedoch nicht zwingend kompatibel zu größeren Datenmengen mit einer netzwerkartigen Struktur, da Abhängigkeiten unter den Daten deutlich schwieriger dargestellt werden können. Dies resultiert aus der blockartigen Anordnung, welche ab einer gewissen Datenmenge eine optische Clusterung nur bedingt zulässt.

■ Portfolio-Grafik

Bei dieser Grafik können besonders gut sogenannte Wertigkeiten von Bebauungselementen oder auch Strategien für Bebauungselemente visualisiert werden. Ebenso eignet sie sich für die Abbildung von Nutzenpotenzialen oder Risiken. Hierbei kann der Ist-Zustand oder auch der Soll-Zustand bzw. deren Differenz dargestellt werden. Laut (Hanschke, 2013, 2016) sollten maximal fünf verschiedene Kriterien in einer Portfolio Grafik abgebildet werden. Diese sind jeweils die Achsenkriterien sowie die Größe, Farbe und der Kantentyp der Füllelemente. Der Kantentyp kann hierbei in der Form (Kreise, Dreiecke, Vierecke, etc.) bzw. in der Struktur der Linien (Stärke der Linie, Art der Linie) verändert werden. Mehr als fünf unterschiedliche Kriterien sollten zwecks Übersichtlichkeit nicht gewählt werden. Die tiefe und Komplexität der Grafik sollte auch vom Empfängerkreis abhängen. Für interne Zwecke kann die Grafik komplexer sein als für fachfremde Personenkreise wie beispielsweise den Vorstand. Hierbei sollte auf weniger Dimensionen geachtet werden.

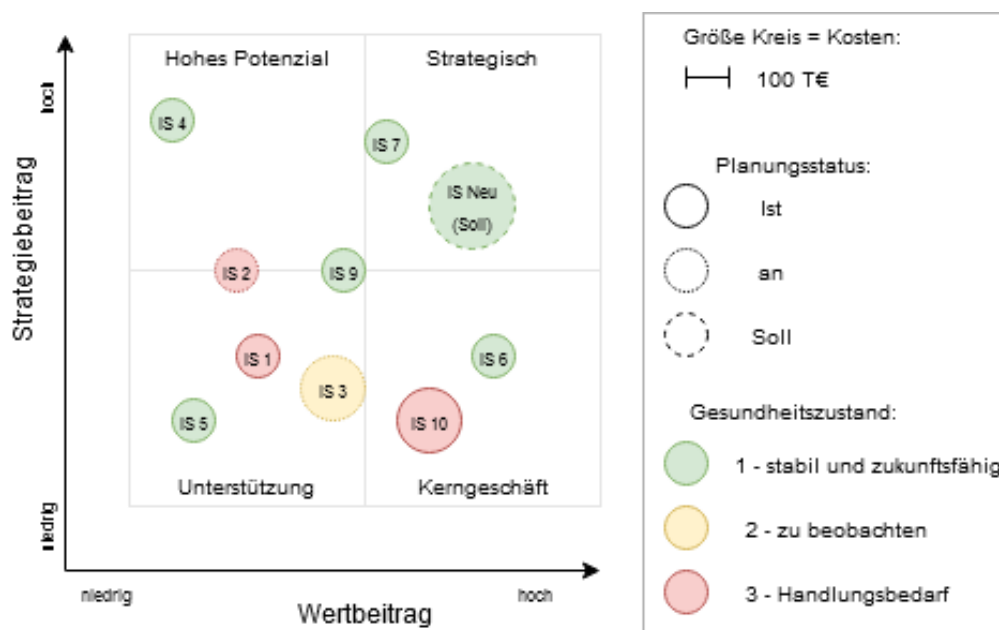


Abbildung 3: Portfolio-Grafik

Durch diese Grafik kann schnell ein Überblick anhand der Positionierung verschafft werden. Neben der Position gibt auch die Größe der einzelnen Elemente Aufschluss über beispielsweise Kosten. Diese Grafik eignet sich besonders gut, um messbare Eigenschaften oder eine kategorische Einordnung von Elementen darzustellen. Abhängigkeiten zwischen Elementen können maximal über die kategorische Einordnung realisiert werden.

- Cluster-Grafik

Das Konzept der Cluster-Grafik beschreibt das Aufteilen und Gruppieren von Bebauungselementen, anhand von Eigenschaften, Beziehungen oder definierten Kriterien. Bekannte Ausprägungen der Cluster-Grafik sind z. B. das fachliche Domänenmodell, welches die Ausprägung einer Prozesslandkarte hat oder eines funktionalen Referenzmodells. Im Fachlichen Domänenmodell werden die Kernstrukturen der Geschäftsarchitektur festgelegt, daher gibt es laut (Hanschke, 2013) innerhalb eines Unternehmens auch nur ein fachliches Domänenmodell.

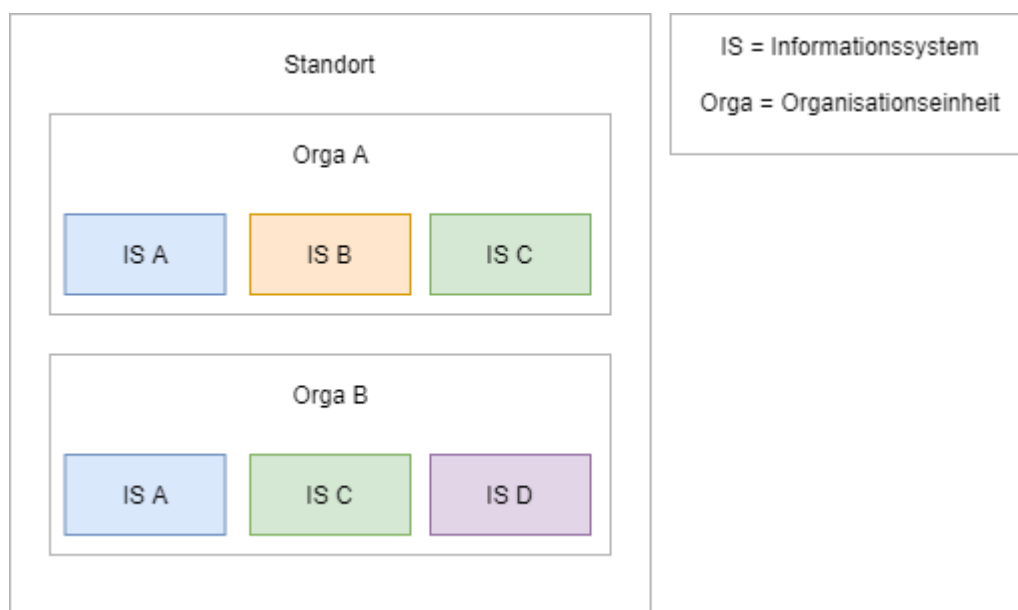


Abbildung 4: Funktionales Referenzmodell

Die Abbildung 4 zeigt ein funktionales Referenzmodell. Dieses eignet sich bis zu einem gewissen Grad, Abhängigkeiten zwischen Bebauungselementen darzustellen. Dies wird in diesem Fall auf Basis einer verschachtelten Darstellung realisiert. Somit eignet sich diese Darstellungsart, um die Zusammengehörigkeit zwischen Elementen der IT-Landschaft eines Unternehmens darzustellen. Die Grenzen werden jedoch erreicht, sobald Elemente in mehreren Gruppierungen zugehörig sind, da dies maximal durch eine redundante Aufführung kompensiert werden kann, wie die Abbildung 4 zeigt. Dies kann jedoch bei größeren Darstellungen wiederum die Lesbarkeit bzw. Interpretierbarkeit der Grafik beeinträchtigen. Es kann somit auch nicht auf einfachem Wege dargestellt werden, wie stark ein Softwareprodukt in ein Unternehmen integriert ist.

2.3 Funktionsweise einer Graph-Datenbank

Jeder ist bereits mit einem Graphen in Berührung gekommen. Sei es bei der Erstellung eines Mindmaps oder beim Skizzieren von Symbolen und Linien auf einer Tafel. Graphen sind nach (Hunger, 2014) sehr einfach, verständlich und vielseitig einsetzbar. Eine der bekanntesten Graphdatenbanken ist Neo4j. Der Name Neo4j leitet sich aus „Network Engine for Objects“ ab. Die Zahl „4“ hat in diesem Zusammenhang keine Verbindung zur Versionsnummer. Das „j“ steht dafür, dass die frühere Java-API entsprechend weiterentwickelt wurde. Zu Beginn war die Bedeutung von „4j“ „für Java“, führt (Trelle, 2017) aus. Mittlerweile kann, neben Java, auch in der Programmiersprache Scala entwickelt werden. Daher ist die Bedeutung „for Java“ nicht mehr ganz aktuell.

Eine Graphdatenbank gehört zu der Gruppe der NoSQL Datenbanken. Dies bedeutet, dass die Datenbank in der Regel einen nicht relationalen Ansatz verfolgt. Jedoch wird die Verwendung von SQL nicht gänzlich ausgeschlossen, da NoSQL für „Not only SQL“ steht.

Eine Relationale Datenbank verwendet Tabellen, welche Spalten und Zeilen für die Speicherung der Daten nutzt. Die NoSQL Datenbanken hingegen nutzen für die Organisation der Daten beispielsweise Attribut-Wert-Paare (Properties), Objekte, Dokumente oder Listen und Reihen. Ein großer Vorteil von NoSQL Datenbanken ist, dass sie dort ansetzen, wo SQL-basierte relationale Datenbanken an ihre Grenzen stoßen. NoSQL Systeme eignen sich besonders gut für große, exponentiell wachsende Datenmengen, um diese performant zu verarbeiten, wie beispielsweise im Bereich von Big Data. Dies resultiert aus der einfachen Unterstützung der Datenbankfragmentierung, welche durch den Verzicht auf ein Datenbankschema und auf die referenzielle Integrität ermöglicht wird. Im speziellen besagt die referenzielle Integrität, dass Datensätze nur auf bestehende Datensätze verweisen dürfen. Entsprechend kann auch nur ein Datensatz gelöscht werden, wenn auf diesen kein anderer Datensatz verweist.

Graphdatenbanken werden nicht tabellarisch geführt, sondern in Form von Labeln organisiert. Diese sind zwar vergleichbar mit den Tabellen der relationalen Datenbanken, haben jedoch den Unterschied, dass diese ohne eine feste Vorgabe der enthaltenen Attribute auskommt. Die Speicherung der Daten erfolgt somit schemafrei. Dies erlaubt zusätzliche Informationen wie mangelnde Attribute zu einem späteren Zeitpunkt problemlos nachzutragen. Ein implizites sturkturiertes Schema sei dennoch durch das Graphenmodell innerhalb einer Graphdatenbank gegeben. Das Graphenmodell kann als Äquivalent zum Relationenmodell der Relationalen Datenbanken interpretiert werden. Es resultiert analog zum Relationenmodell aus dem Entitäten-Beziehungsmodell. Dieser Zusammenhang sei mit Abbildung 5 nach (Meier et al., 2016) näher verdeutlicht.

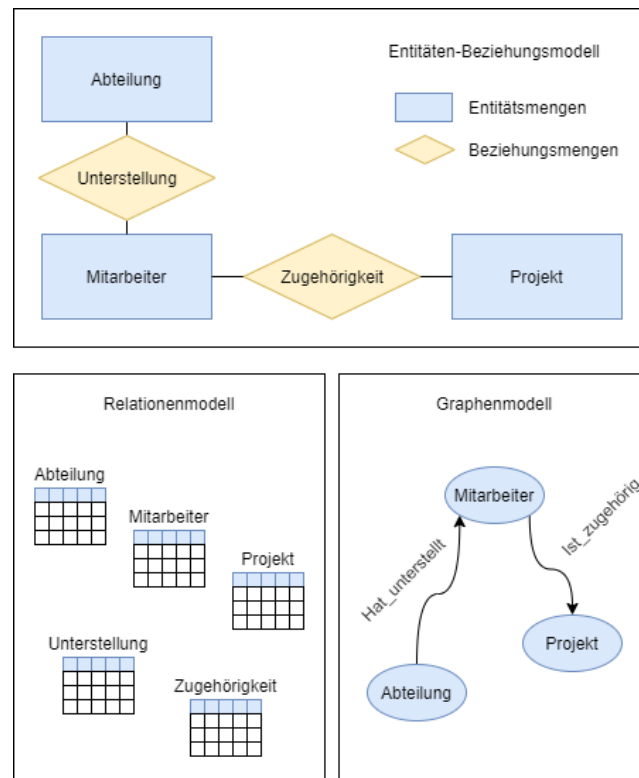


Abbildung 5: Zusammenhang Entitäten-Beziehungsmodell, Relationenmodell, Graphenmodell

Laut (Maier et al., 2016) werden die zu speichernden Daten mithilfe von sogenannten Knoten und Kanten dargestellt. Hier werden Objekte in Knoten und Beziehungen zwischen Knoten in Kanten abgebildet. In einem Knoten ist die Beziehung nicht in ihrer Art oder Anzahl beschränkt, ergänzt (Matzer, 2019).

In der Literatur werden Knoten häufig auch als Vertex (V) und Kanten als Edges (E) benannt. Dies sei an dieser Stelle bezüglich der Vollständigkeit erwähnt. Folgend wird diese Terminologie jedoch nicht weiterverwendet.

Die Kante stellt eine Verbindung zwischen den Knoten dar. Diese Verbindung kann als eine Linie oder als ein Pfeil visualisiert werden. Eine Kante ist gerichtet, das bedeutet, dass sie einen Start- sowie einen Endpunkt besitzt. Die Richtung der Kante wird bei der Initialisierung der Beziehung definiert. Die Beziehungen zwischen den Knoten können Eigenschaften besitzen. Diese Daten können analog zu den Attributen der Knoten abgefragt werden. Nach (Luber et al., 2017) verzichtet eine Graphdatenbank auf verschachtelte Beziehung, was die hohe Performance für die Speicherung, sowie Abfrage der Informationen erklärt.

Die Information, welche Beziehungen zwischen Daten vorliegen, ist unter gewissen Umständen besonders förderlich. Diese Beziehungen kann ein Graph ausgesprochen gut darstellen. Hieraus ergibt sich der hohe Nutzen der Verwendung einer Graphdatenbank für Daten, welche eine netzwerkartige Struktur aufweisen.

Eine Besonderheit der Graphdatenbanken ist nach (Maier et al., 2016), die Eigenschaft der indexfreien Nachbarschaft. Das Datenbanksystem kann somit die direkten Nachbarn eines Knotens ermitteln, ohne sämtliche Kanten

berücksichtigen zu müssen. Dies ist beispielsweise in relationalen Datenbanksystemen nötig, welche sogenannte Beziehungs- oder Verknüpfungstabellen verwenden. Aus diesem Sachverhalt ergibt sich, dass die Abfrage von Beziehungen auf Knoten unabhängig von der gespeicherten Datenmenge sind und somit eine konstante Geschwindigkeit besitzen.

Laut (Matzer, 2019) nutzen bereits große Unternehmen wie z. B. Google, Facebook, Microsoft und LinkedIn die Graphdatenbank. Ebenso haben auch Unternehmen wie die NASA oder die Schweizer Bank UBS die Graphdatenbank für sich entdeckt.

Die gängigsten Abfragesprachen für Graphdatenbanken sind Apache TinkerPop Gremlin, SPARQL und Cypher. Da, in dieser Arbeit mit der Graphdatenbank Neo4j gearbeitet wird, betrachtet diese Arbeit folgend nur die Abfragesprache Cypher näher. Cypher wurde von Neo4j entwickelt und erfüllt somit die besten Voraussetzungen, was Kompatibilität angeht, nach (Matzer, 2019). Zudem ist Cypher sehr verbreitet und in seiner Struktur der Abfragesprache SQL von Relationalen Datenbanken recht ähnlich. Durch diese ähnliche Struktur ist eine Umstellung von SQL auf Cypher nicht zu zeitaufwendig und komplex. In Kapitel 2.4 wird auf die Abfragesprache Cypher näher eingegangen.

2.4 Cypher

Cypher gehört zu der Gruppe der „deklarativen Abfragesprachen“. Die bekannteste Abfragesprache dieser Gruppe ist SQL. Der Fokus der deklarativen Abfragesprachen ist die Beschreibung eines Problems. Gegenüber den deklarativen Abfragesprachen steht die Gruppe der „imperativen Sprachen“, wie beispielsweise Java oder C++. Nach (Böhm, 2005) ist die imperative Programmierung ein Programmierstil, nach welchem „ein Programm aus einer Folge von Anweisungen besteht, die vorgeben, in welcher Reihenfolge was vom Computer getan werden soll“.

Wird folgend Cypher als problembeschreibende Sprache in Bezug auf Graphen betrachtet, kann von der Beschreibung von Mustern innerhalb von Graphen gesprochen werden. An folgendem Beispiel wird dieser Sachverhalt verdeutlicht. Es wird ein Graph angenommen, bestehend aus zwei Klassen von Knoten und einer gerichteten Beziehung zwischen den Knoten. Exemplarisch seien dies für die Knoten „Projekt“ und „Person und für die Beziehung „arbeitet_in“. Entsprechend beschreibt der exemplarische Graph, welche Person, in welchem Projekt arbeitet. Die Beschreibung des Musters des dargelegten Graphen mittels Cypher stellt sich wie folgt dar.

```
Match (a:Person) – [arbeitet_in] -> (b:Projekt) Return a, b
```

Die entsprechende Rückgabe sind alle Knoten und deren Verbindungen untereinander, wie in Abbildung 6 dargestellt. Projekte, zu denen keine Person in der Datenbank hinterlegt sind, werden nicht ausgegeben, da sie dem abgefragten Muster nicht entsprechen.

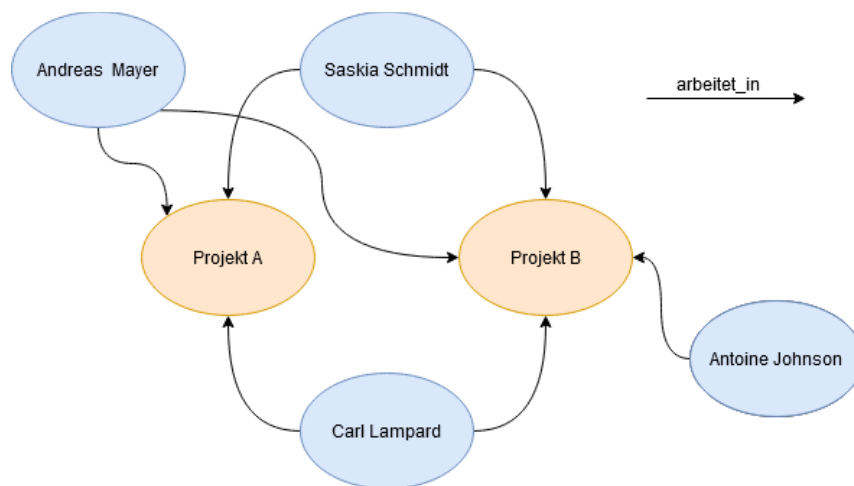


Abbildung 6: Ergebnis Relationenabfrage

Analog zu SQL bietet Cypher die Option, anhand von Attributen der Knoten und Kanten mittels einer Where-Bedingung das Muster zu filtern. Folgend sei dies mittels Cypher dargestellt.

```
Match (a:Person) – [arbeitet_in] -> (b:Projekt)
Where (a.name = „Saskia Schmidt“)
OR (a.name = „Antoine Johnson“)
Return a, b
```

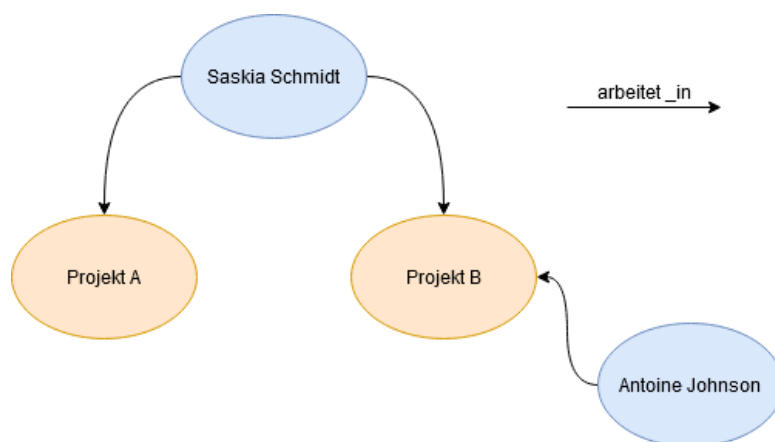


Abbildung 7: Ergebnis Relationenabfrage mit Einschränkungen auf Attributwerte

2.5 Node.js

Node.js ist eine plattformübergreifende Laufzeitumgebung basierend auf JavaScript. Verwaltet wird das Open-Source-Projekt NodeJS durch die OpenJS Foundation mit Unterstützung der Linux Foundation. Die ursprüngliche Hauptaufgabe von NodeJS besteht in der Ausführung von JavaScript-Code außerhalb eines Browsers. Es wird somit möglich, serverseitig dynamische Webseiteninhalte zu erstellen, bevor die Seite an den Benutzer übertragen wird. Entsprechend wird deutlich, dass NodeJS primär für die Entwicklung hoch performanter Webserver verwendet wird. Nativ wird von NodeJS nur JavaScript unterstützt. Es gibt jedoch Compiler, die es ermöglichen andere Sprachen, wie zum Beispiel CoffeeScript oder TypeScript in JavaScript zu übersetzen und somit für NodeJS nutzbar zu machen.

Auf der funktionalen Ebene ist NodeJS bezüglich der dynamischen Erstellung von Webseiteninhalten am ehesten mit PHP zu vergleichen, welches dies analog zu NodeJS ebenso unterstützt. Auf der technischen Ebene unterscheidet sich NodeJS jedoch deutlich von PHP. NodeJS führt Funktionen in der Regel parallel aus und verwendet sogenannte „Callbacks“ um die Fertigstellung oder das Scheitern einer Funktion zu signalisieren. PHP hingegen blockiert die meisten Funktionen bis zur Fertigstellung der vorherigen.

Neben der beschriebenen Hauptaufgabe wird NodeJS jedoch auch sukzessive als Werkzeug für diverse andere Aufgaben eingesetzt. In dieser Arbeit wurde NodeJS beispielsweise indirekt als Bestandteil der Entwicklungs- und Deploymentumgebung verwendet. Dies wird ermöglicht, durch den Node Package Manager (npm), welcher ein integrierter Bestandteil von NodeJS ist. Der npm ist ein Paketmanager für die Laufzeitumgebung von NodeJS, was letztlich als freies Repository für Paketerweiterungen für NodeJS interpretiert werden kann. Am Beispiel dieser Arbeit wurde eine Reihe von Paketerweiterungen wie beispielsweise „webpack“ oder „babel“ für die Entwicklung und das Deployment eingesetzt. Diese Erweiterungen reichen von Compileren bis hin zu Webservern für die lokale Entwicklung. Die im Rahmen dieser Arbeit eingesetzten Paketerweiterungen sind in Kapitel 4.4 im Detail aufgelistet.

2.6 Webserver

Die Hauptfunktion eines Webserver ist nach (Killelea, 2002) das Speichern, Verarbeiten und Übermitteln von Webseiten an Clients. Er operiert somit im sogenannten „Client-Server Modell“, welches in Abbildung 8 näher veranschaulicht ist. Ein Webserver kann im Allgemeinen eine oder mehrere Webseiten enthalten. Bei den bereitgestellten Seiten handelt es sich am häufigsten um HTML-Dokumente, die neben dem Textinhalt auch Bilder, Stylesheets und Skripte enthalten können.

Für Clientanfragen und Serverantworten ist das Hypertext Transfer Protocol (HTTP) von zentraler Bedeutung. Der Client stellt eine HTTP-Anfrage mit einem Uniform Resource Locator (URL), durch welchen die angeforderte Resource serverseitig identifiziert werden kann. Anschließend antwortet der Webserver nach der Anfragenverarbeitung dem Client mit einer entsprechenden Nachricht. Diese enthält wiederum Informationen zum Status der Verarbeitung (erfolgreich oder gescheitert) und gegebenenfalls den angeforderten Inhalt beziehungsweise die angeforderte Resource.

Während die Hauptfunktion darin besteht, Inhalte bereitzustellen, umfasst eine vollständige Implementierung von HTTP auch Möglichkeiten zum Empfangen von Inhalten. Diese Funktionalität wird zum Senden von Webformularen verwendet bis einschließlich des Hochladens von Dateien.

Viele generische Webserver unterstützen wie schon in Kapitel 2.5 aufgegriffen das serverseitige Skripting mit beispielsweise PHP (Hypertext Preprocessor) oder anderen Skriptsprachen. Dies bedeutet letztlich, dass das Verhalten des Webserver mittels separater Skripte erweitert beziehungsweise definiert werden kann, während die tatsächliche Serversoftware komplett unberührt bleibt. Die dynamische Generierung von Webseiteninhalten wird überwiegend zum Abrufen oder Ändern von Informationen aus Datenbanken verwendet.

Die aktuell am weitesten verbreiteten Webserver nach (w3techs.com, 2020) sind der Apache HTTP Server und der Nginx Webserver.

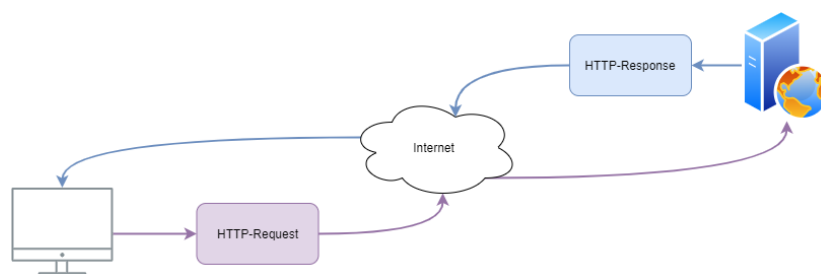


Abbildung 8: Client-Server Modell

2.7 Webtechnologien zur Visualisierung

Der Sinn einer Visualisierung ist, dass diese auch gesehen wird. Der schnellste und einfachste Weg eine Visualisierung für mehrere Menschen zugänglich zu machen ist über eine Webseite. Ein Webbrowser hat laut (Murray, 2017) den Vorteil, dass er nicht an ein Betriebssystem wie beispielsweise Windows, Mac OS, Linux, Android oder iOS gebunden ist. Somit wird lediglich für die Verwendung der Anwendung ein Internetzugang und einen Browser benötigt. Ein weiterer Vorteil ist, dass diese Variante kaum beziehungsweise keine Lizenz- und Schulungskosten mit sich bringt.

Für die Visualisierung von Graphen auf Webseiten gibt es diverse Bibliotheken. Unter anderem sei an dieser Stelle Data-Driven Documents (D3), sigma.js und Cytoscape genannt. D3 ist jedoch die am weitesten verbreitete Bibliothek der genannten Bibliotheken und auch Bestandteil der Aufgabenstellung dieser Arbeit. Aufgrund dieses Sachverhalts wird folgend nur D3 näher betrachtet.

D3 bietet neben der Möglichkeit, Graphen zu visualisieren noch zahlreiche weitere Anwendungsfälle. Alle gängigen Diagramme werden von D3 bereitgestellt, um Daten zu visualisieren. Die Bibliothek setzt hierbei auf standardisierte und bewährte Technologien wie Hypertext Markup Language (HTML), Scalable Vector Graphics (SVG) und Cascading Style Sheets (CSS). Der Einsatz von Webstandards bietet dem User nach (Bostock, 2019) die Möglichkeit mit D3 zu entwickeln, ohne ein proprietäres Framework nutzen zu müssen.

Das zentrale Prinzip von D3 ist das Binden von Daten an Objekte des Document Object Model (DOM). Diese wiederum können nach (Bostock, 2019) selektiert, manipuliert und transformiert werden. Der Fokus der Bibliothek liegt auf Flexibilität, Performance und Wiederverwendbarkeit von Code. Die Bibliothek eignet sich zudem sehr gut für große Datenmengen, was bezüglich der Thematik dieser Arbeit nicht gänzlich vernachlässigt werden darf. Für einen weiterführenden, tieferen Einblick in D3 sei auf die Online Dokumentation dieser verwiesen.

3 Konzeption

Dieses Kapitel befasst sich mit den theoretischen Überlegungen, welche in dieser Arbeit erarbeitet wurden. Konzepte bezüglich des Graphenmodell und Visualisierungen werden thematisiert. Es bildet somit die theoretische Basis, auf welcher der praktische Teil dieser Arbeit aufsetzt. Dieser wird in Kapitel 4 näher betrachtet.

3.1 Graphenmodell

Ein Graphenmodell ist bei der Erstellung einer Datenbank sehr wichtig, da dies die einzelnen Verbindungen und Abhängigkeiten abbildet. Die folgende Grafik zeigt ein Beispiel für ein Graphenmodell einer abstrahierten IT-Landschaft. Das Graphenmodell wurde so erweitert, dass es möglichst realistisch ist und viele eventuelle Szenarien abdecken kann, welche die Entscheidungsfindung unterstützen.

Der Aufbau des Graphenmodells ist in drei Spalten gegliedert. Die linke Spalte, welche in der Farbe Blau dargestellt ist, zeigt die Unternehmensstruktur. Bei der mittleren Spalte handelt es sich um das Informationssystem. Dieses ist die Verbindung zwischen der Unternehmensstruktur und der in der Farbe Grün dargestellten rechten Spalte, dem technischen Aspekt. In manchen Fällen kommt es vor, dass der Name der Relation sich wiederholt. Dies ist der Fall, wenn Verbindungen dieselbe Art der Beziehung aufweisen. Beispiele hierfür können der Abbildung 10 entnommen werden. Beziehungen zwischen Elementen innerhalb der Kategorie der Unternehmensstruktur (blau) sowie der Kategorie des technischen Aspekts (grün) sind zulässig. Des Weiteren werden auch Beziehungen innerhalb einer Objektklasse (Knoten) erlaubt. Als Beispiel sei hierfür eine Beziehung zwischen zwei Informationssystemen an dieser Stelle genannt. Diese könnte beispielsweise durch eine implementierte Schnittstelle hervorgerufen werden. Weiterhin sei erwähnt, dass eine Objektklasse sowie eine Relation mit Attributen erweitert werden kann. Dies hängt in der Realität von der entsprechenden Datengrundlage ab.

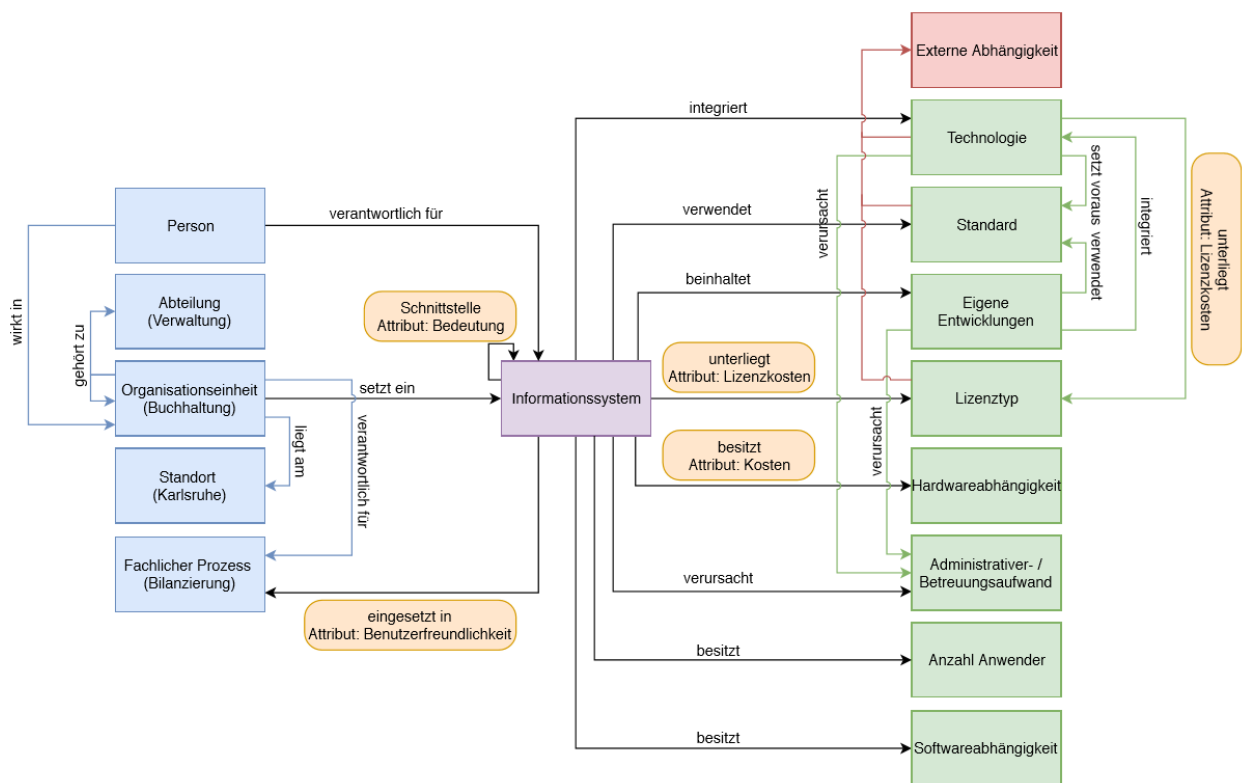


Abbildung 9: Graphenmodell

Graphenmodelle sind sehr aussagekräftig. Durch das visuelle Verständnis fungiert das Modell als eine Art Kommunikationstool zwischen Entwickler und Fachleuten. Wenn man ein konzipiertes Graphenmodell sieht, ist es zumeist selbsterklärend und man kann die Zusammenhänge des Modells schnell erkennen, ohne thematisch tiefe Kenntnisse des Projektes zu haben.

Laut (Wirtschaftslexikon Gabler, 2018) beschreibt ein Graphenmodell die zu verarbeitenden Daten eines Anwendungsbereichs mittels Grafik. Zudem zeigt dieses die Beziehungen, welche die Daten untereinander aufweisen.

Durch ein Graphenmodell können Redundanzen aufgedeckt werden. Ebenso wird dadurch ersichtlich, ob die Daten eindeutige Informationen aufzeigen. Unter Berücksichtigung dieser Punkte können Rückschlüsse über die Datenqualität getroffen werden. Ein qualitativ hochwertiges Graphenmodell ist somit ein essenzieller Bestandteil eines Softwareprojekts, welches eine Graphdatenbank als Ressource einbindet. Aus diesem Grund ist es wichtig sich vor der Implementierung mit dem Graphenmodell auseinanderzusetzen. Zusätzlich unterstützt der visuelle Entwurf der Datenstruktur, den Prozess der Implementierung.

3.2 Visualisierungskonzepte für Graphen

Neben den in Kapitel 2.2 genannten Best-Practice-Visualisierung ist nach (Hanschke, 2013) eine weitere der Graph. Ein Graph stellt die Beziehung zwischen mehreren Informationen visuell dar. Die Verbindungslinien, auch Kanten genannt, können durch verschiedene Farben und Linientypen beschaffen sein. Außerdem können diese auch mit einem Label versehen werden und somit weitere Informationen liefern. Die dargestellten Knoten zeigen Beziehungen und mögliche Abhängigkeiten untereinander auf, welche visuell schneller wahrgenommen werden als durch eine Liste, so (Hanschke, 2013) weiter.

Ein Graph ist im Gegensatz zu den in Kapitel 2.2 vorgestellten Visualisierungskonzepten deutlich agiler in der Darstellung. Er unterliegt keiner vorgegebenen Struktur wie es beispielsweise bei der Cluster-Grafik der Fall ist. Das einzige strukturierende Element des Graphen sind die Beziehungen der Knoten. Es liegt somit eine datengetriebene Struktur vor, mehr jedoch nicht. Des Weiteren impliziert die Verwendung einer Graphdatenbank zur Datenverwaltung auch die Visualisierung mittels eines Graphen näher zu betrachten.

Die Darstellung von Graphen innerhalb einer Webanwendung erwies sich als sehr performant. Eine Darstellung des gesamten Graphen bei einer großen Datenmenge kann jedoch schnell unübersichtlich werden und ist somit nicht zwingend für eine Entscheidungsgrundlage geeignet. Aus diesem Grund empfiehlt es sich, den Graphen in Subgraphen zu unterteilen, um die Übersichtlichkeit zu verbessern. Die folgenden Mockups stellen mögliche Subgraphen dar, welche in dem Szenario der IT-Konsolidierungsprojekten einen Mehrwert liefern können. Wie sich dieser Mehrwert gestaltet ist folgend auf die Mockups näher beschrieben. Exemplarisch wurden zwei der Konzepte in dem praktischen Teil der Arbeit realisiert.

Grad der Vernetzung:

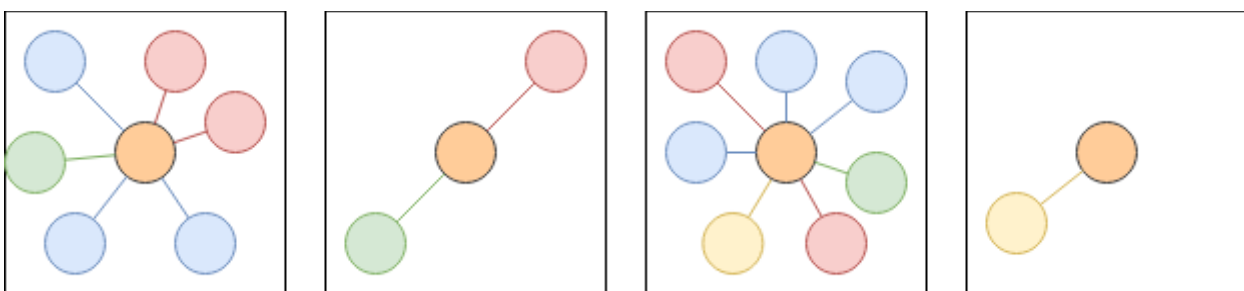


Abbildung 10: Grad der Vernetzung

Das Mockup aus Abbildung 10 stellt den Grad der Vernetzung eines Knotens dar. Die Abbildung zeigt dies für vier unterschiedliche Knoten eines Typs. Der Grad der Vernetzung beschreibt, wie viele Beziehungen ein Knoten besitzt.

In diesem Szenario werden alle Arten von Beziehungen berücksichtigt. Eine Filterung nach einem speziellen Typ von Beziehung wird somit nicht vorgenommen. Entsprechend kann dieses Kriterium herangezogen werden, um zu beurteilen, wie stark ein Bestandteil der IT-Landschaft in ein Unternehmen eingebunden ist.

Der orange farbige Knoten könnte beispielsweise ein Informationssystem darstellen, welches diverse Relationen zu Technologien, Organisationseinheiten und Fachprozessen besitzt. Möchte man nun das Informationssystem durch ein anderes ersetzen, kann diese Darstellung als Indikator für die Komplexität dieses Prozesses herangezogen werden. Sie gibt entsprechend direkt Aufschluss über den Aufwand bzw. die Machbarkeit ein Element der IT-Landschaft zu ersetzen. Ein weiterer Vorteil dieser Grafik wird in Bezug auf die Analyse einer IT-Landschaft deutlich. Der Grad der Vernetzung gibt auch eine Tendenz für die Bedeutung eines Elements der IT-Landschaft in einem Unternehmen.

Darstellung einer spezifischen Relation:

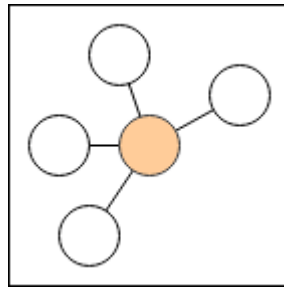


Abbildung 11: Darstellung einer spezifischen Relation

Ein weiteres Beispiel, wie Graphen die Analyse einer IT-Landschaft unterstützen können, sei durch das Mockup in Abbildung 11 dargestellt. Hier wird im Gegensatz zum Grad der Vernetzung nur ein Typ der Beziehung eines Knotens visualisiert. Dies wird beispielsweise relevant, wenn nicht vorhandene Redundanzen oder spezielle Beziehungen aufgedeckt werden sollen. Ein passendes Beispiel hierfür sei ein Mitarbeiter mit diversen Verantwortlichkeiten für mehrere Informationssysteme für den kein adäquater Ersatz im Unternehmen existiert. Dies würde zweifelsfrei zu Problemen führen, wenn dieser Mitarbeiter länger aus dem Unternehmen ausscheidet. Daher müssen seine Aufgaben auf andere Mitarbeiter umgelegt werden.

Ein weiterer Fall könnte das Aufdecken von Systemen sein, welche eine veraltete Technologie einsetzen. Diese sollten entsprechend zeitnah ersetzt werden. Gründe hierfür seien zum Beispiel die Sicherheit oder die Lauffähigkeit eines Systems. In diesem Szenario würden die weißen Knoten die Informationssysteme repräsentieren, welche diese Technologie verwenden. Durch diese Abfrageart wird schnell deutlich, welche Informationssysteme davon betroffen sind.

Ampelsystem:

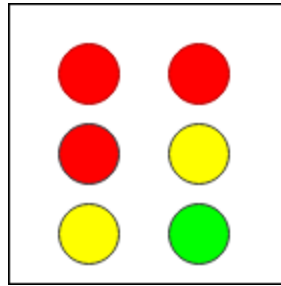


Abbildung 12: Ampelsystem

Das Mockup in Abbildung 12 stellt ein Ampelsystem dar. Es dient der Gliederung in Klassen. Es könnte zum Beispiel für die Darstellung von Risikopotenzialen zum Einsatz kommen. Ein passendes Beispiel sei durch das Risikopotenzial gegeben, welches durch das Supportende einer Technologie entsteht. Sollte eine Technologie keinen Support mehr erfahren, ist dies ein hohes Sicherheitsrisiko für das Unternehmen und die Technologie könnte entsprechend mit der Farbe Rot gekennzeichnet werden. Eine Technologie, welche ein Supportende innerhalb einer definierten Zeitspanne aufweist, würde folglich mit der Farbe Gelb gekennzeichnet werden. Technologien ohne ein bekanntes Risiko bezüglich eines Supportendes entsprechend mit der Farbe Grün.

Darstellung in Abhängigkeit zu quantitativen Attributen:

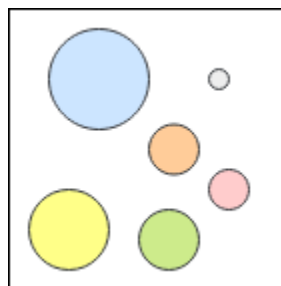


Abbildung 13: Darstellung in Abhängigkeit zu quantitativen Attributen

Das Mockup in der Abbildung 13 zeigt ein Visualisierungskonzept, um messbare Quantitäten zu verdeutlichen. Zum Beispiel könnte die Anzahl der Benutzer eines Informationssystems veranschaulicht werden. Je größer ein Knoten dargestellt wird, desto mehr Nutzer verwenden ein Informationssystem. Dies kann beispielsweise eine Tendenz, über die verursachten Lizenzkosten eines Informationssystems geben oder Aufschluss über den potenziellen Aufwand dieses zu ersetzen.

Dieses Visualisierungskonzept und das des Ampelsystems sind sehr gut geeignet, um sie mit anderen Graphvisualisierungen zu kombinieren, da sie die Hauptinformation nicht durch den Graphen darstellen, sondern jeweils durch eine weitere Dimension wie Größe und Farbe von Objekten.

3.3 Kombiniertes Visualisierungskonzept

Eine Realisierung eines Ampelsystems, welches aktuell in Unternehmen eingesetzt wird, sei durch den Tech Radar gegeben. Dieses ordnet die Knoten in einzelne definierte Phasen ein. Dies kann eine gute Entscheidungsgrundlage darstellen, wenn es beispielsweise darum geht, welche Systeme Risiken aufweisen oder welche Technologien für neue Projekte verwendet werden sollten. Unternehmen wie Zalando haben bereits eine eigene Visualisierung ihrer Daten anhand dieses Konzeptes im Web veröffentlicht. Die Befüllung der Grafik mit Daten erfolgt für das Tech Radar von Zalando manuell. Bei einer großen Datenmenge ist dieses Vorgehen jedoch nicht effizient. In dieser Arbeit wird dieses Vorgehen durch den Einsatz und die Anbindung an eine Graphdatenbank ersetzt. Durch dies lässt sich ein Tech Radar automatisiert befüllen.

Abbildung 14 zeigt diese Umsetzung des Unternehmens Zalando. Dieser Tech Radar besitzt insgesamt 4 Quadranten, welche unterschiedlichen Themengebieten zugeordnet sind. So repräsentieren alle Objekte des oberen linken Quadranten Frameworkbestandteile, welche im Unternehmen Zalando zum Einsatz kommen. Eine Aufteilung des Tech Radars in mehr oder weniger Quadranten ist möglich. Es sollte jedoch die Lesbarkeit der Darstellung berücksichtigt werden.

Analog zu einem klassischen Ampelsystem wird beim Tech Radar die Zugehörigkeit zu einer bestimmten Klasse mittels verschiedener Farben realisiert. Folgend seien die Klassen aus Abbildung 14 näher beschrieben und mit Beispielen aus dem Tech Radar von (Zalando, 2019) verdeutlicht.

Zalando Tech Radar — 2019.12

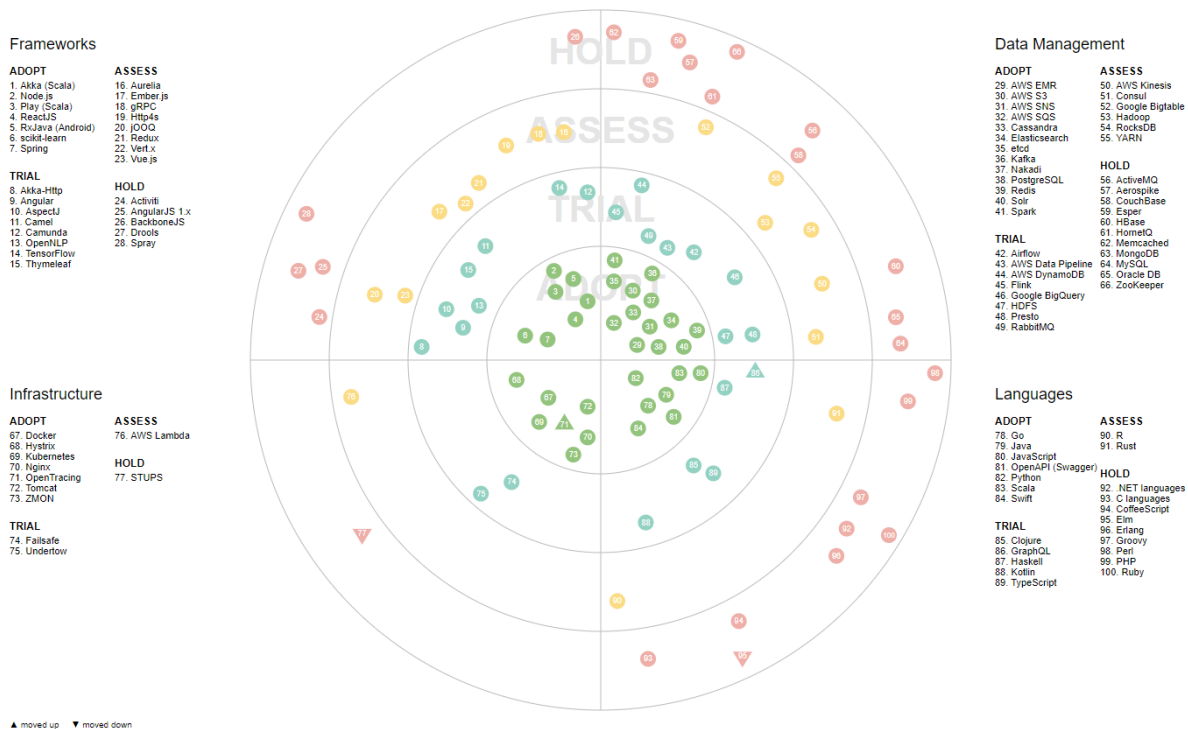


Abbildung 14: Tech Radar

■ Adopt = Übernehmen

In diesem Feld befinden sich Technologien, auf welche großes Vertrauen gesetzt wird, um langfristig mit diesen zu arbeiten. Ebenso sind diese weitestgehend risikofrei und sind recht verbreitet. Ein Auszug dieser am Beispiel von Zalando sind u.a.:

- Frameworks: Node.js, ReactJS, Spring
- Infrastructure: Docker, Nginx, Tomcat
- Data Management: Cassandra, Kafka, PostgreSQL, Spark
- Languages: Java, JavaScript, Python, Scala

■ Trial = Test

Technologien werden in diesem Bereich getestet, u. a. in Projekten. Die Technologie hat sich teilweise bereits in der Problemlösung bewährt. Testtechnologien sind riskanter, da sie unvorhersehbare Auswirkungen haben können. Ein Auszug dieser am Beispiel von Zalando sind u.a.:

- Frameworks: Angular, Camunda, Thymeleaf
- Infrastructure: Failsafe, Underlow
- Data Management: Airflow, AWS DynamoDB, Flink, Google BigQuery, Presto
- Languages: GraphQL, Haskell, Kotlin, TypeScript

▪ Assess = Bewertung

Technologien aus diesem Bereich besitzen ein hohes Potenzial aber auch ein hohes Risiko. Die Technologien dieser Gruppe sollten näher untersucht werden. Mit einer solchen Technologie hat das Unternehmen beispielsweise nur wenige Erfahrungen gesammelt, welche jedoch als positiv und vielversprechend zu werten sind. Ein Auszug dieser am Beispiel von Zalando sind u.a.:

- Frameworks: Aurelia, Ember.js, Redux, Vert.x, Vue.js
- Infrastructure: AWS Lambda
- Data Management: Consul, Google Bigtable, Hadoop, RocksDB, YARN
- Languages: R, Rust

▪ Hold = Halten

In diesem Bereich befinden sich Technologien, welche nicht für neue Projekte präferiert werden. Diese sind lediglich für aktuell bestehende Projekte notwendig. Daher sind diese Technologien keine weiteren Investitionen mehr wert. Ein Auszug dieser am Beispiel von Zalando sind u.a.:

- Frameworks: Activiti, AngularJS 1.x, BackboneJS, Drools, Spray
- Infrastructure: STUPS
- Data Management: ActiveMQ, CouchBase, Esper, MongoDB, MySQL, Oracle DB
- Languages: .NET languages, Elm, Groovy, Perl, PHP, Ruby

Die in Abbildung 15 und Abbildung 16 dargestellten Grafiken zeigen die Erweiterung des in Abbildung 14 abgebildeten Tech Radars von Zalando um die Komponente eines Graphen. Dieses Konzept wurde im Zuge dieser Arbeit entwickelt. Mit der Ergänzung des Graphen sind die Beziehungen unter den dargestellten Elementen gegeben. Das Sichtbarmachen der Beziehungen lässt in Bezug auf die eingesetzten Technologien diverse Rückschlüsse über die Gesamtsituation eines Unternehmens auf diesem Gebiet zu.

Unter der Annahme die Gesamtsituation des zugrunde liegenden Unternehmens ist positiv, so sollte der durch die Ergänzung der Beziehungen entstandene Graph, im Zentrum des Tech Radars sehr stark vernetzt sein. Entsprechend hierzu sollte die Vernetzungsdichte des Graphen nach außen hin deutlich abnehmen. Dieses Schema sei in Abbildung 15 visualisiert.

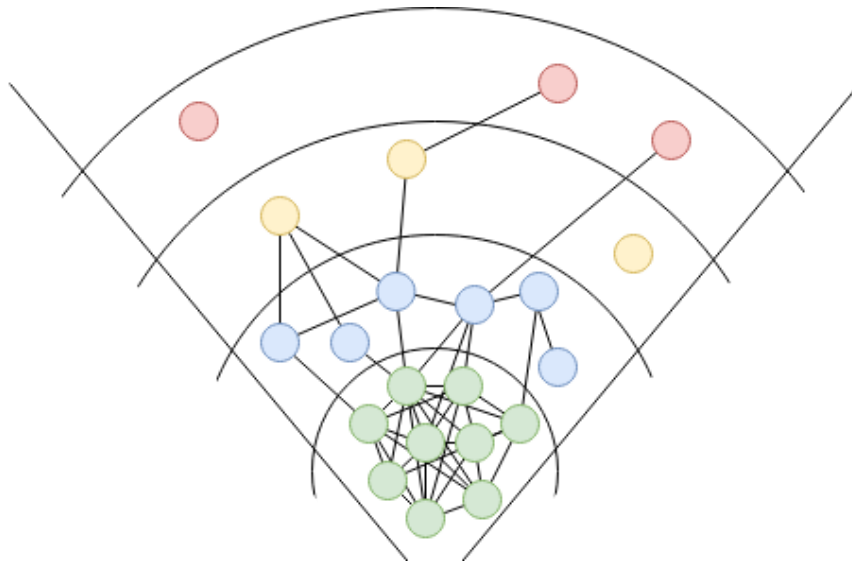


Abbildung 15: Erweitertes Tech Radar Beziehung

Existieren zu dem beschriebenen Schema Abweichungen, kann dies direkt mittels einer solchen Grafik deutlich gemacht werden. Eine Abweichung sei beispielsweise eine starke Vernetzung in einem Randbereich des Tech Radars. Befindet sich diese Abweichung innerhalb eines Quadranten würde dies darauf hindeuten, dass in dem Themengebiet des entsprechenden Quadranten verstärkt Technologien zu ersetzen sind.

Ein weiteres denkbare Muster, welches durch ein Tech Radar als Graph vorkommen kann, ist das Vorkommen einer höheren Dichte der Vernetzung innerhalb einer Klasse als zwischen unterschiedlichen Klassen. Ein einfaches Beispiel hierfür sei der Einsatz einer veralteten Technologie, welche wiederum den Einsatz weiterer veralteter Technologien voraussetzt. Beim Auftreten eines solchen Musters könnten Rückschlüsse über die Kompatibilität der Technologien zu neueren Technologien gezogen werden. Des Weiteren lassen sich hierdurch Technologien aufdecken, welche für das Auftreten eines solchen Musters verantwortlich sind. In Abbildung 16 sei dieses beschriebene Muster schematisch visualisiert.

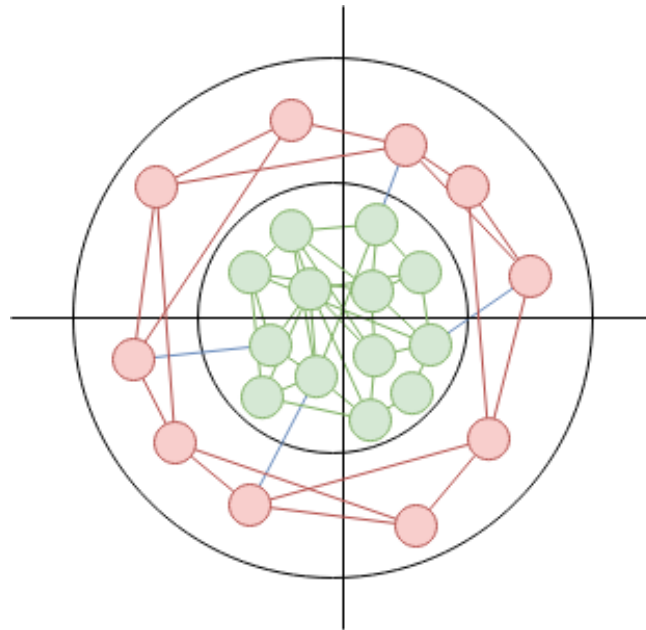


Abbildung 16: Erweitertes Tech Radar Dichte der Vernetzung

Die roten Knoten beischreiben hier die veralteten Technologien und die grünen Knoten die vom Unternehmen bevorzugten Technologien. Es wird ersichtlich, dass die veralteten Technologien überwiegend nur andere veraltete Technologien referenzieren. Entsprechend lässt sich hieraus ableiten, dass die meisten der veralteten Technologien obsolet werden, insofern ein Ersatz für diverse veraltete Technologien eingeführt wird. Weiterführend wird jedoch auch deutlich, dass bei einer Ersetzung einer veralteten Technologie gegebenenfalls weitere veraltete Technologien aufgrund ihrer Abhängigkeiten ersetzt werden müssen. Die Abbildung 16 ist hierbei eine schematische Darstellung, welche dieses Graphmuster visualisiert. Um diesen Sachverhalt zu verdeutlichen, wurde das Muster in der Abbildung etwas überspitzt dargestellt.

4 Implementierung

In diesem Kapitel wird gezeigt, was für das Vorgehen bei der Umsetzung der technischen Seite der Arbeit erforderlich ist. Im speziellen wird die Infrastruktur, als auch die Entwicklungsumgebung dargestellt. Zudem wird der grobe Ablauf des Programms aufgezeigt, auf diverse Schwierigkeiten eingegangen, sowie die Inbetriebnahme des Ergebnisses des praktischen Teils erläutert. Abschließend erfolgt eine kurze Auflistung der in dieser Arbeit eingesetzten Ressourcen und Technologien.

4.1 Infrastrukturaufbau

Die folgende Grafik zeigt den Aufbau der Infrastruktur für diese Arbeit. Als Server kann hier beispielsweise ein Windows Server verwendet werden. Innerhalb dieses Servers befinden sich die Datenbank und der Webserver. Als Datenbank wurde in dieser Arbeit Neo4j angedacht und als Webserver beispielsweise der Apache HTTP-Server. Der Webserver beinhaltet den entsprechenden HTML-Code mit den dazugehörigen Skripten.

Um auf diesen Inhalt zuzugreifen, muss zunächst der User an seinem Computer einen Browser öffnen. Dies kann wie hier abgebildet der Firefox sein. Anschließend wird über das Internet bzw. ein Intranet auf die Webseite zugegriffen.

Die einzelnen Komponenten sind in dieser Grafik nur beispielhaft aufgeführt. Diese können durch vergleichbare Komponenten ersetzt werden. Hier ein kleiner Auszug, welche gängigen Komponenten dies sein könnten:

- Server: Windows, Linux, Unix
- Datenbank: Neo4j, MySQL, PostgreSQL, CouchDB
- Webserver: Apache HTTP-Server, Tomcat, Nginx
- Internet / Intranet
- Browser: Firefox, Google Chrome, Internet Explorer, Apple Safari

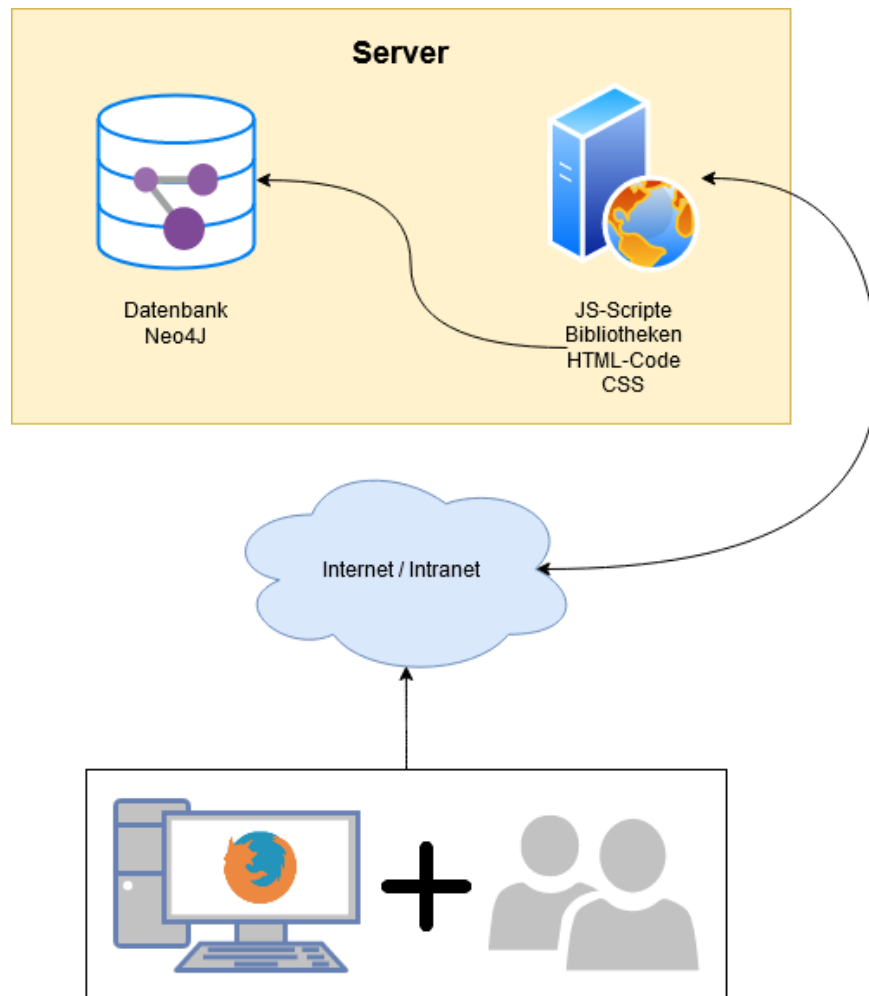


Abbildung 17: Infrastrukturaufbau

4.2 Entwicklungsumgebung

Bei der Erstellung für die genutzte Entwicklungsumgebung, war neben den technischen Anforderungen dieser Arbeit auch die Nachhaltigkeit, die einfache Verwaltung wie auch die einfache Portierung des Systems auf andere Geräte von Bedeutung. In der Abbildung 18 sind die einzelnen Komponenten der Entwicklungsumgebung und deren Verknüpfungen untereinander visualisiert.

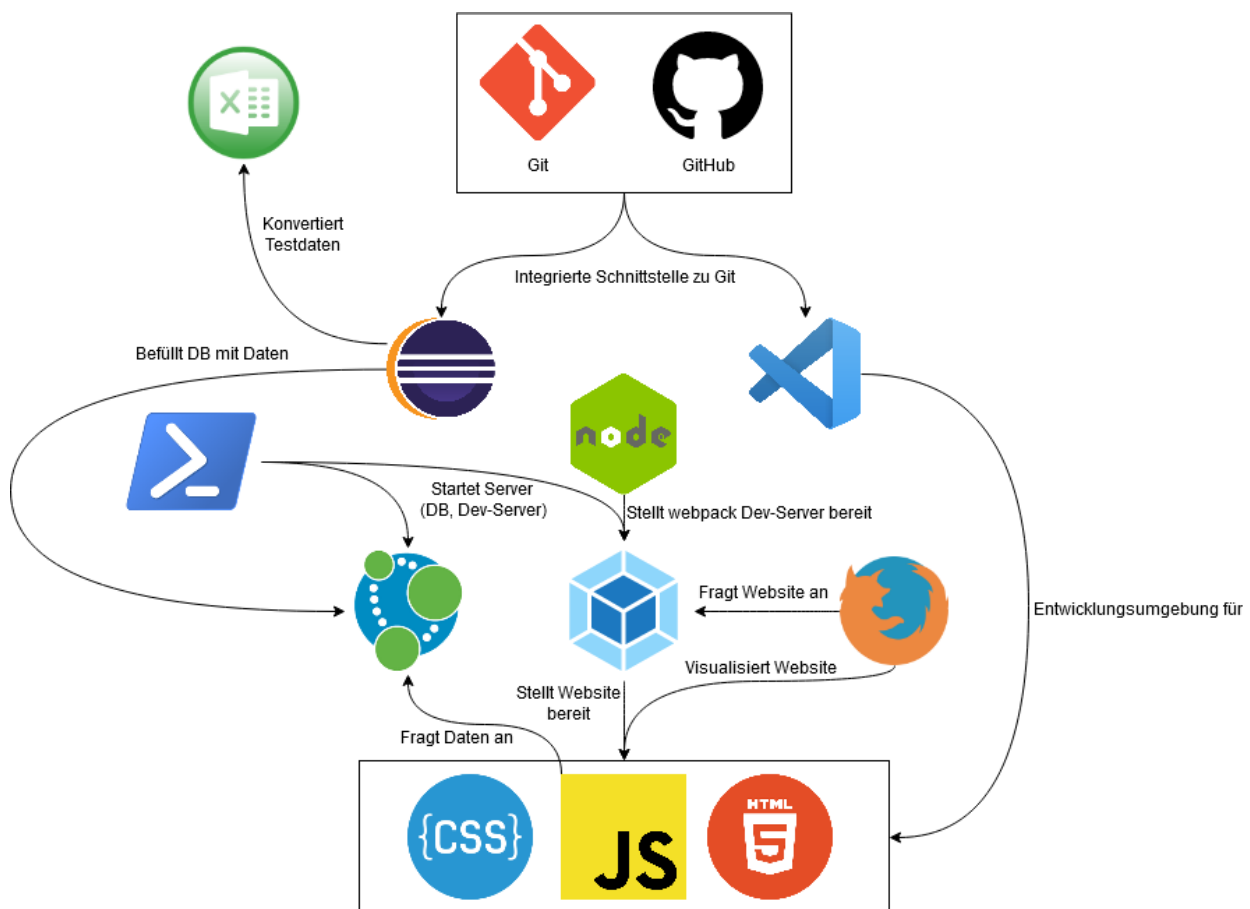


Abbildung 18: Entwicklungsumgebung

Als zentraler Ablageort für den Code wurde GitHub verwendet. GitHub basiert auf dem Versionsverwaltungssystem Git, welches sich auch in der Namensgebung von GitHub wiederfindet. Die Wahl fiel auf Git, da es diverse Schnittstellen zu populären integrierten Entwicklungsumgebungen (IDE) bietet und zudem den aktuellen Standard im Bereich der Versionsverwaltungssysteme definiert. Der Einsatz eines Repositories innerhalb eines Softwareprojekts bietet neben der Datensicherheit auch noch den Vorteil der einfachen Verteilung des Systems auf andere Geräte.

Als IDE wurde Visual Studio Code für die Entwicklung für CSS, JavaScript und HTML5 und für die Entwicklung für Java wurde Eclipse gewählt. Visual Studio Code zeichnet sich durch eine sehr schlanke IDE aus, welche modular durch zahlreiche Erweiterungen ergänzt werden kann. Erweiterungen sind für diverse Sprachen vorhanden. Eclipse ist eine sehr weitverbreitete IDE für Java und aufgrund ihrer langjährigen Historie sehr bewährt.

Der Code des Softwareprojekts wird mittels eines Webservers bereitgestellt und über einen Browser visualisiert. Der Webserver wird durch die Paketerweiterung webpack zur Verfügung gestellt, welche wiederum aus der Laufzeitumgebung NodeJS stammt. Die Installation von webpack erfolgt über den in NodeJS integrierten Node Package Manager. Dieser bietet auch die Option in einer zentralen Konfigurationsdatei (package.json) Abhängigkeiten des

Softwareprojekts zu anderen Paketerweiterungen zu definieren. Diese können dann wiederum mittels des Node Package Managers und der Konfigurationsdatei auf einem anderen System automatisiert installiert werden. Entsprechend trägt der Node Package Manager erheblich zur einfachen Verteilung des Systems auf anderen Geräten bei.

Die eigentliche Hauptaufgabe von webpack liegt jedoch nicht darin, einen Webserver für die Entwicklung bereitzustellen, sondern in dem Bereich des Deployments. Ist ein lauffähiger Zustand des Softwareprojekts erreicht, kann mittels webpack ein Build erzeugt werden. Hierbei extrahiert webpack den gesamten projektrelevanten Quellcode, Ressourcen und Stylesheets und führt dies in entsprechenden Dateien zusammen. Dies führt beispielsweise dazu, dass eine komplette Bibliotheksdatei von mehreren Megabyte in Abhängigkeit, des verwendeten Quellcodes auf mehrere Kilobyte reduziert werden kann. Der Build kann somit mit einer Art Standaloneimplementierung verglichen werden.

Als abschließende Komponente der Entwicklungsumgebung sei an dieser Stelle der Datenbankserver Neo4J erwähnt. Dieser ist für die Datenspeicherung und Datenbereitstellung verantwortlich. Er kommuniziert direkt mit der Webseite unter Zuhilfenahme spezieller Datenbanktreiber für JavaScript. Auf eine Middleware zwischen Datenbank und Webseite wurde bewusst verzichtet, da es das Gesamtsystem spürbar komplexer gemacht hätte, jedoch im aktuellen Zustand des Prototyps keinen erheblichen Mehrwert liefert. Das System kann jedoch ohne größeren Aufwand aufgrund des modularen Aufbaus zu einem späteren Zeitpunkt um eine Middleware nachgerüstet werden.

4.3 Beschreibung der implementierten Anwendung

Der grobe Ablauf des Programms sei durch die Abbildung 19 skizziert. Der Einstieg in die Anwendung erfolgt durch den Aufruf der implementierten Webseite. Hierdurch erfolgt der Start einer JavaScript-Routine mit der eigentlichen Programmlogik. Der Start der Routine ist mit dem Lademechanismus der Webseite codeseitig verknüpft. Sie wird somit immer bei einem Aufruf oder Reload der Webseite ausgeführt. Die Routine lässt sich grob in sechs Abschnitte untergliedern, wie es auch in der Abbildung 19 verdeutlicht wird. In dem Bereich der Deklaration werden Abhängigkeiten zu JavaScript-Bibliotheken oder zum verwendeten Stylesheet definiert, globale Programmvariablen angelegt, Verbindungsparameter der Datenbank deklariert und initialisiert, sowie diverse Statements in der Abfragesprache Cypher definiert.

Für die Initialisierung der Datenbankverbindung wird ein spezieller Treiber benötigt. Dieser wird durch die JavaScript-Bibliothek „neo4j-web“ zur Verfügung gestellt. In der Regel wird für jede Datenbank ein eigener spezieller Treiber benötigt. Eine Verwendung dieses Treibers mit einer MySQL Datenbank ist somit nicht möglich. Damit der Treiber eine Verbindung zu einer Datenbank aufbauen kann, benötigt er mindestens zwei Informationen. Zum einen wäre hier die URL, unter welcher die Datenbank zu erreichen ist und zum anderen die Authentifizierungsparameter für die Datenbank. Diese setzen sich zusammen aus dem Namen des Datenbanknutzers, mit welchem die Verbindung hergestellt werden soll, und dessen Passwort. Folgend sei die für die Initialisierung der Datenbankverbindung benötigte Codezeile dargestellt.

```
var driver = neo4j.driver("bolt://localhost", neo4j.auth.basic("neo4j", "myPWD"));
```

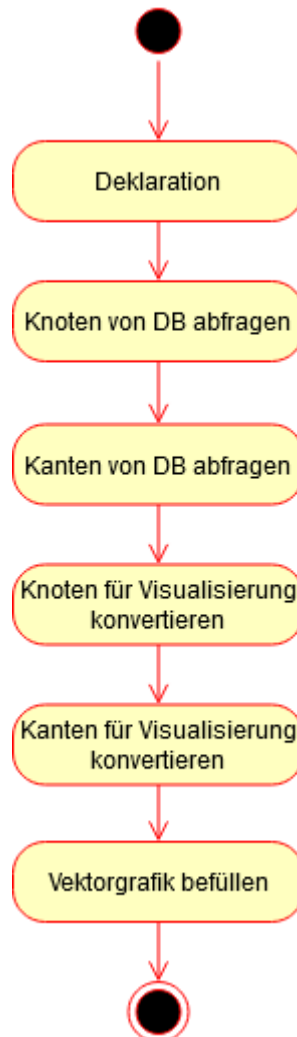


Abbildung 19: Ablauf des Programms

In den Abschnitten „Knoten von DB abfragen“ und „Kanten von DB abfragen“ werden mittels der initialisierten Datenbankverbindung Cypher-Abfragen an die Datenbank übermittelt. Nach der serverseitigen Verarbeitung der Anfrage durch die Datenbank wird ein entsprechender Rückgabewert an das Programm übergeben. Die Abfrage von Daten einer Neo4J Datenbank mittels JavaScript ist eine sogenannte „non-blocking“ Operation. Dies bedeutet, dass die Ausführung weiterer Operationen nicht verhindert wird, solange kein Rückgabewert zurückgeliefert wurde. Die Operation der Datenbankabfrage kann somit als asynchron verstanden werden. In dem Fall dieser Arbeit, würde das bedeuten, dass die Grafik zusammengebaut wird, ohne jedoch die Daten zu besitzen, da diese schlicht noch nicht von der Datenbank übermittelt wurden. Dies führt zu Fehlern während der Laufzeit. Um diesen Mechanismus zu unterbinden, musste in dieser Arbeit mit sogenannten „Promisses“ und mit „chaining“ gearbeitet werden, um die asynchronen

Prozesse in einen prozeduralen Ablauf zu bringen. Analog hierzu wäre auch ein zweites Vorgehen denkbar, welches die Prozesse parallel verarbeitet und an der benötigten Stelle des Programms zusammenführt. Dies wurde jedoch in dieser Arbeit nicht angewendet.

Promisses werden in sogenannten Callback-Methoden eingesetzt. Eine solche Methode führt nach Erhalt des Rückgabewertes eine definierte, nachfolgende Methode aus. In Abbildung 20 sei dieser Sachverhalt visualisiert.

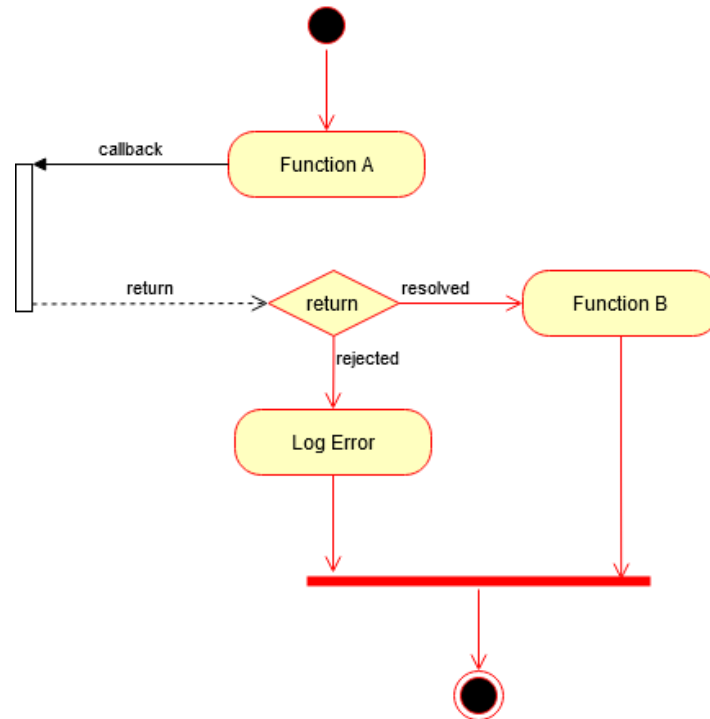


Abbildung 20: Call-Back Methode

Das chaining wird durch den Sachverhalt relevant, dass in der implementierten Anwendung in gewissen Szenarien mehrere Abfragen in Folge an die Datenbank übermittelt werden müssen. In diesen Szenarien setzt sich das zu visualisierende Ergebnis aus mehreren Abfrageergebnissen zusammen. Die technische Umsetzung gestaltet sich wie folgt. Es sei ein Array mit einer definierten Anzahl an Cypher-Abfragen gegeben. Dieses wird an eine Funktion übergeben, welche alle Abfragen abarbeitet und nach Erhalt aller Rückgabewerte diese an die aufrufende Methode übergibt. Das chaining erfolgt beim Übermitteln der einzelnen Abfragen an die Datenbank. Dies bedeutet letztlich, dass der Prozess aus Abbildung 20 in Abhängigkeit der Anzahl der Abfragen n-mal prozedural ausgeführt wird. Das chaining verhindert damit, dass parallele Abfragen an die Datenbank übermittelt werden und resultierend daraus den Fehlerfall einer Prozessierung der Anwendung ohne den Erhalt aller angefragten Informationen verhindert. Folgend sei dieser Sachverhalt als Codebeispiel dargestellt und in der Abbildung 21 visualisiert.

```

function parseStatementArray(stm) {
  return new Promise(resolve => {
    var test = [];
    stm.reduce((chain, currentStatement) => {
      var t = chain.then(() => parseCypherToDB(currentStatement));
      test.push(t);
      return t;
    }, Promise.resolve())
    resolve(Promise.all(test))
  })
}

```

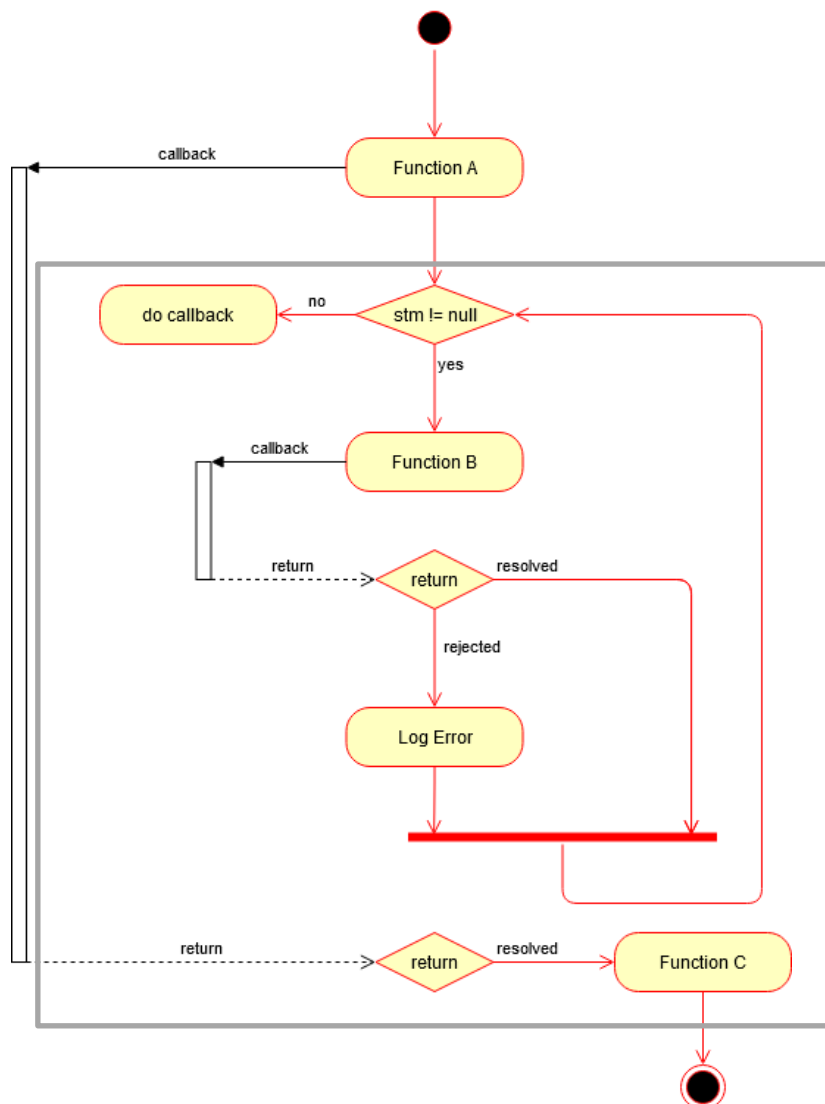


Abbildung 21: Chaining

In den Abschnitten „Knoten für Visualisierung konvertieren“ und „Kanten für Visualisierung konvertieren“ werden die Ergebnisse der Datenbankabfrage entsprechend für die Visualisierung aufbereitet. Alle Knoten und alle Kanten werden in separierten Arrays verwaltet. Der Inhalt des entsprechenden Abfrageergebnisses für einen Knoten wird in ein sogenanntes assoziatives Array konvertiert und die Ergebnisse einer Kante in ein Array, bestehend aus Quelle, Ziel und Typ der Beziehung. Quelle und Ziel bestehen hierbei aus Integerwerten, welche jeweils den Index des entsprechenden Knoten im Array der Knoten repräsentiert. Die Beziehungen zwischen den Arrays der Knoten und Kanten sei in Abbildung 22 nochmals verdeutlicht.

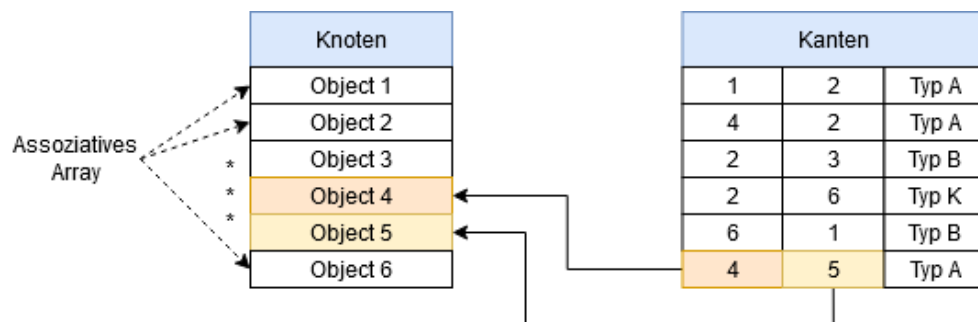


Abbildung 22: Zusammenhang Knoten und Kanten

Ein assoziatives Array unterscheidet sich zu einem normalen Array in dem Punkt, dass es Key-Value Paare speichert. Der Key übernimmt somit die gleiche Funktion wie der Index. Es muss jedoch gewährleistet sein, dass der Key innerhalb des Arrays eindeutig ist. Durch folgendes Codebeispiel sei ein solches assoziatives Array exemplarisch dargestellt.

```
i = {
    Code: record.properties.Code,
    Name: record.properties.Name,
    Art: record.properties.Art,
    type: "Standard"
};
```

Der letzte Abschnitt der Anwendung, „Vektorgrafik befüllen“, befasst sich mit dem Styling der darzustellenden Objekte, den Eigenschaften der Objekte und dem Hinzufügen der Objekte zu der auf der Webseite visualisierten Vektorgrafik. Das Styling wurde aus JavaScript ausgegliedert und erfolgt über ein Stylesheet im Format CSS. Mit den Eigenschaften der Objekte sind unter anderem funktionale Eigenschaften wie ein Drag-Event oder ein Hover-Event gemeint. Beispielsweise ruft der Hover-Event ein Tooltip-Popup auf, welches die Informationen des Knotens anzeigt. Des Weiteren werden in diesem Abschnitt noch Eigenschaften des Forcelayouts von D3 definiert. Das Forcelayout wird auf den Graphen angewendet und beschreibt zum Beispiel welchen Abstand die Knoten zueinander besitzen sollen, oder ob diese sich überlagern dürfen.

4.4 Verwendete Technologien

Um die in der Arbeit verwendeten Software- und Entwicklungskomponenten in ihrer Gänze darzustellen, seien diese nachfolgend aufgelistet und mit den entsprechend verwendeten Versionsnummern versehen:

Software:

- Betriebssystem: Windows 10 Enterprise
- Google Chrome Version 79.0.3945.130 (64-Bit)
- Notepad++ Version 6.5.4
- GitHub Desktop Version 2.2.4
- Microsoft Visual Studio Code (User) Version 1.41.1
- Eclipse Java 2019-09
- XAMPP Version 7.3.9-0
- Apache HTTP-Server 2.4.41
- Neo4j Server Version 3.5.9
- Node.js Version 10.16.3
- PowerShell
- Java TM SE Development Kit 13.0.1.0 (64-Bit)

Bibliotheken:

- D3.js Version 5
- jquery 1.11.0
- neo4j-web 1.0.4
- lodash 4.15.0

NodeJS Paketerweiterungen:

- @babel/core 7.8.4
- babel-loader 8.0.6
- babel-preset-env 1.7.0
- babel-preset-stage-0 6.3.13
- css-loader 3.4.2
- eslint 6.8.0
- file-loader 5.0.2
- html-webpack-plugin 3.2.0
- rimraf 3.0.2
- style-loader 1.1.3
- webpack 4.41.5
- webpack-cli 3.3.10
- webpack-dev-server 3.10.3

Aufgrund des Umfangs und der Relevanz wird an dieser Stelle nicht näher auf die einzelnen Komponenten eingegangen. Es sei diesbezüglich auf die Onlinedokumentation der jeweiligen Produkte verwiesen.

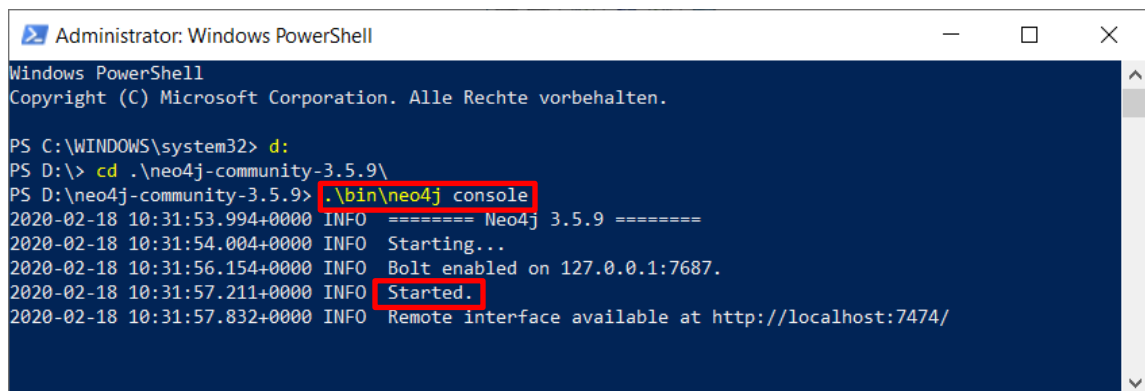
4.5 Inbetriebnahme der erstellten Anwendung

Dieses Kapitel thematisiert die Inbetriebnahme der erstellten Anwendung. Hierbei wird unterschieden zwischen der Inbetriebnahme des Entwicklungssystems und der Inbetriebnahme des erzeugten finalen Build. Für beide Varianten wird jedoch vorausgesetzt, dass eine lauffähige Installation eines Neo4J-Datenbankservers vorhanden ist, welcher auch die entsprechenden Daten bereitstellt. Für die Einspielung der Testdaten dieser Arbeit kann ein Java-Programm verwendet werden, welches im Zuge dieser Arbeit erzeugt wurde und der Arbeit beiliegt. Das Programm konvertiert in einem ersten Schritt die von der ISB bereitgestellten Daten in Cypher-Statements für den Import. Im zweiten Schritt werden die generierten Import-Statements an die Datenbank gesendet. Hierzu wird die Java-Bibliothek JDBC verwendet.

Ein Starten des Datenbankservers kann mittels Windows PowerShell erfolgen. Hierzu muss der Anwender in das Installationsverzeichnis des Datenbankservers wechseln. Im Anschluss kann die Datenbank mittels folgenden Befehles gestartet werden.

```
.\bin\neo4j console
```

Abbildung 23 zeigt diesen Vorgang im Detail. Neben dem Status der Datenbank, welcher in Abbildung 23 durch das Wort „Started“ repräsentiert wird, erhält der Anwender zudem eine Information, unter welcher URL sowie Port das Webinterface des Datenbankservers zu erreichen ist.



```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. Alle Rechte vorbehalten.

PS C:\WINDOWS\system32> d:
PS D:\> cd .\neo4j-community-3.5.9\
PS D:\neo4j-community-3.5.9> .\bin\neo4j console
2020-02-18 10:31:53.994+0000 INFO ===== Neo4j 3.5.9 =====
2020-02-18 10:31:54.004+0000 INFO Starting...
2020-02-18 10:31:56.154+0000 INFO Bolt enabled on 127.0.0.1:7687.
2020-02-18 10:31:57.211+0000 INFO Started.
2020-02-18 10:31:57.832+0000 INFO Remote interface available at http://localhost:7474/
```

Abbildung 23: Start Datenbankserver durch PowerShell

Das Webinterface des Datenbankservers ist in Abbildung 24 dargestellt. Es bietet neben administrativen Funktionen auch die Möglichkeit, Abfragen an die Datenbank zu senden. Diese werden nach der entsprechenden Rückgabe der Datenbank innerhalb des Webinterface visualisiert, wie es in der Abbildung 24 dargestellt ist.

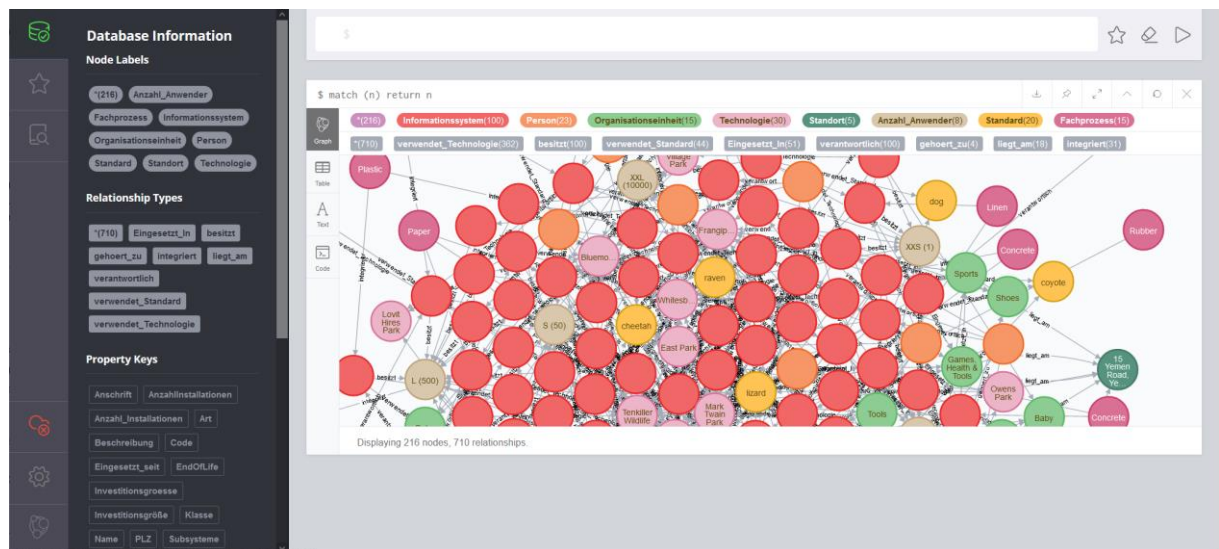


Abbildung 24: Webinterface Datenbankserver

4.5.1 Inbetriebnahme der Entwicklungsumgebung

Für die Inbetriebnahme der Entwicklungsumgebung müssen neben einer Installation des Datenbankservers noch weitere Voraussetzungen erfüllt sein. Zwingend notwendig sind hierfür Teile der Projektstruktur, welche im digitalen Anhang dieser Arbeit im Ordner „Implementierung“ zu finden sind. Alle relevanten Bestandteile sind folgend aufgelistet.

- src (Ordner)
- package.json
- webpack.config.js

Wie schon in Kapitel 2.5 erwähnt wird in dieser Arbeit NodeJS als Teil der Deploymentumgebung genutzt. Diesbezüglich setzt die Entwicklungsumgebung eine möglichst aktuelle, lauffähige Installation von NodeJS und die im Projekt definierten Paketerweiterungen voraus. Die Paketerweiterungen sind in der Datei „package.json“ definiert. Mittels npm und der Datei „package.json“ lassen sich die Paketerweiterungen gebündelt und automatisiert installieren. Hierzu muss der Anwender lediglich mit zum Beispiel Windows PowerShell in das Verzeichnis der Datei „package.json“ wechseln und im Anschluss den Befehl „npm install“ ausführen. Dieses Vorgehen vereinfacht das Ausrollen der „Entwicklungsanwendung“ deutlich, da die gesamten Abhängigkeiten an einem zentralen Ort definiert sind und auch nicht zwingend bekannt sein müssen.

Nachdem alle Voraussetzungen erfüllt sind, kann der Webserver der Entwicklungsumgebung gestartet werden. Dieser wird von der Paketerweiterung „webpack“ zur Verfügung gestellt. Analog zu der Datenbank kann dieser mittels

Windows PowerShell gestartet werden. Dies erfolgt über das Wechseln in das Verzeichnis der Datei „webpack.config.js“ und anschließendem Ausführen folgenden Befehles.

```
npm run dev
```

Ist der Webserver gestartet und der Quellcode konnte erfolgreich kompiliert werden, wird dies im Fenster von Windows PowerShell durch den Eintrag „Compiled successfully“ deutlich gemacht. Der beschriebene Startvorgang des Webserver ist in Abbildung 25 veranschaulicht.

```

PS C:\WINDOWS\system32> cd D:\Thesis\Implementierung\
PS D:\Thesis\Implementierung> npm run dev

> Thesis_Katharina_Schemel@0.0.1 dev D:\Thesis\Implementierung
> webpack-dev-server

  @wdm: Project is running at http://localhost:8080/
  @wdm: webpack output is served from /
  @wdm: Content not from webpack is served from D:\Thesis\Implementierung
  @wdm: Hash: 863ae29b5b43d1a331ab
Version: webpack 4.41.5
Time: 1144ms
Built at: 2020-02-18 12:56:32 PM
    Asset      Size  Chunks             Chunk Names
acd9f492e5ecf92f45f6e32fa138e0bc.png  21.5 KiB          0  [emitted]
          app.js    908 KiB          0  [emitted]
          index.html  523 bytes          0  [emitted]
        neo4j-web.min.js  190 KiB          0  [emitted]
Entrypoint app = app.js
[0] multi (webpack)-dev-server/client?http://localhost:8080 ./src/app.js 40 bytes {app} [built]
    multi entry
  [./node_modules/ansi-html/index.js] 4.16 KiB {app} [built]
    cjs require ansi-html [./node_modules/webpack-dev-server/client/overlay.js] (webpack)-dev-server/client/overlay.js 4:15-35
  [./node_modules/file-loader/dist/cjs.js?name=[name].[ext]!./node_modules/neo4j-driver/lib/browser/neo4j-web.min.js] 60 bytes {app} [built]
    cjs require file-loader?name=[name].[ext]!./node_modules/neo4j-driver/lib/browser/neo4j-web.min.js [./src/app.js] 3:0-98
  [./node_modules/lodash/lodash.js] 528 KiB {app} [built]
    cjs require lodash [./src/app.js] 5:8-25
  [./node_modules/strip-ansi/index.js] 161 bytes {app} [built]
    cjs require strip-ansi [./node_modules/webpack-dev-server/client/index.js?http://localhost:8080] (webpack)-dev-server/client?http://localhost:8080 6:16-37
  [./node_modules/webpack-dev-server/client/index.js?http://localhost:8080] (webpack)-dev-server/client?http://localhost:8080 4:29 KiB {app} [built]
    single entry D:\Thesis\Implementierung\node_modules\webpack-dev-server\client\index.js?http://localhost:8080 [0] multi (webpack)-dev-server/client?http://localhost:8080 ./src/app.js
app[0]
  [./node_modules/webpack-dev-server/client/overlay.js] (webpack)-dev-server/client/overlay.js 3:51 KiB {app} [built]
    cjs require ./overlay [./node_modules/webpack-dev-server/client/index.js?http://localhost:8080] (webpack)-dev-server/client?http://localhost:8080 10:14-34
  [./node_modules/webpack-dev-server/client/socket.js] (webpack)-dev-server/client/socket.js 1:53 KiB {app} [built]
    cjs require ./socket [./node_modules/webpack-dev-server/client/index.js?http://localhost:8080] (webpack)-dev-server/client?http://localhost:8080 8:13-32
  [./node_modules/webpack-dev-server/client/utils/createSocketUrl.js] (webpack)-dev-server/client/utils/createSocketUrl.js 2:91 KiB {app} [built]
    cjs require ./utils/createSocketUrl [./node_modules/webpack-dev-server/client/index.js?http://localhost:8080] (webpack)-dev-server/client?http://localhost:8080 20:22-56
  [./node_modules/webpack-dev-server/client/utils/log.js] (webpack)-dev-server/client/utils/log.js 964 bytes {app} [built]
    cjs require ./utils/log [./node_modules/webpack-dev-server/client/index.js?http://localhost:8080] (webpack)-dev-server/client?http://localhost:8080 12:15-37
    cjs require ./log [./node_modules/webpack-dev-server/client/utils/reloadApp.js] (webpack)-dev-server/client/utils/reloadApp.js 4:15-31
  [./node_modules/webpack-dev-server/client/utils/reloadApp.js] (webpack)-dev-server/client/utils/reloadApp.js 1:59 KiB {app} [built]
    cjs require ./utils/reloadApp [./node_modules/webpack-dev-server/client/index.js?http://localhost:8080] (webpack)-dev-server/client?http://localhost:8080 18:16-44
  [./node_modules/webpack-dev-server/client/utils/sendMessage.js] (webpack)-dev-server/client/utils/sendMessage.js 402 bytes {app} [built]
    cjs require ./utils/sendMessage [./node_modules/webpack-dev-server/client/index.js?http://localhost:8080] (webpack)-dev-server/client?http://localhost:8080 16:18-48
  [./node_modules/webpack/hot sync ^\\.\\/log$] (webpack)-dev-server/client/index.js?http://localhost:8080 [built]
    require.context webpack/hot [./node_modules/webpack-dev-server/client/index.js?http://localhost:8080] (webpack)-dev-server/client?http://localhost:8080 76:17-67
  [./src/app.js] 16.7 KiB {app} [built]
    single entry ./src/app.js [0] multi (webpack)-dev-server/client?http://localhost:8080 ./src/app.js app[1]
    single entry ./src/app.js app
  [./src/assets/stylesheets.css] 572 bytes {app} [built]
    cjs require ./assets/stylesheets.css [./src/app.js] 1:0-34
    + 26 hidden modules
Child html-webpack-plugin for "index.html":
    Asset      Size  Chunks             Chunk Names
    index.html  535 KiB          0
Entrypoint undefined = index.html
  [./node_modules/html-webpack-plugin/lib/loader.js!./src/assets/index.html] 703 bytes {0} [built]
    single entry D:\Thesis\Implementierung\node_modules\html-webpack-plugin\lib\loader.js!D:\Thesis\Implementierung\src\assets\index.html
  [./node_modules/lodash/lodash.js] 528 KiB {0} [built]
    cjs require !./node_modules/lodash/lodash.js [./node_modules/html-webpack-plugin/lib/loader.js!./src/assets/index.html] 1:8-56
  [./node_modules/webpack/buildin/global.js] (webpack)-dev-server/buildin/global.js 472 bytes {0} [built]
    cjs require global [./node_modules/lodash/lodash.js] 1:0-41
  [./node_modules/webpack/buildin/module.js] (webpack)-dev-server/buildin/module.js 497 bytes {0} [built]
    cjs require module [./node_modules/lodash/lodash.js] 1:0-41
  @wdm: Compiled successfully.
  
```

Abbildung 25: Start Webservers durch PowerShell

Nach dem erfolgreichen Kompilieren kann der Prototyp über den Webserver mittels der URL „localhost:8080“ in einem Webbrowser aufgerufen werden. Für die Visualisierung des Prototyps sei auf Kapitel 4.6 Ergebnisse verwiesen.

4.5.2 Inbetriebnahme des finalen Build

Mittels der Entwicklungsumgebung kann ein Build der Anwendung erstellt werden. Dies bedeutet, dass die Paketerweiterung „webpack“ eine Routine aufruft, welche alle relevanten Quellcodebestandteile und Ressourcen zusammenführt. Dies beinhaltet neben dem eigenen Quellcode auch den von Fremdbibliotheken. Hierdurch wird die Anwendung portabel und der benötigte Speicherplatz kann signifikant reduziert werden.

Ein erstellter Build der Anwendung ist im Ordner „Implementierung/build“ des digitalen Anhangs der Arbeit zu finden. Diese Dateien beinhalten alle relevanten Bestandteile mit Ausnahme der Datenbank, um die Anwendung auszuführen. Entsprechend können diese Dateien über einen Webserver bereitgestellt werden. Am Beispiel eines Apache Webserver würde dies bedeuten, dass sie im Verzeichnis „htdocs“ des Apache Webserver abgelegt werden. Nach dem Starten des Webserver ist die Anwendung über einen Webserver vollumfänglich erreichbar. Der größte Vorteil dieses Vorgehens kann darin gesehen werden, dass keine Abhängigkeiten wie zum Beispiel zusätzliche Software serverseitig installiert werden muss.

4.6 Ergebnisse

Folgend ist der Prototyp mit der Visualisierung des gesamten Graphen in Abbildung 26 exemplarisch visualisiert.

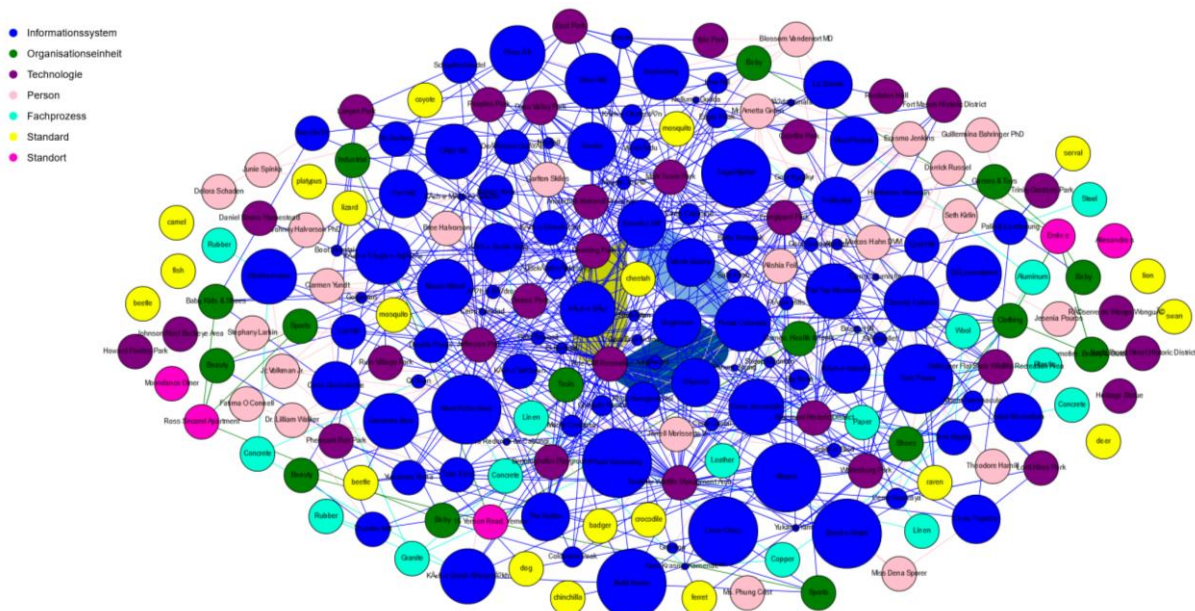


Abbildung 26: Visualisierung des gesamten Graphen

Angezeigt werden die Testdaten einer fiktiven IT-Landschaft der ISB AG. Es wird deutlich, dass die Abbildung des gesamten Graphen sehr schnell unübersichtlich wirkt, wie es schon in Kapitel 3.2 erwähnt wurde. Speziell die vielen Beziehungen der Knoten untereinander können nicht mehr überblickt werden. Eine manuelle Anordnung der Knoten kann in diesem Fall die Übersichtlichkeit noch bis zu einem gewissen Grad verbessern. Eine Unterteilung in Subgraphen erscheint jedoch sinnvoller, wie es in Abbildung 27 und Abbildung 28 zu sehen ist.

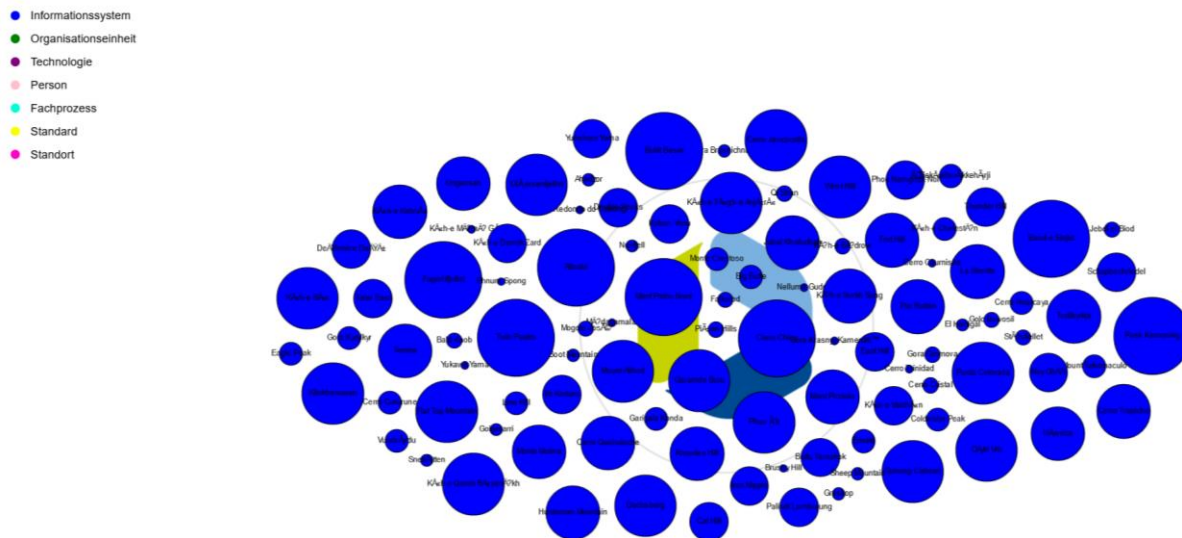


Abbildung 27: Objekte in Abhängigkeit zu quantitativen Attributen

Abbildung 27 zeigt Informationssysteme, welche in Abhängigkeit der Anzahl von Nutzern dargestellt werden. Je mehr Nutzer ein Informationssystem besitzt, desto größer wird der entsprechende Knoten dargestellt. Für dieses Beispiel ist zu erwähnen, dass hier mit Klassen für die Anzahl von Nutzern gearbeitet wurde. Dies bedeutet, dass Informationssysteme auf Basis der Nutzeranzahl einer Klasse zugeordnet werden und der Knoten erhält in Abhängigkeit der Klasse einen spezifischen Radius. Wie in Kapitel 3.2 erwähnt kann dieses Visualisierungskonzept sehr gut mit anderen Visualisierungskonzepten kombiniert werden. Ein Beispiel einer solchen Realisierung ist in Abbildung 28 dargestellt.

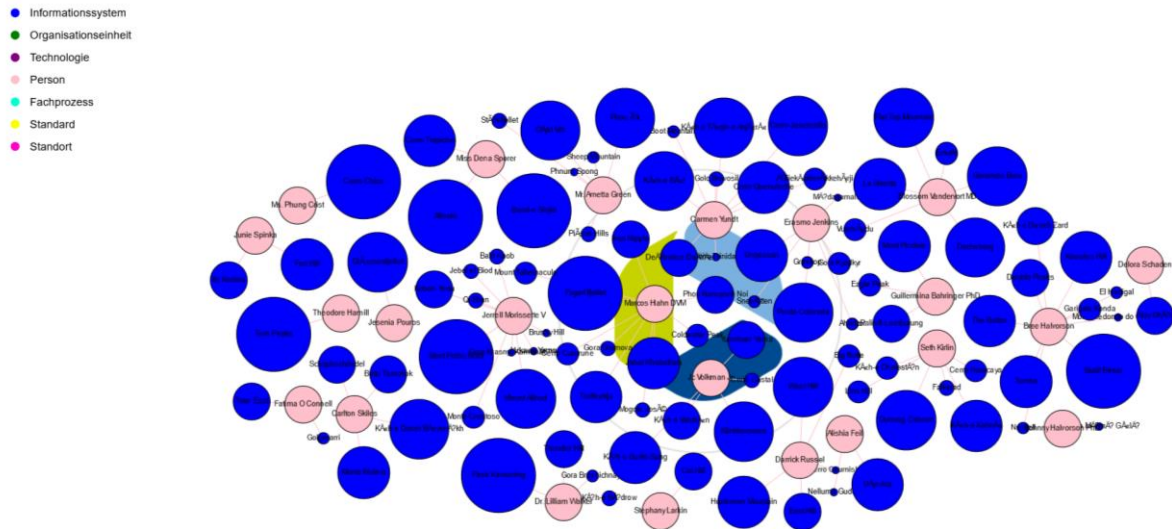


Abbildung 28: Erweiterung durch Relation

Abbildung 28 erweitert die Darstellung der Informationssysteme aus Abbildung 27 um die Darstellung der für das Informationssystem verantwortlichen Personen. Anhand dieses Beispiels können nicht vorhandene Redundanzen bezüglich der Verantwortlichkeit aufgedeckt werden. Des Weiteren kann an diesem Beispiel veranschaulicht werden, wie gut die Verantwortlichkeiten aufgeteilt sind. Diese und weitere Schlussfolgerungen können direkt auf Basis der Grafik des visualisierten Subgraphen erfolgen. Es sei somit exemplarisch mit diesem Beispiel aufgezeigt, welchen Mehrwert eine Darstellung eines Subgraphen gegenüber der Darstellung des gesamten Graphen besitzt.

5 Evaluation

Die in dieser Arbeit erarbeiteten Konzepte erwiesen sich weitestgehend als belastbar. Während der Implementierung zeigte sich des Öfteren, dass das umgesetzte Konzept für die Entwicklungsumgebung einen deutlichen Mehrwert erbracht hat. Die Portierung des gesamten Softwareprojekts auf unterschiedliche Rechner konnte mehrfach während der Erstellung dieser Thesis problemlos vollzogen werden. Die zentralen Elemente für den reibungslosen Prozess, waren hierbei der Einsatz des Versionsverwaltungssystems Git und der Einsatz des Node Package Manager. Das Versionsverwaltungssystem wurde in diesem Zusammenhang für das Verteilen des Softwareprojekts genutzt und der Node Package Manager für die automatisierte Installation aller im Softwareprojekt abhängigen Fremdbibliotheken.

Die erarbeiteten Konzepte bezüglich der Visualisierung konnten nicht vollumfänglich validiert werden. Eine Validierung erfolgte lediglich innerhalb einer kleinen Befragung auf Basis von erzeugten Mockups, welche jedoch nicht als repräsentativ angesehen werden kann. Im Zuge weiterer Arbeiten sollte dieser Punkt erneut aufgegriffen werden und eine detailliertere Untersuchung der Konzepte sollte stattfinden. Das Ergebnis der Befragung sei dennoch an dieser Stelle erwähnt. Es zeigte sich die Tendenz, dass die Basiskonzepte aus Kapitel 3.2 sehr gut zu interpretieren waren. Im speziellen seien hier die Darstellung des Grades der Vernetzung, des Ampelsystems und die Darstellung der Objekte in Abhängigkeit zu quantitativen Attributen genannt. Das Konzept des um einen Graphen erweiterten Tech Radar erwies sich als deutlich komplexer zu interpretieren. Eine korrekte Interpretation der visualisierten Muster des Graphen, wie sie in Kapitel 3.3 beschrieben sind, konnte ohne zusätzliche Kenntnisse im Rahmen der Befragung nicht erfolgen. Nach einer Erläuterung wurde der Hintergrund der Muster verstanden und weitere exemplarische Beispiele konnten folgend korrekt interpretiert werden. Entsprechend deutet dies daraufhin, dass die Visualisierung des erweiterten Tech Radar nur mit Spezialwissen zu interpretieren ist.

Die Verwendung von Webtechnologien zur Visualisierung erweist sich als äußerst positiv. Diese Erkenntnis resultiert überwiegend aus der Qualität der generierten Grafiken und dem hohen Potential bezüglich der Wiederverwendbarkeit des erstellten Quellcodes für andere Projekte. Dennoch sei erwähnt, dass sich die Hürde für den Einstieg ohne größere Vorkenntnisse als recht hoch erwies. Was sich letztlich in dem zeitlichen Aufwand für den praktischen Teil dieser Arbeit niedergeschlagen hat.

6 Zusammenfassung und Ausblick

In diesem Kapitel wird die Abschlussarbeit kurz zusammengefasst und legt die wichtigsten Erkenntnisse der Arbeit dar. Der abschließende Ausblick zeigt auf, welches Potenzial noch in der Arbeit steckt und wie diese Arbeit in Zukunft noch weiter ausgebaut werden kann.

6.1 Zusammenfassung

Diese Arbeit befasst sich mit der Thematik der Visualisierung von IT-Konsolidierungsprojekten unter Zuhilfenahme von Graphdatenbanken. In diesem Kontext dient die Visualisierung der Sichtbarmachung von Beziehungen und Abhängigkeiten zwischen Komponenten einer IT-Landschaft eines Unternehmens. Sie kann somit direkt als Werkzeug in dem Prozess der Analyse einer IT-Landschaft eingesetzt werden. Weiterhin kann sie als Entscheidungsgrundlage für Maßnahmen herangezogen werden, welche aus dem Konsolidierungsverfahren entstanden sind.

Zu Beginn dieser Arbeit werden die notwendigen Grundlagen diskutiert. Um die Thematik der IT-Konsolidierung einordnen zu können, wurde als erstes das Enterprise Architecture Management (EAM) näher beschrieben. Folgend darauf geht die Arbeit auf gängige Visualisierungskonzepte mit typischen EAM Visualisierungen ein. In Bezug auf Graphdatenbanken ist die Funktionsweise dieser sowie die Abfragesprache Cypher erläutert. Neben den genannten Grundlagen werden die Themen Webserver, NodeJS und Webtechnologien für die Visualisierung angesprochen. Diese bilden die Grundbausteine für die Konzeption der Entwicklungsumgebung und des Prototyps, welcher im Zuge dieser Arbeit erstellt wurde.

Ein essenzieller Bestandteil dieser Arbeit ist die Erarbeitung von graphbasierten Visualisierungskonzepten. Diese erweitern das Portfolio der gängigen EAM Visualisierungskonzepte. Innerhalb dieser Arbeit wurden diverse solcher graphbasierten Visualisierungskonzepte erarbeitet. Unter anderem fokussieren sich diese Konzepte auf die Darstellung von Beziehungen und Abhängigkeiten, die Darstellung von Objekten in Abhängigkeit zu quantitativen Werten und die Darstellung von attributiven Klassifizierungen. Es wird anhand von Beispielen gezeigt, in welchen Anwendungsfällen diese graphbasierten Visualisierungskonzepte zum Einsatz kommen können und welchen Mehrwert diese

potenziell bieten. Im weiteren Verlauf der Arbeit wird ein sogenanntes Tech Radar konzeptionell um die Komponente eines Graphen erweitert. Thematisiert wird in diesem Kontext im speziellen, welche Muster des Graphen innerhalb der Darstellung des erweiterten Tech Radars auftreten können und welche Schlussfolgerungen bezogen auf die Muster erhoben werden können.

Neben der Erarbeitung von Visualisierungskonzepten ist ein weiterer wichtiger Bestandteil dieser Arbeit, die Implementierung eines Prototyps, in welchem von den erarbeiteten Konzepten ausgewählte Konzepte realisiert sind. Für die Implementierung wurde im Rahmen der Arbeit eine Entwicklungsumgebung konzipiert und umgesetzt. Der Fokus bei der Planung der Entwicklungsumgebung wurde auf die Nachhaltigkeit und die einfache Verwaltung dieser gelegt, sowie auf eine einfache Portierung des Systems auf andere Geräte. Folgend geht die Arbeit im Detail auf die Implementierung ein. In diesem Zusammenhang wird der grobe Ablauf der Anwendung beschrieben und anhand von Ablaufdiagrammen visualisiert. Des Weiteren wird im speziellen auf Promisses und auf chaining im Kontext von JavaScript eingegangen.

Abschließend werden die Ergebnisse der Arbeit innerhalb der Evaluation reflektiert und bewertet. Es zeigte sich, dass die erarbeiteten Konzepte überwiegend belastbar waren. Lediglich das erweiterte Tech Radar konnte ohne vertiefende Kenntnisse nicht in vollem Umfang von einem befragten Personenkreis korrekt interpretiert werden.

6.2 Ausblick

Die Arbeit hat für das Unternehmen ISB AG den Grundstein auf dem Gebiet der Visualisierung von IT-Konsolidierungsprojekten gelegt. Zahlreiche Themengebiete wurden aufgezeigt, welche ein hohes Potential aufweisen. Diese sollten zukünftig näher untersucht werden. Im speziellen ist es vorstellbar die Untersuchung der Graphmuster aus Kapitel 3.3 zu intensivieren. Sie stellen eine neue Entwicklung auf dem Gebiet dar und repräsentieren somit den essenziellen Mehrwert dieser Arbeit. Des Weiteren ist es sehr gut vorstellbar, dass weiterer Muster mit einer Beziehung zu speziellen Situationen der IT-Landschaft eines Unternehmens existieren.

Auch sollten die in dieser Arbeit erstellten Visualisierungskonzepte auf ihre Anwendbarkeit näher betrachtet werden. Vor allem der Vergleich zu aktuell gängigen Visualisierungskonzepten im Bereich des EAM könnte von Interesse sein. Eine erste Erprobung der erstellten Konzepte im Rahmen eines Projekts könnte diesbezüglich wichtige Informationen und Erkenntnisse liefern. In diesem Zusammenhang könnte beispielsweise auch das Potential der Informationsvermittlung der Grafiken untersucht werden.

Für zukünftige Arbeiten basierend auf dieser Thesis, sei abschließend eine Entwicklung eines Visualisierungswerkzeugs genannt. Der in dieser Arbeit implementierten Prototyps könnte hierbei die Grundlage der Entwicklung darstellen. Entsprechend könnte eine Weiterentwicklung des Prototyps erfolgen, bis hin zu einem Visualisierungswerkzeug. Eine komplette Neuentwicklung, welche sich auf die Erfahrungen des praktischen Teils dieser Arbeit stützt,

wäre auch denkbar. Ein Mehrwert für das Unternehmen wäre durch ein solches Werkzeug jedoch in beiden Fällen gegeben.

Quellenverzeichnis

2020. [Online] 2020. [Zitat vom: 03. 01 2020.] <https://w3techs.com/>.

Andreas M. Böhm, Bettina Jungkuntz. 2005. *Grundkurs IT-Berufe*. s.l. : Vieweg-Verlag, 2005.

Andreas Maier, Michael Kaufmann. 2016. *NoSQL-Datenbanken*. s.l. : Springer Vieweg, 2016. 8. Auflage.

BITKOM - Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e. V. 2011. bitkom.
bitkom.org. [Online] 2011. [Zitat vom: 11. 10 2019.] <https://www.bitkom.org/sites/default/files/file/import/EAM-Enterprise-Architecture-Management-BITKOM-Leitfaden.pdf>.

Bostock, Mike. 2019. *D3.js.org*. *d3js.org*. [Online] 2019. [Zitat vom: 08. 01 2020.] <https://d3js.org/>.

2020. Business-IT (Grafik). *business-it.link*. [Online] 2020. [Zitat vom: 07. 01 2020.] <https://www.business-it.link/unternehmen/vision-leitbild-strategie>.

Hanschke, Inge. 2016. *Enterprise Architecture Management - einfach und effektiv: Ein praktischer Leitfaden für die Einführung von EAM*. München : Carl Hanser Verlag, 2016. 2. Auflage.

—. **2013.** *Strategisches Management der IT-Landschaft: Ein praktischer Leitfaden für das Enterprise Architecture Management*. München : Carl Hanser Verlag, 2013. 3. Auflage.

Hunger, Michael. 2014. *Neo4j 2.0: Eine Graphdatenbank für alle*. Paderborn : entwickler-press, 2014.

Innern, Bundesministerium des. BMI.Bund. [Online] [Zitat vom: 08. 10 2019.] <https://www.bmi.bund.de/DE/themen/it-und-digitalpolitik/it-des-bundes/it-konsolidierung/it-konsolidierung-node.html>.

Killelea, Patrick. 2002. *Web performance tuning*. s.l. : O'Reilly, 2002.

Klein, Manfred. 2017. eGovernment Computing. *egovernment-computing.de*. [Online] 13. August 2017. [Zitat vom: 11. 10 2019.] <https://www.egovernment-computing.de/was-ist-it-konsolidierung-in-der-oeffentlichen-hand-a-741563/>.

Mario Winter, Mohsen Ekssir-Monfared, Harry M. Sneed, Richard Seidl, Lars Borner. 2013. *Der Integrationstest: Von Entwurf und Architektur zur Komponenten- und Systemintegration*. München : Carl Hanser Verlag, 2013.

Michael Matzer, Big Data Insider (E-Book). 2019. *Graph-Datenbanken*. Augsburg : Vogel IT-Medien, 2019.

- Murray, Scott. 2017.** *Interactive Data Visualization for the Web: An introduction to designing with D3*. United States of America : O'Reilly, 2017. 2. Auflage.
- 2019.** Opensource.Zalando. *opensource.zalando.com*. [Online] 12. 2019. [Zitat vom: 05. 11. 2019.] <https://opensource.zalando.com/tech-radar/>.
- Richard Lackes, Markus Sieper. 2018.** Wirtschaftslexikon Gabler. *wirtschaftslexikon.gabler.de*. [Online] 19. 02. 2018. [Zitat vom: 08. 11. 2019.] <https://wirtschaftslexikon.gabler.de/definition/datenmodell-28093/version-251730>.
- 2019.** RYTE WIKI. [Online] 2019. [Zitat vom: 21. 02. 2020.] <https://de.ryte.com/wiki/Mockup>.
- Stefan Luber, Nico Litzel. 2017.** bigdata-insider. *bigdata-insider.de*. [Online] 12. 06. 2017. [Zitat vom: 04. 11. 2019.] <https://www.bigdata-insider.de/was-ist-nosql-a-615718/>.
- Trelle, Tobias. 2017.** Codecentric Blog: IT Expertenwissen von Entwicklern für Entwickler. *blog.codecentric.de*. [Online] 19. 06. 2017. [Zitat vom: 16. 10. 2019.] <https://blog.codecentric.de/2017/06/graphen-visualisierung-mit-neo4j/>.
- Wolff, Eberhard. 2016.** *Continuous Delivery: Der pragmatische Einstieg*. s.l. : dpunkt.verlag, 2016. 2. Auflage.

Glossar

Begriff	Definition / Erklärung
Ampel	Hier werden Kennzahlen durch Farben visualisiert. In der Praxis werden häufig dreifarbige Ampel („rot“ – „gelb“ – „grün“) verwendet.
Bebauung	Sie füllt die von der Architektur vorgegebenen Strukturen.
Best Practice	Ist die Bezeichnung für eine Zusammenstellung von Wissen, Methoden und Standards zu Praktiken, welche sich in der Vergangenheit bewährt haben.
Deployment	Softwareauslieferungsprozess
Enterprise Architecture (EA)	Dieser Begriff ist gleichbedeutend mit Unternehmensarchitektur. Diese generiert eine gesamthafte Sicht auf das Unternehmen. Hier werden die wesentlichen fachlichen und IT-Strukturen festgelegt und mit einander verknüpft. Somit lassen sich Business und IT, sowie deren Zusammenhänge beschreiben.
Enterprise Architecture Management (EAM)	Dies ist ein systematischer und ganzheitlicher Ansatz, um das Verständnis, Kommunizieren, Gestalten und Planen der fachlichen und technischen Strukturen innerhalb des Unternehmens. Dadurch kann die Komplexität einer IT-Landschaft beherrscht und weiterentwickelt werden.
Fachliches Domänenmodell	Gibt eine übergeordnete fachliche Struktur vor. Aktuelle oder zukünftige fachliche Elemente (z. B. Geschäftsprozesse) werden hier einsortiert. Es definiert den Rahmen für die Weiterentwicklung von Business und IT.
Funktionales Referenzmodell	Dies ist die Ausprägung eines fachlichen Domänenmodells. Dokumentiert werden hier die fachlichen Funktionen und Fähigkeiten des Unternehmens.

Geschäftsprozess	Eine zielgerichtete Abfolge von Aktivitäten. Diese leisten einen essenziellen Beitrag zur Wertschöpfung des Unternehmens bei.
IT-Konsolidierung	Der Einsatz von Menschen, Prozessen und Technologien wird optimiert, für einen effektiveren/effizienteren Betrieb. Ziele sind die Optimierung von IT-Ressourcen, Kosten und des Service-Levels als auch die Flexibilität des Unternehmens zu erhöhen.
IT- Landschaft	Umfasst die Gesamtheit aller IT-Systeme (Informationssysteme, Daten, Schnittstellen, technische Bausteine und Betriebsinfrastrukturen).
Mockup	Dies stellt einen digitalen Entwurf dar. Zu Beginn der Konzeptionsphase dienen diese zur Visualisierung von Ideen und Konzepten.
Prozesslandkarte	Ausprägung eines fachlichen Referenzmodells. Sie stellt die Geschäftsprozesse in ihrem Zusammenwirken grafisch dar. Dadurch entsteht eine fachliche Strukturierung für das Unternehmen, welche sich in der Organisation widerspiegelt.
Unternehmensarchitektur	Dieser Begriff ist ein Synonym für Enterprise Architecture (EA).

Die Erklärungen des Glossars sind von Wolff, 2016; Hanschke, 2013, als auch ryte.com, 2019 entnommen.

Anhang

In der angehängten Daten-CD befinden sich folgende Punkte:

- schriftliche Arbeit
- kompletter Quellcode
- aktuelle Datenbestände