Entwurf und Implementierung einer Werkzeugunterstützung zur sprachlichen Analyse und automatisierten Transformation von Projektlastenheften im Kontext der Automobilindustrie

An der Fachhochschule Dortmund

im Fachbereich Informatik

Studiengang Informatik

Vertiefung Praktische Informatik

erstellte Thesis

zur Erlangung des akademischen Grades

Bachelor of Science

B. Sc.

von Aaron Schul,

geboren am 24.06.1997

und Felix Ritter

geboren am 31.08.1997

Betreuung durch:

Prof. Dr. Sebastian Bab und Prof. Dr. Steffen Helke

Dortmund, 28.02.2019

Zusammenfassung

Ein weitläufiges Problem in der Autoindustrie ist die effiziente Verarbeitung von Pflichtenheften. Einer der Gründe dafür ist, dass viele verschiedene Bereiche bei der Entwicklung der Fahrzeuge und sogar kleinster Einzelteile beteiligt sind. So müssen zu Beginn einer Produktentwicklung etwa Betriebswirtschaftler, Designer, Techniker und Ingenieure zusammen ein Dokument entwerfen, das die Produktmerkmale widerspiegelt. Darin müssen die Anforderungen an das gewünschte Produkt so genau beschrieben sein, sodass es anhand dieses Dokuments entwickelt werden kann. Als Experten ihrer jeweiligen Domäne weiß dabei jeder genau, was dazu nötig ist. Sobald jedoch Auswirkungen über die eigene Domäne hinausgehen, kann es schnell passieren, dass Widersprüche oder Abhängigkeiten entstehen. Diese werden später schnell übersehen, da solche Zuständigkeiten nicht eindeutig geklärt sind. Das Resultat ist dann ein Fehler in der Entwicklung, welcher von zeitlicher Verzögerung über zusätzliche Kosten bis hin zum Abbruch des Projekts führen kann. Methoden aus dem Natural Language Processing (NLP) können dabei helfen, die Überprüfung der Pflichtenhefte auf Konsistenz stark zu vereinfachen oder sogar teilweise zu automatisieren. Dafür kann beispielsweise der Text in den Lastenheften analysiert und dessen Inhalt innerhalb einer Wissensbasis, sog. Ontologie, abgespeichert werden. In dieser Arbeit wird daher die Entwicklung zweier NLP-basierter Werkzeuge zur Vereinfachung bzw. Lösung dieser Probleme vorgestellt werden. Zum einen ein *Requirements-to-Boilerplate-Converter*-Werkzeug (R2BC), welches dem Nutzer helfen soll, das Pflichtenheft eines Auftraggebers in die betriebsinternen Richtlinien und Standards zu überführen. Zum anderen wird der *Delta-Analyser* vorgestellt, welcher auf dem R2BC aufbaut, indem er automatisch zwei homogene Lastenheft vergleicht und dadurch im Kontext des gesamten Pflichtenhefts Widersprüche und Abhängigkeiten herausstellt. Ziel ist es dabei, jeweils einen Prototypen als „*proof of concept*“, also als Machbarkeitsstudie, zu implementieren. Nach dieser Vorstellung wird zusätzlich das weitere Potential der Programme erläutert, welches sich bei der Entwicklung der Prototypen gezeigt hat. Diese bietet Vorschläge zur weiteren bzw. vollständigen Implementierung der Programme.

Abstract

A widespread problem in the auto industry is the efficient processing of specifications. One of the reasons for this is that many different areas are involved in the development of vehicles and even the smallest of parts. For example, at the beginning of product development, managers, designers, technicians, and engineers all need to design a single document together. The requirements for the desired product must be described in such detail that it can be developed using this document. As experts in their domain, everyone knows exactly what is needed. However, as soon as effects go beyond the own domain, it can quickly happen that contradictions or dependencies arise. These are quickly overlooked later, as such responsibilities are not clearly clarified. The result is then a mistake in the development, which can lead from time delay over additional costs up to the demolition of the project. Methods from Natural Language Processing (NLP) can help to simplify or even partially automate the review of functional specifications for consistency. For example, the text in the specifications can be analyzed and its contents stored within a knowledge base, so-called ontology. In this work, therefore, the development of two NLP-based tools to simplify or solve these problems will be presented. On the one hand, there is a program called Requirements-to-Boilerplate Converter (R2BC), which should help the user to transfer the specifications of a client into the company's internal guidelines and standards. On the other hand, the so-called delta analyzer is shown, which is based on the R2BC, in that it automatically compares two homogeneous specification sheets and thereby highlights contradictions and dependencies in the context of the entire specification. The goal here is to implement a prototype as a proof of concept of the program concepts from ZIC19. Due to the limited resources of this work, a list with further potential of the programs, which has been shown during the development of the prototypes, is additionally explained. This offers suggestions for further or complete implementation of the programs.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

NLP	Natural Language-Processing
OWL	Web-Ontology-Language
POS	Part-Of-Speech
R2BC	Requirements-To-Boilerplate-Converter
DA	Delta-Analyser
CRS	Customer Requirement Specification
OEM	Original Equipment Manufacturer
E-AE	Electronics-Advanced Engineering
PMT	Projects, Methods and Tools

Danksagung

An dieser Stelle möchten wir uns bei allen Personen bedanken, die uns bei unserer Bachlorarbeit und während des Studiums begleitet und unterstützt haben.

Der Dank gebührt unseren Familien und Freunden aus der Heimat sowie all jenen tollen Menschen, die wir durch das Studium an der FH Dortmund erst kennenlernen durften.

Wir danken auch allen Kollegen aus der Abteilung E-AE bei der Firma Hella in Lippstadt für ihr Interesse und die großartige Zusammenarbeit. Die einzigartigen Erfahrungen, die wir während unseres Praktikums und später während der Bachelorarbeit in der Firma sammeln konnten, haben uns wirklich motiviert. Besonderer Dank geht dabei an Konstantin Zichler, der weit mehr als nur Betreuer in der Firma, sondern auch Mentor und Ideengeber unserer Arbeit war.

Schlussendlich danken wir unseren Betreuern Prof. Dr. Sebastian Bab und Prof. Dr. Steffen Helke für die zahlreichen Tipps und Anregungen während der Bachelorarbeit.

Kapitel 1

Einführung

1.1 Motivation und thematische Grundlagen

In einer Vielzahl von Firmen stellt die Erhebung von Anforderungen den ersten Schritt bei der Entwicklung von neuen Produkten dar. Diese Anforderungen manifestieren sich später in der zu entwickelnden Hard- und Software. Spätere Produktmerkmale müssen dabei weit vor der tatsächlichen Entwicklung berücksichtigt werden und fließen in das Anforderungsdokument ein. Die Erhebung und Identifikation dieser Anforderungen wird als *Requirements-Engineering* bezeichnet [BAL10].

Dieser Prozess wird dabei sowohl bei kompletten Neuentwicklungen, als auch bei der Adaption eines bereits vorhandenen Produktes auf neue Projekte durchlaufen. Zur Rolle von Requirements-Engineering in Projekten siehe etwa [MW02] und [HL01].

Dem Betrieb liegen anschließend die Aufgaben und Ziele für die Entwicklung als Dokumente vor, auf deren Basis die Produktentwicklung beginnen kann. Die Entwicklung unterliegt dabei bestimmten Kriterien und Faktoren, die den unternehmerischen Erfolg beeinflussen. Neben betriebswirtschaftlichen Einflüssen wie der Einordnung des Produktes in der Wertschöpfungskette sind es dabei besonders technische Anforderungen an das Produkt, die definiert und während der Produktentwicklung eingehalten werden müssen.

Anforderungen im betrieblichen Entwicklungsprozess

Das Resultat ist das sogenannte Pflichtenheft, in Unternehmen häufig als CRS (*Customer Requirement Specification*) bezeichnet [DGE19]. Mithilfe

dessen kann nun die Entwicklung eines Produktes begonnen werden. Ein später hergestelltes Produkt sollte möglichst allen beschriebenen Anforderungen entsprechen. Da Betriebe jedoch nur selten alle einzelnen Komponenten des späteren Produktes selbst herstellen können, sind sie auf Zulieferer angewiesen, die einzelne Komponenten entwickeln und produzieren. Diese Hersteller einzelner Komponenten werden als *OEM* (*Original Equipment Manufacturer*), zu Deutsch *Erstausrüster*, bezeichnet [IR09].

In diesem Fall wird dem Zulieferer ein Pflichtenheft übergeben, dass allen Anforderungen des Kunden genügt. Das bedeutet jedoch, dass auch die Produktentwicklung anhand des Pflichtenheftes an den Zulieferer ausgelagert wird. Dem Zulieferer werden damit Möglichkeiten geboten und Verantwortung übertragen, das Produkt weiter zu definieren. Einerseits können etwa interne Richtlinien und Erfahrungen auf dem Gebiet der Produktentwicklung aus früheren Projekten integriert werden, andererseits aber auch weitere Optimierungen sowie Merkmale, die noch nicht im Pflichtenheft dokumentiert sind. Somit agiert der Zulieferer nicht nur als produzierende Fabrik, sondern nimmt am Entwicklungsprozess teil.

Die Anforderungen an ein Produkt, etwa technische Rahmenbedingungen und Qualitätsanforderungen, setzen sich dabei aus einer Auflistung von funktionalen und nicht-funktionalen Anforderungen zusammen [BAL10]. Häufig enthält das Pflichtenheft auch weitere Informationen und abseits von Text Abbildungen, die einzelne Anforderungen ergänzen.

Umstände der Anforderungsdefinition

Die Anforderungen werden dabei von vielen verschiedenen Domänenexperten beim OEM formuliert und in das Pflichtenheft für die Entwicklung eingetragen. Verschiedenste Akteure aus einem Unternehmen sind dabei an der Festlegung der Anforderungen an ein Entwicklungsprojekt bzw. Produkt beteiligt. Beteiligte sind etwa Produktdesigner, Ingenieure und Systemtechniker, die an verschiedenen Stellen im Lastenheft Anforderungen an eine Komponente festlegen und an entsprechenden Stellen vermerken. Diese Beteiligten sind in der Regel auf ihren Bereich spezialisiert und nicht interdisziplinär, da häufig an verschiedenen Orten und an spezifischen Aspekten eines Produktes in mehreren Projektteams entwickelt wird.

Projektteams aus einem Bereich tauschen sich häufig nicht untereinander aus und legen unabhängig von anderen Beteiligten Anforderungen fest. Demzufolge sammeln sich im Lastenheft schon bei der Entstehung beim OEM

verschiedenste Merkmale einer Komponente, die aber nicht im Bezug zueinander stehen und sich im schlimmsten Fall gegenseitig ausschließen. Ferner gibt es sprachliche Eigenheiten der Autoren und sprachliche Fehler, die innerhalb des Teams nicht auffallen und später nicht mehr korrigiert werden. Auch gibt es unternehmensinterne Richtlinien für die Formulierung von Anforderungen beim OEM, die das Verständnis auf Seite des Zulieferers erschweren können. Christian Allmann, Lydia Winkler und Thorsten Kölzow haben in [AWK06] solche Herausforderungen für das Requirements-Engineering zwischen OEMs und Zulieferern identifiziert.

Durch die entstehende große fachliche Breite und Tiefe der Spezifikationen im Pflichtenheft, können mitunter Dokumente mit einem Umfang von von mehreren tausend Seiten entstehen. Dies bedeutet einen hohen Aufwand bei der Transformation und Bündelung der Anforderungen in ein tatsächliches Produkt beim Zulieferer.

Prozessoptimierung durch Werkzeuge

Bei der Untersuchung von Projektlastenheften muss also nach Zusammenhängen und Bezügen zwischen mehreren Anforderungen gesucht werden, damit die Korrektheit des späteren Produktes gewährleistet ist. Die Analyse von Zusammenhängen zwischen Anforderungen stellt dabei aus Gründen der Effizienz ein Problem dar, wenn jeder Projektbeteiligte von Hand die für ihn relevanten Anforderungen aus dem Lastenheft extrahieren muss. Auch müssen die Lastenhefte an die Formulierungen und Ausdrucksweisen für Requirements-Management im Unternehmen angepasst werden.

Bislang gibt es jedoch kaum Werkzeugunterstützung, die effiziente Möglichkeiten zur automatisierten Überarbeitung und Anpassung einzelner Anforderungen aus dem Dokument bietet. Ansätze aus dem *Natural-Language-Proessing*, kurz NLP, stellen gleichzeitig vielversprechende Forschungsfelder in der Informatik dar, die eine solche automatisierte Verarbeitung auf Basis von Sprachanalyse ermöglichen. Diese werden in Kap. 2 ausführlich erläutert. Syntax und Semantik der einzelnen Sätze und Zusammenhänge in Texten können auf Basis aktueller Trends wie Machine-Learning und dynamischer Programmierung zunehmend besser abgebildet werden.

1.2 Unternehmenskontext der Arbeit

Besonders betroffen von dem geteilten Entwicklungsprozess zwischen OEMs und Zulieferern ist die Automobilindustrie. Eine Vielzahl von Zulieferern beliefert die Automobilhersteller mit Komponenten für ihre Fahrzeuge. Einzelne Komponenten bestehen dabei häufig auch aus mehreren Einzelteilen, die wiederum von mehreren verschiedenen Zulieferern geliefert werden. [AWK06]

1.2.1 Einsatz bei HELLA

Einer dieser Zulieferer ist die HELLA GmbH & Co. KGaA (im folgenden (die) HELLA genannt), die sich auf die Entwicklung und Produktion von Licht- und Elektronikkomponenten in Hard- und Software spezialisiert hat. HELLA ist ein international operierender deutscher Automobilzulieferer mit Hauptsitz in Lippstadt und zählt zu den Top 100 Automobilzulieferern weltweit [LV10]. Für weitere Information siehe [HE19] sowie die Firmenwebsite.

Zum Verfassen dieser Arbeit waren die Autoren fünf Monate bei der Firma beschäftigt. Dies beinhaltete zwei Monate Arbeit als Praktikanten und die verbleibenden drei Monate als Bacheloranden. Die Hauptaufgabe dort war die Mitarbeit im Bereich Requirements-Engineering für die Abteilung der Elektronik-Vorentwicklung *E-AE* (*Electronics-Advances Engineering*).

Die Abteilung ist zuständig für die Planung und Umsetzung von Vorentwicklungsprojekten um die Ergebnisse an die Serienentwicklung oder verschiedene OEMs weiterzuleiten. Dies beinhaltet häufig, wie zuvor beschrieben, den Austausch von Informationen der Firma HELLA mit diversen OEMs bezüglich gemeinsamer Projekte. Die genaue Darstellung dieser Prozesse erfolgt in Kap. 3.1. Somit erhält diese Abteilung auch oft Lastenhefte der OEMs, welche dann an die Projektteams weitergeleitet werden.

1.2.2 Anforderungsmanagement im Unternehmen

Die Gewichtung einzelner Anforderungen in einem größeren Systemkontext fällt dort schwer, da nun Projektteams beim Zulieferer, die an der Entstehung des Lastenheftes nicht beteiligt waren, dieses verstehen sollen und auf ihre Teilkomponente anwenden müssen. Zusätzlich zu den teilweise widersprüchlichen Anforderungen aus dem Pflichtenheft kommen weitere Anforderungen des Zulieferers, wie etwa die Gewinnmarge und unternehmensinternte Richtlinien hinzu. Schlussendlich soll jedoch ein Produkt entwickelt werden, dass

möglichst alle Anforderungen berücksichtigt. [MW02]

Zu diesem Zweck können bei Zulieferern spezielle Abteilungen existieren, die Projektteams bei der Identifikation und Gewichtung von Anforderungen unterstützen. Diese Abteilungen sind auf die Auswertung der Projektlastenhefte spezialisiert und erstellen Modelle zu Merkmalen des späteren Produktes. Die Experten stellen dabei die Eindeutigkeit und Verständlichkeit sicher und identifizieren relevante Anforderungen für die Mitglieder des Projektteams. Somit kann ein eigenes Anforderungsdokument für das Projektteam beim Zulieferer erstellt werden. Falls nicht vorhanden, muss das Team selbst über die Bewertung von Anforderungen entscheiden und Rücksprache mit dem OEM halten.

Bei HELLA existiert eine dementsprechende Abteilung, genannt *Projects, Methods and Tools*, kurz *PMT*. Im Rahmen unserer Beschäftigung konnte in enger Zusammenarbeit mit PMT genau ermittelt werden, wie Lastenhefte momentan verarbeitet werden, welche Verbesserungspotentiale dabei bestehen und wie diese genutzt werden können.

1.3 Hypothese

Hypothese dieser Arbeit ist, dass sich mithilfe von NLP Lastenhefte effizient automatisiert verarbeiten lassen. Damit kann die Arbeit von Requirements-Engineers, aber auch von Beteiligten an der Entwicklung beim Verständnis der Anforderungen, erleichtert werden. Dabei ist die Auswertung der Syntax und Semantik für ein tieferes Verständnis von Textzusammenhängen, also von verschiedenen Anforderungen, relevant.

Die sprachliche Analyse kann Requirements erkennen und kategorisieren, sowie in Beziehung zueinander setzen. Insbesondere lassen sich die beschriebenen Probleme bei der Überprüfung der Pflichtenhefte auf Konsistenz lösen oder zumindest vereinfachen. Dies kann mithilfe der in dieser Arbeit beschriebenen Programme geschehen, welche konzeptuell dargestellt und prototypisch implementiert werden.

Weiterhin soll geprüft werden, wie stark ein Mehrwert bei der Nutzung solcher Programme für ein Firma ausfallen kann. Dies beinhaltet ersparte Arbeitszeit, Kosten und vermiedene Fehler.

1.4 Methodik

Verschiednene Methoden aus dem Bereich Natural Language-Processing werden zunächst anhand des Rückblicks auf die Projektarbeit theoretisch und anhand praktischer Beispiele erläutert. Der betriebliche Kontext der Firma Hella wird durch genaue Darstellung der dortigen Verfahren beleuchtet.

Mit diesen Grundlagen wird dann die Unterstützung der dortigen Prozesse im Bereich Requirements-Engineering auf Basis von natürlichsprachlicher Textanalyse evaluiert. Die gewonnenen Erkenntnisse aus der Zusammenarbeit mit der Abteilung PMT wurden in Anforderungen an die zu entwickelnden Werkzeuge zusammengefasst, die als Basis für unsere Softwareentwicklung dienten. Dadurch entstand eine genaue Vorstellung der zu entwickelnden Software, welche mit Hilfe der Programmkonzepte von Konstantin Zichler ZIC19 entworfen und implementiert werden konnte.

Das Ergebnis stellten zwei Software-Prototypen auf *java*-Basis dar, deren Struktur und Funktionsweisen wir anhand von Modellen und Codebeispielen darstellen. Die Prototypen wurden mit originalen Pflichtenheften von verschiedenen OEMs und der HELLA getestet und evaluiert.

1.5 Aufbau der Arbeit

In der folgenden Ausarbeitung wird zunächst in Kapitel 2 eine Überblick über den Stand der Technik gegeben. Dies beinhaltet einerseits einen Rückblick auf die zuvor verfasste Projektarbeit zu NLP und einer Erläuterung der Grundlagen von Ontologien und andererseits einen Überblick von Arbeiten, die sich der gleichen oder ähnlichen Problemstellungen gewidmet haben.

Kapitel 3 beschäftigt sich für eine bessere Einordnung der beiden Programme mit dem betrieblichen Umfeld bei der HELLA. Hierzu wird zunächst, vor dem Hintergrund des momentanen Vorgehens, der Anwendungsfall der Programme genauer beschrieben. Somit lassen sich die betrieblichen Anforderungen an die Werkzeuge bestimmen, welche dann von den vorgestellten Konzepten abgedeckt werden sollen.

In Kapitel 4 wird das erste Werkzeug - der R2B-Converter - ausführlich beschrieben. Dies beinhaltet die Darstellung dessen Architektur, eine Vorstellung der Implementierung, eine Auflistung möglicher Erweiterungen und eine genaue Beschreibung des Testverfahrens der Software. Zu den Tests werden dabei präzise die Methodik, die Durchführung und die Ergebnisse

vorgestellt, welche in dem späteren Evaluations-Kapitel aufgegriffen und eingeordnet werden.

Das zweite Werkzeug - der Delta-Analyser - wird in Kapitel 5 behandelt. Dabei werden ähnlich zu Kapitel 4 die Architektur, die Implementierung, mögliche Erweiterungen und durchgeführte Tests beschrieben. Es wird genau dargelegt wie die Testergebnisse erzielt wurden, die später zusammen mit den Ergebnissen von Kapitel 4 evaluiert werden, um das Resultat der beiden Werkzeuge bewerten zu können.

Kapitel 6 stellt das zuvor erwähnte Evaluations-Kapitel dar. Hier findet zunächst die Auswertung der erhobenen Testergebnisse statt. Anschließend wird anhand dessen dargestellt, ob und inwiefern das Nutzen der Werkzeuge einen Mehrwert für die Firma erwirtschaften kann und somit im betrieblichen Umfeld eingesetzt werden sollte. Somit lässt sich die Korrektheit der in 1.2 gestellten Hypothese überprüfen.

Die Arbeit wird von Kapitel 7 abgeschlossen. Dieses beinhaltet eine Zusammenfassung der geleisteten Arbeit und der Ergebnisse. Außerdem wird zuletzt noch ein Ausblick für die Zukunft der Werkzeuge und die Lösung des Problems erläutert.

1.6 Autorenverzeichnis

1.: Ritter, Schul
- 1.1: Ritter
- 1.2: Ritter
- 1.3: Ritter, Schul
- 1.4: Schul
- 1.5: Schul

2.: Ritter, Schul
- 2.1: Schul

Kapitel 2

Stand der Technik

In diesem Kapitel wird zunächst ein Rückblick auf die zuvor von den Autoren verfasste Projektarbeit und die darin enthaltenen Grundlagen von NLP gegeben. Aufgrund ihrer besonderen Relevanz für die Programme werden anschließend zusätzliche Ontologien als Teil von NLP genauer beleuchtet. Außerdem wird mit Hilfe von verwandten Arbeiten ein Überblick über die Problematik und verschiedene Lösungsansätze geboten.

2.1 Rückblick auf die Projektarbeit

In [RS18] sind die Grundlagen von NLP im Rahmen einer Projektarbeit zusammengefasst. In diesem Abschnitt werden die für die Arbeit relevanten Aspekte kurz wiederholt.

2.1.1 Grundlagen zu Natural Language Processing

Definition und Ziel

Natural Language Processing ist zu verstehen als die automatisierte oder halb-automatisierte Verarbeitung von natürlicher Sprache mit Hilfe eines Computers. In NLP sind verschiedenste wissenschaftliche Bereiche verknüpft.[1] Hauptsächlich handelt es sich dabei um Linguistik und Informatik, jedoch

[1] Es ist anzumerken, dass verschieden weit gefasste Definitionen von NLP existieren. Manche schließen etwa auch die Erfassung von Sprache mit ein, etwa durch Interaktion mit Sprachcomputern. Andere beschränken sich ausschließlich auf den Verarbeitungsprozess der Sprachdaten im Computer selbst, siehe 2.1.2.

gibt es auch viele Verbindungen zur Psychologie, Philosophie Mathematik und Logik. [COP04]

Ziel von NLP ist üblicherweise die Extraktion von Informationen aus einem Text durch kleinschrittige Textanalyse. Diese Informationen werden dem Text direkt über sogenannte Annotationen angehängt oder anders abgespeichert, etwa in Ontologien, welche in einem späteren Abschnitt genauer erläutert werden.

Beispiele von Forschung und Entwicklung

Die wissenschaftliche Erforschung von NLP entstand Mitte des 20. Jahrhunderts, als der Linguist Noam Chomsky zeigte, dass sich Merkmale der englischen Sprache formalisieren und somit automatisieren lassen [CHO57].

Frühe Projekte wie der Sprachcomputer ELIZA aus den 1960er Jahren waren somit bereits in der Lage eine natürlichsprachliche Frage als Texteingabe entgegenzunehmen und üblicherweise eine plausible Antwort zu liefern.

Heutzutage gibt es ein Vielzahl von Spracherkennungssoftware auf Smartphones oder Smart-Devices wie ALEXA, welche in der Lage sind gesprochene Kommandos zu erkennen und über das Internet an Server zu leiten, welche in der Lage sind diese Kommandos schnell und effizient zu verarbeiten [HAO14].

Herausforderungen von NLP

Durch die Verbindung von Linguistik und Informatik stellt sich NLP generell der Herausforderung sowohl inhärente Probleme der Sprache, als auch Probleme der Automatisierung lösen zu müssen.

Zu den sprachlichen Problemen gehören Aspekte wie Mehrdeutigkeit, Sarkasmus und Umgangssprache bzw. Redewendungen. Diese lassen sich oft nicht einzig und allein durch den gegebenen Text erklären, sondern beruhen auf Kontext und Situation.

Vor allem bei der Erfassung von Semantik, also dem Inhalt der Texte, können komplexe Probleme auftreten. So gibt es beispielsweise Mehrdeutigkeiten, welche selbst für den Menschen nicht einfach zuzuordnen sind. So kann „*Die Betrachtung des Studenten*“etwa bedeuten, dass der Student etwas betrachtet. Genauso kann es jedoch sein, dass jemand anderes den Studenten betrachtet.

Eine andere häufige Fehlerquelle ist das korrekte Zuordnen von sprachlichen Bezügen. Betrachtet man die Frage „*Verkaufen Sie Handys und Compu-*

ter von Samsung“, so ist diese eindeutig grammatikalisch korrekt. Dennoch ist es uneindeutig, ob sich die Frage auf nach der Marke der Handys richtet, oder ob dies lediglich auf die Computer bezogen ist.

Noch komplexer wird es, wenn sich diese Bezüge über verschieden Sätze bzw. Teilsätze erstrecken. So ist der Satz „*Tom schenkt Tim ein Fahrrad, weil er nett ist*“ebenfalls auf zwei verschieden Arten zu deuten. Einerseits kann es sein, das Tom das Fahrrad verschenkt, weil Tom ein netter Mensch ist, andererseits kann es sein, das Tom das Fahrrad verschenkt, weil Tim nett zu ihm ist.

Weiterhin können auch bei der korrekten Erfassung des Sprachinhalts weitere Probleme die Kommunikation stark beeinflussen. So kann etwa die Intention der Sprache dem Inhalt widersprechen, wie es bei Sarkasmus üblicherweise der Fall ist.

Bei der Implementierung ergeben sich dann Probleme, trotz der sprachlichen Besonderheiten konsistente Regelmäßigkeiten als Grundlage zur Automatisierung zu finden.

Linguistische Analyse

Um einen Text möglichst Fehlerfrei verarbeiten zu können bedarf es also einem kleinschrittigen und präzisen Verfahren. Dieses bezeichnet man als linguistische Analyse.

Sie ist aufgeteilt in vier wichtige Schritte, welche aufeinander aufbauen und an Komplexität zunehmen. Im folgenden sind diese Schritte aufgeführt und kurz erläutert [RS18]:

1. Morphologische Analyse - Die Zerlegung von Wörtern bezüglich ihrer Struktur. Die Komposition aus Präfix, Wortstamm und Suffix von Wörtern wird erkannt, um Fall, Tempus, Numerus etc. des Wortes zu bestimmen. Im Englischen zeigt so die Endung -ed bei Verben die Vergangenheitsform an. Dabei ist zu beachten, dass Mehrdeutigkeiten entstehen können.

2. Syntax - Aufbau von Sätzen durch einzelne Wörter. Jede Sprache besitzt syntaktische Regelungen bezüglich der auftauchenden Wörter; im Deutschen folgt etwa auf einen Artikel irgendwann ein Substantiv oder bestimmte Signalwörter geben Satztyp und Tempus an. Auf Basis dieses Wissens können formale und strukturelle Analysen der Textbestandteile erfolgen.

3. Semantiken - Die Identifikation der Bedeutung von einzelnen Sätzen und Wörtern. Die Semantik eines Textabschnittes wird häufig als „Logik“ bezeichnet und fragt nach dessen Bedeutung bzw. Thema. Semantische Deutungen können anhand von Kompositionen einzelner Wörter und Sätze auf Basis der semantischen Analyse erkannt werden.

4. Kontext - Darstellung von satzübergreifenden Zusammenhängen syntaktischer und semantischer Natur. Mehrere Sätze können über Konstrukte wie etwa Pronomen verbunden sein, wenn sich das Pronomen des einen Satzes auf ein Subjekt des anderen bezieht.

Diese Aufteilung lässt sich jedoch in noch weitere Teilschritte für die Spracherkennung und -generierung als NLP-Architektur unterteilen. Diese wird im folgenden Abschnitt genauer erläutert.

2.1.2 Umsetzung der Verarbeitungsschritte natürlicher Sprache

Dieser Abschnitt beschäftigt sich primär mit den verschiedenen Schritten der sprachlichen Analyse. Diese Differenzierung kann dabei, wie zuvor beschrieben, anhand der linguistischen Analyse unterschiedlicher Bestandteile von Sprache vorgenommen werden. Diese sukzessive Verarbeitung wird im folgenden anhand verschiedener Algorithmen und Verfahren erläutert, die im wesentlichen in [RS18] zu finden sind.

Das Ziel jeder Analyse ist es, Wissen über einen Teilbereich von Sprache zu generieren. Das Wissen über den Zusammenhang von Wörtern eines Textes kann eindeutig sein, jedoch können besonders bei der morphologischen Analyse sprachliche Mehrdeutigkeiten auftreten. Dies wird in 2.1.3-Morphologie näher erläutert.

Die Präzision dieser Aussagen fällt dementsprechend abhängig von der Treffergenauigkeit und dem Abdeckungsgrad der linguistischen Analyse durch die verwendeten Verfahren ab. Es ist dabei zu beachten, dass die Analyse typischerweise als Architektur aufgebaut ist, da die verschiedenen Schritte aufeinander aufbauen [COP04]. Somit können, basierend auf den Ergebnissen, inhaltliche Aussagen und Zusammenhänge über den zugrundeliegenden Text getroffen werden.

[COP04] identifiziert dabei eine allgemeine NLP-Architektur, die basierend auf der linguistischen Analyse diese Teilbereiche voneinander abgrenzt.

Für ein besseres Verständnis der Verfahren werden die für diese Arbeit relevanten Schritte in [RS18] wie folgt kurz erläutert:

1. Input Processing - Erkennung der Dokumentensprache und Normalisierung des Textes. Im ersten Schritt geht es um das korrekte Format des Eingabedokumentes für die Verarbeitung. Dieser Vorgang stellt jedoch noch keine Analyse dar, sondern dient lediglich der Vorbereitung.

2. Morphologische Analyse - Die meisten Sprachen sind im Bezug auf ihre Grammatik und syntaktische Struktur der Wörter in Systemen abgeschlossen, etwa durch Modellierung mittels Automaten zur Erzeugung der Worte. Auch können Vokabeln in Wörterbüchern/Lexika gesammelt und Regeln auf ihnen formuliert und gespeichert werden.

3. Part-of-Speech-Tagging - Die einzelnen Wörter eines Satzes werden im Hinblick auf den Sach- oder Satzzusammenhang, wie auch auf die Stellung im Satz hin analysiert. Basierend auf Kontext und/oder Erfahrungswerten bzw. Heuristiken können die Wörter korrekt erfasst und deren Fall getaggt werden. Subjekte, Prädikate usw. werden identifiziert. Dazu kann eine Wissensbasis mit Trainingsdaten und die Einordnung des Kontextes darin verwendet werden.

4. Parsing - Die Ergebnisse der vorherigen Schritte werden weiter verarbeitet und in ein standardisiertes Format gebracht. Syntaktische Zusammenhänge, wie etwa das Zusammenfassen von Wörtern zu einer gemeinsamen Bedeutungsphrase und das Zuordnen von Verben zu einem Nomen können nach dem Parsing dargestellt werden.

5. Disambiguation - Die Entfernung von Mehrdeutigkeiten auf Basis der Ergebnisse des Parsings stellt einen entscheidenden Schritt für die semantischen Analysen dar. Nur wenn die Aussage und der Kontext eines Satzes klar erkennbar sind, kann dessen Semantik gezielt abgeleitet werden.

6. Context Module - Textbausteine, deren Interpretation semantisch auf dem Kontext anderer Sätze oder Wörtern beruhen, können erfasst werden. Bei Anaphern ist etwa das Verständnis des aktuellen Kontexts abhängig von einer vorherigen Deutung.

7. Text Planning - Die Sachzusammenhänge und Semantiken, die aus dem zugrundeliegenden Text extrahiert wurden, werden für die Darstellung im fertigen Text definiert. Es wird festgelegt, welche der Bedeutungen qualitativ übertragen und vermittelt werden sollen. Die Reihenfolge und Umfang der behandelten Themen wird geschätzt und definiert.

8. Tactical Generation - Die Bedeutungen werden in Form konkreter Zeichenketten generiert, die die gewünschte Bedeutung enthalten. Diese Textbausteine basieren häufig direkt auf den Ergebnissen des vorherigen Parsings, da dort schon Bedeutungen zusammengefasst werden können. Der quantitative Teil des Textes wird erzeugt.

9. Morpholigical Generation - Die erzeugten Sätze werden morphologisch an die Rahmenbedingungen der verwendeten Sprache angepasst. Grammatiken und Regeln der Satzkonstruktion werden auf die erzeugten Wörter angewendet, sodass ein lesbarer und nachvollziehbarer Text entsteht.

10. Output Processing - Der Text wird nun, nachdem dieser inhaltlich komplett erzeugt vorliegt, an das Design und Format des jeweiligen Einsatzzwecks angepasst. Sollte Text, anders als in dieser Arbeit der Einfachheit halber angenommen, nicht als geschriebenes Wort ohne besondere Formatierung vorliegen, kann dieser einem *Formatter* zur Designanpassung übergeben werden. Ferner kann etwa mihilfe von Text-to-Speech eine Audioausgabe erfolgen.

Wie später gezeigt wird, kann die in dieser Arbeit beschriebene Entwicklung der Software-Prototypen als eine Umsetzung eben dieser NLP-Architektur verstanden werden.

2.1.3 Syntaktische Analyse von Wörtern und Sätzen

In diesem Abschnitt wird die Zerlegung der Syntax anhand logischer Modelle von Sprache beschrieben. Begonnen wird mit der einfachen Struktur und Zusammensetzung einzelner Wörter aus der Grammatik. Die Kombination aus Wortstämmen mit Prä- und Suffixen kann etwa in einem Lexikon gesucht werden, dass alle Wörter einer Sprache enthält. Basierend auf dieser Morphologie eines Wortes kann dann in der Analyse mehrerer Wörter auf die Rolle von Wörtern im Satzzusammenhang geschlossen werden.

Morphologie

Dieser Abschnitt befasst sich genauer mit der Analyse der Struktur und Zusammensetzung von Wörtern. Dies wird als morphologische Analyse bezeichnet. Wie im folgenden beschrieben, können auf dieser Basis logische Modelle wie etwa Automaten erstellt werden, die zulässige Wörter aus einer Sprache auf Basis grammatikalischer Regeln erzeugen können. Für ein besseres Verständnis der im Rahmen dieser Arbeit vorgestellten Ergebnisse beschränkt sich diese Arbeit auf die vergleichsweise einfach strukturierte englische Sprache. Aufgrund der großen Fortschritte und Erfahrungen in der Sprachforschung, werden die folgenden Beispiele auf Englisch behandelt.

In der englischen Sprache besteht jedes Wort aus einem Wortstamm, sowie den sogenannten Affixen. In einem Satz könnte etwa das Wort „*believable* "auftauchen. Das Wort stammt von der Grundform „*(to) believe* "(zu Deutsch: „glauben ") ab. Das -*able* stellt ein Suffix dar, also eine Wortendung. Zusammen mit den Präfixen, die dem Wortstamm vorangestellt sind, gehört -*able* somit zur Kategorie der Affixe.

Das Ziel der morphologischen Analyse ist nun, die Zusammensetzung des Wortes nachzuvollziehen und die Information für die weitere Verarbeitung bereitzustellen. Die Generierung des Wortes *believable* kann anhand von Buchstabierregeln erfolgen, die alle zulässigen Wörter auf englisch anhand der Wortstämme und Affixe erzeugen können. Es fällt auf, dass in unserem Beispiel *believable* vor dem Hinzufügen des Suffixes -*able* zunächst das -*e* von der Grundform *(to) believe* entfernt werden muss. Eine Grammatik, die zur Erzeugung dieser Wörter genutzt werden kann, muss dementsprechend eine solche Regel vorhanden sein. Gemäß [RS18] bedeutet dies formal ausgedrückt:

$$\texttt{e} \rightarrow \varepsilon \hat{} \texttt{\_able}$$

In der Logik kann man diese Regeln als nicht-deterministischer *Zustandswandler* visualisieren.[2] In Abb. 2.1 ist der Zustandswandler gezeigt, die das Suffix -*able* an entsprechende Wörter anhängen kann.

[2]In [RS18] wurde darauf hingewiesen, dass nicht-deterministische Zusatandswandler trotzdem mithilfe der *Potenzmengenkonstruktion* in einen äquivalenten deterministischen Zustandswandler überführt werden kann. Sie kommen somit zu einem eindeutigen Ergebnis *erzeugbar* oder *nicht erzeugbar* in der Sprache.

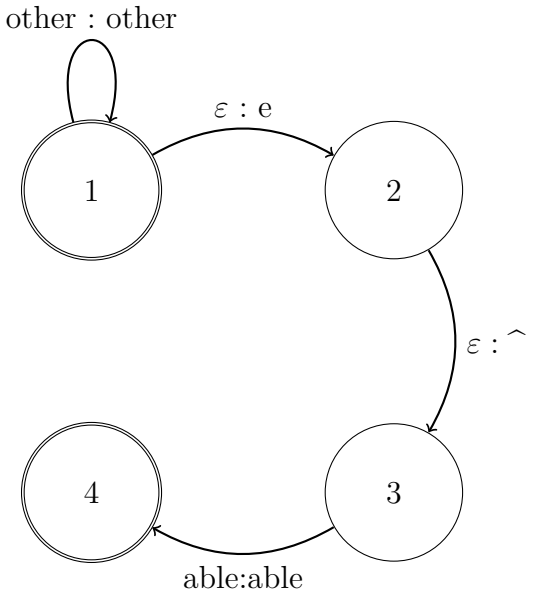

Abbildung 2.1: Nicht-deterministischer Zustandswandler der able-Regel [RS18]

Um alle zulässigen Wörter sammeln und identifizieren zu können, muss die morphologische Analyse über eine Menge an Informationen über die Sprache verfügen. Die Bestandteile von Wörtern können dabei in verschiedenen Lexika enthalten sein, welche die entsprechenden Informationen bereitstellen. Gemäß [COP04] bedarf es dabei drei verschiedener Lexika:

1. Liste von *Affixen* (zusammen mit den Informationen, die damit verbunden sind, z.B. Negation „`un-"`)
2. Liste von *unregelmäßigen Wörtern* (zusammen mit den Informationen, die damit verbunden sind, z.B. „`went":simple past„(to) go"`)
3. Liste von *Wortstämmen* (zusammen mit syntaktischer Kategorie, z.B. „`believe":verb`)

Es ist zu beachten, dass die in der morphologischen Analyse verwendeten Buchstabierregeln zwar zu jeweils eindeutigen Ergebnissen kommen können, die Analyse selbst jedoch nicht eindeutig sein muss. Dies kann, wie bereits zuvor beschreiben, durch sprachliche Mehrdeutigkeiten entstehen.

Das Wort „*even*“könnte isoliert etwa einmal als *gleichmäßig*, also als Adjektiv kategorisiert, ins Deutsche übersetzt werden, jedoch auch sinngemäß *sogar*, also das Adverb mit einem oder mehreren Bezugswörtern meinen.[3] Die mehreren möglichen Bedeutungen des einzelnen Wortes können jedoch in der Regel durch die Position und grammatikalische Funktion im Satz aufgelöst werden. Diese Eigenschaften werden durch das *Part-of-Speech-Tagging* identifiziert.

Part-Of-Speech-Tagging

Die Ergebnisse der morphologischen Analyse von mehreren Wörtern, etwa in einem Satz, können zur Analyse von Zusammenhängen zwischen diesen Wörtern genutzt werden. Somit kann die grammatikalische Struktur des Satzes auf Basis der einzelnen Wörter dargestellt werden. Der grammatikalische Typ jedes Wortes wird durch das Part-Of-Speech-Tagging festgelegt und jedes Wort wird *annotiert*. Wie später in dieser Arbeit beschrieben wird, dienen solche Annotationen in NLP-Werkzeugen der Sammlung der bekannten Informationen zu einem Wort oder Satz.

Zur Festlegung dieser grammatikalischen Funktionen werden sog. *POS-Tagger* eingesetzt, die verschiedene Worttypen als Kategorien beinhalten. Diese werden dann jedem Wort eines Satzes als Annotation hinzugefügt. Die Aufgabe ist es nun, die korrekte Funktion eines Wortes im Satz auf Basis der morphologischen Analyse zu identifizieren.

Als Beispiel soll der Satz

```
I love trains.
```

dienen. Das Part-of-Speech tagging zum Wort „*trains*“soll hier veranscheulicht werden. Das Wort „*trains*“allein ist ein Nomen im Plural, jedoch könnte es sowohl das Subjekt, als auch das Objekt im Satz darstellen. im Im Englischen sollte ein POS-Tagger aber erkennen, dass *Subjekt-Prädikat-Objekt* eine gültige Satzkonstruktion darstellt, die zu dem Beispielsatz passt. Dementsprechend entfällt die Möglichkeit von *trains* als Subjekt, da es hinter einem Verb als Akkusativobjekt des Satzes auftaucht.

[3] Aus Gründen der Vereinfachung wurde sich hier auf zwei erkennbar verschiedene Bedeutungen beschränkt. Tatsächlich stellt sich neben Bestimmung des grammatikalischen Typs auch die Frage, ob sich die verschiedenen Übersetzungen derselben Kategorie, z.B. *even* als *gerade* und *gleichmäßig*, in ihrer Bedeutung im Sachzusammenhang unterscheiden.

Es existieren verschiedene Techniken für diese Beurteilungen der POS-Tagger. Verschiedene gültige Satzkonstruktionen sind häufig als sog. *corpora* gespeichert. Diese stellen repräsentative Trainingsdaten dar, die beispielsweise aus der Zeitung *Wall-Street-Journal* stammen. In diesem Corpus könnten etwa die annotierten Sätze

```
People(subj) love(vb) trains(obj) .(punc) und
I(subj) go(vb) home(obj) .(punc)
```

enthalten sein.[4]

Mithilfe dieser Menge von Daten kann POS-Tagging auch heuristisch, d.h. auf Basis von Wahrscheinlichkeiten bzw. Metriken, erfolgen, indem das Auftauchen verschiedener Satzbauten im Trainingscorpus gezählt wird. Auf dieser Basis können dann Wahrscheinlichkeiten für verschiedene Möglichkeiten berechnet werden, wenn es mehrere mögliche Zuordnungen gibt.

Nutzen syntaktischer Analyse für NLP

Ein wichtiges Einsatzgebiet für POS-Tagger ist auch die Wortvorhersage bzw. *prediction* auf Basis solcher Heuristiken. Diese kann im Kontext der zuvor vorgenommen Annotationen von Wörtern erfolgen. Beispielsweise werden sog. *n-gramme* verwendet, um unter Berücksichtigung der Annotation von n gelesenen Wörtern die Möglichkeiten für das folgende Wort zu berechnen. In [RS18] findet sich ein vollständiges Beispiel anhand eines *Bigramms* mit Corpus.

Die heutigen Verfahren zur syntaktischen Analyse nutzen zur Steigerung der Genauigkeit u.a. Verfahren aus dem maschinellen und automatisierten Lernen und können somit hohe Trefferquoten erzielen.[5] Um ein tieferes Verständnis von Texten zu erzielen, reicht die in diesem Abschnitt beschriebene syntaktische Analyse anhand der Grammatik nicht aus. Die tatsächliche

[4] Die tatsächliche Annotation kann durch POS-Tagger i.d.R. feiner erfolgen, sodass etwa Pronomen von Subjekten unterschieden werden. Auch können, wie später in dieser Arbeit gezeigt wird, eigene Annotationen bei Bedarf nach neuen Kategorien hinzugefügt werden. Der Corpus hier stellt ein stark vereinfachtes Beispiel dar. Ein ausführliches Beispiel unter der Verwendung der CLAWS-7-POS-Taggers findet sich in [RS18].

[5] In [SHE07] wird ein POS-Tagger auf der Basis von bidirektionaler Analyse und Machine-Learning zum Erstellen der Trainingsdaten beschrieben. Obwohl nur auf Basis der Textsyntax erzielt, sind die Ergebnisse zu 97,33% korrekt.

Abbildung von Wissen über semantische Zusammenhänge stellt eine verlässlichere Möglichkeit dar, sprachliche Zusammenhänge zu identifizieren.

2.1.4 Umgang mit Semantik

Wie im letzten Abschnitt aufgezeigt wurde, können nicht alle Informationen aus einem Text eindeutig anhand der Erfassung der Grammatik eines Satzes extrahiert werden. Die Syntax eines Satzes ist häufig mehrdeutig, da der Zusammenhang einzelner Wörter verschieden interpretiert werden kann. Auf der semantischen Ebene, d.h. der Bedeutung, existieren jedoch bereits bei einzelnen Wörtern verschiedene Deutungen.

Schon bei der Mensch-Mensch-Kommunikation kommt es somit häufig zu Missverständnisse, wenn die Bedeutung hinter einer Aussage nicht eindeutig ist. Auf Basis des Kontextes einer Aussage bzw. der Domäne des Inhalts kann jedoch meist zu einer mehrdeutigen Aussage eine eindeutige Deutung identifiziert werden.

NLP beschäftigt sich daher auch intensiv mit der Abbildung und Modellierung der Semantik von Sprache, die in der Logik häufig als „*Logik einer Aussage*“bezeichnet wird.[6] Die Formalisierung mehrerer Bedeutungen kann dabei mithilfe von Symbolen der Aussagenlogik erfolgen, die Symbole dienen dabei als Platzhalter für Wörter bzw. Satzteile. Etwa das Beispiel aus [RS18]:

`Every man loves a woman.`

Zu deutsch *„Jeder Mann liebt...“* ist semantisch mehrdeutig zwischen

1. *„...die eine Frau“*: $\exists y[woman'(y) \bigwedge \forall x[man'(x) \Rightarrow love'(x,y)]]$ und
2. *„...irgendeine Frau“*: $\forall x[man'(x) \Rightarrow \exists y[woman'(y) \bigwedge love'(x,y)]]$

Bereits durch den Menschen fällt die Deutung anhand von Symbolen der Aussagenlogik nicht leicht. Daher wurden verschiedene Arten von Zusammenhängen, u.a. auf Basis von Symbolen der Mengenlehre, anhand welcher die Deutung vereinfacht und klarer erkennbar wird.

Meaning-Postulates

Linguisten und Philosophen haben Regeln für die Abbildung von Zusammenhängen auf eindeutige Kategorien formalisiert. Dabei wurden verschiedene

[6] Bereits in den 1980er Jahren wurden Konzepte auf Basis logischer Modelle, mit denen Semantik abgebildet werden kann, mithilfe von Theorembeweisern und Model-Checking beschrieben. [SB88] [KN85]

Bedeutungen von Wortzussammenhängen identifiziert, die als (Bedeutungs-)Mengenbeziehung bzw. Relation zwischen Symbolen visualisiert werden können [CAR52]:

1. *Meaning-Postulates*: A beinhaltet B und C

   Der Begriff A setzt sich logisch aus der Relation zwischen B und C zusammen, etwa:

   $\forall x[bachelor(x) \Rightarrow man(x) \bigwedge unmarried(x)]$

   sinngemäß:„*Ein Junggeselle ist ein Mann und nicht verheiratet.*“

2. *Hyponomy*: A ist ein B

   ```
   Ein Hund ist ein Tier.
   ```

   Also hat ein Hund auch alle Eigenschaften eines Tieres. Auf Basis solchen Wissens kann eine Kategorisierung erfolgen bzw. lassen sich verscheidene Ausprägungen der gleichen Sache zusammengefassen [RS18].

3. *Meronymy*: A $\subseteq$ B, A ist Teil von B

4. *Synonymy*: A $\equiv$ B, A ist semantisch äquivalent zu B

5. *Antonymy*: A $\equiv \neg$B, A bedeutet das Gegenteil von B

Feinheiten der Deutung, wie etwa der Unterschied von „*A semantisch äquivalent zu B*“ zu „*A ist ein B*“ müssen dabei während der Erstellung semantischer Modelle unterschieden werden, da die Kategorisierung sonst zu grob ausfällt. Es können sonst falsche Zusammenhänge und Beziehungen entstehen, die Fehlinterpretationen der Aussagen erzeugen.

In der Praxis stellt sich daher die Frage, in welchem Umfang Semantiken für NLP tatsächlich für eine Verbesserung der Textanalyse sorgen und wie detailliert eine Modellierung dazu erfolgen muss. Mit diesem Ansatz beschäftigen sich die *Ontologien* in Kapitel 2.2.

2.2 Ontologien

Dieser Abschnitt befasst sich genauer mit dem Begriff der Ontologien im Kontext der Informatik.

2.2.1 Definition

Ontologien lassen sich allgemein definieren als „formale, schematische Abbildungen eines Wissensbereichs, bestehend aus einem Vokabular und Regeln zu seiner Zusammensetzung“[WE13].

Zunächst bedeutet dies, dass eine Ontologie stets auf nur Wissen aus einer bestimmten Domäne bezieht und nicht versucht das gesamte über die Welt bzw. sehr allgemeines Wissen abzubilden.

Des weiteren Besteht eine Unterteilung in ein Vokabular und in eine Menge von Regeln.

Einer der Aspekte von Ontologien, welche sie für die Informatik so geeignet machen ist, dass sie in maschienenlesbaren Ontologiesprachen verfasst werden. Somit können Computersysteme diese automatisch auslesen und ggf. interpretieren und daraus schlussfolgern.

Da Terminologie und bestimmte Formalitäten zwischen Ontologiesprachen variieren können, wird im Folgenden der Bezug auf die weit verbreitete Ontologiesprache „Web Ontology Language“ (OWL) angenommen.

2.2.2 Aufbau

Der Aufbau von Ontologien lässt sich grob in zwei Aspekte unterteilen:

1. Das Vokabular: Hier sind alle Dinge innerhalb des Wissensbereiches aufgeführt, welcher von der Ontologie abgedeckt werden soll. Dazu gehören sowohl die Oberbegriffe dieser Dinge, als auch die Instanzen selbst.

2. Die Regeln: Diese beschreiben die Dinge genauer und bilden deren Beziehung untereinander ab. Sie können eigenständig existieren oder im Bezug zu den vorhandenen Dingen stehen.

Klassenhierarchie

Die Hierarchie einer Ontologie, also der strukturelle Aufbau, lässt sich in die folgenden drei Komponenten unterteilen:

1. Begriffe: Die Grundlage für die Struktur einer Ontologie stellt der hierarchische Aufbau von *Begriffen* (*concepts*). Diese Repräsentieren eine Art Klassenstruktur und werden daher oft auch als Klassen bezeichnet. Sie können in aus der Informatik bekannten Ansätzen wie Ober-

und Unterklassen angeordnet werden. Begriffe sind üblicherweise sehr generell gehalten und könnten etwa die Klasse *Auto* abbilden.

2. Typen: In vielen Fällen lässt sich ein Begriff in verschiedene *Typen* aufteilen. Ein Auto kann beispielsweise nach verschiedenen Automarken typisiert werden. Diese Typen werden dann ebenfalls durch Klassen verkörpert und bilden in dem hierarchischen Aufbau eine Unterklasse des jeweiligen Begriffs. Da sowohl Begriffe, als auch Typen in Klassen verkörpert werden, wird in der Praxis oftmals nicht zwischen den beiden unterschieden. Man kann jedoch üblicherweise über einer Unterklasse als Typ der Oberklasse sprechen.

3. Instanzen: Die sogenannten *Instanzen* (*instances*) verkörpern konkrete Objekte dieser Klassen und werden nicht weiter aufgespalten bzw. typisiert. Ein Instanz wäre dann etwa ein bestimmtes Modell, einer bestimmten Marke von Autos.

Diese Hierarchie ist ggf. beliebig erweiterbar. So könnte man für eine bessere Klassifizierung z.B. die Marke eines Autos zunächst in eine Produktreihe oder nach einer Produkteigenschaft wie *Combi* typisieren und erst davon abhängig die verschiedenen Instanzen zuordnen.

In Abbildung X.X ist eine stark vereinfachte Klassenhierarchie aus Oberklassen, Unterklassen und Instanzen beispielhaft dargestellt.

HIER KLASSENHIERARCHIEBILD EINFÜGEN

Auf der obersten Ebene der Klassen-Hierarchie befindet sich die Klasse *owl: Thing*. Von dieser erben alle Begriffe durch die *hasSubclass*-Relation. Dies ist vergleichbar mit der objektorientierten Programmiersprache *Java* bei welcher alle Klassen direkt oder indirekt von der Oberklasse *Object* erben.

Auf der nächsten Ebene befinden sich die beiden Begriffe *Automobil* und *Person*. Diese können entweder direkt in Instanzen übergehen, oder zunächst durch Typisierung unterteilt werden.

Somit finden sich in der nächsten Ebene die drei nach Marke getrennten Automobil-Typen *VW*, *BMW* und *Audi*. Außerdem werden Autofahrer und Ingenieure als Personen unterschieden. Es ist anzumerken, dass es nicht ausgeschlossen ist, das ein Ingenieur auch Autofahren ist. Wegen der möglichen Mehrfachvererbung lassen sich jedoch Typen oder Instanzen erzeugen, welche sowohl von Autofahrer, als auch von Ingenieur erben. Somit stellt eine

Unterteilung in Typen ohne gegenseitigen Ausschluss in der Praxis kein Problem dar und kann sogar nützlich sein, wie später anhand von Relationen gezeigt wird.

In der vierten Ebene sind einzig zu Demonstrationszwecken einige der Typen weiter unterteilt und andere nicht. Man sieht, dass die Hierarchie also keineswegs gleichmäßig sein muss, um eine sinnvolle Unterteilung zu gewährleisten.

In der letzten Ebene befinden sich einige Instanzen von Autos welche durch konkrete Modelle dargestellt werden. Theoretisch ließen sich auch diese weiter unterteilen, etwa nach Innenausstattung. Diesbezüglich ist je nach Verwendungszweck der Ontologie genau festzulegen, wie die Hierarchie zu strukturieren und wie weit sie auszuführen ist. Beispielsweise macht es für einen großen Autohersteller keinen Sinn, wenn in einer konzernübergreifenden Ontologie jeder Ingenieur instantiiert ist.

Regeln

Die Regeln in einer Ontologie lassen sich in die folgenden zwei Komponenten unterteilen:

1. Relationen: Einen wichtigen Bestandteil von Ontologien bilden die sogenannten *Relationen* (*properties*). Relationen beschreiben Beziehungen zwischen Begriffen und werden oft auch als Eigenschaften verwendet und bezeichnet. Ähnlich zu vielen Programmiersprachen können diese Relationen und Eigenschaften eines Begriffs an die Instanzen des Begriffs vererbt werden. Grundsätzlich ist dabei auch eine Mehrfachvererbung möglich.

2. Axiome: Des Weiteren können Ontologien durch sogenannte *Axiome* (*restictions*) genauer modelliert bzw. eingeschränkt werden. Axiome verkörpern dabei Regeln in der Ontologie welche stets zutreffen müssen. Üblicherweise werden diese dazu verwendet, Einschränkungen vorzunehmen, die nicht effizient durch den abgebildeten Wissensbereich darzustellen sind. Dies könnte Beispielsweise in einer Ontologie über verschiedene Automodelle verschiedenster Marken eine generelle Definition von Automobilen sein. So ließe sich die Voraussetzung, dass ein Automobil motorisiert sein muss durch eine Klasse *Bauteil* mit der Unterklasse *Motor* darstellen, welche dann über eine *muss verbaut sein in*-Relation mit der Klasse *Automobil* verbunden ist. Ist es jedoch nicht

Zweck der Ontologie Bauteile genauer abzudecken, wäre dies unnötiger Aufwand, der außerdem die Übersichtlichkeit der Ontologie einschränken könnte.

Abbildung X.X zeigt eine beispielhafte, stark vereinfacht Ontologie zu Automobilen und bestimmten im Kontext relevanten Personen.

HIER RELATIONEN BILD

Die Relationen zwischen den Konten der Ontologie sind dargestellt durch Linien mit einem mittigen Pfeil. Die durchgezogenen Linien zeigen den Grundaufbau der Hierarchie. Die blauen Linien verkörpern die *has subclass*-Beziehung, welche auf Vererbung hinweist. Die violetten Linien stehen für die *has individual*-Beziehung und zeigen auf eine Instanz der Klasse. Außerdem wurden zwei Objekt-Attribute (*object properties*) manuell hinzugefügt. Diese können verwendet werden um Relationen zwischen Klassen aufzuzeigen, welche nicht voneinander erben, sondern sich in unterschiedlichen „Teilbäumen“ der Ontologie befinden. So verweist, dargestellt durch den braunen Pfeil eine *benutzt*-Relation von dem Autofahrer auf das Automobil, bedeutet also das Autofahrer Automobile verwenden. Ähnlich dazu zeigt von der Klasse Ingenieur eine *entwickelt*-Beziehung auf Automobil.

NACHSCHAUEN WEGEN VERERBUNG AUF INSTANZEN.

Verwendung von Ontologien

Wie eingehend bereits erwähnt sind Ontologien vor allem in der Informatik durch ihre Maschinenlesbarkeit wertvoll. Sie sind außerdem auch geeignet komplexere Sachzusammenhänge abzubilden, was mit primitiven Datentypen wie man sie in einer relationalen Datenbank abspeichern kann nicht möglich wäre. Simple Verbindungen wie sie etwa in den in 2.1.4 beschriebenen Bedeutungspostulaten beschrieben sind lassen sind in Ontologien schnell und effizient darstellen.

Betrachte man zum Beispiel das Satzschema „B ist ein A“ so würde dies durch eine einfache *hasSubclass*-Beziehung von der Klasse *A* zu der Klasse *B* dargestellt. Erstellt man nun die beiden Instanzen *b1* und *b2* der Klasse B, so liegen implizit bereits die Informationen vor, dass sie beide auch ein *A* sind. Außerdem ist ihre Homogenität durch die Abstammung von der gleichen Klasse gewährleistet. Daher brauchen die Instanzen hierfür weder eine

Beziehung untereinander, noch zu der Oberklasse A ihrer Elternklasse B.

Grafik mit A,B,b1 und b2 und Tier, Vierbeiner, Hund und Katze

Im Zusammenhang dieser Arbeit sollen später Ontologien dazu verwendet werden den Inhalt betrieblichen Anforderungen aus einem Pflichtenheft abzuspeichern. Pflichtenhefte sind dabei stark domänenbezogen und somit optimal geeignet um in Ontologien abgespeichert zu werden. Des Weiteren lassen sich stets Relationen zwischen den einzelnen Teilbereichen ziehen, was für einen einzelnen Domänenexperten schwierig wäre. Eine üblicherweise vorausgesetzte Eigenschaft für Anforderungen ist Atomarität [QUELLE], was bedeutet, dass jeder Satz nur genau eine Aussage beinhalten soll. Dies hat zur Folge, dass viele betriebliche Anforderungen tatsächlich den vorgestellten, knappen Satzschemata entsprechen.

2.3 Verwandte Arbeiten

Dieses Kapitel stellt einen Überblick über bereits veröffentlichte Arbeiten dar, die sich mit dem Themen Natural Language Processing für Anforderungsmanagement, sowie die Unterstützung durch Werkzeuge beim Requirements-Engineering befassen.

Grundlage dieser Arbeit

K. Zichler und S. Helke stellen in [**?**] und [**?**] die wesentlichen Aspekte für den Entwurf einer Softwareunterstützung für Requirements-Engineering in Firmen vor.

In [**?**] wird zunächst der Bedarf nach solcher Unterstützung besonders für die Automobilindustrie beschrieben, da die Verbesserung herkömmlicher Methoden im Requirements-Engineering dort wesentliches Potential bietet. Der Vorschlag beinhaltet die Unterstützung von Projektteams durch ein Expertenteam aus Requirements-Experten, die Anforderungen aufbereiten und deren konsistente Umsetzung verbessern.

Auf dieser Basis stellen sie in [**?**] konkret einen Entwurf eines Werkzeugs für die automatisierte Analyse von Projektlastenheften vor. Das Werkzeug gliedert sich dabei in eine Komponente zur Überführung natürlichsprachli-

cher Anforderungen in vordefinierte Sprachschablonen, genannt Requirements-to-Boilerplate-Converter (*R2BC*). Auf dieser Basis arbeitet dann der *Delta-Analyser* der die Unterschiede zwischen mehreren Lastenheften, die in Schablonen überführt sind, darstellen kann. Dabei kommen auch Ontologien und NLP-Methoden zum Einsatz. Die Prototypen basieren dabei auf der Open-Source-Software *GATE* zur sprachlichen Auswertung und sollen als 3-Schichten-Architektur umgesetzt werden.

Diese Arbeiten stellen die wesentliche Grundlage für die in unserer Arbeit entworfenen Prototypen dar. Die Implementierung erfolgte in Zusammenarbeit mit dem Autoren Konstantin Zichler bei der Firma HELLA, die den konkreten Anwendungsfall für seine Entwürfe und unsere Umsetzung dargestelt hat.

Es ist anzumerken, dass eine Vielzahl betrachteter Arbeiten das Verbessern von Requirements mithilfe von Schablonen zum Thema haben, sich diese jedoch auf eine Unterstützung bei der Formulierung von Anforderungen beschränken. Eine Umsetzung einer automatisierten Verbesserung und Anpassung bestehender Anforderungen wird häufig zwar umrissen, jedoch selten implementiert bzw. real getestet.

Literatur

In [KR93] untersuchte Kevin Ryan bereits vor modernen Entwicklungen im Bereich NLP in die Möglichkeiten einer Unterstützung von Requirements-Engeneering durch natürlichsprachliche Analyse (NLP). Dabei weist er auf die Vorteile einer formalen Einschränkung bereits bei der Spezifikation von Anforderungen hin, womit die Möglichkeit einer Fehlinterpretation vermindert wird. Ryan stuft jedoch auch die Verifikation von Anforderungen durch NLP als geeignet ein, um Konsistenz und Eindeutigkeit der Anforderungen zu gewährleisten. Dazu generieren sie Schemas aus der Analyse der Anforderungen, die dann inhaltlich mit neu dazukommenden Anforderungen im späteren Entwicklungsprozess abgeglichen werden können. Auch schlägt er die Einteilung besonders von Text im Anforderungsdokument in Kategorien vor, damit die Sichtbarkeit von Anforderungen verbessert wird.

In [**?**] beschäftigen sich Farfelder et. al mit der Unterstützung von Requirements-Experten bei der Anforderungsspezifikation durch vorgefertigte Sprachschablonen und Ontologien, womit Fehler bei der Konsistenz von Anforderungen

vermeiden werden. Sie stellen das Tool *DODT* vor, dass als Alternative zur natürlichsprachlichen Formulierung das Erstellen von Anforderungen anhand von Schablonen anbietet, aus denen der Nutzer im Programm auswählen kann. Dabei greift das Werkzeug auf eine Ontologie zu, welche alle möglichen Begriffe zum Einsetzen in die Schablone enthält. Die Eintragungen des Nutzers stellen dann eine Anforderung dar, die in der Ontologie gespeichert wird.

In [FH14] wird das Vorgehen für die Dokumentation und logische Einordnung von funktionalen Anforderungen mithilfe von Sprachschablonen beschrieben. Dazu werden aus den Anforderungen für jedes Gesamtsystem zunächst die einzelnen Komponenten benannt und einzeln anhand ihrer Eigenschaften gespeichert. Die einzelnen Komponenten des Modells können dann in Beziehung zueinander gesetzt werden. Auf Basis der einzelnen Komponenten und der Beziehungen kann dann ein Netz aus allen Komponenten des Gesamtsystems erstellt werden. Zur Erstellung dieser formalen Modelle mittels verschiedener Sprachschablonen haben sie das Werkzeug *ReqPat* entwickelt, das als Erweiterung in Anforderungsmanagament-Programmen wie *IBM Rational-DOORS* integriert werden kann. Ferner kann zur grafischen Darstellung des erstellten Modells ein Export aus Reqpat in UML-Diagramme erfolgen.

[CA15] beinhaltet einen Vorschlag für das automatisierte Überführen von natürlichsprachlichen Anforderungen in Sprachschablonen mittels NLP. Sie stellen eine Methode vor, die sprachliche Muster in den Anforderungen erkennt und daraus eine Grammatik in Backus-Naur-Form (BNF) erzeugt. Diese Grammatik kann somit zulässige Sätze in einem Anforderungsdokument anhand der Satzbausteine erzeugen bzw. finden. Durch die Übersetzung in eine JAPE-Regel kann auf dieser Basis die Erkennung der gefundenen sprachlichen Regeln durch das NLP-Werkzeug GATE erfolgen, das die Konformität der Anforderungen auf Basis der Regeln prüft.

Heßeler et. al untersuchen in [HE13] formale Prozesse für das Anforderungsmanagement in Unternehmen. Sie zeigen die möglichen Verbesserungen des projektbezogenen Requirements-Engineerings durch die Definition verschiedener Rollen für die beteiligten Entwickler auf. Dabei beschreiben sie explizit Kriterien für die Auswahl von Tools für eine Werkzeugunterstützung der Requirements-Engineers in Firmenr. Konkret beschreiben sie Werkzeuge

für die Unterstützung der Formulierung von Anforderungen bzw. für die Verifikation von Anforderungen mithilfe grafischer Modelle. Das Tool *DOORS Telematic* dient dabei zur Sammlung von Anforderungen und der Verknüpfung durch Relationen in einer Datenbank, die alle Anforderungen hierarchisch enthält. Tau Generation2 wird für die Visualisierung von Sytemeigenschaften in UML2.0-Diagrammen und für den Export als DOORS-Objekte in die Anforderungs-Datenbank genutzt.

Kapitel 3

Betriebliches Umfeld - Hella Use-Case

Im betrieblichen Umfeld liegen zu Beginn jeden Entwicklungsprojektes für neue Produkte die Aufgaben und Ziele für die Entwicklung als Dokumente vor. Forschungsergebnisse finden Anwendung in der Vorentwicklungsphase, in der die Eignung der Erkenntnisse für neue Produkte eines Unternehmens evaluiert wird. Die Produktentwicklung unterliegt dabei bestimmten Kriterien und Faktoren, die den unternehmerischen Erfolg beeinflussen. Neben betriebswirtschaftlichen Einflüssen wie der Einordnung des Produktes in der Wertschöpfungskette sind es dabei besonders technische Anforderungen an das Produkt, die definiert und während der Produktentwicklung eingehalten werden müssen. Verschiedenste Akteure aus einem Umternehmen sind dabei an der Festlegung der Anforderungen an ein Entwicklungsprojekt bzw. Produkt beteiligt.

In der Automobilindustrie betrifft dieser Ablauf zumeist die Entwicklung neuer Fahrzeugkomponenten, heutzutage meist elektronische und mechanische Bausteine. Diese Bausteine werden dabei nicht sämtlich vom Fahrzeughersteller (OEM) selbst, sondern durch eine Vielzahl von Zulieferern produziert und entwickelt. Die Produktspezifikationen liegen meist digital als Texte, Tabellen und Grafiken vor und werden an den Zulieferer übermittelt. Nach dem Entwicklungsprozess steht dann die (Serien-)entwicklung und -fertigung des Produktes für das Ausrollen in großen Stückzahlen an den Hersteller, der das zugelieferte Produkt dann in seinen Produkten verwendet. Um dies zu erreichen, müssen während des gesamten Prozesses die Anforderungen, die das Systemumfeld des Fahrzeugherstellers hat, berücksichtigt

und eingehalten werden.

Die Anforderungen an das Produkt, etwa technische Rahmenbedingungen, werden dabei von vielen verschiedenen Domänenexperten beim OEM formuliert und in das sogenannte Pflichtenheft für die Entwicklung eingetragen. Beteiligte sind etwa Produktdesigner, Ingenieure und Systemtechniker, die an verschiedenen Stellen im Lastenheft Anforderungen an eine Komponente festlegen. Diese Beteiligten sind in der Regel auf ihren Bereich spezialisiert und nicht interdisziplinär, zudem gibt es sprachliche Eigenheiten der Autoren und unternehmensinterne Richtlinien für die Formulierung, die das Verständnis erschweren können. Demzufolge sammeln sich im Lastenheft verschiedenste Merkmale einer Komponente, die aber nicht im Bezug zueinander stehen und sich im schlimmsten Fall gegenseitig ausschließen.

Durch diese fachliche Breite und Tiefe der Spezifikationen im Pflichtenheft, aber auch durch den Umfang des Lastenheftes von mehreren tausend Seiten, kommt es häufig insbesondere zu Verständnisproblemen auf Seite des Zulieferers. Die Gewichtung einzelner Anforderungen in einem größeren Systemkontext fällt dort schwer, da nun Projektteammitglieder, die an der Entstehung des Lastenheftes nicht beteiligt waren, dieses verstehen und ein Produkt entwickeln sollen, dass möglichst alle Anforderungen berücksichtigt. In Texten muss also nach Zusammenhängen und Bezügen zwischen mehreren Anforderungen gesucht werden, damit die Korrektheit des späteren Produktes gewährleistet ist.

Die Analyse von Zusammenhängen zwischen Anforderungen stellt dabei aus Gründen der Effizienz ein Problem dar, wenn jeder Beteiligte von Hand die für ihn relevanten Anforderungen aus dem Lastenheft extrahieren muss. Auch müssen die Lastenhefte an die Formulierungen und Ausdrucksweisen für Requirements-Management im Unternehmen angepasst werden. Bislang gibt es jedoch kaum Werkzeugunterstützung, die effiziente Möglichkeiten zur automatisierten Überarbeitung und Anpassung einzelner Anforderungen aus dem Dokument bietet. Ansätze aus dem *Natural-Language-Proessing* (NLP) stellen gleichzeitig vielversprechende Forschungsfelder in der Informatik dar, die eine solche automatisierte Verarbeitung auf Basis von Sprachanalyse ermöglichen. Syntax und Semantik der einzelnen Sätze und Zusammenhänge in Texten können auf Basis aktueller Trends wie Machine-Learning und dynamischer Programmierung zunehmend besser abgebildet werden.

3.1 Allgemeines zu HELLA

In diesem Abschnitt wird zur besseren Einordnung die Firma HELLA kurz beschrieben. Dies beinhaltet die Geschichte von HELLA und die momentane Aufstellung des Unternehmens. [HE18] [HE19]

3.1.1 Geschichte

Die HELLA GmbH & Co. KGaA wurde 1899 von Sally Windmüller unter dem damaligen Namen „Westfälische Metall-Industrie Aktien- Gesellschaft(WMI)“ in Lippstadt gegründet, wo sich bis heute auch ihr Hauptsitz befindet.

Die Hergestellten Produkte beschränkten sich zu diesem Zeitpunkt auf Ballhupen und Lampen für Kutschen. Im Jahr 1908 tauchte mit dem ersten entwickelten elektrischen Scheinwerfer auch der Name *HELLA* zum ersten mal als Warenzeichen auf. Dieser wurde jedoch erst 1986 offiziell in die Firmenbeschreibung aufgenommen. Es wird davon ausgegangen, dass der Name eine Mischung aus der Anlehnung an den Namen von Windmüllers Frau Helene und dem Wortspiel mit „heller“ ist.

Die erste Tochtergesellschaft gründete HELLA 1951 in Todtnau und dem Namen „Metallwerke Todtnau“. 1961 begann das Unternehmen sich auch international aufzustellen. Dies geschah durch die Gründung der ersten internationalen Produktionsstätte in Mentone in Australien. Das Zentrallager, welches auch heute noch unter dem Namen *HELLA Distribution GMBH* besteht, wurde 1973 in dem Lippstadt nahegelegenen Erwitte errichtet.

3.1.2 Aktueller Stand des Unternehmens

Die HELLA ist in die vier Geschäftsbereiche Elektronik, Scheinwerfer, *Aftermarket* und *Special Applications* unterteilt und verfügt über 125 Standorte in mehr als 35 Ländern.

Nach dem Geschäftsjahr 2017/2018 werden insgesamt über 40.000 Mitarbeiter beschäftigt. Davon arbeiten mehr als 7000 Mitarbeiter in der Forschung und Entwicklung, wodurch sich das Unternehmen mehr Sicherheit und Wachstum für die Zukunft verspricht.

So konnte HELLA in diesem Geschäftsjahr fast 7,1 Milliarden Euro Umsatz verbuchen, wovon ca. ein drittel außerhalb des europäischen Marktes

erzielt wurde. Damit gehört HELLA international zu den Top 40 der weltweiten Automobilzulieferer und befindet sich unter den Top 100 der größten deutschen Industrieunternehmen.

3.2 Requirements-Engineering innerhalb der Firma

pmt abteilung, status quo der prozesse

status quo des re-prozesses, schaubilder des todes interner shit

use-case / problematik im geteilten entwicklungsprozess, schaubilder 2 oem-hella-allesläuftuncremig, schlechte absprache...

3.3 Betriebliche Anforderungen

kriterien der qualität von Anforderungen LITERATUR correctness completeness consistency, talk mit colin, flussdiagramm

Aspekte für den entwurf der werkzeugunterstützung, ziele?

konkrete anforderungen für tool

3.4 Ansatz und Konzept unserer Werkzeuge

Kapitel 4

R2B-Converter

4.1 Architektur Klassen und Verteilung der Ressourcen

4.1.1 Umsetzung der NLP-Architektur in unserem Werkzeug

In diesem Abschnitt wird die Einordnung unseres Prototypen als NLP-Werkzeug in die von [COP04] definierte Architektur behandelt. Die verschiedenen Schritte aus Kap. 2.2. haben wir bei der Analyse und Verarbeitung der Lastenhefte in Programmbestandteilen umgesetzt. Im Arbeitsprozess unserer Entwicklung können wir die Aufgaben unseres Programms dort wie folgt einordnen:

4.2 Implementierung (bisschen Code, GUI, Listenarchitektur, Workflow für User

4.3 mögliche Erweiterungen

4.4 Test

4.4.1 Methodik

4.4.2 Durchführung

4.4.3 Ergebnisse

Kapitel 5

Delta-Analyse

5.1 Architektur

5.2 Implementierung

5.3 mögliche Erweiterungen

5.4 Test

5.4.1 Methodik

5.4.2 Durchführung

5.4.3 Ergebnisse

Kapitel 6

Evaluation

6.1 Auswertung der Testresultate

6.2 Mehrwert?

6.3 Ziel erreicht? Hypothese reviewen und schwafeln

Kapitel 7

Fazit

7.1 Zusammenfassung

7.2 Ausblick

Kapitel 8

Notizen

1. UML-Klassendiagramme im Vergleich zu Ontologien (Struktur, Beziehungen, Informatik-Bezug)
2. Nutzen von Ontologien, welcher Mehrwert kommt zur syntax durch sie hinzu?
3. Betriebliche Anforderungen an Tool: ACID aus DB? Quelle zu Anforderungen an Werkzeuge für Betriebe?

Literaturverzeichnis

[AWK06] Christian Allmann, Lydia Winkler, Thorsten Kölzow. „The requirements engineering gap in the OEM-supplier relationship.“, Journal of Universal Knowledge Management 1.2 (2006): 103-111.

[CA15] Arora, Chetan, et al. „Automated checking of conformance to requirements templates using natural language processing.“ IEEE transactions on Software Engineering 41.10 (2015): 944-968.

[BAL10] Helmut Balzert. „Lehrbuch der softwaretechnik: Basiskonzepte und requirements engineering.“, Springer-Verlag, 2010.

[CAR52] Rudolf Carnap, „Meaning Postulates.“ , Philospohical Studies vol.3 S.65-73, 1952.

[CHO57] Noam Chomsky, „Syntactic Structures“ , Mouton & Co., Feb. 1957.

[COP04] Ann Copestake, University Of Cambridge, „Natural Language Processing“, 2004.

[DGE19] Deutsche Gesellschaft für EMV-Technologie e.V. „CRS Customer Requirements Specification“, aus https://www.demvt.de/publish/viewfull.cfm?ObjectID=ba9a0878_e081_515d_74a3411df6771be8, abgerufen am 09.01.19.

[FF11] Farfeleder, Stefan, et al. „DODT: Increasing requirements formalism using domain ontologies for improved embedded systems development. “ Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2011 IEEE 14th International Symposium on. IEEE, 2011.

[FH14] Fockel, Markus, and Jörg Holtmann. „A requirements engineering methodology combining models and controlled natural language.“

Model-Driven Requirements Engineering Workshop (MoDRE), 2014 IEEE 4th International. IEEE, 2014.

[HAO14] Hao Wu, et al. „ILLINOISCLOUDNLP: Text Analytics Services in the Cloud.“ LREC. 2014.

[HE13] Alexander Heßeler, et al. „Anforderungsmanagement: Formale Prozesse, Praxiserfahrungen, Einführungsstrategien und Toolauswahl.“ Springer-Verlag, 2013.

[HE18] HELLA Geschäftsbericht des Geschäftsjahres 2017/2018, von https://www.hella.com/hella-com/assets/media_global/2018.08.10_HELLA_Geschaeftsbericht_2018_geschuetzt.pdf, aufgerufen am 16.01.2019

[HE19] Firma HELLA. „Unternehmensinformationen in Kürze.“, aus https://www.hella.com/hella-com/de/HELLA-im-Ueberblick-723.html, abgerufen am 09.01.19.

[HL01] Hubert F. Hofmann, Franz Lehner. „Requirements engineering as a success factor in software projects.“IEEE software 4 (2001): 58-66.

[IR09] BGH, „Urteil vom 6. Juli 2000, I ZR 244/97“Artikel beim Institut für Rechtsinformatik von der Universität des Saarlandes, 13. Oktober 2009

[KN85] Kapur, Deepak, and Paliath Narendran. „An equational approach to theorem proving in first-order predicate calculus.“ ACM SIGSOFT Software Engineering Notes 10.4 (1985): 63-66

[KR93] Ryan, Kevin. The role of natural language in requirements engineering.Proceedings of the IEEE International Symposium on Requirements Engineering. IEEE, 1993.

[LV10] F. Langenscheidt, B. Venohr. „Lexikon der deutschen Weltmarktführer: Die Königsklasse deutscher Unternehmen in Wort und Bild.“, Deutsche Standards–Gabal Verlag Google Scholar (2010).

[MW02] Matthias Weber, Joachim Weisbrod. „Requirements engineering in automotive development-experiences and challenges.“, Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on. IEEE, 2002.

[RS18] Felix Ritter, Aaron Schul. „Analyse aktueller NLP-Methoden und -Werkzeuge am Beispiel von GATE, Projektarbeit Fachhochschule Dortmund 2018 “, 18.11.2018

[SB88] Muggleton, Stephen, and Wray Buntine. „Machine invention of first-order predicates by inverting resolution.“ Machine Learning Proceedings 1988. 1988. 339-352

[SHE07] L. Shen, G. Satta, A. K. Joshi. In Meeting of the Association for Computational Linguistics (ACL), „Guided learning for bidirectional sequence classification“ , 2007.

[WE13] Weller, Katrin. Öntologien."(2013): 207-218.