Karlsruhe Institute of Technology

Identifikation problematischer Substantivierungen in natürlichsprachigen Anforderungsdokumenten

Bachelorarbeit
von

Jan Keim

An der Fakultät für Informatik
Institut für Programmstrukturen
und Datenorganisation (IPD)

Erstgutachter:	Prof. Dr. Walter F. Tichy
Zweitgutachter:	Prof. Dr. Ralf Reussner
Betr. Mitarbeiter:	Dipl. Inform.-Wirt Mathias Landhäußer
Zweiter betr. Mitarbeiter:	Dr. Sven J. Körner

Bearbeitungszeit: 17.11.2014 – 16.03.2015

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde.

Die Regeln zur Sicherung guter wissenschaftlicher Praxis im Karlsruher Institut für Technologie (KIT) habe ich befolgt.

Karlsruhe, 13.03.2015

. . . . . .

(**Jan Keim**)

Kurzfassung

Nominalisierungen in natürlichsprachigen Anforderungsdokumenten können potentiell gefährlich für das Gelingen des Projekts sein. Bei dem Einsatz von Nominalisierungen wird häufig zusätzliche Information weggelassen, die aber für das vollständige Verständnis benötigt wird. So ist etwa bei der Nominalisierung *Anmeldung* ohne eine nähere Spezifikation unklar, wo sich wer auf welche Weise anmeldet. Daher ist eine Erkennung von unterspezifizierten Nominalisierungen wichtig, damit eine Fehlinterpretation der Anforderung, was eine potentielle Gefahr für das Scheitern des Projekts darstellt, vermieden wird. Die Fachliteratur rät davon ab Nominalisierungen zu verwenden. Bei der bisherigen Erkennung von Nominalisierungen mit dem Werkzeug RESI wurden ebenso alle Nominalisierungen aufgezeigt. Allerdings ist nicht jede Nominalisierung an sich kritisch. Somit erhielt der Nutzer eine unnötig hohe Anzahl an Warnungen, was die Akzeptanz des Werkzeuges schmälert.

In dieser Arbeit wird eine Kategorisierung von Nominalisierungen vorgestellt, die sie in potentiell gefährliche und ungefährliche Nominalisierungen einteilt. Mit dieser Kategorisierung wird eine Möglichkeit gezeigt, wie unproblematische Nominalisierungen erkannt werden können. Dafür wird mit Hilfe von Ontologien und einem Parser die Nominalisierung und der Satz, in dem sie steht, auf spezifizierende Wörter, Nominalphrasen etc. untersucht.

Mit diesem Ansatz wurden Lastenhefte der Daimler AG untersucht. Das Prüfungskorpus beinhaltete knapp 60.000 Wörter, welches über 1.100 Nominalisierungen beinhaltet. Davon konnte ein Großteil als unproblematisch gefiltert und damit 129 als problematisch identifiziert werden. Durch diesen Ansatz konnte somit eine große Anzahl von Fehlwarnungen unterdrückt werden, wodurch die Akzeptanz beim Nutzer und die Praxistauglichkeit gesteigert wird.

Inhaltsverzeichnis

1 Einleitung **1**

2 Grundlagen **3**
2.1 Requirements Engineering . . . 3
2.1.1 Requirements Engineering im Automotive-Bereich . . . 6
2.1.2 Qualität von Anforderungen . . . 7
2.1.3 Natürliche Sprache . . . 8
2.1.4 Nominalisierungen . . . 9
2.2 Ontologien . . . 10
2.3 Werkzeuge . . . 10
2.3.1 RESI . . . 11

3 Verwandte Arbeiten **13**
3.1 Ansätze zur Verbesserung von Anforderungsdokumenten . . . 14
3.1.1 Unterstützung beim Schreiben . . . 15
3.1.2 Unterstützung beim Überprüfen . . . 16
3.1.2.1 Wortlisten und Wörterbücher . . . 17
3.1.2.2 Ontologien . . . 19
3.1.2.3 Sonstige . . . 20

4 Analyse und Entwurf **23**
4.1 Kategorisierung von Nominalisierungen . . . 23
4.1.1 Kategorie 1 . . . 23
4.1.2 Kategorie 2 . . . 24
4.1.2.1 Nominalphrase . . . 24
4.1.2.2 Spezifizierende Satzteile . . . 24
4.1.3 Kategorie 3 . . . 26
4.1.4 Kategorie 4 . . . 26
4.1.5 Statistiken zur Kategorisierung . . . 26
4.2 Entwurf . . . 28

5 Implementierung **33**
5.1 Formatierer . . . 33
5.2 Automatisierter RESI-Starter . . . 35
5.3 Erkennung unproblematischer Nominalisierungen . . . 36
5.3.1 Wortlisten-Test . . . 36
5.3.2 Parser-Test . . . 36
5.3.3 *CompleteProcessWord*-Test . . . 38

6 Evaluation **41**

7 Zusammenfassung und Ausblick **45**

Literaturverzeichnis **47**

Abbildungsverzeichnis

2.1 Das Wasserfallmodell als Prozessmodell 5
2.2 Das V-Modell als Prozessmodell[DHM98] 5
2.3 Der Ablauf und die Aufgaben von Requirements Engineering und Requirements Management[PD] 6
2.4 Korrelation zwischen Aufwand für RE und Kostenüberschreitung[Par08] . . 6
2.5 Die verschiedenen Arten von Werkzeugen 11

3.1 DB/IS Entwicklungszyklus als beispielhaftes Konzept für einen allgemeinen Entwicklungszyklus [RP92] 17
3.2 Verfügbare Prüfungen im Lastenheft-Analyse-Assistenten nach [Hou14] . . . 21

4.1 Beispiel eines Satzes mit einer Nominalphrase, in der sich eine Nominalisierung, hier *selection*, befindet 25
4.2 Syntaxbaum des Satzes, annotiert durch den Stanford-Parser 25
4.3 Visualisierung der Verteilung auf die verschiedenen Kategorien der Ergebnisse der Analyse mit RESI 28
4.4 Visualisierung der einzelnen Ergebnisse der Analyse mit RESI in Prozent . 29
4.5 Visualisierung der einzelnen Ergebnisse der manuellen Analyse in Prozent . 29
4.6 Ablauf der Überprüfung auf unproblematische Nominalisierungen. Rote Pfeile bedeuten, der Test ordnete die Nominalisierung als nicht spezifiziert ein. Grüne bedeuten, die Nominalisierung wurde als ausreichend spezifiziert erkannt 30
4.7 Abhängigkeiten im Satz nach dem Stanford-Parser 31

5.1 Klassendiagramm des RESIStarters 34
5.2 Ablauf des Starts des Verbesserungsprozess von RESI mit dem RESIStarter 34
5.3 Grafische Benutzeroberfläche von RESI mit den verschiedenen Einstellungsmöglichkeiten 35
5.4 Beispiel für einen Satz mit einer Nominalphrase, die die Nominalisierung *development* zusammen mit dem Elternknoten bildet 37
5.5 Nominalphrase mit einer prep_for-Beziehung zwischen Elternknoten und Nominalisierung 37
5.6 Beispiel für eine Nominalisierung (*selection*), um die eine Nominalphrase gebildet ist 38
5.7 Nominalphrase mit einer prep_by-Beziehung zwischen Nominalisierung und Kindknoten 38
5.8 Fenster des *CompleteProcessWord*-Tests, in dem die Argumente ausgewählt werden können 39

6.1 Darstellung der Anzahl an Warnungen: Vergleich der alten und der neuen Regel mit der Erkennung unproblematischer Nominalisierungen 42
6.2 Darstellung der prozentualen Anzahl an Warnungen auf Nominalisierungen nach der neuen Regel mit Erkennung unproblematischer Nominalisierungen 42

Tabellenverzeichnis

3.1 Auswahl von Werkzeugen und Arbeiten im Requirements Engineering mit den jeweiligen Herangehensweisen 15

4.1 Ergebnisse der manuellen Analyse verschiedener Lastenhefte 27
4.2 Einzelne Ergebnisse der Analyse verschiedener Lastenhefte mit RESI 27

6.1 Evaluation der Implementierung mit verschiedenen Lastenheften. Für die Ausbeute wurde angenommen, dass die von RESI bei 100% liegt. 43

1. Einleitung

Gute Anforderungen sind entscheidend für das erfolgreiche Entwickeln und Herstellen von Produkten. Dementsprechend können schlechte Anforderungen dies gefährden und sollten daher frühzeitig erkannt und verbessert werden. Die meisten Anforderungen werden in natürlicher Sprache geschrieben [Hei10][LMP04], was neben fachlichen Mängeln die Gefahr von sprachlichen Defekten bringt. Viele Arbeiten und Werkzeuge, welche in Kapitel 3 beschrieben werden, beschäftigen sich daher mit den verschiedenen Defekten in natürlichsprachigen Anforderungsdokumenten.

Nominalisierungen können zu diesen potentiell gefährlichen Defekten zählen, da sie oftmals einen Interpretationsspielraum offen lassen, der vermieden werden sollte. Nominalisierungen im Sinne der Requirements Engineering sind substantivierte Prozessverben, wie das Wort *Anmeldung* im Satz „Nach der Anmeldung wird der Benutzer zu seiner persönlichen Seite weitergeleitet“ [Kör14]. Aus der Gefahr heraus haben Rupp und die SOPHISTen eine Bauernregel aufgestellt: „Will der Stakeholder die Anforderung verderben, macht er Substantive aus den Verben“[RS09]. Sie raten von der Benutzung von Nominalisierungen ab. Außer, man ist sich sicher, dass sie keine Probleme verursachen können. Daher sind nicht alle Nominalisierungen problematisch, da sie ausreichend spezifiziert und somit eindeutig sein können. So wird beispielsweise die Nominalisierung aus obigen Beispiel in folgendem Halbsatz ausreichend spezifiziert: „Nach der Anmeldung des Benutzers über das Login-Formular mit Nutzernamen und Passwort...“. Deshalb wird in dieser Arbeit untersucht, wie problematisch Nominalisierungen in natürlichsprachigen Anforderungsdokumenten sind und wie man die unproblematischen automatisiert erkennen kann.

Der *Requirements Engineering Specification Improver*, kurz RESI, ist ein Werkzeug zur Verbesserung von Anforderungen und wird in Kapitel 2.3.1 näher beschrieben. Bei RESI gibt es noch Verbesserungspotenzial. Vor allem bei der Erkennung von Nominalisierungen wird dies deutlich, da momentan lediglich alle Wörter angezeigt werden, die eine Nominalisierung sind. Es wird nicht unterschieden oder bewertet, ob und wie kritisch die Nominalisierung ist. Dadurch kommt es zu einer hohen Rate an unnötigen Warnungen. Ein Werkzeug wird aber oftmals nicht von den Nutzern akzeptiert, wenn die Rate an Falschmeldungen aus Sicht des Anwenders zu hoch ist. Ziel dieser Arbeit ist es, die Rate an Falschmeldungen mittels der Erkennung unproblematischer Nominalisierungen zu senken und nur noch (potenziell) problematische Nominalisierungen anzuzeigen.

Dafür wird in dieser Arbeit mit Hilfe der Analyse von Lastenhefte der Daimler AG eine Kategorisierung der Nominalisierungen in vier Kategorien erstellt, die in Kapitel 4.1 näher

beschrieben wird. Kategorie 1-3 stellen dabei ausreichend spezifizierte Nominalisierungen dar, während die letzte Kategorie die unterspezifizierten Nominalisierungen enthält. Die weitere Analyse ergab außerdem, dass die meisten Nominalisierungen in der Praxis zu den unproblematischen Kategorien gezählt werden können. In Kapitel 4.2 wird eine Möglichkeit dargestellt, wie diese unproblematischen Nominalisierungen automatisiert erkannt werden können.

Die Implementierung in Kapitel 5 zeigt, wie der Entwurf umgesetzt wurde. In der Evaluation in Kapitel 6 wurden 10 Anforderungsdokumente der Daimler AG mit insgesamt knapp 60.000 Wörtern untersucht. Das Kapitel 7 gibt eine Zusammenfassung der Arbeit und einen Ausblick auf zukünftige Arbeiten.

2. Grundlagen

Um ein Produkt entwickeln und herstellen zu können, muss man dieses zuerst spezifizieren. Es muss spezifiziert werden, was genau am Ende entstehen soll und was das Produkt nach den Anforderungen des Auftraggebers können soll. Hierbei ist es in der Regel auch hilfreich, Grenzen abzustecken. Somit kann sichergestellt werden, dass auch wirklich das Produkt entsteht, das sich der Auftraggeber vorgestellt hat.

2.1 Requirements Engineering

Damit gewährleistet werden kann, dass das Produkt korrekt entwickelt und hergestellt wird, gibt es Anforderungen, die in Anforderungsdokumenten wie Lastenheften festgehalten werden. Nach IEEE Standard 610.12 [IEE90] ist eine Anforderung:

1. Eine Bedingung oder Fähigkeit, die von einem Benutzer benötigt wird, um ein Problem zu lösen oder ein Ziel zu erreichen.
2. Eine Bedingung oder Fähigkeit, die ein System oder eine Systemkomponente erfüllen oder besitzen muss, um einen Vertrag, eine Norm, eine Spezifikation oder andere, formell vorgegebene Dokumente zu erfüllen.
3. Eine dokumentierte Repräsentation einer Bedingung oder Eigenschaft nach 1. oder 2.

Eine weitere Definition einer Anforderung stammt von den SOPHISTen: „Eine Anforderung ist eine Aussage über eine Eigenschaft oder Leistung eines Produktes, eines Prozesses oder der am Prozess beteiligten Personen“[RS09]. Anforderungen decken also sowohl das Produkt an sich als auch den kompletten Entwicklungsprozess ab.

Bei der Disziplin der Anforderungsermittlung und Anforderungsverwaltung (*Requirements Engineering*) beschäftigt man sich mit ebendiesen Anforderungen und den *Stakeholdern*, die damit zu tun haben. Ein *Stakeholder* ist eine Person, die direkten oder indirekten Einfluss auf die Anforderungen hat [RS09]. Als Beispiel für *Stakeholder* kann man Auftraggeber, Entwickler und Tester nennen.

Das *International Requirements Engineering Board (IREB)* definiert dabei die Disziplin des Requirements Engineering wie folgt: “ Das Requirements Engineering ist ein systematischer und disziplinierter Ansatz zur Spezifikation und zum Management von Anforderungen mit den folgenden Zielen:

1. Die relevanten Anforderungen zu kennen, Konsens unter den Stakeholdern über die Anforderungen herzustellen, die Anforderungen konform zu vorgegebenen Standards zu dokumentieren und die Anforderungen systematisch zu verwalten.
2. Die Wünsche und Bedürfnisse der Stakeholder zu verstehen, zu dokumentieren sowie die Anforderungen zu spezifizieren und zu verwalten, um das Risiko zu minimieren, dass das System nicht den Wünschen und Bedürfnissen der Stakeholder entspricht.“ [PR11]

Somit ist die Disziplin des Requirements Engineerings sowohl für das Erheben und Dokumentieren von Anforderungen zuständig als auch für das Prüfen und Verwalten. Außerdem kann man aus dem zweiten Punkt der Definition des *IREB* herauslesen, dass der Anforderungsingenieur (*Requirements Engineer*) auch als Mittler zwischen den verschiedenen *Stakeholdern* tätig ist, da er nach der Definition „Wünsche und Bedürfnisse der *Stakeholder* [...] verstehen“[PR11] und die Anforderungen so spezifizieren muss, dass das System alle *Stakeholder* möglichst zufrieden stellt. Deshalb ist das Requirements Engineering eine eigene Disziplin und für die Entwicklung von Produkten essentiell.

Es findet in jedem Projekt Requirements Engineering statt. Das Requirements Engineering kann aber auf verschiedene Arten und Weisen und damit unterschiedlich professionell durchgeführt werden. Die Qualität der Anforderungen hat direkten Einfluss auf nachfolgende Phasen in der Entwicklung des Produkts. Deutlich wird das zum Beispiel durch das *Wasserfallmodell* als Vorgehensmodell für insbesondere die Softwareentwicklung, das in der Abbildung 2.1 zu sehen ist. Nach der Ermittlung der Anforderungen in der Planungsphase wird aus diesen der Entwurf erstellt und daraufhin implementiert, bevor die Implementierung getestet und später gewartet wird. Je früher ein Fehler unbemerkt auftritt, desto weitreichender können auch die Folgen sein und das Projekt gefährden. Da die Anforderungsermittlung stets ganz zu Beginn des Projekts stattfindet, geht hier eine besonders große Gefahr für das Projekt aus, da sich ein Fehler durch alle folgenden Phasen fortpflanzt. Auch beim V-Modell, das in Abbildung 2.2 zu sehen ist, findet zu Beginn die Planung und Definition statt. Die Weiterentwicklung des V-Modells, das V-Modell XT, ist sogar Entwicklungsstandard für IT-Systeme des Bundesministeriums des Innern (BMI)[Bun]. Auch hier finden die Planungen stets zu Beginn einer Phase statt. Das Requirements Engineering soll hier helfen, Fehler und deren Fortpflanzung in nachfolgende Phasen möglichst zu vermeiden.

In Abbildung 2.3 sieht man die Aufgabenfelder und den Ablauf bei der Anforderungsermittlung und -verwaltung. Die Anforderungen werden beim Requirements Engineering ermittelt, analysiert, spezifiziert und dann validiert, bis sie qualitativ hochwertig genug sind. Das Requirements Management kümmert sich um die Organisation und Verwaltung dieses Prozesses und der Anforderungsdokumente. Hier sind diese beiden Teilgebiete klar getrennt, allerdings ist nach der Definition des Requirements Engineering, wie oben bereits geschrieben, ebenso wie nach Meinung von Rupp et al. in [RS09], das Requirements Management ein Teilgebiet des Requirements Engineering. Die Trennung dient hier dem Verständnis und der Differenzierung der Teilaspekte dieser Disziplin.

Ein Aufgabenbereich des Requirements Engineerings ist das Prüfen und Verbessern von Anforderungsdokumenten. Idealerweise folgt auf bessere Anforderungen eine bessere Dokumentation der Anforderungen und daraus dann letztlich ein besseres Produkt. Die erste Phase des Projekts, das Ermitteln von Anforderungen und Erstellen der Anforderungsdokumente sowie Analyse, ist nach Partsch eine Schlüsselphase [Par08]. Fehler können hier noch sehr einfach und schnell – und damit kostengünstig – erkannt und beseitigt werden. Geschieht das nicht, können sich Fehler wie in einem Schneeballsystem potenzieren. Nach Rupp lassen sich ca. 65% der schwerwiegenden Fehler in Programmen auf Unzulänglichkeiten in der Analyse zurückführen [RS09]. Partsch ist ebenso der Meinung, dass die meisten

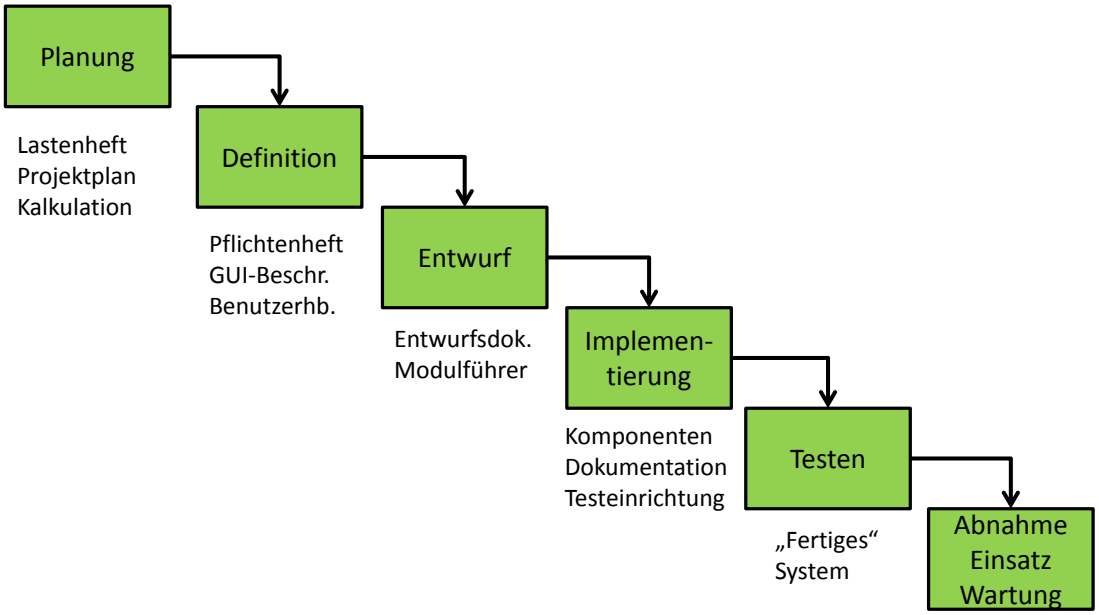

Abbildung 2.1: Das Wasserfallmodell als Prozessmodell

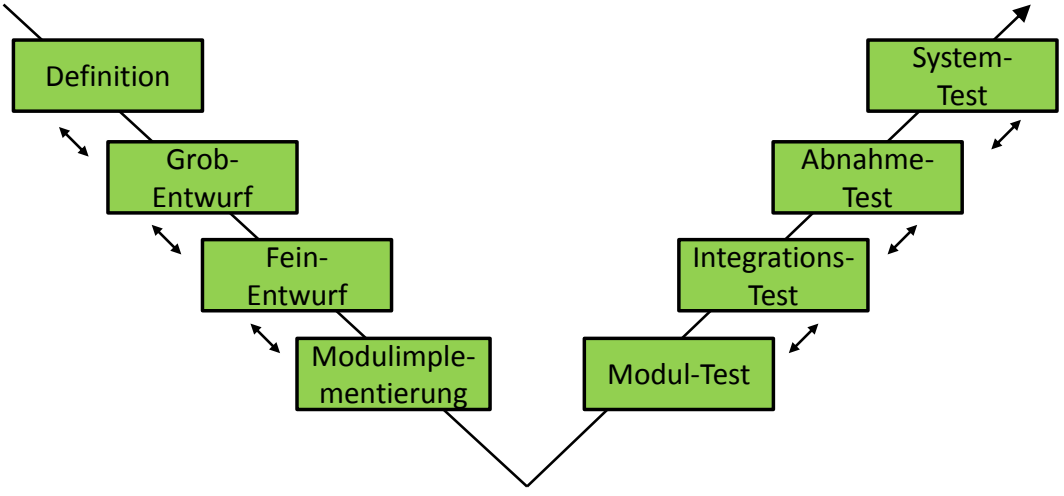

Abbildung 2.2: Das V-Modell als Prozessmodell[DHM98]

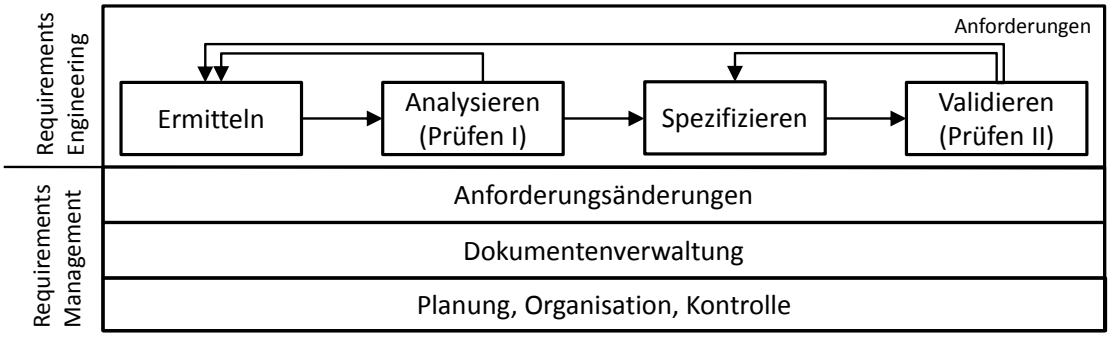

Abbildung 2.3: Der Ablauf und die Aufgaben von Requirements Engineering und Requirements Management[PD]

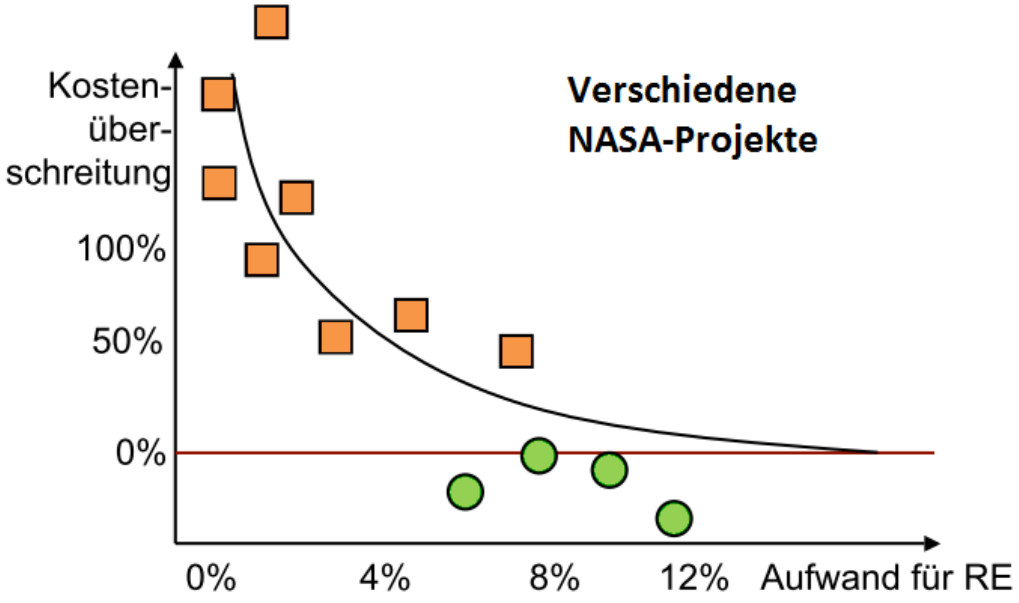

Abbildung 2.4: Korrelation zwischen Aufwand für RE und Kostenüberschreitung[Par08]

Fehler in softwarebasierten und vermutlich auch in anderen Systemen ihren Ursprung in Anforderungen haben [Par08].

Dadurch wird klar, dass sich durch das frühe Beseitigen von Fehlern Zeit und Geld sparen lässt. Abbildung 2.4 zeigt die Korrelation zwischen dem Aufwand für das Requirements Engineering und der Kostenüberschreitung bei Projekten der NASA. Deutlich zeichnet sich ab, dass sich Mehraufwand im Requirements Engineering lohnt. Heidenreich schreibt sogar, dass die Fehlerfolgekosten in frühen Phasen im Vergleich zur Behebung von Fehlern erst in der Marktphase um bis zu 90% niedriger sein können [Hei10].

2.1.1 Requirements Engineering im Automotive-Bereich

Betrachtet man im konkreten Fall dieser Arbeit die Automobilbranche näher, stellt man fest, dass in Autos sich heute viele unterschiedliche Systeme und Geräte befinden, die direkt oder indirekt miteinander verbunden sind. Das Ausfallen eines dieser Systeme kann große Auswirkungen auf das gesamte Auto haben.

Ein aktuelles Beispiel dafür, welche Auswirkungen Fehler in der Entwicklung haben können, liefert General Motors. Im Februar 2014 rief das Unternehmen die ersten Autos auf Grund eines defekten Zündschlüssels zurück. Dieser konnte bei der Fahrt in die *Aus*-Position springen. Der Vorfall weitete sich aus und eine Rückrufaktion von mehreren Millionen Autos und dementsprechend hohe Kosten im Milliardenbereich entstanden. Schwerwiegend ist außerdem, dass durch diesen Defekt mindestens 42 Menschen starben und noch weitere Personen in Unfällen verletzt wurden.

Selbst ohne Verletzte oder Tote ist bei einem Ausfall eines Systems in der Regel eine Rückrufaktion die Folge, die in mehreren Hinsichten schlecht für das Unternehmen ist, da damit u.a. Kosten und ein Verlust an Reputation einhergehen. Requirements Engineering soll hier helfen gute Systeme zu spezifizieren und die Interaktion zwischen Systemen zu normieren und damit zu stabilisieren.

Ein weiterer Punkt für das Requirements Engineering ist, dass Anforderungsdokumente oftmals, auch außerhalb des Automotive-Bereichs, als Vertragsdokumente beispielsweise gegenüber Zulieferern eingesetzt werden. Nach DIN69901 ist eine Hauptaufgabe der Lastenhefte die Nutzung in Ausschreibungen und als Vertragsdokument [Deu09]. So heißt es unter anderem, dass Lastenhefte die Gesamtheit der Forderungen an die Lieferungen und Leistungen innerhalb eines (Projekt-)Auftrags seien. Gerade für öffentliche Aufträge sind Ausschreibungen auch verpflichtend.

Die Qualität der Anforderungsdokumente hat dadurch direkte juristische Implikationen, da diese Dokumente im Streitfall vor Gericht standhalten müssen. So stehen am Anfang jedes Lastenhefts der Daimler AG Sätze, die auf die rechtlichen Verbindlichkeiten hinweisen. Einer dieser Sätze ist der folgende: „Das vorliegende Lastenheft und alle weiteren hier genannten Dokumente bilden zusammen die Grundlage des durch den Auftragnehmer zu erbringenden Leistungsumfangs“. Auch innerhalb der Anforderungsdokumente finden sich viele vertragliche Regelungen und Verpflichtungen für beide Vertragsseiten. Dies ist ein weiterer Grund, warum Anforderungsdokumente und deren Qualität wichtig sind und daher das Requirements Engineering eine wichtige Disziplin ist.

2.1.2 Qualität von Anforderungen

Damit Anforderungen qualitativ hochwertig sind, müssen sie verschiedene Kriterien erfüllen: Die Definition von Anforderungen unterscheidet zwischen Anforderungen und dokumentierten Anforderungen. Das bedeutet, dass es Anforderungen gibt, die ein System nach Wünschen von *Stakeholdern* haben soll und sogar essentiell für die Funktionalität eines Systems sein können, aber nicht in Dokumenten etc. festgehalten wurden. Zum großen Problem werden solche unvollständigen Anforderungen, wenn auf Grund der nicht ausreichend dokumentierten Anforderungen das Projekt fehlschlägt. Ebenso sind Änderungen später im Projekt häufig kompliziert und teuer, da dadurch auch weitere Änderungen in der Produktspezifikation oder in Tests folgen können[Ben83].

Bei Anforderungen ist es somit wichtig, dass diese klar, eindeutig und präzise sind und möglichst keinen Interpretationsspielraum lassen. Falsch interpretierte Anforderungen führen unweigerlich zu Problemen. Interpretationsspielraum entsteht unter anderem durch mögliche Unterschiede zwischen gewünschten und tatsächlich dokumentierten Anforderungen. Damit möglichst kein Interpretationsspielraum existiert und auch andere Probleme minimiert werden, hat das IEEE Qualitätskriterien für Anforderungsspezifikationen aufgestellt, die von Rupp und den SOPHISTen [RS09] für einzelne Anforderungen angepasst und erweitert wurden. Demnach müssen Anforderungen wie folgt sein:

- Vollständig
- Korrekt
- Abgestimmt
- Klassifizierbar
- Konsistent
- Prüfbar
- Eindeutig
- Verständlich
- Gültig und aktuell
- Realisierbar
- Notwendig
- Verfolgbar
- Bewertet

Interpretationsspielraum entsteht oft durch die unterschiedliche Wahrnehmung der verschiedenen Personen auf Grund von Erfahrung und Wissen. Beim Formulieren von Anforderungen, besonders wenn dafür die natürliche Sprache verwendet wird, findet ein Transformationsprozess statt. Die Realität wird von einer Person auf ihre eigene Weise wahrgenommen und dann versucht sprachlich darzustellen. Dabei ist die Person durch das eigene Wissen limitiert. Durch diesen Prozess wird die Realität nun transformiert, wodurch unter anderem Informationen verloren gehen können.

Bei der Darstellung der (wahrgenommenen) Realität gibt es nach Rupp und den SOPHISTen in [RS09] drei Arten der Transformation. Zum einen gibt es die Tilgung, bei der Informationen unvollständig sind und Details weggelassen werden, da diese als selbstverständlich angesehen werden. Als zweites existiert das Problem der Generalisierungen. Hierbei ist die Gefahr, dass durch zu starke Verallgemeinerungen Anforderungen zu allgemein werden und sich dadurch statt auf nur einen Teil des Systems auf das gesamte System auswirken. Insbesondere bei Sonderfällen kann dies gefährlich sein. Die letzte Transformation ist die Verzerrung. Dabei wird die Realität – meist für das bessere Verständnis – so umgeformt, dass jemand anderes die Informationen nicht korrekt versteht. Dabei geht im Allgemeinen, wie bei der Tilgung, Information verloren, die essentiell sein kann.

Dies alles führt letztlich dazu, dass Anforderungen falsch verstanden werden können und somit das Projekt gefährdet wird. Daher haben die SOPHISTen auch in [RS09] ein Regelwerk veröffentlicht, mit dem Anforderungen „therapiert“ werden und somit bessere Anforderungsdokumente entstehen können.

2.1.3 Natürliche Sprache

Natürliche Sprache ist die Art und Weise, wie wir uns alltäglich unterhalten und textuell kommunizieren. Jeder *Stakeholder* kann Anforderungen in natürlicher Sprache lesen oder erstellen, ohne bestimmte Techniken erlernen zu müssen. Heutzutage sind ein Großteil der Anforderungsdokumente natürlichsprachig verfasst, trotz des großen Angebots an formalen Techniken[SDGDG13][Hei10]. Eine Marktstudie von Luisa et al. ermittelte beispielsweise, dass 95% der Anforderungsdokumente in natürlicher Sprache geschrieben sind [LMP04].

Natürliche Sprache schränkt nicht ein und erlaubt Freiheiten beim Schreiben der Anforderungen. Diese Freiheit bringt aber Probleme mit sich, denn Anforderungen können neben fachlichen Mängeln auch sprachliche besitzen. Viele Mehrdeutigkeiten und andere Mängel entstehen durch mangelnden Fokus und unzureichende Sorgfalt bei der Formulierung. Man muss darauf achten, sprachliche Fehler zu vermeiden, damit Anforderungen die oben genannten Qualitätskriterien erfüllen. Daher sollten Anforderungen immer überprüft werden und, wie Rupp vorschlägt, „therapiert“ werden[RS09].

Zur Vermeidung der Probleme natürlicher Sprache gibt es viele Techniken und Formalismen. Diese formalen Techniken sind beispielsweise Schablonen, nach denen die Anforderungen formuliert werden können. Da diese einem festen Muster folgen, sind sie oftmals

leichter zu digitalisieren und mit automatischen Prozessen zu überprüfen und weiter zu verarbeiten. Beispielsweise kann aus Anforderungen automatisch Quellcode erzeugt werden [Par08]. Allerdings besitzen automatisierte Prozesse den Nachteil, dass sie die Freiheiten bei der Formulierung stark einschränken und manche Anforderungen dadurch gar nicht oder nur umständlich formuliert werden können.

Formale Techniken müssen erlernt werden. Dadurch werden eventuell *Stakeholder* ausgeschlossen. Es gibt allerdings auch geeignete, weniger formelle Schablonen für die Erstellung von natürlichsprachigen Anforderungen, wie beispielsweise die Schablonen für die Erstellung natürlichsprachiger Anforderungen der SOPHISTen [RS09]. Dadurch kann man zum einen ein wenig mehr Sicherheit hineinbringen, auf der anderen Seite aber auch einfach zu verstehende Anforderungen schreiben.

2.1.4 Nominalisierungen

In dieser Arbeit soll untersucht werden, wie unproblematische Nominalisierungen erkannt werden können. Dafür ist wichtig zu verstehen, was genau Nominalisierungen sind. Nach der allgemeinen linguistischen Definition sind Nominalisierungen, oder auch Substantivierungen genannt, substantivisch gebrauchte Worte einer anderen Wortart wie Verben oder Adjektive[Dud]. Im Requirements Engineering sind nach Rupp[RS09] Nominalisierungen jedoch lediglich diejenigen Substantive, die aus einem substantivierten Verb entstehen und einen komplexen Prozess zusammenfassen. Ein Beispiel ist das Verb *anmelden*, das zur Nominalisierung *Anmeldung* wird. Solche Nominalisierungen findet man, indem man bei Substantiven hinterfragt, welcher Prozess hinter diesem Wort stehen könnte. Als Beispiel für eine Nominalisierung präsentiert Körner in [Kör14] den folgenden Satz mit der Nominalisierung *Anmeldung*: „Nach der Anmeldung wird der Benutzer zu seiner persönlichen Seite weitergeleitet.“

Das Problem bei Nominalisierungen ist die Tilgung von Informationen, also der Verlust von wichtigen Details. Nominalisierungen fassen komplexe Prozesse zusammen, dabei kann wichtige Information verloren gehen, ohne die die Anforderung nicht ausreichend spezifiziert ist und zu Problemen führen kann. [SOP14] Nicht vollständig spezifizierte Nominalisierungen lassen einen zu großen Interpretationsspielraum, der verhindert werden sollte.

In dem obigen Satz könnte ein Problem bei der Nominalisierung *Anmeldung* existieren. Zumindest alleine in diesem Satz wäre das Wort nicht ausreichend spezifiziert. So bleiben beispielsweise die Informationen verborgen, wer daran beteiligt ist, wie die Anmeldung funktionieren soll, etc. Wenn man eine solche Nominalisierung auflösen will, muss man nicht zwangsweise die Nominalisierung als Vollverb ausdrücken. Alternativ kann man im Satz oder in zusätzlichen Anforderungen die Nominalisierung weiter spezifizieren. Verbessern könnte man diesen Satz also zum einen durch das Erstellen einer weiteren Anforderung, in der die Anmeldung genau beschrieben ist. Auf Satzebene könnte man die Anforderung verbessern, indem man die Nominalisierung in ein Verb umwandelt oder die nötigen Informationen einfügt. Durch letzteres könnte der Satz nach [Kör14] wie folgt aussehen: „Nachdem sich der Benutzer durch Eingabe von Nutzername und Passwort angemeldet hat, wird er zur persönlichen Seite weitergeleitet.“

Da weitere Anforderungen oder andere Wörter im Satz eine Nominalisierung auch spezifizieren können und diese dann unproblematisch ist, ist folglich auch nicht jede Nominalisierung an sich kritisch. Bei Nominalisierungen muss man daher erst hinterfragen, ob alle benötigten Informationen im Satz vorhanden sind oder es Anforderungen gibt, die die Nominalisierung weiter spezifizieren könnten.

2.2 Ontologien

Der Begriff *Ontologie* stammt ursprünglich aus der Philosophie. Die philosophische Ontologie ist die Lehre vom Sein und ist der allgemeinen Metaphysik zuzuordnen. Darin geht es unter anderem um Erkenntnisse über Attribute und Wissen über die Dinge an sich [SS09][Hes02].

Der philosophische Begriff ist dadurch dem informatischen Begriff gar nicht so unähnlich. In der Informatik werden Ontologien dazu verwendet, Wissen in Form von Daten ähnlich wie in einer Datenbank zu speichern. Dadurch kann beispielsweise Domänenwissen wiederverwendet werden.

In Ontologien existieren Begriffe (*concepts*) und Regeln. Zwischen den Begriffen und Regeln gibt es Beziehungen (Relationen). All dies kann auch vererbt werden, wodurch eine Vererbungshierarchie entsteht. Als Ergebnis erhält man eine große Wissensdatenbank, in der die einzelnen Daten ähnlich wie in einem Graphen miteinander verbunden sein können.

Dadurch wird eine Struktur erzeugt, die auch selbstständig auf Konsistenz prüfen und sogar fehlendes Wissen ergänzen kann, indem durch logisches Folgern (*Inferenz*) Informationen erzeugt werden.

Gerade beim *Natural Language Processing*, also der Verarbeitung von natürlicher Sprache, sind Ontologien sehr beliebt, da durch sie Informationen über Semantik und Beziehungen zwischen Wörtern ermittelt werden können. Damit kann beispielsweise die Zuordnung von Wörtern zu Domänen stattfinden oder ermittelt werden, welche Wörter in direkter Verbindung zu den anderen Wörtern im Satz stehen können.

Eine bekannte Ontologie ist Cyc [Cyca] bzw. der wissenschaftliche Ableger ResearchCyc [Cycb]. Diese allgemeinen, bereichsübergreifenden (*top-level*) Ontologien zählen zu den umfangreichsten Datenbanken für Allgemeinwissen [Kör14]. ResearchCyc enthält unter anderem das WordNet, eine Datenbank, die semantische und lexikalische Beziehungen zwischen Wörtern der englischen Sprache beinhaltet [Pri]. Dadurch eignen sie sich auch sehr gut für die Akquise von Wissen über Wörter für die Verarbeitung von Sprache.

Ontologien wie ResearchCyc können somit nützlich sein, da dadurch Wörter in den Ontologien mitsamt den Informationen und Relationen benutzt werden können. Somit kann man beispielsweise ermitteln, ob ein Wort in seiner aktuellen Bedeutung allgemein eine Nominalisierung sein kann.

2.3 Werkzeuge

Der Vorteil von Werkzeugen gegenüber dem manuellen Überprüfen der Anforderungsdokumente ist die Automatisierung. Diese unterstützt den Benutzer in seiner Arbeit durch Vorschläge und Verbesserungen. Viele Werkzeuge überlassen die finale Entscheidung aber oft beim menschlichen Benutzer. Somit können sie auf mögliche Fehler hinweisen, während der Benutzer gleichzeitig kontrollieren und entscheiden kann, ob dieser Fehler tatsächlich Probleme verursachen kann und damit geändert werden muss.

Ein Problem bei vielen Projekten ist der Zeitdruck. Dies belegt unter anderem eine Studie [LLP10] aus China von Liu et al., die in Kapitel 3 näher ausgeführt wird. Oftmals werden Entscheidungen auf Grund des Mangels an Zeit getroffen, die mit mehr Zeit vermutlich anders getroffen worden wären.

Außerdem lässt der Zeitdruck auch oft Stress entstehen und Fehler werden dabei übersehen. Lami spricht auch davon, dass manuelle Überprüfung zeitraubend, ermüdend und oft ineffektiv ist [Lam05]. Um die Konzentration zu erhalten, müssen Pausen eingelegt werden.

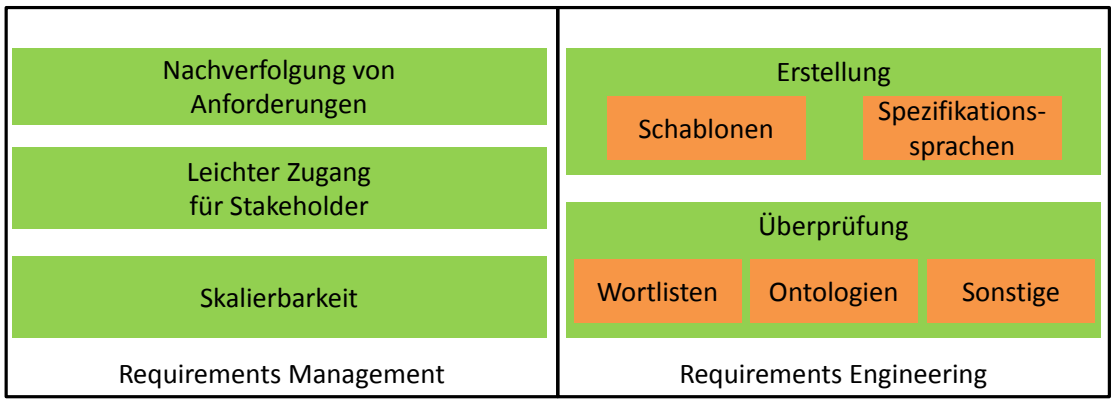

Abbildung 2.5: Die verschiedenen Arten von Werkzeugen

Dadurch kommt es zu einer längeren Überprüfung, was gerade bei engen Zeitplänen problematisch ist. Die Alternative beinhaltet allerdings das Risiko des Konzentrationsabfalls, was in einer geringeren Qualität der Anforderungen resultiert, da Mängel übersehen werden können. Houdek sieht die manuelle Überprüfung als sehr aufwändig und fehleranfällig an, wodurch automatisierte Prüfungen motiviert werden [Hou14].

Den genannten Problemen kann mittels Einsatz von Werkzeugen vorgebeugt werden. Durch sie kann zum einen die Qualität gesteigert werden, zum anderen ist auch eine Zeitersparnis ein Ziel der Werkzeuge.

Werkzeuge können in verschiedene Kategorien eingeteilt werden, wie es die Abbildung 2.5 aufzeigt. Eine Art der Werkzeuge sind solche, die den Benutzer beim Organisieren und Verwalten von Anforderungsdokumenten unterstützen und somit dem Requirements Management zugeordnet werden können. Für die Unterstützung beim Erstellen von Anforderungen gibt es Ansätze, die mit Hilfe von Schablonen oder speziellen Spezifikationssprachen versuchen, das Erstellen der Anforderungen zu lenken und damit die Qualität zu steigern.

Natürlichsprachige Anforderungen werden jedoch meist durch Werkzeuge verarbeitet, die die Anforderungen überprüfen sollen. Eine Art des Ansatzes für das Überprüfen sind Wortlisten. Mit Hilfe von Wortlisten (*dictionaries*) werden die Anforderungen untersucht. Findet sich ein Wort der Wortliste in einer Anforderung wieder, kann aufgrund von zusätzlichen Informationen aus der Wortliste das Wort und damit die Anforderung weiter verarbeitet werden. Andere Ansätze setzen auf die Benutzung von Ontologien, um zusätzliche Informationen zu erhalten und daraufhin die Anforderungen zu analysieren und auf mögliche Probleme hinzuweisen. Neben diesen beiden Arten gibt es noch weitere Ansätze wie bspw. eine statistische Verarbeitung von Anforderungsdokumenten aufgrund der Domäne oder das Auswerten von Ergebnissen aus dem *Part of Speech(POS)-tagging* oder *parsing*, wodurch syntaktische Informationen und ähnliches von Satzteilen ermittelt werden.

Beispiele für die jeweiligen Ansätze sind unter anderem in Kapitel 3.1 zu finden.

2.3.1 RESI

Der *Requirements Engineering Specification Improver*, kurz RESI, ist ein Werkzeug zur Verbesserung von Anforderungen [Kör14][KB09]. Es unterstützt den Anwender dabei, mögliche Schwachstellen in Anforderungen zu ermitteln.

RESI benutzt für diese Arbeiten unter anderem diverse Ontologien wie Cyc [Cyca], um an das nötige Wissen zu kommen. Dabei ist entscheidend, dass RESI von der Güte der Ontologien abhängt.

Momentan kann RESI auf einige Mängel hinweisen. Dazu zählt unter anderem das Aufzeigen von Mehrdeutigkeiten, Nomen ohne Referenzierung und falsch genutzte Mengenwörter. Außerdem ist RESI in der Lage, auf Wörter mit ähnlicher Bedeutung hinzuweisen, also auf verschiedene Wörter, die aber vermutlich für den gleichen Begriff stehen. Letztlich kann auch auf unvollständig spezifizierte Prozessverben und Nominalisierungen hingewiesen werden. Bei den Prüfungen werden auch nicht nur Hinweise gegeben, sondern, wenn möglich, auch ein Vorschlag für die Verbesserung der Anforderung gemacht.

RESI geht dabei wie folgt vor: Zuerst wird das Anforderungsdokument als reine Textdatei eingelesen. Nach dem Auswählen der gewünschten Konfigurationen wird das Dokument vorverarbeitet, indem Wort-Stammformen und Satzelemente mit Markierern (*Taggern*) ermittelt werden. Dabei werden die grammatikalischen Strukturen der Wörter im Satz bestimmt. Danach werden die verschiedenen Regeln, die ausgewählt wurden, angewandt. Am Ende kann das annotierte Anforderungsdokument wieder exportiert werden.

Wie bereits in der Einleitung erwähnt, ist RESI noch ausbaufähig und die Handhabung von Nominalisierungen soll in dieser Arbeit angegangen werden.

3. Verwandte Arbeiten

Viele Arbeiten haben sich mit dem Requirements Engineering und den dort auftretenden Probleme beschäftigt, damit Fehler in Anforderungsdokumenten möglichst vermieden werden können.

Deshalb haben sich Liu et al. in ihrer Arbeit [LLP10] mit der grundsätzlichen Frage beschäftigt, ob Requirements Engineering sinnvoll ist und warum es scheitern kann. Es wurde mit der Hilfe einer Umfrage aus dem Jahr 2009 versucht zu ermitteln, wie Firmen in China das Requirements Engineering ansehen und wo Verbesserungspotential herrscht. Sie fanden heraus, dass wohl der Sinn von Requirements Engineering erkannt wird und die meisten auch glauben, dass ein direkter Zusammenhang zwischen dem Requirements Engineering und der Softwarequalität existiert. Allerdings wird trotzdem ungern Zeit investiert. Immerhin 25% verbringen aber mehr als 20% des Aufwands für ein Projekt mit dem Requirements Engineering, 42% noch mehr als 10%. Laut der Umfrage sehen aber vor allem größere Firmen die Vorteile des Requirements Engineering und verbringen mehr Zeit damit.

Ein anderer relevanter Punkt in dieser Studie war, dass das Requirements Engineering oft durch Führungskräfte und extra geschultem Personal durchgeführt wird. Gleichzeitig lassen sich die Fehler oft auf Unwissen in der Domäne zurückführen. Hier stellt sich die Frage, ob es sinnvoll ist, extra geschultes Personal einzusetzen oder einfach das jeweilige Fachpersonal im Requirements Engineering zu schulen.

Ein weiterer Faktor, der von Liu et al. in [LLP10], aber ebenso oft in anderen Werken, wie in [Mic96] von Mich, genannt wird, ist Zeit. Die Zeitplanung ist oftmals zu eng, obwohl dieses Problem allgemein bekannt ist und was, wie bereits in Kapitel 2.3 genannt, zu schlechteren Anforderungsdokumenten führen kann. Produkte sollen schnellstmöglich entwickelt werden und Kunden üben Druck auf den Projekt- und Zeitplan aus. Die Gefahr, dass das Projekt dadurch eventuell scheitern könnte, wird nicht bedacht.

Trotz alledem benutzen 60% der Befragten keine Werkzeuge. Durch Werkzeugunterstützung lassen sich aber in der Regel die eben genannten Probleme angehen und damit Zeit, Aufwand und Kosten sparen, während gleichzeitig die allgemeine Qualität der Produkte gesteigert werden kann. Sie ermöglichen einen einfacheren Einstieg in das Requirements Engineering, sodass mehr Fachpersonal aus den entsprechenden Bereichen daran teilnehmen kann und dadurch Fehler durch fehlendes Domänenwissen reduziert werden können. Laut Liu et al. lässt sich aber nicht alles durch Werkzeuge verbessern. So führen sie beispielsweise an, dass bei der Anforderungsermittlung die oftmals schlechte Kommunikation

unter den Stakeholdern verbessert werden sollte. Ebenso ist ein häufiges Problem, dass sich Anforderungen und Bedürfnisse ändern können. Deshalb sollte beim Requirements Engineering und dem Erstellen und Verwalten von Anforderungen nach Liu et al. eher proaktiv gedacht werden und mögliche (offensichtliche) Änderungen „vorausgesehen“ – und rechtzeitig angesprochen – werden, anstatt lediglich reaktiv zu handeln. Zu guter Letzt ist ein Tipp der Autoren, der zwar allen bekannt und logisch erscheint, aber häufig vernachlässigt wird: „Erst denken, dann handeln“.

Ryan schreibt in seiner Arbeit über die Rolle von natürlicher Sprache im Requirements Engineering [Rya93], dass Werkzeuge zwar den ganzen Prozess unterstützen können, aber auch nicht alles ersetzen können. Für ihn ist der sozialer Prozess essentiell. Damit ist er ähnlicher Meinung wie Rupp in [RS09], während aber heutzutage viele Ansätze versuchen die informelle, soziale Ebene im Requirements Engineering zu ersetzen. Durch den sozialen Austausch erhält man aber nach Ryan oft ein tieferes Verständnis für das Projekt[Rya93]. Damit lässt sich die Qualität der Anforderungsdokumente weiter steigern und das Endprodukt entspricht mehr den Vorstellungen der Stakeholder. Ryan hält außerdem die Vorstellung von sogenannten *Just tell me*-Systemen, die dem Benutzer jegliche Denkprozesse abnehmen sollen, für eine gefährliche Illusion.

Nach seiner Meinung hat auch die Verarbeitung von natürlicher Sprache, das Natural Language Processing (NLP), viele Schwächen, da es die natürliche Sprache nicht vollständig und eindeutig verstehen und damit formelle Ansätze nicht voll ersetzen kann. Hier liegt meiner Meinung nach aber ein Schwachpunkt der Ausführung. Denn wie bereits im Kapitel 2.1.3 geschrieben, können auch Formalismen die Bedeutung nicht vollständig verstehen, während allerdings die Freiheit so eingeschränkt sein kann, dass das System nicht mehr einfach verständlich in vollem Umfang beschrieben werden kann. Gleichzeitig wird aktuell in verschiedenen Arbeiten und Institutionen an der Verbesserung der Werkzeuge für NLP geforscht. Beispielsweise existiert an der Stanford University eine *Natural Language Processing Group*, die die Forschung in diesem Bereich voranbringt. Ziel der Forschungen ist, dass Ergebnisse korrekter und besser werden. Dadurch werden Einschränkungen durch Formalismen immer weniger benötigt werden.

3.1 Ansätze zur Verbesserung von Anforderungsdokumenten

Die Qualität von Anforderungen und Anforderungsdokumenten lässt sich mit verschiedenen Möglichkeiten verbessern. Im Kapitel 2.1.3 wurden bereits Ansätze genannt, wie man Probleme der natürlichen Sprache umgehen kann. Ein Ansatz ist die Formulierung von Anforderungen mit Hilfe von formalen Methoden wie Schablonen oder Mustern. Dabei gilt im Allgemeinen, dass mit der steigenden Formalität die sprachlichen Mängel sinken, aber gleichzeitig auch die Freiheiten und die Einfachheit der natürlichen Sprache. Deshalb gibt es viele Lösungsansätze, die den Benutzer nicht direkt bei der Erstellung der Anforderungen unterstützen, sondern als Korrektur- und Überprüfungshilfen eingesetzt werden. Diese Werkzeuge lesen die Anforderungen aus den Dokumenten ein und überprüfen diese dann auf bestimmte Qualitätsmerkmale. Sie führen die Überprüfung und „Therapie“ [RS09] der Anforderungen zu großen Teilen automatisch durch.

Unter den verschiedenen Werkzeugen gibt es unter anderem diejenigen, die dem Benutzer beim Organisieren und der allgemeinen Erstellung von Anforderungsdokumenten unterstützen. Zu solchen Werkzeugen gehört unter anderem das *Dynamic Object Oriented Requirement System (DOORS)* [IBM], das mittlerweile von IBM unterhalten wird. Mit DOORS können Anforderungen einfach erstellt und verwaltet werden. *DOORS* findet heute Anwendung in vielen verschiedenen Bereichen und Betrieben, unter anderem auch bei der Daimler AG. Es bietet dem Benutzer eine einfache Möglichkeit Anforderungen zu strukturieren und zu verwalten. Die Verfolgbarkeit von Anforderungen ist wichtiger

Tabelle 3.1: Auswahl von Werkzeugen und Arbeiten im Requirements Engineering mit den jeweiligen Herangehensweisen

	Schreiben		Prüfen		
Werkzeug/Arbeit	form. Sprache	Muster	Wortlisten	Ontologien	Sonstiges
[LDL98]	+				
[Hof12]		+			
DESIRe			+		
QuARS			+		
[YDRG+12]			+		
[AKSS13]				+	
[SDGDG13]				+	
[FLGS14]					+
VARED	+				+
AcRES			+	+	
GATE				+	
NL-OOPS				+	

Bestandteil, weshalb es unter anderem die Möglichkeit gibt Objekte zu verlinken, um Anforderungen zu verfolgen. Durch seine Eigenschaften und die Ausrichtung auf gerade hochkomplexe und sicherheitskritische Umgebungen eignet sich *DOORS* gerade für größere Unternehmen [IBM].

Ein ähnliches Werkzeug wie DOORS ist *Cradle* von 3SL [3SL]. Es ist ein Anforderungsverwaltungswerkzeug, das mit allen Projektgrößen zurecht kommt. Es ist über eine Weboberfläche steuerbar und dadurch leicht für Stakeholder zugänglich. Gleichzeitig bietet es Kontrolle, Sicherheit und Eigenschaften wie Effizienz und die Verfolgbarkeit von Anforderungen. Es soll die die Erstellung von Dokumenten automatisieren und die erstellten Dokumente verwalten.

Ähnlich wie *DOORS* richtet sich *Caliber* von Borland an große und komplexe Systeme[Bor]. Mit seiner Visualisierung von Anforderungen und Testfällen und den jeweiligen Abhängigkeiten, zusammen mit der Verfolgbarkeit von Anforderungen und der Ausrichtung auf die Beteiligung von Stakeholdern über das Internet, ist dieses Werkzeug für diesen Zweck gut ausgestattet.

Es gibt noch viele weitere solche Werkzeuge für die Anforderungsverwaltung. Relevanter für diese Arbeit sind allerdings Werkzeuge, die den Benutzer beim Schreiben oder Überprüfen von Anforderungen unterstützen. Eine Auswahl davon ist in Tabelle 3.1 zu sehen.

3.1.1 Unterstützung beim Schreiben

Den Ansatz den Benutzer beim Schreiben von Anforderungen zu unterstützen verfolgen unter anderem Lamsweerde et al. in [LDL98] mit Hilfe der *KAOS (Knowledge Acquisition in automated Specification) specification language*. Ziel der Arbeit ist, die Erstellung von Anforderungen zu leiten, indem Ziele gesetzt werden. Wenn alle diese Ziele erreicht sind, dann ist die Spezifikation komplett. Hauptaugenmerk wird hier vor allem auf die Inkonsistenzen gelegt. Laut den Autoren sind Inkonsistenzen im Requirements Engineering nicht eine Ausnahme, sondern die Regel. Viele dieser Inkonsistenzen entstehen, wenn Anforderungen von unterschiedlichen Stakeholdern und aus unterschiedlichen Ansichten entstehen. Diese können schnell zu Problemen führen.

Gleichzeitig kann man die Inkonsistenzen aber nutzen, um die Anforderungen zu verbessern, wenn man sie gut verwaltet, denn beim Auflösen kann man aus ihnen weitere wichtige

Informationen erhalten. Dadurch werden die Anforderungen letztendlich verbessert. Bisherige Ansätze hatten nach der Ansicht der Autoren das Problem, dass die spezifische Art der Inkonsistenz nicht klar war. Außerdem war unklar, was ein Konflikt zwischen den Anforderungen genau bedeutet. Hier sah man bisher nur Konflikte zwischen genau zwei Anforderungen. Außerdem wurden Konflikte zwischen nicht-funktionalen Anforderungen oftmals nicht entdeckt. Ein weiteres Defizit bei bisherigen Arbeiten war, dass die Inkonsistenzen nicht durch eine Transformation der Anforderungen aufgelöst wurde. Deshalb wurde in dieser Arbeit die *KAOS specification language* angepasst. Durch sogenannte *divergence patterns* (Abweichungsmuster) und bestimmte Regeln werden somit Inkonsistenzen entdeckt und nach bestimmten Konventionen transformiert. In der Arbeit wurde allerdings nicht beschrieben, wie effizient das funktioniert. Es stellt sich hier die Frage, was für ein Aufwand betrieben werden muss und vor allem, wie groß die Einarbeitungszeit ist. Denn großer Nachteil ist hier, dass eine spezielle, neue Spezifikationssprache erlernt werden muss.

Um keine neue Spezifikationssprache erlernen zu müssen, verfolgt Hoffmann in seiner Arbeit [Hof12] den Ansatz, die Erstellung der Anforderungen durch Muster zu leiten. Es wird aufgeführt, dass musterbasierte Ansätze einige Vorteile bringen. So können sie den Aufwand verringern und damit Ressourcen schaffen, um die allgemeine Qualität von Anforderungsdokumenten zu verbessern. Ein Hauptargument für die Benutzung dieser Muster ist die Wiederverwendung. Im Gegensatz zu der manuellen Kontrolle können Muster immer wieder benutzt werden und man erhält dadurch eine gewisse Routine. Die resultierenden Anforderungen haben auch eine einheitliche Form und sind in der Regel somit einfacher und klarer zu verstehen. Bei der Erstellung von rechtlichen Anforderungen wird außerdem geraten die *KORA(Konkretisierung rechtlicher Anforderungen)*-Methode zu verwenden, eine extra von Rechtsexperten entwickelte Methode.

Die Muster und die resultierenden Anforderungen sind verständlich, allerdings bleibt in der Arbeit offen, wie viele Muster es gibt und wie lange die Einarbeitungszeit dauert und die Muster dem Anwender bekannt sind. Sobald Muster bekannt sind, kann man diese schnell und sicher verwenden und daraus Anforderungen erstellen. In der Einarbeitungszeit sehe ich aber eher das Problem, dass das Heraussuchen und Anwenden der korrekten Muster das Erstellen von Anforderungen eher verlangsamt. Hat man ein wechselndes Personal, dann muss das neue Personal jedes Mal neu eingearbeitet werden, was zeitraubend sein kann.

Das Problem dieser Ansätze ist die Einarbeitungszeit. Neue Ansätze benötigen Einarbeitungszeit und haben u.U. eine flache Lernkurve, wenn vorab Methoden, Muster und spezielle Spezifikationsfragen erlernt werden müssen. Gleichzeitig stellt sich die Frage, wie lange das Erstellen auf diese Art und Weise dauert und ob das Ergebnis dann besser ist. Ein weiterer großer Nachteil bleibt, dass neue Stakeholder erst einmal ausgeschlossen werden können. Bei vielen Unternehmen, wie beispielsweise der Daimler AG, werden Lastenhefte als Ausschreibungen benutzt. Durch den formellen Ansatz könnten Unternehmen aber abgeschreckt werden.

Gerade den Einsatz von Lastenheften als Ausschreibungen sollte man stets im Hinterkopf behalten. Methoden, die verhindern könnten, dass Lastenhefte als Ausschreibung genutzt werden können, haben ein klares Defizit.

3.1.2 Unterstützung beim Überprüfen

Einen anderen Ansatz verfolgen Werkzeuge, die die Qualität von Anforderungen verbessern wollen, indem sie die natürlichsprachigen Anforderungen auf Mängel untersuchen. Hier gibt es eine ganze Bandbreite von Werkzeugen, die jeweils den Fokus auf bestimmte Aspekte wie Konsistenz, Eindeutigkeit etc. setzen.

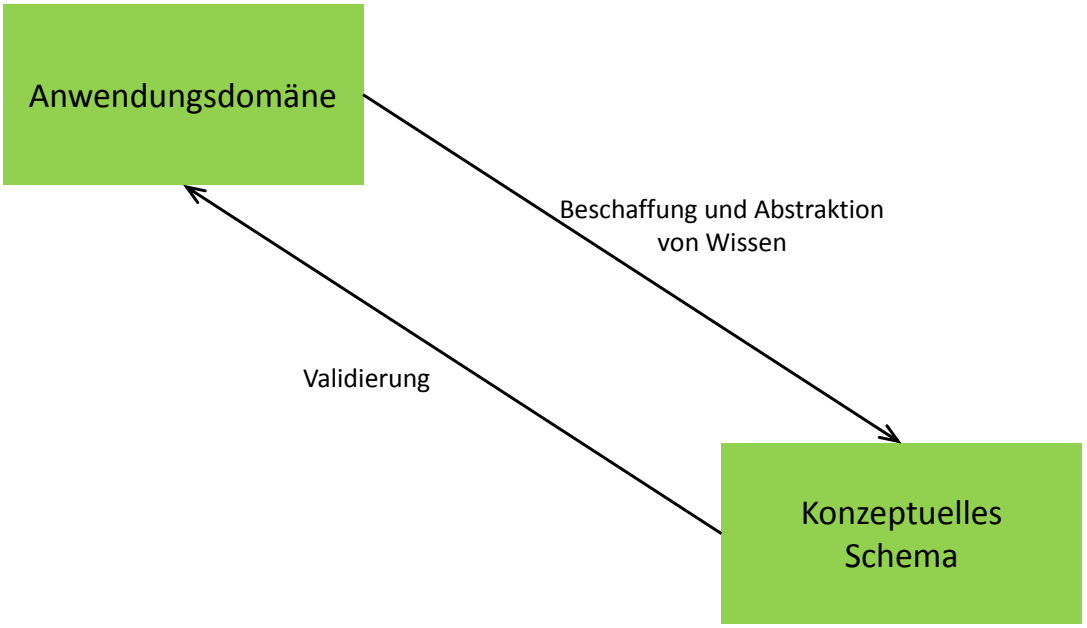

Abbildung 3.1: DB/IS Entwicklungszyklus als beispielhaftes Konzept für einen allgemeinen Entwicklungszyklus [RP92]

Rolland und Proix schreiben in ihrer Arbeit [RP92] aus dem Jahr 1992 davon, dass es einen Bedarf an Werkzeugen für die natürliche Sprache im Requirements Engineering gibt. In Abbildung 3.1 sieht man beispielhaft einen Teil des Entwicklungszyklus von Datenbank- und Informationssystemen (DB/IS). Das Problem sehen die Autoren hier, aus der reellen Welt zu abstrahieren und dadurch zum konzeptuellen Schema zu kommen. In der Abbildung stellt dies der Pfeil „Beschaffung und Abstraktion von Wissen“ dar. Der Konflikt entsteht bei der „Validierung“ der abstrahierten Domäne. Hier ziehen die Autoren einen Vergleich mit Schülern. Diese können oftmals eine Gleichung lösen, aber diese nicht aus der (textuellen) Aufgabenstellung erstellen. Das selbe Phänomen existiert nach ihrer Ansicht auch beim Requirements Engineering. Daher muss eine Überprüfung statt finden, um die Unterschiede zwischen Realität und Konzept zu minimieren. Werkzeuge können diesen Prozess unterstützen.

3.1.2.1 Wortlisten und Wörterbücher

Es gibt verschiedene Ansätze, die mit Hilfe von Wortlisten die Anforderungen verbessern wollen. Nach Stöckel et al. in [SSUE09] hängt die Wirksamkeit dieser Methoden von der Wortliste ab. Dabei zählt neben der Qualität der Einträge auch die Quantität. Der Vorteil der Wortlisten ist, dass sie einfach an die verschiedenen Einsatzgebiete angepasst werden können. Verschiedene Sprachen, Domänen, Produktwelten und andere Besonderheiten können leicht mit dem Austausch der Wortlisten angegangen werden. Sie sind in der Regel leicht erweiterbar, aber der große Nachteil liegt daran, dass sie zuerst einmal erstellt werden müssen, was anfangs viel Zeit in Anspruch nehmen kann. In ihrem Werkzeug *DESIRe (Dynamic Expert System for Improving Requirements)* [SSUE09] haben Stöckel et al. versucht, eine Methode zu entwickeln, die für den Benutzer intuitiv, immer und für alle verfügbar und lernfähig ist und die Unterstützung direkt an den dokumentierten Anforderungen zur Verfügung stellt.

An anderen Methoden hatten sie auszusetzen, dass diese oftmals kompliziert sind, besondere Schulungen benötigt haben und dies damit zeitlich ein Problem darstellt. Deshalb

beruht das Werkzeug mit Wortlisten auf 3 einfachen, wiederholbaren Schritten. Zuerst wird die Anforderung analysiert, dann verbessert und zuletzt werden die Erkenntnisse abgespeichert. Dabei existieren definierte Fragen und Hinweise für definierte Sprachelemente, die aus der Wortliste kommen. Somit kann auf häufig gemachte Fehler und vorher definierte Probleme gezielt hingewiesen werden. Dies geht soweit, dass die Nutzer nach einer gewissen Zeit die Wortliste verinnerlichen und dadurch selbstlernend bestimmte Fehler gar nicht mehr produzieren. Nachteil jedoch ist, dass nicht zuerst automatisiert geprüft wird, ob die Nachfrage überhaupt benötigt wird. Somit wird auf sehr vieles hingewiesen bzw. Nachfragen werden gestellt, die überhaupt nicht problematisch sind, wodurch der Nutzer frustriert werden kann. Verbesserungspotential besteht außerdem bei der Art und Weise der Benutzung der Wortlisten. Der Text gibt keinen Hinweis darauf, dass Wortstämme o.ä. benutzt werden, sodass (konjugierte/deklinierte etc.) Wörter entweder nicht erkannt werden oder die Wortliste aufgebläht werden muss, da Wörter dann in ihren verschiedenen Formen in der Wortliste stehen müssen.

Der *Quality Analyzer for Requirements Specification (QuARS)*, den Lami in seiner Arbeit [Lam05] vorstellt, soll das Suchen nach Defekten, das „langweilig, zeitraubend und oft ineffektiv“ [Lam05] ist, erleichtern und verbessern. Der Autor teilt die verschiedenen Qualitätseigenschaften in drei Kategorien ein: Ausdruckskraft, Konsistenz und Vollständigkeit. Anforderungen sollen auf genau diese drei Kategorien untersucht werden. Die Arbeit verfolgt dafür einen Ansatz mit einer Art Wortliste, setzt diese aber auf eine etwas andere Weise ein.

Für die Ausdruckskraft werden die Anforderungen auf Wörter und Konstrukte untersucht, die diese einschränken können. Die anderen beiden Kategorien werden auf eine andere Weise sicher gestellt. Dafür wurden zwei neue Begriffe eingeführt: *V-Dictionaries* und *Views*. *V-Dictionaries* sind Wörterbücher, die Wörter, die im Kontext eines speziellen Themas stehen, enthalten. Diese Wörterbücher müssen für die jeweiligen Themen manuell erstellt werden. *Views* sind eben solche gebündelten Sätze zu einem bestimmten Thema.

Die Sätze der Anforderungen werden mit Hilfe von *V-Dictionaries* untersucht und zu *Views* zugeordnet, wenn die jeweiligen Wörter von der vorherigen Syntax-Analyse als Subjekte oder Objekte des Satzes markiert wurden. Die folgenden Analysen von Sätzen können daraufhin die jeweiligen gebündelten Sätze besser untersuchen, da Thema und Kontext bekannt sind. Außerdem können so auch Sätze erkannt werden, die sich nicht im richtigen, aktuellen Kontext des Abschnitts im Anforderungsdokument befinden und dadurch zu bspw. Inkonsistenz führen können. Weiterhin werden bei *QuARS* Zusammenhänge bzw. Widersprüche automatisch ermittelt, indem Subjekt-Objekt-Aktion-Tripel mit Hilfe der *V-Dictionaries* ausgewertet werden.

Nach Ansicht von Lami liefert die Benutzung dieses Werkzeugs in Verbindung mit menschlichen Inspektionen das bestmögliche Ergebnis. Der Einsatz ist effizienter als menschliche Kontrolle alleine und spart gleichzeitig Zeit und damit Geld. Durch die Wortlisten etc. ist das Programm auch sehr anpassungsfähig für verschiedene Domänen. Allerdings stellt sich auch hier wieder die Frage, wie aufwändig ein Wechsel in eine neue Domäne ist, da dadurch neue Wörterbücher und *V-Dictionaries* angelegt werden müssen. Allerdings sind die Ergebnisse der Tests mit echten Anforderungen aus der Industrie laut Lami vielversprechend.

Ein häufiges Problem bei natürlichsprachigen Anforderungen ist außerdem die vage und spekulative Formulierung. Diese ist teilweise beabsichtigt, doch im Allgemeinen entstehen hier Probleme durch die offen gelassenen Freiheiten und der daraus entstehenden Gefahr der Missinterpretation. Unsicherheiten gehen oft einher mit Mehrdeutigkeiten, denn nach Yang et al. [YDRG+12] werden Mehrdeutigkeiten oft benutzt, um Unsicherheiten zu

verbergen. Es existiert jedoch ein Unterschied zwischen beiden. Demnach entstehen Mehrdeutigkeiten durch die Form der Anforderung, während Unsicherheiten durch den Inhalt der Anforderung entstehen. Daher sollten sie frühestmöglich angegangen werden, damit die Anforderungen inhaltlich eine ausreichend hohe Qualität aufweisen können.

Yang et al. haben daher einen Ansatz erarbeitet, um solche Anforderungen zu erkennen und ausbessern zu können [YDRG+12]. Sie untersuchen die Dokumente auf Wörter, die bekanntermaßen Unsicherheiten auslösen können. Da diese auch in nicht spekulativen Sätzen vorkommen können, setzen sie außerdem einen Parser ein und klassifizieren die Sätze mit Hilfe weiterer Heuristiken. Das Problem des momentanen Ansatzes ist allerdings, dass Stichworte, die aus mehreren Wörtern bestehen, nur rudimentär per direkten *String Matching*, also per direkten Wortabgleich, gefunden werden können. Gleichzeitig hängt die Güte dieses Ansatzes auch immer von der Güte des POS-Taggers und Parsers ab. Dennoch sind die von Yang et al. präsentierten Zahlen der Evaluation gut und vielversprechend.

3.1.2.2 Ontologien

Ontologien können beim Überprüfen und Verarbeiten von natürlichsprachigen Anforderungen sehr nützlich sein. Man kann sie, wie in Kapitel 2.2 beschrieben, unter anderem dafür verwenden, Domänenwissen in die Überprüfung einfließen zu lassen. Ebenso sind Informationen über die Semantik für die Verarbeitung nützlich.

Annervaz et al. haben allerdings in ihrer Arbeit [AKSS13] angemerkt, dass es zu wenige Ansätze gibt, die natürlichsprachige Anforderungsdokumente mit der Hilfe von Ontologien angehen. Ihrer Ansicht nach machen es sich Lösungen, die vorformulierte Vorlagen benutzen, zu einfach, um in komplexen, echten Anwendungen erfolgreich eingesetzt werden zu können. Gleichzeitig sehen sie die Problematik bei spezifischen Regeln für jeweilige Domänen darin, dass diese oftmals schwer und teuer zu erstellen sind. Bei der natürlichen Sprache sehen sie auch verschiedene Probleme, die sie aber in ihrer Arbeit angehen wollen. Zum einen gibt es die Probleme durch Mehrdeutigkeiten, die durch die unpräzise Art der natürlichen Sprache automatisch vorkommen. Außerdem kann die natürliche Sprache zu unvollständigen Anforderungen führen, da die Vollständigkeit und eine Validierung nicht automatisch erzwungen wird.

Der Ansatz von Annervaz et al. sieht vor, dass anfangs ein Modell der Domäne mit Hilfe von Ontologien erstellt wird. Durch lexikalische und semantische Ähnlichkeiten können Wörter nun zu dem Elementen in diesem Modell zugewiesen werden. Dadurch können Anforderungen zugeordnet und daraufhin auch überprüft werden. Zum einen wird dabei überprüft, ob die Anforderung im Sinne der Mehrdeutigkeit eindeutig verständlich ist. Danach kann auch noch überprüft werden, ob die Anforderung andere Mängel hat, da sie beispielsweise unvollständig oder unterspezifiziert ist. Problem dieses Ansatzes ist allerdings nach Aussage der Autoren die Performanz, wenn die Größe der Ontologien zunimmt. Außerdem kostet das (erste) Erstellen der spezifischen Modelle auch Zeit. Hier ist also die Frage, wie zeit- und damit kostenintensiv die Erstellung ist und wie flexibel diese Modelle dann eingesetzt werden können.

Einen ähnlichen Ansatz verfolgen Sadoun et al.[SDGDG13]. Um natürlichsprachige Anforderungen zu verarbeiten, wird bei ihnen ein Tripel von Informationen verarbeitet. Dieses besteht aus lexikalischen, syntaktischen und zusätzlich semantischen Informationen, während andere Ansätze meist nur die ersten beiden Informationen verarbeiten. Dadurch muss sich diese Lösung nicht nur auf die lexikalische Erkennung verlassen, sondern kann auch auf die Semantik zurückgreifen und die Analyse bekommt mehr „Tiefgang“. Mit diesem Ansatz können sie die Eigenschaften eines Terms bestimmen und Mehrdeutigkeiten ausschließen. Dadurch können die natürlichsprachigen Anforderungen formalisiert und formelle Methoden angewandt werden.

Die Ergebnisse ihrer Arbeit anhand eines Tests bestätigen, dass dieser Ansatz vielversprechend ist. Doch auch hier bleibt die Frage offen, wie viel Mehraufwand durch die Erstellung der Ontologien und Modelle entsteht.

3.1.2.3 Sonstige

Ferrari et al. haben in ihrer Arbeit [FLGS14] einen etwas anderen Ansatz gewählt. Sie stellen einen pragmatischen Ansatz zur Entdeckung von Mehrdeutigkeiten in natürlicher Sprache vor. Mehrdeutigkeiten hängen zum einem von ihrem Kontext ab, aber auch vom Hintergrundwissen des Lesers. Aus diesem Grund sind die Autoren der Meinung, dass regelbasierte Lösungsansätze hier nicht ausreichend sind, denn die Hintergründe können unzählbar viele sein und man bräuchte für diese jeweils Regeln, was unmöglich ist. Deshalb haben die Autoren eine Methode gewählt, die Hintergrundwissen als Graphen modelliert. Die jeweilige Interpretation erfolgt dann durch eine Kürzester-Pfad-Suche in diesem Graphen. Um den Graphen zu modellieren, werden zunächst verschiedene Dokumente aus der selben Domäne analysiert. Somit kann ermittelt werden, in welchen Kontexten Wörter oft verwendet werden und somit auf die Bedeutung geschlossen werden.

Das Problem dieses Ansatzes ist aber, dass es sich hier lediglich um eine statistische Methode handelt und die Ergebnisse (noch) nicht zuverlässig genug sind. Laut der Analyse von Ferrari et al. kommen sie durch diese Methode auf eine Trefferquote von 63% und eine Präzision von 51%. Die Autoren sehen hierin zumindest einen guten Anfang, aber solange diese Werte nicht höher sind, sehe ich ein Problem, da zum einen Mehrdeutigkeiten einfach übersehen werden können und zum anderen der Zweck des Programms noch nicht ganz erfüllt ist. Neben den falsch-negativen Ergebnissen ist nach Aussage der Autoren noch ein weiteres Problem die Effizienz. Außerdem müssen für die Dokumente jeweils vorher ausreichend Dokumente der selben Domäne gefunden werden, damit das Werkzeug trainiert und angewandt werden kann. Gerade bei projektspezifischen, allgemeinen und Mehrwort-Termen hat das Werkzeug noch Probleme, vor allem wenn manche Terme in den anderen Dokumenten nicht vorkommen, da dann keine Abschätzung darüber abgeliefert werden kann. Die Idee hinter der Lösung ist nachvollziehbar, allerdings muss hier noch einiges verbessert werden, bevor der Nutzen dieses Ansatzes zur Geltung kommen kann.

Bei der Daimler AG wird aktuell der Lastenheft-Analyse-Assistent (LH-AnA) genutzt [Hou14]. Der LH-AnA ist ein Werkzeug mit verschiedenen Tools zur Überprüfung von Lastenheften, die auch in Abbildung 3.2 zu sehen sind. So sind verschiedene sprachliche Prüfungen enthalten, aber auch Konsistenz- und Plausibilität-Checks. Dadurch können Lastenhefte automatisch überprüft werden, die Ausgabe der Tests erfolgt entweder interaktiv oder durch Prüfberichte, die aus Excel-Tabellen bestehen. Ein Beispiel für eine sprachliche Prüfung ist die Analyse von sogenannten *Weak Words*, d.h. von Wörtern oder Phrasen mit unpräziser Bedeutung [Kri14]. Ein Test der Plausbilitätsprüfung ist die Analyse von Referenzen. Darin wird überprüft, ob die Referenzen des Lastenhefts auch existieren bzw. dem Lastenheft angehängt sind.

Die Nutzung des LH-AnA bringt eine größere Verbesserung der Lastenhefte, wie die Analyse von repräsentativen Lastenheften ergab. Momentan wird jedoch eher selten die sprachliche Überprüfung genutzt, da die Prüfungen momentan noch viele falsch-positiven Ergebnisse erzeugt.

Die Modularisierung des LH-AnA erlaubt außerdem die einfache Erweiterung, sodass weitere Prüfungen hinzugefügt werden können. Das eröffnet verschiedene Möglichkeiten, wie beispielsweise die Überprüfung auf Nominalisierungen von RESI einzubauen. Für die Akzeptanz müsste allerdings die Anzahl der falsch-positiven Ergebnissen reduziert werden, was ein Ziel in dieser Arbeit ist.

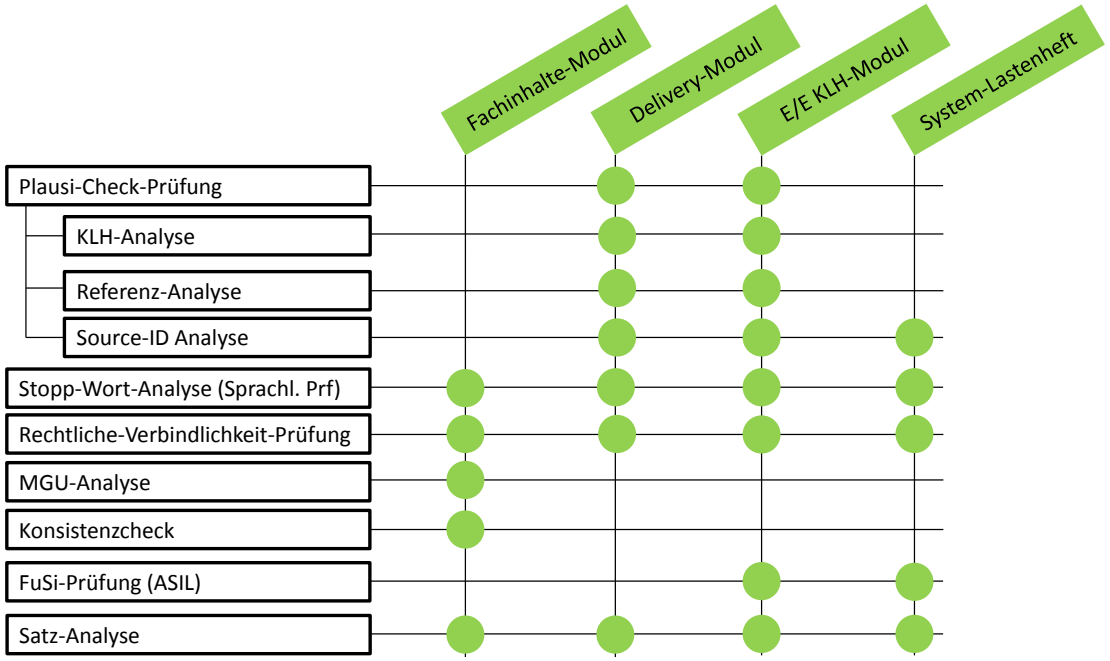

Abbildung 3.2: Verfügbare Prüfungen im Lastenheft-Analyse-Assistenten nach [Hou14]

4. Analyse und Entwurf

Um festzustellen, wie problematisch Nominalisierungen in der Praxis sind und ob sich ein Regelwerk ableiten lässt, das die Kritikalität von Nominalisierungen abbilden kann, wurden Lastenhefte aus dem Fahrzeugumfeld der Daimler AG analysiert. Dabei wurden manuell fünf Lastenhefte mit insgesamt 40.460 Wörtern untersucht, mit Hilfe von RESI acht Dokumente mit insgesamt 33.496 Wörtern. Dabei hat sich herausgestellt, dass sich bei Nominalisierungen das Subjekt, auf das sich die Nominalisierung bezieht, stets aus dem Kontext ableiten ließ. Im Allgemeinen war dies stets das zu entwickelnde Produkt oder derjenige, der das Produkt entwickeln und herstellen soll. In den übrigen Fällen war der Bezug stets klar, da der Kontext eindeutig genug war. Für diese Arbeit wurde daher explizit nicht auf das Subjekt geprüft bzw. sichergestellt, dass die Nominalisierung nicht alleine durch das Subjekt spezifiziert werden kann.

4.1 Kategorisierung von Nominalisierungen

Aus der Analyse der Lastenhefte konnten vier Kategorien von Nominalisierungen abgeleitet werden. Hierbei stellen die ersten beiden Kategorien Nominalisierungen dar, die ausreichend spezifiziert sind. Die dritte Kategorie ist eine Grenzkategorie. Hier hängt die Einschätzung, ob die Nominalisierung problematisch ist, davon ab, wie eindeutig man Kontext und Domäne des Dokuments und der einzelnen Anforderungen bestimmen kann. Die vierte Kategorie ist in jedem Fall problematisch, da sie die stets nicht ausreichend spezifizierten Nominalisierungen enthält.

4.1.1 Kategorie 1

Die Nominalisierungen der Kategorie 1 sind die stets unproblematischen. Die Nominalisierungen an sich sind durch sich selbst ausreichend spezifiziert, sodass keine weiteren Bezüge benötigt werden. Dadurch sind die Nominalisierungen auch völlig unabhängig von dem restlichen Satz schon spezifiziert und bilden nie ein Problem, da sie nicht falsch interpretiert werden können. In diese Kategorie fallen der Erfahrung nach wenige Nominalisierungen, da wenige völlig ohne zusätzliche Informationen auskommen.

Ein Beispiel für diese Kategorie ist das Wort *troubleshooting*. Da das Subjekt, wie oben bereits erläutert, klar ist, benötigt dieses Wort keine weiteren Informationen. Das Wort *troubleshooting* ist so allgemein in seiner Bedeutung, dass es durch keine zusätzlichen Informationen näher spezifiziert werden muss. Hilfreich hierbei ist sicher, dass das Wort

zusammengesetzt ist aus der eigentlichen Nominalisierung *shooting* und dem Bezugswort *trouble*. Das Wort ist allerdings sehr allgemein, weshalb sich gleichzeitig die Frage stellt, ob das Wort dem Satz einen ausreichenden Mehrwert an Informationen bringt.

Es ist auch denkbar in diese Kategorie Wörter einzuordnen, die durch die spezifische Domäne des Dokuments zu selbstspezifizierenden Nominalisierungen werden. Hierfür könnte man spezielle Wortlisten für die unterschiedlichen Domänen verwenden.

4.1.2 Kategorie 2

In Kategorie 2 fallen diejenigen Nominalisierungen, die durch den Kontext im Satz spezifiziert werden. Im Satz, in dem die Nominalisierung vorkommt, existieren weitere Wörter beziehungsweise Satzteile, auf die sich die Nominalisierung bezieht und dadurch spezifiziert wird. Die Kategorie lässt sich dabei in zwei Unterkategorien aufteilen. Die erste behandelt die direkte Umgebung der Nominalisierung, da eine Nominalphrase gebildet wird, während die zweite Unterkategorie sich auf entferntere Satzteile und Wörter bezieht.

4.1.2.1 Nominalphrase

Nominalisierungen der Kategorie 2, die unter diese Kategorie fallen, bilden mit anderen Satzteilen eine sogenannte Nominalphrase. Hierbei wird ein Substantiv durch zusätzliche Wörter weiter beschrieben. In unserem Fall sind solche Nominalphrasen von Interesse, die Nominalisierungen weiter spezifizieren. Dies ist der Fall, wenn eine Nominalisierung zusammen mit einem anderen Substantiv ein Kompositum bildet. Dadurch wird aus der Nominalisierung auch oftmals ein Teil eines anderen Substantivs, wodurch die eigentliche Nominalisierung verschwindet.

Ein Beispiel für eine Nominalphrase ist in dem folgenden (Halb-)Satz zu finden: „If the user changes the channel selection of the headphones, ...“. Die Nominalisierung *selection* wird hier durch die Nominalphrase weiter spezifiziert. Zum einen durch das Wort *channel*, wodurch klar wird, was ausgewählt werden soll. Zum anderen auch noch durch den Teil, das aus dem Wort *headphones* gebildet wird und durch die Präposition *of* mit der Nominalisierung verbunden wird und sie weiter spezifiziert. Der Syntaxbaum, der die Abhängigkeiten, die durch den Stanford-Parser ermittelt wurden, darstellt, wird in der Abbildung 4.1 dargestellt. *Selection* befindet sich direkt in einer Nominalphrase (*Noun Phrase (NP)*) mit dem Wort *channel*. Die gebildete Nominalphrase wiederum bildet zusammen mit der Präpositionalphrase (*PP*) um das Wort *headphones* eine weiteren Nominalphrase. Dadurch wird die Nominalisierung *selection* gleich durch mehrere Wörter und Phrasen spezifiziert.

Ein weiteres Beispiel kann man in folgendem Satz finden: „The contractor shall coordinate with the clients development manager“. Hier bilden die Wörter *development* und *manager* eine Nominalphrase, wodurch die eigentliche Nominalisierung in *development* zum Teil eines Substantivs wird und das Substantiv weiter spezifiziert. Die Abbildung 4.2 verdeutlicht diese Verbindung zwischen *manager* und *development* über die Nominalphrase.

4.1.2.2 Spezifizierende Satzteile

In vielen Fällen steht das Wort, auf das sich eine Nominalisierung im Satz bezieht, jedoch nicht direkt daneben und ist nicht Teil der Nominalphrase. Durch diese Wörter können die Nominalisierungen jedoch auch spezifiziert werden. Als Beispiel sei der folgende Satz aufgeführt: „For a given model series, the implementation of this function shall be discussed during development.“ Die Nominalisierung *development* kann sich auf das Wortpaar *model series* beziehen, wodurch sie ausreichen spezifiziert wird.

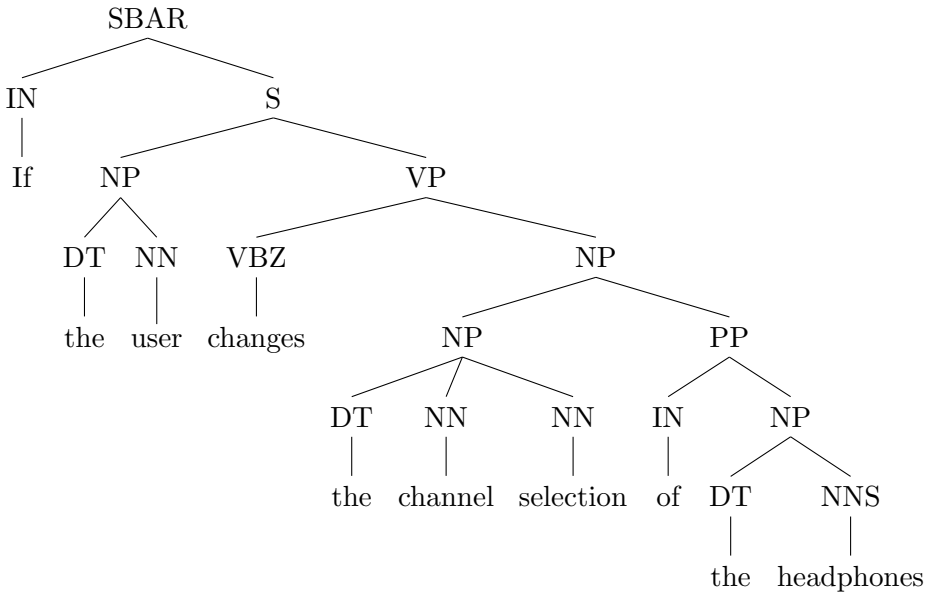

Abbildung 4.1: Beispiel eines Satzes mit einer Nominalphrase, in der sich eine Nominalisierung, hier *selection*, befindet

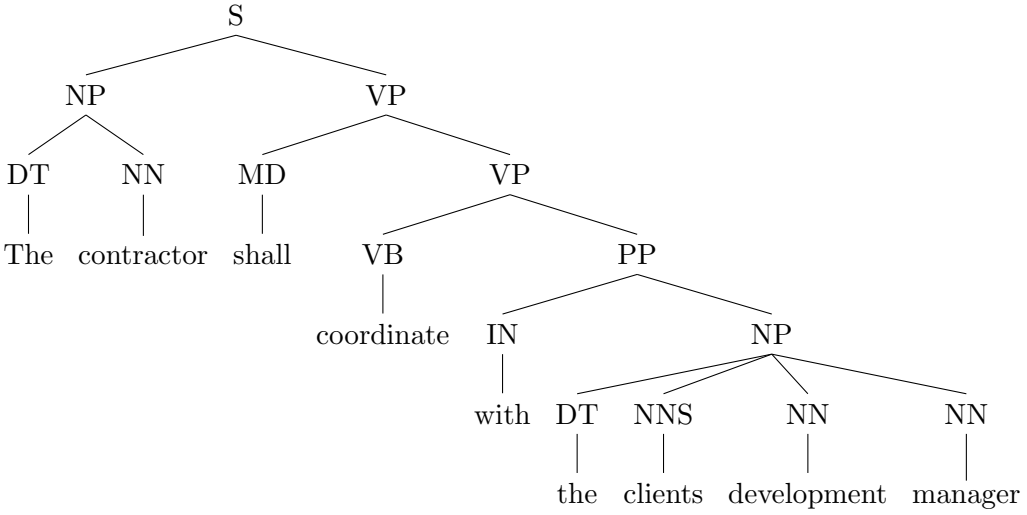

Abbildung 4.2: Syntaxbaum des Satzes, annotiert durch den Stanford-Parser

4.1.3 Kategorie 3

Nominalisierungen der Kategorie 3 werden durch die Domäne oder den Kontext der Anforderung spezifiziert. Dadurch sind die Nominalisierungen nicht problematisch. Dies stellt aber die Frage auf, wann der Kontext beziehungsweise die Domäne für den Leser ersichtlich ist. Außerdem muss man bedenken, dass der Kontext beziehungsweise der Bezug der Nominalisierung mehrdeutig sein kann. Deshalb ist diese Kategorie die Grenzkategorie. Sind die Bezüge der Nominalisierung eindeutig und für den Leser in jedem Fall ersichtlich, dann gehört die Nominalisierung zu dieser Kategorie und ist nicht problematisch. Andernfalls muss die Nominalisierung der Kategorie 4 angerechnet werden.

Als Beispiel kann man folgenden Satz nennen: „The exact design will be determined during development“. Aus dem Kontext, in dem die Anforderung steht, wird ersichtlich, dass sich die Nominalisierung *development* auf das zu entwickelnde Produkt bezieht und daher eindeutig genug ist.

4.1.4 Kategorie 4

Nominalisierungen dieser Kategorie sind problematisch und sollten überarbeitet werden. Der Kontext, in dem die Nominalisierung steht ist nicht eindeutig oder es existiert überhaupt kein Kontext, auf den sich die Nominalisierung beziehen kann. Dadurch wird sie nicht ausreichend spezifiziert. Die Informationen, die dadurch vorenthalten werden oder nicht eindeutig sind, können jedoch entscheidend für das korrekte Verständnis sein.

Im Beispielsatz „For fine optimizations and verification an additional cycle in the B-phase car will be permitted“ ist durch den Kontext, in dem die Nominalisierungen *optimization* und *verification* stehen, nicht ganz eindeutig, auf was sich die *optimization* und *verification* beziehen.

4.1.5 Statistiken zur Kategorisierung

Anhand der obigen Kategorisierung wurden Lastenhefte analysiert und die gefundenen Nominalisierungen eingeordnet. Dadurch konnte eine Übersicht über die Verteilung der verschiedenen Kategorien erstellt und abgeschätzt werden, wie viele unnötige Hinweise auf Nominalisierungen theoretisch unterdrückt werden können.

Die Ergebnisse der Analyse sind in den Tabellen 4.1 und 4.2 zu finden. In den Tabellen sind die Namen der Dokumente jeweils anonymisiert. In 4.1 findet man die Ergebnisse der manuellen Analyse anhand von 5 Lastenheften mit insgesamt 40.460 Wörtern. Dokument 1 und 2 wurden nur manuell analysiert, da diese viele Anforderungen beinhalteten, die lediglich vertragliche Verpflichtungen darstellten und daher nicht sehr ergiebig waren. Fehlende Prozente auf 100% sind durch falsch-positive Warnungen vor Wörtern, die gar keine Nominalisierungen sind, geschuldet.

Die Tabelle 4.2 zeigt die Ergebnisse der Verarbeitung der Lastenhefte mit der Regel für Nominalisierungen von RESI anhand 33.496 Wörtern in 8 verschiedenen Lastenheften. Allgemein kann man sagen, dass zumindest in diesen Lastenheften keine Wörter der Kategorie 1 zu finden waren. Wie bereits erwähnt, sind solche Wörter selten, was durch diese Ergebnisse untermauert wird. Bei diesen Analysen wurde lediglich auf allgemein unproblematische Wörter der Kategorie 1 geachtet. Wörter, die lediglich aufgrund der Domäne unproblematisch sind, wurden hier nicht zu Kategorie 1 gezählt.

In den untersuchten Dokumenten wurden ebenfalls kaum Nominalisierungen der Kategorie 4 gefunden. Die gefundenen Nominalisierungen waren durch Satz oder Kontext in der Regel ausreichend spezifiziert.

Tabelle 4.1: Ergebnisse der manuellen Analyse verschiedener Lastenhefte

	Wörter	Nomin.	Kat 1	Kat 2	Kat 3	Kat 4
Dok. 1	9942	85	0 0.0%	60 70.6%	25 29.4%	0 0.0%
Dok. 2	23104	158	0 0.0%	96 60.8%	58 36.7%	4 2.5%
Dok. 3	2129	21	0 0.0%	17 81.0%	4 19.0%	 0.0%
Dok. 4	3687	62	0 0.0%	59 95.2%	3 4.8%	 0.0%
Dok. 5	1598	30	0 0.0%	17 56.7%	13 43.3%	 0.0%
Gesamt	40460	356	0 0.0%	247 69.4%	102 28.7%	4 1.12%

Tabelle 4.2: Einzelne Ergebnisse der Analyse verschiedener Lastenhefte mit RESI

	Wörter	Nomin.	Kat 1	Kat 2	Kat 3	Kat 4
Dok. 3	2158	25	0 0.0%	22 88.0%	2 8.0%	0 0.0%
Dok. 4	3687	56	0 0.0%	47 83.9%	1 1.8%	0 0.0%
Dok. 5	1598	15	0 0.0%	11 73.3%	1 6.7%	1 6.7%
Dok. 6	6069	116	0 0.0%	106 91.4%	6 5.2%	0 0.0%
Dok. 7	12581	133	0 0.0%	112 84.2%	7 5.3%	0 0.0%
Dok. 8	7403	154	0 0.0%	115 74.7%	25 16.2%	0 0.0%
Gesamt	33496	499	0 0.0%	413 82.8%	42 8.42%	1 0.2%

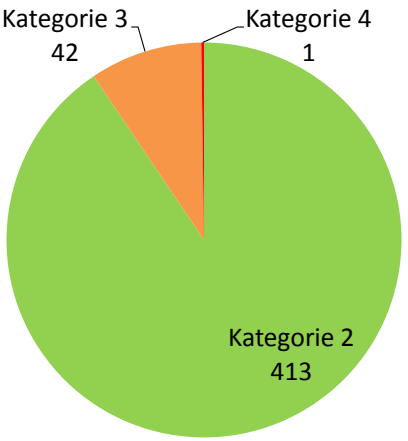

Abbildung 4.3: Visualisierung der Verteilung auf die verschiedenen Kategorien der Ergebnisse der Analyse mit RESI

Die meisten Nominalisierungen findet man in Kategorie 2. Dementsprechend sollte auch ein Fokus darauf gelegt werden, solche Nominalisierungen zu erkennen und als unproblematisch zu behandeln. Deutlich wird dies, wenn man die Verteilung im Diagramm in Abbildung 4.3 betrachtet. Da Nominalisierungen der Kategorie 2 per Definition spezifiziert sind, kann man durch das Beseitigen dieses Blockes bereits einen Großteil der unnötigen Meldungen unterdrücken, wodurch eine Akzeptanz des Werkzeuges durch die Nutzer wahrscheinlicher wird.

Die Ergebnisse aus Tabelle 4.2 zusammen mit der Darstellung als Diagramme in den Abbildungen 4.4 und 4.5 zeigen, dass die Verteilung in den einzelnen Dokumenten konsistent ist und jeweils kaum von den obigen Gesamtergebnissen abweicht. Man kann davon ausgehen, dass in anderen Dokumenten, zumindest im Fahrzeugumfeld der Daimler AG, diese Verteilung ähnlich bleibt. Die Abweichung der manuellen Analyse von der Analyse durch RESI kommt durch das Übersehen von Nominalisierungen und ähnlichen menschlichen Fehlern, wie sie beispielsweise in Kapitel 2.3 genannt wurden. Mangelnde Erfahrung zum Zeitpunkt dieser Analyse ist ebenfalls ein Faktor, weshalb die Ergebnisse abweichen.

4.2 Entwurf

In den obigen Ergebnissen der Analyse war zu erkennen, dass gerade durch die Beseitigung der Nominalisierungen der Kategorie 2 viele falsche Nominalisierungsmeldungen unterdrückt werden können. Daher soll auf diese Kategorie besonders Rücksicht genommen werden.

Um entscheiden zu können, ob eine Nominalisierung unproblematisch ist, muss man beim Erkennungsprozess direkt an der Stelle eingreifen, an der festgestellt wird, dass eine Nominalisierung vorliegt. Die Warnung muss zurückgehalten werden und soll nur erfolgen, wenn die Tests auf die verschiedenen Kategorien ergaben, dass die Nominalisierung nicht ausreichend genug spezifiziert ist. Ergibt einer der unten aufgeführten Tests, dass die Nominalisierung ausreichend spezifiziert ist, wird die Überprüfung an dieser Stelle unterbrochen und die Warnung zurückgehalten. Ansonsten wird der nächste Teil der Überprüfung ausgeführt. Der Ablauf ist in Abbildung 4.6 zu sehen.

Zuerst wird auf die Kategorie 1 geprüft. Nominalisierungen dieser Kategorie können per einfachem Vergleich mit einer vorher erstellten Wortliste angegangen werden. Da momen-

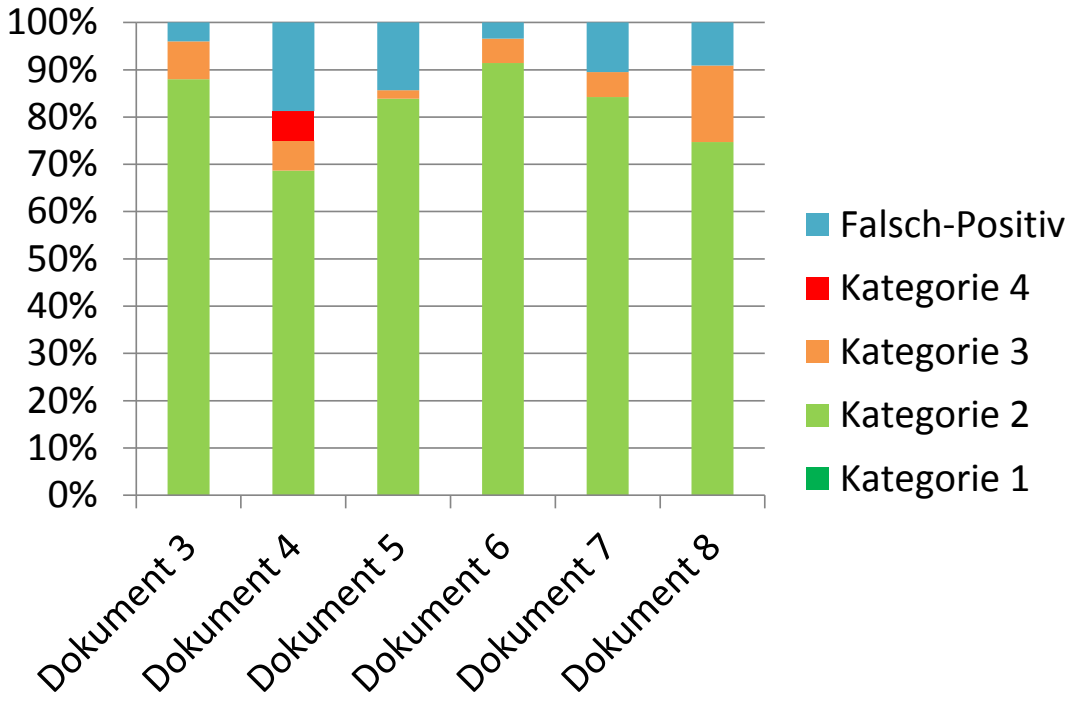

Abbildung 4.4: Visualisierung der einzelnen Ergebnisse der Analyse mit RESI in Prozent

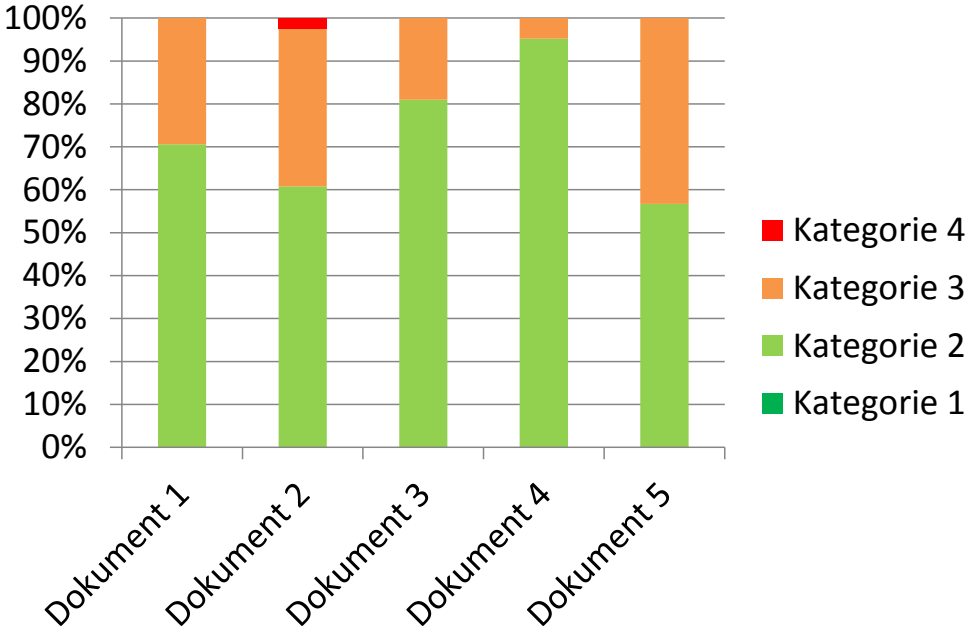

Abbildung 4.5: Visualisierung der einzelnen Ergebnisse der manuellen Analyse in Prozent

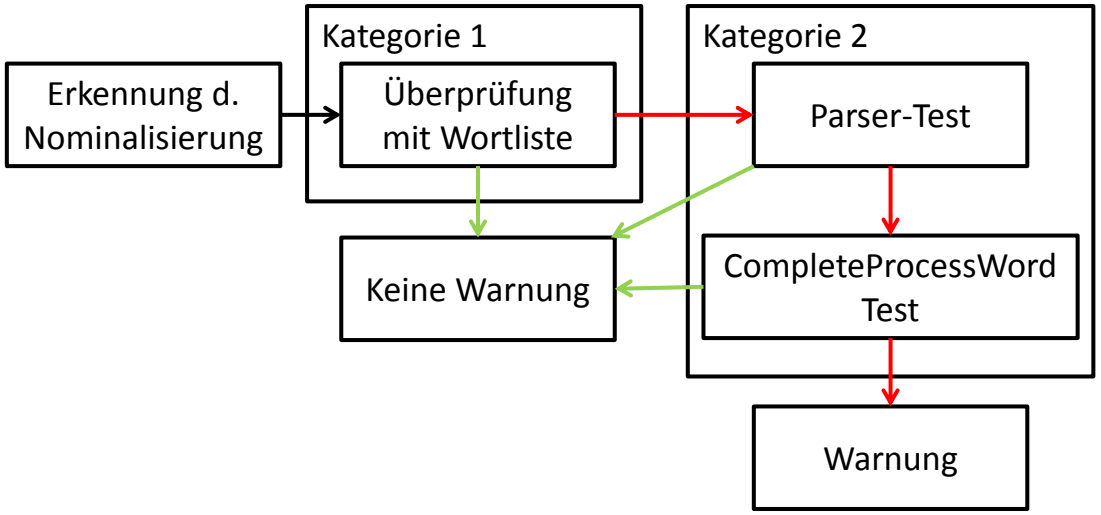

Abbildung 4.6: Ablauf der Überprüfung auf unproblematische Nominalisierungen. Rote Pfeile bedeuten, der Test ordnete die Nominalisierung als nicht spezifiziert ein. Grüne bedeuten, die Nominalisierung wurde als ausreichend spezifiziert erkannt

tan nur wenige Nominalisierungen dieser Kategorie gefunden wurden, wird in diesem Ansatz vorgesehen, sie direkt in das Programm als Liste hinein zu programmieren.

Daraufhin wird auf die Kategorie 2 getestet. Da die Kategorie an sich in zwei Unterkategorien unterteilt ist, muss man hier auch mehrere Prüfungen durchführen.

Zuerst soll auf die Nominalphrase mit einem Parser geprüft werden. Ein Parser für natürliche Sprache untersucht die grammatikalische Struktur von Sätzen und analysiert dabei beispielsweise, welche Wörter zusammen eine Phrase bilden. Um zu prüfen, ob sich die Nominalisierung in einer Nominalphrase befindet, muss ermittelt werden, ob eine Verbindung, die auf eine Nominalphrase hinweist, zwischen der Nominalisierung und verbundenen Wörter existiert. Diese Verbindungen kann man beispielsweise in der Abbildung 4.7 von *selection* zu *channel* und zu *headphones* sehen.

Der Parser erstellt zur grammatikalischen Struktur eine repräsentierende Baumstruktur. Bei der Verarbeitung werden statt den Syntaxbäumen, die oben in der Erläuterung zu den Kategorien gezeigt wurden, die direkten Abhängigkeiten (*dependencies*) der verschiedenen Wörter im Satz, die der Parser ermittelt, verwendet. Für die einfachere Verarbeitung werden dabei die sogenannten *collapsed CC-processed dependencies* ermittelt. Dabei werden bei Abhängigkeiten Präpositionen, Konjunktionen und ähnliches aufgelöst, damit eine direkte Verbindung zwischen den durch diese Wörter verbundenen Satzteile existiert. Ein Beispiel ist die Verbindung *prep_of* zwischen *selection* und *headphones*, die in Abbildung 4.7 zu sehen ist. Ohne die *collapsed dependencies* wäre in diesem Fall eine Verbindung vom Typ *prep* zwischen *selection* und *of* und eine weitere Verbindung zwischen *of* und *headphones*.

Diese direkten Abhängigkeiten bringen uns die benötigten Informationen, um auf Nominalphrasen schließen zu können. Dabei sind sie für unsere Zwecke einfacher zu nutzen als die Syntaxbäume.

In der Abbildung 4.7 ist die Nominalisierung *selection* der Elternknoten zu den spezifizierenden Kindknoten *channel* und *selection*. Da sowohl Kindknoten der Nominalisierung, wie es dort der Fall ist, als auch Elternknoten die Nominalisierung spezifizieren können,

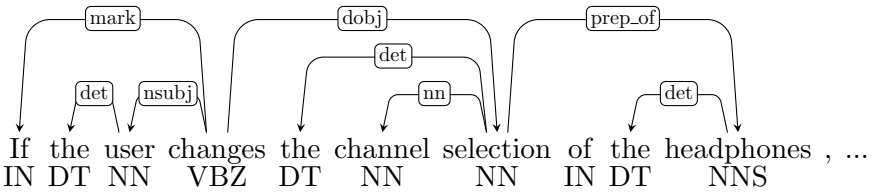

Abbildung 4.7: Abhängigkeiten im Satz nach dem Stanford-Parser

muss beides überprüft werden.

Sind ein Knoten und die Nominalisierung miteinander verbunden und jeweils als Nomen markiert, dann liegt eine Nominalphrase vor. Dabei muss sichergegangen werden, dass die Nominalphrase auch wirklich die Nominalisierung spezifiziert, da beispielsweise eine Verbindung mit der Präposition *by* (*prep_by*-Verbindung), wie sie beispielsweise in „selection by the user“ vorkommt, nur das Subjekt der Nominalisierung anhängt, dieses aber, wie bereits beschrieben, nicht ausreicht.

Nach dem Parser-Test müssen noch die Wörter geprüft werden, die an anderer Stelle im Satz stehen und die Nominalisierung spezifizieren können. Dafür werden Ontologien zur Hilfe genommen. Diese liefern die Information, welche Argumente das Wort benötigt, um ausreichend spezifiziert zu sein, weshalb die eingesetzten Ontologien diese Informationen liefern können müssen. Daraufhin wird der Satz auf diese Argumente untersucht.

Auf Nominalisierungen der Kategorie 3 wird in dieser Arbeit vorerst kein Augenmerk gelegt. Bei der Einführung der Kategorien wurde bereits angemerkt, dass die Festlegung auf einen Kontext bei Nominalisierungen der Kategorie 3 nicht trivial ist und daher grundsätzlich eine Gefahr darstellt. Der Aufwand der Erkennung der Kategorie 3 steht der geringen Anzahl an Nominalisierungen gegenüber, die nach der Analyse letztendlich der Kategorie 3 zuzuordnen sind. Daher werden Kategorie 3 und Kategorie 4 Nominalisierungen gleich behandelt und davor gewarnt, wenn der letzte Test nicht ergab, dass die Nominalisierung unproblematisch ist.

Hier könnte man jedoch auch weiter ansetzen und mit der Prüfung der Nominalisierungen auf Kategorie 3 fortsetzen. Neben der schon oben erwähnten domänenspezifischen Wortliste wäre eine Idee, Anforderungen in einem bestimmten Umfeld der Nominalisierung auf diese Nominalisierung zu untersuchen. Dort muss die Nominalisierung ausreichend spezifiziert vorkommen, indem dort die Nominalisierung mitsamt aller zusätzlich benötigter Informationen steht. Gleichzeitig muss verifiziert werden können, dass beide Existenzen der Nominalisierung sich auf das selbe beziehen. Trifft beides zu, ist die Nominalisierung unproblematisch. Hier könnte man zur Bestimmung des Kontextes auch die Idee mit den *V-dictionaries* aus der Arbeit von Lami [Lam05], die in Kapitel 3.1.2.1 kurz erläutert wird, verwenden.

Der Vorteil des obigen Entwurf ist die Unabhängigkeit von bereichsspezifischen Ontologien. Die Erstellung von Ontologien benötigt jeweils Zeit. Spezifische Ontologien haben das Potential dabei zu helfen, unproblematische Nominalisierungen zu erkennen und falsche Warnungen zu unterdrücken. Der obige Ansatz ist jedoch allgemein einsetzbar und direkt in andere Domänen übertragbar. Tiefer gehende Evaluationen müssen zeigen, ob domänenspezifische Ontologien und Ansätze hier Vorteile bringen.

5. Implementierung

Die zu verarbeitenden Lastenhefte werden aus DOORS im Textformat extrahiert, dann formatiert und anschließend automatisch in RESI eingefügt und geprüft. Dafür wurde ein Formatierer geschrieben, der die exportierten Dokumente in die gewünschte Form bringt. Für den automatisierten Start wurde der *RESIStarter* entwickelt.

Das Klassendiagramm des RESIStarters ist in Abbildung 5.1 zu sehen. Der graue Teil ist dabei schon in RESI vorhanden gewesen. Der grüne Teil ist im Zuge dieser Arbeit entstanden und dient dem automatischen Start von RESI und der verbesserten Überprüfung auf Nominalisierungen. Der Starter nutzt dabei den SpecificationImprover. Dieser hält die verschiedenen Einstellungen und die verbesserte Regel für Nominalisierungen (*RuleImprovedCheckForNominalization*), die von der ursprünglichen Regel (*RuleCheckForNominalization*) erbt. Beim Verbesserungsprozess nutzt der SpecificationImprover dann die Regeln und verschiedenen Einstellungen. Dabei wird die verbesserte Regel aufgerufen, die die beiden Klassen *CompleteProcessWordTest* und *ParserTest* nutzt, um die Überprüfungen durchzuführen.

5.1 Formatierer

Die aus DOORS exportierten Textdateien, die mir vorlagen, mussten noch vor der eigentlichen Verarbeitung formatiert werden. Der Formatierer liest die Textdateien ein und vorverarbeitet diese, damit sie die richtige Form für den Analyseprozess haben und überprüft werden können. So werden nicht benötigte Zusatzinformationen entfernt, die von RESI und den benötigten vorangestellten Werkzeugen nicht verarbeitet werden können. Dazu zählen beispielsweise, neben dem Kürzel der Anforderungskategorie und der Anforderungsnummer, das Wort *NEWLINE*, das am Ende einer jeden Anforderung steht. Beim Einlesen der einzelnen Zeilen werden noch zusätzlich Zeichen entfernt oder angepasst, bei denen das Werkzeug *plaintextemf*, das eine Textdatei in die gewünschte Form für das in RESI eingesetzte Dateiformat bringt, Probleme hat. Zeichen wie + oder @ konnten es beispielsweise zum Abbruch der Verarbeitung führen und mussten dementsprechend ersetzt werden. In den Anforderungen wurden teilweise verschiedene Symbole für Aufzählungen und ähnliches benutzt, die mit diesem Formatierer vereinheitlicht wurden. Dadurch hat man zum einen einheitliche Zeichen, zum anderen waren einige dieser Zeichen nur in bestimmten Textkodierungen verfügbar und damit nicht praktikabel. Bei dem ganzen Prozess wurde darauf geachtet, dass keine wichtigen Informationen verloren gehen.

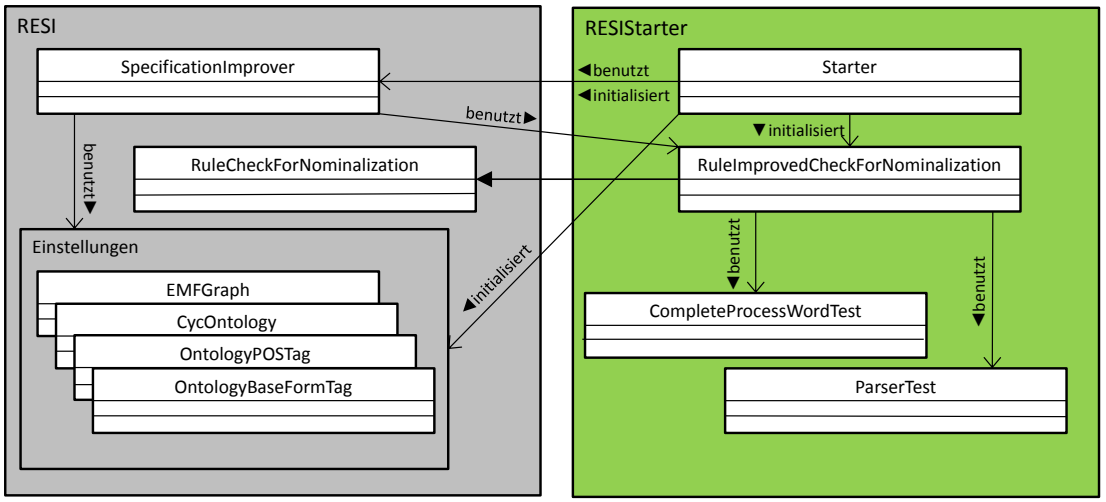

Abbildung 5.1: Klassendiagramm des RESIStarters

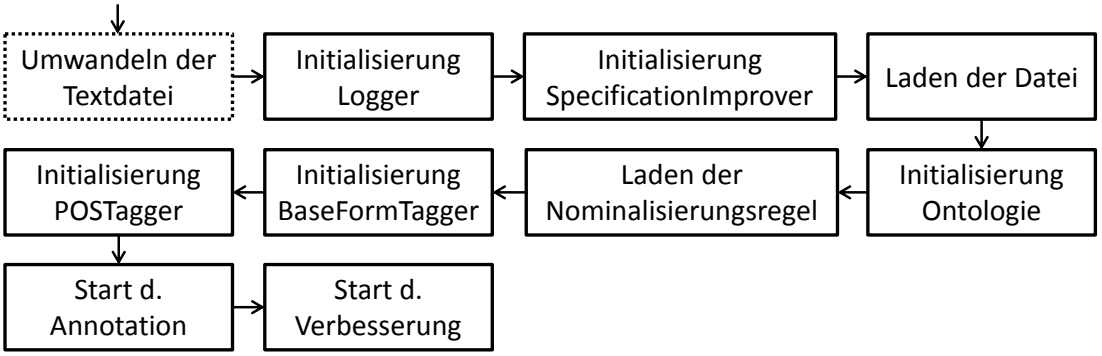

Abbildung 5.2: Ablauf des Starts des Verbesserungsprozess von RESI mit dem RESIStarter

Anforderungen, die lediglich vertragliche Inhalte besaßen, haben eine bestimmtes Anforderungskürzel. Da die vertraglichen Bestandteile explizit durch die Rechtsabteilung geprüft werden sollte und gleichzeitig für unsere Zwecke wenig ergiebig sind, wurden diese Anforderungen entfernt.

Ein Beispiel für die Formatierung einer Anforderung ist die Zeile „///ABCD-3659### requirement///For fine optimizations and verification an additional cycle in the B-phase car will be permitted. “, bei der der erste Teil entfernt und die in die Zeile „For fine optimizations and verification an additional cycle in the B-phase car will be permitted“ umgewandelt wurde.

Damit gleich eine größere Anzahl an Textdateien formatiert werden kann, ist der Formatierer in der Lage sowohl einzelne Textdateien als auch alle Textdateien in einem Ordner nacheinander zu formatieren. Der Formatierer ist unabhängig vom restlichen Teil wie dem RESIStarter und muss extra aufgerufen werden. Sollten die Anforderungsdokumente schon in einer Form vorliegen, die von RESI verarbeitet werden kann, kann der Aufruf des Formatierers so leicht übersprungen werden.

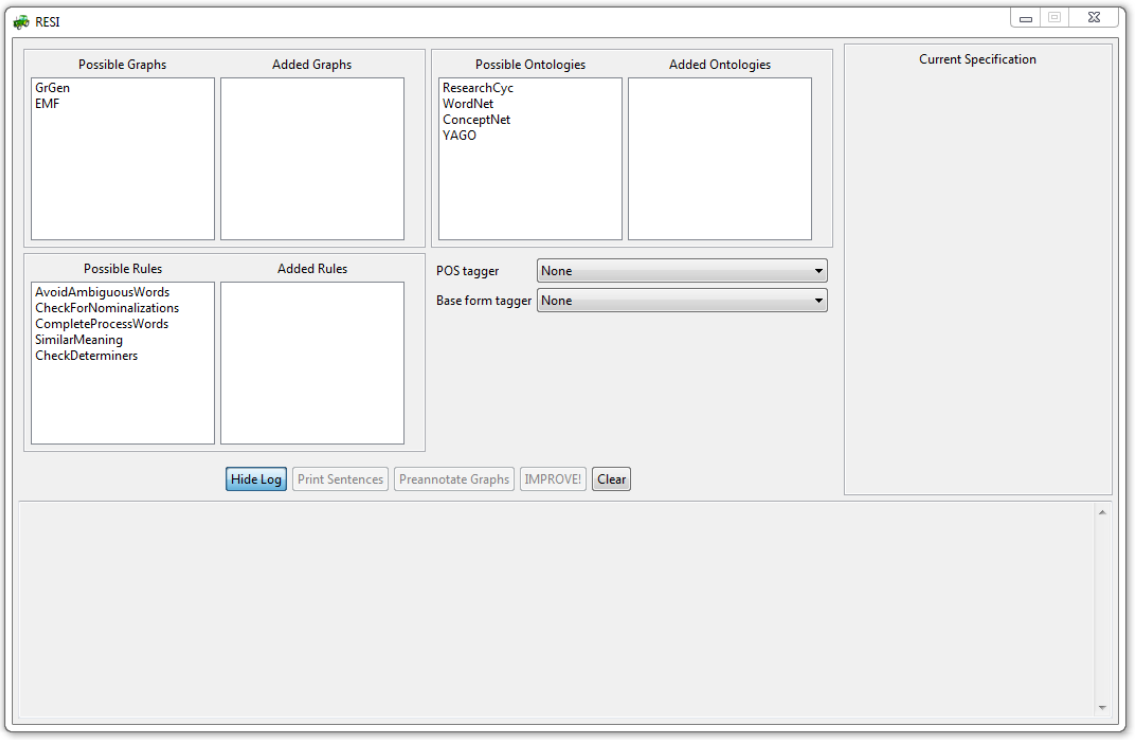

Abbildung 5.3: Grafische Benutzeroberfläche von RESI mit den verschiedenen Einstellungsmöglichkeiten

5.2 Automatisierter RESI-Starter

RESI besitzt eine grafische Benutzeroberfläche, die in Abbildung 5.3 zu sehen ist. In der grafischen Benutzeroberfläche kann der Benutzer auswählen, welche Einstellungen und Regeln er benutzen will, bevor mittels verschiedener Schaltflächen die Vorannotation des Graphen und der Verbesserungsprozess gestartet werden kann. Dieser Start wurde automatisiert, da in unserem Fall immer die gleichen Grundeinstellungen nötig sind. Der Prozess ist in der Abbildung 5.2 dargestellt.

Wird dem Starter eine Textdatei übergeben, so wird diese zuerst in das XMI-Format für das RESI-interne Dateiformat umgewandelt. Ist die übergebene Datei bereits im XMI-Format, wird dies übersprungen.

Im Anschluss wird ein eigener Logger initialisiert, damit Ergebnisse und Informationen über die gefundenen Nominalisierungen direkt in eine Protokolldatei geschrieben werden können.

Dann werden die von RESI benötigten Klassen initialisiert und Einstellungen geladen. Die Einstellungen, wie beispielsweise die Adresse der Ontologien, stammen dabei teilweise aus der beim Programm liegenden *config.xmi*-Datei, die bereits bei dem originalen RESI existierte und von dort übernommen wurde.

Zunächst wird der *SpecificationImprover* initialisiert. Ihm werden später die meisten Objekte und Einstellungen übergeben.

Als nächstes wird der *EMFGraph* für das interne Dateiformat mit dem Pfad zur (umgewandelten) XMI-Datei erstellt. Damit wird das Anforderungsdokument in das Programm geladen.

Als Ontologie wird die CycOntologie mit der entsprechenden Server-Adresse und dem entsprechenden Port geladen.

Um dem Programm ein Regelwerk für die Nominalisierungen mitzugeben, wird die verbesserten Nominalisierungsregel geladen, die noch in Kapitel 5.3 näher beschrieben wird. Hier gibt es die Option, dass bei gesetzter boolescher Variable zusätzlich die alte Nominalisierungsregel geladen werden kann. RESI führt dann beide Regeln nacheinander aus, was den Vorteil hat, dass die Ergebnisse beider Prüfungen besser verglichen werden können, da die selben Benutzereingaben zugrunde liegen.

Als letztes werden der Stanford-*POSTagger* mit den gewünschten Modellen (*english bidirectional distsim*) und Konfigurationen und der BaseFromTagger mit der Adresse und dem Port zum WordNet-Server initialisiert. Diese werden benutzt, um die geladenen Sätze der Datei zu annotieren.

Nachdem alle Klassen geladen wurden, wird überprüft, ob die Objekte jeweils korrekt erstellt werden konnten, also beispielsweise keine NullPointer existieren, bevor mit der Verarbeitung der Eingabe begonnen wird. Die Verarbeitung beginnt mit der Annotation der geladenen Sätze durch den *POSTagger*. Dann startet der SpecificationImprover das Verbessern des übergebenen Dokuments mit den geladenen Regeln.

Wenn alle Regeln abgearbeitet wurden und das Dokument analysiert wurde, werden letzte Informationen, wie die Anzahl an Nominalisierungen, in die Protokolldateien geschrieben und das Programm wird beendet.

5.3 Erkennung unproblematischer Nominalisierungen

Die Erkennung unproblematischer Nominalisierungen greift ein, nachdem von RESI erkannt wurde, dass ein Substantiv eine Nominalisierung ist. Die Übersicht über den gesamten Ablauf sieht man in der Abbildung 4.6 aus dem vorigen Kapitel. Dabei wird jeweils der gesamte Vorgang unterbrochen, sobald ein Test das Ergebnis lieferte, dass die Nominalisierung ausreichend spezifiziert ist.

In der beschriebenen Implementierung wurde jeweils Rücksicht darauf genommen, dass das Subjekt der Nominalisierung nicht benötigt wird und auch nicht alleine ausreicht, um sie zu spezifizieren. Die Mechanismen dafür sind jedoch leicht zu ändern, sollte man in zukünftigen Analysen zu einem anderen Schluss kommen.

5.3.1 Wortlisten-Test

Für die Erkennung von Nominalisierungen der Kategorie 1 wird, wie im Entwurf geplant, ein Wortvergleich durchgeführt. Momentan wird dafür lediglich auf das Wort *troubleshooting* geprüft. Es ist jedoch denkbar und möglich, mit einer größeren Anzahl an Wörtern in dieser Kategorie die Implementierung anzupassen und auf Listen, Hashmaps, Datenbanken oder ähnliches zurückzugreifen.

5.3.2 Parser-Test

Der Parser-Test für die Erkennung der Nominalphrasen in Kategorie 2 findet mit Hilfe des Stanford-Parsers statt. Der Stanford-Parser stammt aus der *CoreNLP*-Bibliothek, in der auch der *POSTagger*, der von RESI bereits genutzt wird, stammt. Daher ist es naheliegend auf den Parser des gleichen Pakets zurückzugreifen.

Der Parser-Test ist in zwei kleinere Tests unterteilt, um einmal die Verbindungen zu überprüfen, die auf die Nominalisierung zeigen (*Parent-Test*), und die, auf die die Nominalisierung an sich zeigt (*Children-Test*).

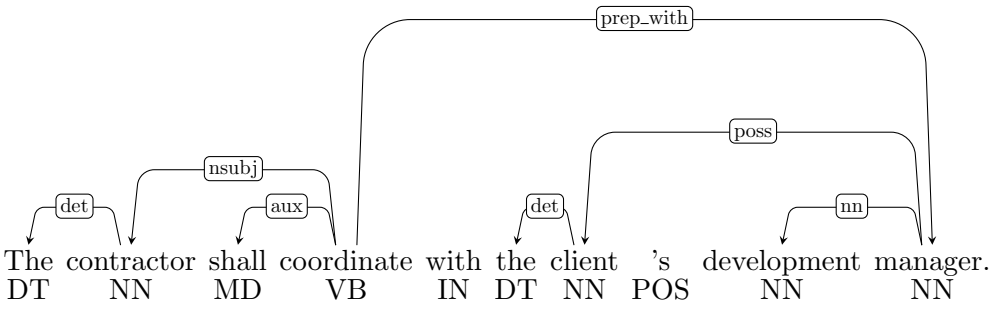

Abbildung 5.4: Beispiel für einen Satz mit einer Nominalphrase, die die Nominalisierung *development* zusammen mit dem Elternknoten bildet

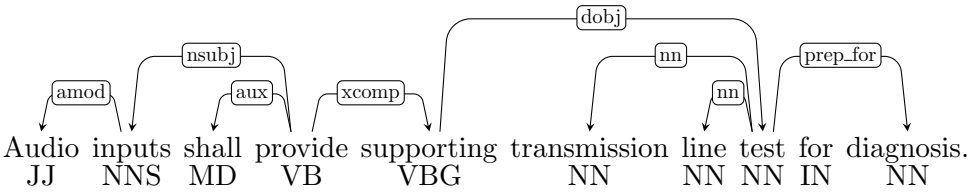

Abbildung 5.5: Nominalphrase mit einer prep_for-Beziehung zwischen Elternknoten und Nominalisierung

Parent-Test

Beim Parent-Test wird überprüft, ob der Elternknoten der Nominalisierung als Nomen markiert wurde, da dadurch eine Nominalphrase gebildet werden kann. Erkenntlich wird das im Parser durch die Markierungen, die *NN* enthalten. In dem Beispiel aus Abbildung 5.4 bildet der Elternknoten *manager* zusammen mit der Nominalisierung *development* eine Nominalphrase.

Außerdem wird geprüft, ob die grammatikalische Beziehung zwischen dem Elternknoten und der Nominalisierung nicht *prep_for* oder *prep_of* ist, da dadurch keine ausreichend spezifizierende Nominalphrase gebildet wird. Deutlich wird das in dem Beispielsatz: „Audio inputs shall provide supporting transmission line test for diagnosis“. Die Verarbeitung dieses Satzes durch den Parser ist in Abbildung 5.5 zu sehen. Die Nominalisierung *diagnosis* ist durch die Präposition *for* mit dem Wort *test* verbunden und beide sind als *NN* markiert. Alleine durch das Wort *test* wird die Nominalisierung jedoch nicht ausreichend spezifiziert, da durch die Präposition der Elternknoten näher spezifiziert wird und nicht die Nominalisierung. Das selbe gilt für die Präposition *of* beispielsweise in dem Ausdruck „at the beginning of the development“, weshalb auch Verbindungen mit dieser Präposition zum Elternknoten ausgenommen werden.

Children-Test

Der Children-Test läuft ähnlich ab. Es wird überprüft, ob ausgehend von der Nominalisierung mindestens ein Kindknoten als Kennzeichen für ein Nomen ein *NN* in der Markierung enthält. In der Abbildung 5.6 sieht man ein Beispiel dafür. Die Kindknoten *channel* und *headphones* bilden jeweils mit der Nominalisierung *selection* eine Nominalphrase, wodurch die Nominalisierung spezifiziert wird.

Nominalisierungen werden nicht alleine durch das Subjekt, auf das sie sich beziehen, spezifiziert. Daher wird zusätzlich geprüft, ob die grammatikalische Beziehung *prep_by* zwischen beiden existiert. In Abbildung 5.7 ist ein Beispiel dafür zu sehen. Der Nominalisierung

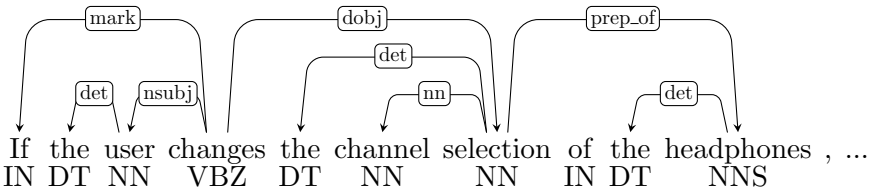

Abbildung 5.6: Beispiel für eine Nominalisierung (*selection*), um die eine Nominalphrase gebildet ist

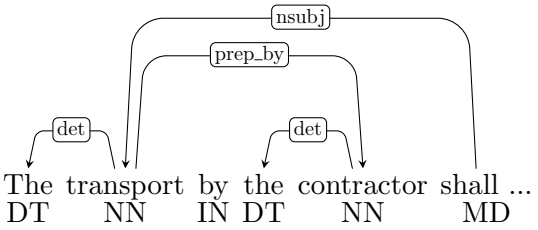

Abbildung 5.7: Nominalphrase mit einer prep_by-Beziehung zwischen Nominalisierung und Kindknoten

transport fehlt in diesem Fall das Objekt, also die Information, was genau transportiert werden soll.

5.3.3 *CompleteProcessWord*-Test

RESI besitzt bereits eine Regel, die Verben auf ihre Vollständigkeit prüft. Dabei wird mit Hilfe der Ontologie ermittelt, welche Argumente das Verb benötigt, um spezifiziert zu sein. Der Satz wird auf diese Argumente untersucht und in einem Fenster wird dem Nutzer angezeigt, welches Wort untersucht wird und welche Argumente benötigt werden. Dabei werden dem Benutzer auch Vorschläge für die Argumente gegeben.

Diese Regel wird nun auch für die Nominalisierungen genutzt. Dafür muss zuerst die Nominalisierung in ein Verb umgewandelt werden. Die nötigen Informationen existieren bereits. Sie stammen aus der Ontologie und wurden bei der ersten Überprüfung auf Nominalisierung ermittelt und gespeichert. Nachdem der Satz analysiert wurde und dem Benutzer das Fenster gezeigt wird, kann der Benutzer die Auswahl noch einmal bei Bedarf korrigieren. Das Auswahlfenster hierfür kann man in Abbildung 5.8 sehen. Nachdem die Auswahl bestätigt wurde, wird sie analysiert. Damit eine Nominalisierung ausreichend spezifiziert ist, müssen alle Argumente zugewiesen, also keine *null*-Werte zurückgegeben worden sein. *Null*-Werte entstehen, wenn der Nutzer *None of the above* im Auswahlfenster auswählt. Wie bereits bei anderen Tests wird hier erneut das Subjekt der Nominalisierung ausgenommen, indem ignoriert wird, ob es spezifiziert ist.

Nach dem *CompleteProcessWord*-Test können noch weitere Tests eingebaut werden, die beispielsweise auf Kategorie 3 testen. Dies wurde jedoch aus Gründen, die in Kapitel 4.2 erläutert wurden, nicht implementiert.

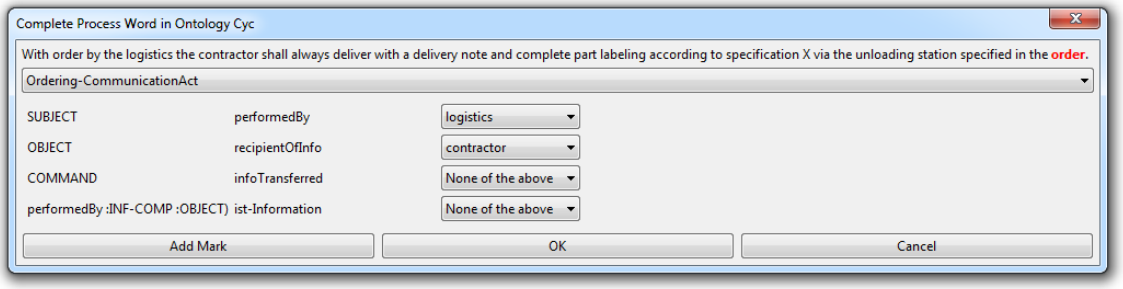

Abbildung 5.8: Fenster des *CompleteProcessWord*-Tests, in dem die Argumente ausgewählt werden können

6. Evaluation

In der Analyse der Lastenhefte kam heraus, dass durch eine Implementierung, die die Nominalisierungen der Kategorie 2 filtert, über 80% der Warnungen unterdrückt werden können. Nun soll geprüft werden, wie effektiv die Implementierung ist und wie groß die Verbesserung tatsächlich ist. Dafür wurden erneut Lastenhefte der Daimler AG geprüft. Das Prüfungskorpus umfasste zehn Lastenhefte mit einem Umfang von insgesamt knapp 60.000 Wörtern. Für den besseren Vergleich wurden simultan die neue, verbesserte Regel und die alte Regel aus RESI angewandt. Die Verbesserung ist in der Abbildung 6.1 zu sehen und zeigt, dass ein Großteil der Warnungen tatsächlich unterdrückt werden konnte.

Tabelle 6.1 zeigt die Details der Evaluation. Bei dem obigen Testkorpus konnte mit Hilfe der Erkennung unproblematischer Nominalisierungen die Anzahl an Meldungen sogar um über 88% reduziert werden.

Die Implementierung arbeitet mit den von RESI gefundenen Nominalisierungen, weshalb bei der Ausbeute davon ausgegangen wird, dass RESI zuvor alle Nominalisierungen gefunden hat. Die Präzision konnte dabei um 56.75% verbessert werden. Die Ausbeute liegt dabei gesamt bei über 88%.

Gerade beim Vergleich der Verteilung der Nominalisierungen der verschiedenen Kategorien, die man in der Abbildung 6.2 sehen kann, mit der ursprünglichen aus Abbildung 4.3 aus der Analyse, erkennt man, dass die meisten Nominalisierungen der Kategorie 2 gefiltert werden konnten und nun größtenteils Nominalisierungen der Kategorie 3 übrig blieben.

Die Analyse der nicht gefilterten Nominalisierungen der Kategorie 2 ergab, dass dies meist an schlecht formulierten Sätzen, schlechtem Englisch oder Schreibfehlern lag. Durch Anforderungen mit einer besseren rein sprachlichen Qualität könnte man die Präzision also noch weiter verbessern.

Außerdem gab es an einigen Stellen Probleme mit der korrekten Erkennung durch den Parser in Verbindung mit unglücklich gewählten Formulierungen. In dem Satz „After the highest current selection the selection should go to the lowest level“ wurde beispielsweise das entscheidende Wort *current*, das die Nominalisierung *selection* spezifizieren würde, vom Parser nicht in der Bedeutung *Spannung*, sondern mit der Bedeutung *aktuell* verarbeitet. Dadurch wurde diese Nominalisierung bei diesem Test als nicht ausreichend spezifiziert gewertet.

Ein anderes Beispiel ist in dem folgenden Satz zu finden: „The design of the wheel sensor must ensure that a tyre can be easily mounted and removed during a tyre change without

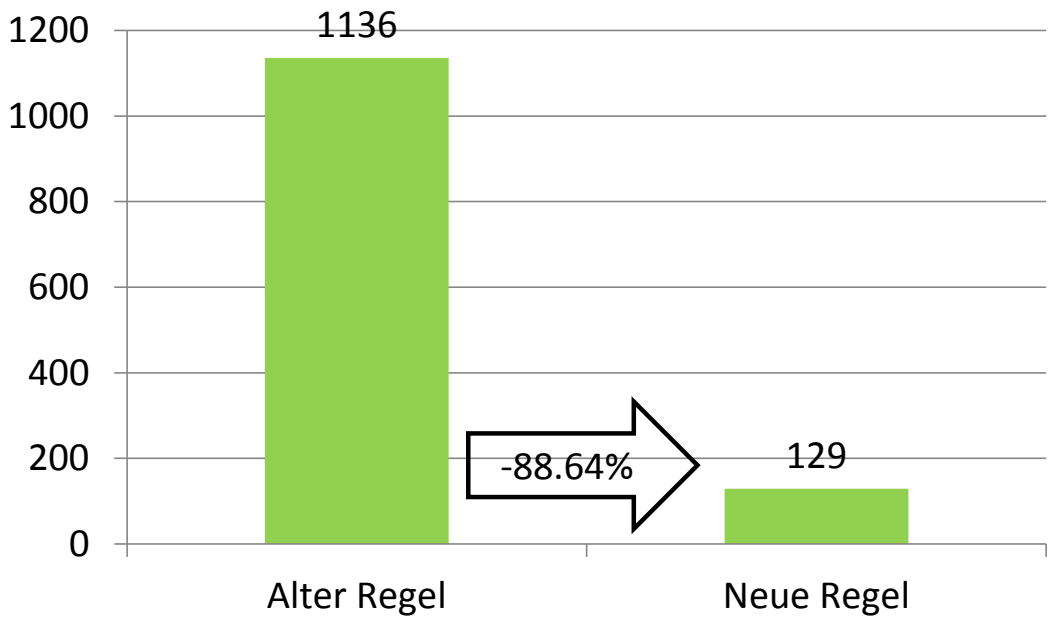

Abbildung 6.1: Darstellung der Anzahl an Warnungen: Vergleich der alten und der neuen Regel mit der Erkennung unproblematischer Nominalisierungen

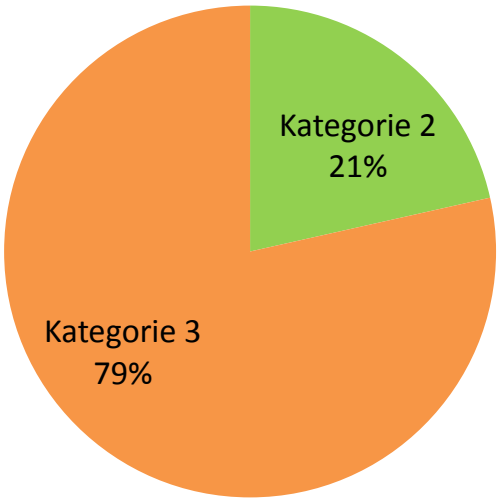

Abbildung 6.2: Darstellung der prozentualen Anzahl an Warnungen auf Nominalisierungen nach der neuen Regel mit Erkennung unproblematischer Nominalisierungen

Tabelle 6.1: Evaluation der Implementierung mit verschiedenen Lastenheften. Für die Ausbeute wurde angenommen, dass die von RESI bei 100% liegt.

		gemeldete Nomin.		problem.	Präzision			Ausbeute
	Wörter	Alte Impl.	Neue Impl.	Nomin.	Alt	Neu	Verbesserung	Neu
Dok. 4	3.687	81	5	2	2,47%	40,00%	37,53%	100,00%
Dok. 5	1.598	15	2	1	6,67%	50,00%	43,33%	100,00%
Dok. 6	6.069	108	6	4	3,70%	66,67%	62,96%	100,00%
Dok. 7	12.580	246	12	3	2,03%	25,00%	22,97%	60,00%
Dok. 8	7.403	167	34	20	13,77%	58,82%	45,05%	86,96%
Dok. 9	2.923	28	4	4	14,29%	100,00%	85,71%	100,00%
Dok. 10	8.098	130	17	13	13,08%	76,47%	63,39%	76,47%
Dok. 11	2.590	57	10	7	12,28%	70,00%	57,72%	100,00%
Dok. 12	10.444	243	26	21	9,47%	80,77%	71,30%	91,30%
Dok. 13	4.094	61	13	9	14,75%	69,12%	54,48%	100,00%
Gesamt	**59.486**	**1136**	**129**	**84**	**8,36%**	**65,12%**	**56,75%**	**88,42%**

damage to the wheel sensor.“ Hier ist das Wort *tyre* in seiner britischen Schreibweise geschrieben. Der benutzte Parser erkennt jedoch nur die amerikanische Version *tire*, wodurch die Nominalisierung *change* in dem Satz nicht korrekt als spezifiziert angesehen wird.

Ein weiteres Problem war, dass durch die Vorverarbeitung durch das *plaintextemf*-Werkzeug, das die Textdateien in das XMI-Format für das interne Dateiformat umwandelte, viele Sonderzeichen verloren gegangen sind. Unter anderem fehlten dadurch Klammern, wodurch dem Parser weitere Probleme bereitet wurden.

Zusätzliche Probleme existieren beim *CompleteProcessWords*-Test. Dort kann in seltenen Fällen das Wort durch die Umwandlung in das Verb im Satz nicht korrekt markiert werden, wodurch die Prüfung dieser Nominalisierung abbricht.

Diese Fehler sorgten jedoch meist nur dafür, dass eine Nominalisierung fälschlicherweise als problematisch angesehen und dadurch dem Nutzer gemeldet wurde. Dadurch wird die Präzision gesenkt, aber es werden weniger Nominalisierungen fälschlicherweise gefiltert. Die Anzahl an Fehlermeldungen konnte dennoch stark reduziert werden.

In wenigen Fällen wurde jedoch fälschlicherweise eine Nominalisierung gefiltert, die mindestens der Kategorie 3 angehört und daher nicht ausreichend spezifiziert sein könnte. Die Implementierung ist abhängig von der Güte der Ergebnisse, die von der Vorverarbeitung, dem Parser und RESI erzeugt werden. Der Entwurf und die Implementierung funktionieren zuverlässig, wenn die Sätze eine ausreichende Sprachqualität haben und die benutzten Werkzeuge dadurch korrekte Ergebnisse liefern. Beispielsweise erkennt RESI das Wort *implementation* grundsätzlich nicht als Nominalisierung, da es nicht korrekt in der benutzten Ontologie zugeordnet werden kann. Daher kann auch diese eigentliche Nominalisierung nicht geprüft werden.

Ein Beispiel für eine Nominalisierung, vor der nicht gewarnt wird, da der Satz nicht gut formuliert ist, ist in folgendem Satz zu finden: „Other automotive wireless systems including systems for broadcast and data transfer must not be disturbed by the product“. Der Parser zieht eine Verbindung zwischen *broadcast* und *transfer* und die Nominalisierung wird dadurch als ausreichend spezifiziert angesehen. Der Satz in diesem Ausdruck ist aber nicht eindeutig genug. Es ist zu vermuten, dass der Autor dieses Satzes ein eher deutsches Englisch (*Denglisch*) verwendet und der Satz dadurch eine etwas andere Bedeutung hat.

Die Überprüfung der gefilterten Nominalisierungen ergab, dass exakt elf Nominalisierungen fälschlicherweise gefiltert wurden, wovon einige in mehrmals gleich auftretenden Anforderungen vorkamen. Die Ursache lag dabei in jedem Fall bei der schlechten Sprachqualität der Sätze oder ungeschickten Formulierungen und den dadurch bedingten fehlerhaften Ergebnissen des Parsers. Eine weitere Ursache ist die falsche Eingabe von Daten durch den Benutzer bei Interaktionen wie dem *CompleteProcessWords*-Test. Durch eine Wiederholung des Tests mit den korrekten Eingaben traten die fehlerhafte Filterungen nicht mehr auf. Die Ausbeute kann daher ebenfalls durch eine bessere sprachliche Qualität der Sätze gesteigert werden.

Von den übrig gebliebenen Nominalisierungen sind exakt neun von RESI fälschlicherweise als solche erkannt worden und konnten bei den Tests ebenfalls nicht gefiltert werden.

7. Zusammenfassung und Ausblick

In dieser Arbeit wurde eine Klassifizierung für die Beurteilung von Kritikalität von Nominalisierungen in natürlichsprachigen Anforderungsdokumenten erstellt. Sowohl die manuellen Analyse als auch die Analyse mit RESI ergab, dass ein Großteil der Nominalisierungen der unproblematischen Kategorie 2 angehört und damit viele Nominalisierungen weniger kritisch sind als bisher angenommen.

Es wurde daraufhin eine Methode zur Erkennung von Nominalisierungen der unproblematischen Kategorien 1 und 2 erstellt, die zur Filterung von unproblematischen Nominalisierungen in RESI eingebaut werden konnte. Die Implementierung hat gezeigt, das dies mit einfachen Mitteln möglich ist und die Ergebnisse dabei sehr vielversprechend sind. In einem Prüfungskorpus mit zehn Lastenheften aus dem Fahrzeugumfeld der Daimler AG mit einem gesamten Umfang von knapp 60.000 Wörtern konnten über 88% der Nominalisierungen gefiltert werden, wobei eine gesamte Präzision von 65,12% erreicht werden konnte. Diese Präzision kann aber weiter gesteigert werden, wenn die Sätze jeweils eine ausreichend gute Sprachqualität aufweisen. Dennoch kommt die Erkennung von Nominalisierungen im Rahmen dieser Arbeit in einen Bereich, in dem die Benutzer nicht durch die große Anzahl an falsch-positiven (*false positive*) Warnungen abgeschreckt werden und das Werkzeug akzeptieren könnten.

In dieser Arbeit wurde angenommen, dass das Subjekt, auf das sich die Nominalisierung bezieht, stets eindeutig ist. Dieses Ergebnis stammt aus der Analyse der vorliegenden Anforderungsdokumente. Hierbei könnte es sich um ein Phänomen handeln, das aufgrund der Domäne des Fahrzeugumfelds oder durch das Unternehmen Daimler AG auftritt. Diese Annahme müsste daher mit anderen Anforderungsdokumenten aus anderen Bereichen und anderen Unternehmen bestätigt werden. Sollte sich herausstellen, dass das Subjekt durchaus problematisch sein kann, dann muss darauf im Entwurf und vor allem in der Implementierung geachtet und dementsprechende Änderungen getätigt werden.

Ebenso wurde in dieser Arbeit die automatisierte Erkennung von Nominalisierungen der Kategorie 3 nicht angegangen. Dafür müsste festgelegt werden, wann der Kontext eindeutig ist und als solcher zuverlässig automatisiert festgestellt werden kann. Daraufhin kann man versuchen Nominalisierungen, die durch den Kontext spezifiziert werden, zu erkennen und als unproblematisch einzustufen. Daher wurden in dieser Arbeit Nominalisierungen der Kategorie 3 genauso problematisch behandelt wie Nominalisierungen der Kategorie 4.

Weiterhin gibt es noch Raum für Verbesserungen an RESI. Der Umgang ist in dem Sinne nicht nutzerfreundlich genug, dass momentan noch viele Abfragen durch den Benutzer

benötigt werden. Damit der Mechanismus zum Erkennen von Nominalisierungen beispielsweise bei der Daimler AG Anklang findet, sollte der Prozess jedoch möglichst automatisiert laufen und an möglichst wenigen Stellen eine Benutzerinteraktion fordern.

Viele der Abfragen, um die Bedeutung des Wortes in der Ontologie zu bestimmen, sind auch nicht unbedingt nötig. Beispielsweise wird für die Erkennung von Nominalisierungen die Bedeutung in der Ontologie für jedes Substantiv nachgefragt. Dabei sind jedoch nur ein Bruchteil der Substantive solche, die in einer bestimmten Bedeutung eine Nominalisierung sein können. Daher wäre es eine Verbesserung, wenn nur noch Substantive hinterfragt werden, die grundsätzlich auch eine Nominalisierung sein könnten, da sie in einer ihrer Bedeutungen eine Nominalisierung sein könnten. Dafür müsste jedoch die Architektur von RESI für diesen Einsatz angepasst werden, da man nun zuerst jede Bedeutung des Wortes auf eine Nominalisierung abfragen müsste, um zu entscheiden, ob der Nutzer gefragt werden muss. Eine weitere Möglichkeit die Anzahl an Nachfragen zu senken wäre der Einsatz von Listen für die Bedeutung in der Domäne. Dadurch könnte man sich die Nachfrage und die Überprüfung in der Ontologie für einige Wörter sparen. Eine dritte Möglichkeit hierfür wäre der Einsatz von bereichsspezifischen Ontologien, die vermutlich auch die Erkennungsrate verbessern können.

Zuletzt sollte man nicht vergessen, dass Verbesserungen an den benutzten Werkzeugen und Ontologien auch die Erkennung von unproblematischen Nominalisierungen verbessern. Daher sollte auch ein Fokus hierauf gelegt werden.

Literaturverzeichnis

[3SL] 3SL: *Cradle*. http://www.threesl.com/index.php

[AKSS13] Annervaz, K.M. ; Kaulgud, V. ; Sengupta, S. ; Savagaonkar, M.: Natural language requirements quality analysis based on business domain models. In: *Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on*, 2013, S. 676–681

[Ben83] Benington, Herbert D.: Production of Large Computer Programs. In: *Annals of the History of Computing* 5 (1983), Oct, Nr. 4, S. 350–361. http://dx.doi.org/10.1109/MAHC.1983.10102. – DOI 10.1109/MAHC.1983.10102. – ISSN 0164–1239

[Bor] Borland: *Caliber*. http://www.borland.com/Products/Requirements-Management/Caliber

[Bun] Bundesministerium des Innern (BMI): *V-Modell XT*. http://www.v-modell-xt.de/

[Cyca] Cycorp: *Cyc*. http://www.cyc.com/

[Cycb] Cycorp: *ResearchCyc*. http://www.cyc.com/platform/researchcyc

[Deu09] Deutsches Institut für Normung e.V. (Hrsg.): *Projektmanagement – Projektmanagementsysteme – Teil 5: Begriffe*. 9. Berlin, Wien, Zürich : Beuth Verlag GmbH, 2009 http://www.beuth.de/en/standard/din-69901-5/113428752

[DHM98] Dröschel, Wolfgang ; Heuser, Walter ; Midderhoff, Rainer: *Inkrementelle und objektorientierte Vorgehensweisen mit dem V-Modell 97*. Oldenbourg, 1998. – ISBN 978–3–486–24276–8

[Dud] Duden: *Substantivierung*. http://www.duden.de/rechtschreibung/Substantivierung#b2-Bedeutung-2

[FLGS14] Ferrari, A. ; Lipari, G. ; Gnesi, S. ; Spagnolo, G.O.: Pragmatic ambiguity detection in natural language requirements. In: *Artificial Intelligence for Requirements Engineering (AIRE), 2014 IEEE 1st International Workshop on*, 2014, S. 1–8

[Hei10] Heidenreich, Martin: Metriken und Werkzeugunterstützung zur Überprüfung von Anforderungen. In: *Objektspektrum* (2010)

[Hes02] Hesse, Wolfgang: *Ontologie(n)*. https://www.gi.de/service/informatiklexikon/detailansicht/article/ontologien.html. Version: 2002

[Hof12] Hoffmann, A.: A Pattern-based approach for analysing requirements in socio-technical systems engineering. In: *Requirements Engineering Conference (RE), 2012 20th IEEE International*, 2012. – ISSN 1090–750X, S. 341–344

[Hou14] Houdek, Dr. F.: Erfahrungen beim Einsatz automatisierter Lastenheft-Qualitätsbewertungen. In: *REConf2014*, 2014

[IBM] IBM Software: *Rational DOORS*. http://www-03.ibm.com/software/products/en/ratidoorfami

[IEE90] IEEE Standard Glossary of Software Engineering Terminology. In: *IEEE Std 610.12-1990* (1990), Dec, S. 1–84. http://dx.doi.org/10.1109/IEEESTD.1990.101064. – DOI 10.1109/IEEESTD.1990.101064

[KB09] Körner, S. J. ; Brumm, T.: RESI - A Natural Language Specification Improver. In: *Semantic Computing, 2009. ICSC '09. IEEE International Conference on*, 2009, S. 1–8

[Kör14] Körner, Sven J.: *RECAA - Werkzeugunterstützung in der Anforderungserhebung*, Karlsruher Institut für Technologie, Fakultät für Informatik (INFORMATIK), Institut für Programmstrukturen und Datenorganisation (IPD), Diss., 2014. http://dx.doi.org/10.5445/KSP/1000039460. – DOI 10.5445/KSP/1000039460

[Kri14] Krisch, Jennifer: Kontextbasierte lexikalische Kontrolle von Anforderungsdokumenten. In: Abel, Andrea (Hrsg.) ; Vettori, Chiara (Hrsg.) ; Ralli, Natascia (Hrsg.): *Proceedings of the 16th EURALEX International Congress*. Bolzano, Italy : EURAC research, jul 2014. – ISBN 978–88–88906–97–3, S. 647–656

[Lam05] Lami, Giuseppe: QuARS: A Tool for Analyzing Requirement / Software Engineering Institute, Carnegie Mellon University. Version: 2005. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=7681. Pittsburgh, PA, 2005 (CMU/SEI-2005-TR-014). – Forschungsbericht

[LDL98] Lamsweerde, A. van ; Darimont, R. ; Letier, E.: Managing conflicts in goal-driven requirements engineering. In: *Software Engineering, IEEE Transactions on* 24 (1998), Nov, Nr. 11, S. 908–926. http://dx.doi.org/10.1109/32.730542. – DOI 10.1109/32.730542. – ISSN 0098–5589

[LLP10] Liu, Lin ; Li, Tong ; Peng, Fei: Why Requirements Engineering Fails: A Survey Report from China. In: *Requirements Engineering Conference (RE), 2010 18th IEEE International*, 2010. – ISSN 1090–705X, S. 317–322

[LMP04] Luisa, Mich ; Mariangela, Franch ; Pierluigi, Inverardi: Market Research for Requirements Analysis Using Linguistic Tools. In: *Requir. Eng.* 9 (2004), Februar, Nr. 1, 40–56. http://dx.doi.org/10.1007/s00766-003-0179-8. – DOI 10.1007/s00766–003–0179–8. – ISSN 0947–3602

[Mic96] Mich, L.: NL-OOPS: from natural language to object oriented requirements using the natural language processing system LOLITA. In: *Natural Language Engineering* 2 (1996), 6, 161–187. http://dx.doi.org/null. – DOI null. – ISSN 1469–8110

[Par08] Partsch, Prof. Dr. H.: *Modell-basiertes Requirements Engineering: alles schon da gewesen (?)*. http://2008.reconf.de/fileadmin/PDF_Dateien/REConf_2008/Vortraege/KEYNOTE_Prof_Partsch.pdf. Version: 2008

[PD] Patig, Susanne ; Dibbern, Jens: *Requirements Engineering*. http://www.enzyklopaedie-der-wirtschaftsinformatik.de/wi-enzyklopaedie/lexikon/is-management/Systementwicklung/Hauptaktivitaten-der-Systementwicklung/Problemanalyse-/Requirements-Engineering/index.html

[PR11] Pohl, Klaus ; Rupp, Chris: *Basiswissen Requirements Engineering - Aus- und Weiterbildung zum "Certified Professional for Requirements Engineering" ; Foundation Level nach IREB-Standard*. 3. korrigierte Auflage. Dpunkt-Verlag, 2011. – ISBN 978–3–898–64771–7

[Pri] Princeton University: *WordNet*. `http://wordnet.princeton.edu/`

[RP92] Rolland, C. ; Proix, C.: A natural language approach for Requirements Engineering. Version: 1992. `http://dx.doi.org/10.1007/BFb0035136`. In: Loucopoulos, Pericles (Hrsg.): *Advanced Information Systems Engineering* Bd. 593. Springer Berlin Heidelberg, 1992. – DOI 10.1007/BFb0035136. – ISBN 978–3–540–55481–3, 257-277

[RS09] Rupp, Chris ; SOPHISTen, die: *Requirements-Engineering und -Management - professionelle, iterative Anforderungsanalyse für die Praxis*. 5. aktualisierte und erweiterte Auflage. Hanser, 2009. – ISBN 978–3–446–41841–7

[Rya93] Ryan, K.: The role of natural language in requirements engineering. In: *Requirements Engineering, 1993., Proceedings of IEEE International Symposium on*, 1993, S. 240–242

[SDGDG13] Sadoun, D. ; Dubois, C. ; Ghamri-Doudane, Y. ; Grau, B.: From Natural Language Requirements to Formal Specification Using an Ontology. In: *Tools with Artificial Intelligence (ICTAI), 2013 IEEE 25th International Conference on*, 2013. – ISSN 1082–3409, S. 755–760

[SOP14] SOPHISTen, Die: *Die kleine RE-Fibel*. Version: 2014. `https://www.sophist.de/fileadmin/SOPHIST/Puplikationen/Broschueren/RE-Broschuere_Komplett_Final.pdf`

[SS09] Staab, Steffen ; Studer, Rudi: *Handbook on Ontologies*. 2nd. Springer Publishing Company, Incorporated, 2009. – ISBN 3540709991, 9783540709992

[SSUE09] Stöckel, Frank ; Stolz, Philip ; Uddin, Ifthaker ; Endriss, Larissa: DESIRe - Dynamic Expert System for Improving Requirements / HOOD Group. 2009. – Forschungsbericht

[YDRG+12] Yang, Hui ; De Roeck, A. ; Gervasi, V. ; Willis, A. ; Nuseibeh, B.: Speculative requirements: Automatic detection of uncertainty in natural language requirements. In: *Requirements Engineering Conference (RE), 2012 20th IEEE International*, 2012. – ISSN 1090–750X, S. 11–20