Peter Dencker et. al. (Hrsg.): Automotive Safety & Security 2017,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 121

Ontologiebasierte Abhängigkeitsanalyse im Projektlastenheft

Konstantin Zichler[1] und Steffen Helke[2]

Abstract: Zu Beginn eines Projekts dokumentieren interdisziplinäre Domänen-Experten die Anforderungen an alle Lebensphasen eines Nutzfahrzeugs und die entsprechenden Realisierungskonzepte im Projektlastenheft. Die Kenntnis der Abhängigkeiten zwischen Anforderungen bietet den Vorteil, fehlerhafte Produktkonzepte bereits in der frühen Projektphase zu vermeiden. Bei der Durchführung von Abhängigkeitsanalysen besteht für die Experten der einzelnen Abteilungen die Schwierigkeit darin, von den dokumentierten Einzelbeiträgen auf domänenübergreifende Abhängigkeiten zwischen den Anforderungen zu schließen. Bisher werden diese Analysen für gewöhnlich manuell durchgeführt, da es dafür kaum Werkzeugunterstützung gibt. Wir stellen ein neuartiges Verfahren vor, bei dem das für die Abhängigkeitsanalyse erforderliche, fachspezifische Wissen zu einer gemeinsamen Wissensbasis in Form einer Ontologie aggregiert wird. Zusammen mit Axiomen, einem Reasoner und Werkzeugen aus dem Natural Language Processing wird eine automatisierte Abhängigkeitsanalyse im Projektlastenheft realisiert, mit der es möglich ist, bisher nicht berücksichtigte Abhängigkeiten zwischen Anforderungen zu identifizieren.

Keywords: Anforderungsmanagement, Abhängigkeitsanalyse, Ontologie, Reasoner, Natural Language Processing.

1 Einleitung

Die Entwicklung von Nutzfahrzeugen startet gewöhnlich nach einem Frontloading-Prinzip. Ziel bei diesem Ansatz ist es, bereits in einer frühen Phase der Produktentstehung Fahrzeugkonzepte so zu entwickeln, dass diese im späteren Projektverlauf kaum noch verändert werden müssen. Dies wird dadurch erreicht, dass alle relevanten Marktanforderungen an ein Produkt von Beginn an bei der Konzeption berücksichtigt werden. Dieses Vorgehen birgt vor allem den Vorteil, dass nach Abschluss der Frontloading-Phase nur noch über ökonomisch sinnvolle und technisch realisierbare Alternativen von Fahrzeugkonzepten entschieden werden muss. Dadurch werden viele Verzögerungen im weiteren Prozessverlauf vermieden und die gesamte Produktentwicklung stabiler und kostengünstiger.

Als Voraussetzung für die Entwicklung marktgerechter Produkte gilt das Anforderungsmanagement. In der frühen Phase des Projekts erheben Projektteams, bestehend aus interdisziplinären Domänen-Experten, zunächst die Anforderungen an alle

[1] Daimler AG, T/OGP, Mercedesstr. 132/1, 70546 Stuttgart, konstantin.zichler@daimler.com
[2] Brandenburgische Technische Universität Cottbus-Senftenberg, steffen.helke@b-tu.de

Lebensphasen eines Nutzfahrzeugs. Jeder Domänen-Experte definiert die Anforderungen, die aus Sicht der eigenen Abteilung erforderlich sind. Dabei handelt es sich um funktionale und nichtfunktionale Anforderungen. Anschließend entwickelt das Projektteam auf Basis der Anforderungen Realisierungskonzepte und bestätigt die technische Machbarkeit, sowie die Wirtschaftlichkeit des Projekts.

Die frühe Phase in der Produktentstehung von Nutzfahrzeugen erfordert insbesondere eine präzise Abstimmung zwischen den Domänen-Experten, wobei unterschiedlichste Einflussfaktoren berücksichtigt werden müssen. Als maßgebliches Instrument dient dabei das Projektlastenheft. Darin dokumentieren Projektteams Anforderungen, Realisierungskonzepte und andere für die Projektabwicklung wichtige Informationen. Das Projektlastenheft dient vor allem als Kommunikationsgrundlage zwischen den Domänen-Experten und soll ein gemeinsames Verständnis für das angestrebte Projektziel schaffen. Um ein Verständnis der Inhalte für alle Projektteammitglieder gleichermaßen zu ermöglichen, werden Anforderungen und Realisierungskonzepte in natürlicher Sprache in Office-Dokumenten dokumentiert. Das Projektlastenheft ist ferner die Grundlage für den weiteren Spezifikationsprozess und die anschließende Produktentwicklung. Alle Abhängigkeiten zwischen den Anforderungen müssen durch sorgfältige Analysen identifiziert und bewertet werden, da sie zu Widersprüchen und damit fehlerhaften Fahrzeugkonzepten führen können.

Bisher suchen Projektteams manuell nach Abhängigkeiten im Projektlastenheft, da es hierfür kaum Werkzeugunterstützung gibt. Neben der Tatsache, dass das Projektlastenheft ein sehr umfangreiches Dokument ist, stellen die unterschiedlichen Perspektiven der Fachexperten dabei eine der wesentlichen Herausforderungen für eine erfolgreiche Abhängigkeitsanalyse dar. Das Wissen über Zusammenhänge im Projekt liegt verteilt vor, hauptsächlich in den Köpfen der Domänenexperten. Mit seinem Beitrag im Projektlastenheft dokumentiert der jeweilige Fachexperte nur einen Teil seines Wissens. Für das Projetteam besteht die Herausforderung folglich darin, von den dokumentierten Einzelbeiträgen auf domänenübergreifende Abhängigkeiten zwischen den Anforderungen zu schließen. Ein weiterer Faktor sind die impliziten Annahmen der einzelnen Projektteammitglieder. Pohl et al. verweisen bspw. darauf, dass der häufigste Grund für unvollständige Anforderungen falsche Annahmen der Stakeholder sind. Sie setzten z.B. voraus, dass bestimmte Informationen selbstverständlich sind und deshalb gar nicht explizit genannt werden müssen [PR11].

Anhand des folgenden Beispiels wird diese Problematik deutlich. Ein Plug-in-Hybrid-Fahrzeug soll auf der Grundlage einer vorhandenen Fahrzeugplattform entwickelt werden. Das Projektteam entscheidet sich bei der Definition des Realisierungskonzepts dafür, das Bordnetz der vorhandenen Fahrzeugplattform zu übernehmen. Wenn das Bordnetz zuvor in einem Fahrzeug mit Verbrennungsmotor eingesetzt wurde, wäre die Lebensdauer des Bordnetzes für ein Plug-in-Hybrid-Fahrzeug nicht ausreichend. Der Grund dafür ist, dass Teile des Bordnetzes eines Plug-in-Hybrid-Fahrzeugs auch im Stillstand des Motors aktiv sind, nämlich während der Beladung des Akkumulators über eine externe Stromquelle. In diesem Beispiel muss das Projektteam also die

Abhängigkeiten zwischen den Anforderungen an die Nutzung des Plug-in-Hybrid-Fahrzeugs, wie das Fahren und das Laden des Akkumulators, und den Anforderungen an die Lebensdauer des Bordnetzes ausreichend berücksichtigen. Da diese Anforderungen aber in einem realen Projekt von unterschiedlichen Domänen-Experten aus Vertrieb, Entwicklung und Qualität erhoben werden, können bei einer manuellen Analyse solche Abhängigkeiten auch leicht unerkannt bleiben.

Um derartige Anforderungen besser aufdecken zu können, schlagen wir vor, das verteilte Wissen über die Produktentstehung von Nutzfahrzeugen in einer Wissensbasis zu konzentrieren und basierend auf den gesammelten Informationen domänenübergreifende Abhängigkeiten zwischen Anforderungen abzuleiten. Das Wissen soll dabei mit Hilfe von Ontologien repräsentiert werden. Eine Ontologie ist eine formale, explizite Spezifikation einer gemeinsamen Konzeptualisierung [SBF98]. Sie kann als eine Wissensbasis verstanden werden, die das Wissen einer Anwendungsdomäne beschreibt. Mit Hilfe von Methoden zur Schlussfolgerung ist es außerdem möglich, neue, implizite Informationen aus dem bereits spezifizierten Wissen abzuleiten [HKR+08]. Für die Prüfung der Ontologie auf Konsistenz und zur Ableitung von implizitem Wissen werden Inferenzmaschinen (engl. *Reasoner*) eingesetzt.

Der Einsatz von Ontologien im Anforderungsmanagement wurde in der Vergangenheit bereits in einigen Arbeiten vorgestellt, vgl. dazu [FMK+11], [SB16], [SP14], [Si14] und [SHS14]. Dabei fokussieren diese Ansätze eher auf detaillierte und vor allem technische Anforderungen, wie sie für Feinspezifikationen im späteren Verlauf der Produktentstehung typisch sind. Demgegenüber enthält ein Projektlastenheft vor allem grobe Anforderungen und strategische Zielbeschreibungen an alle Produktlebensphasen. Das bedeutet, dass dabei auch Anforderungen aus den Abteilungen, wie Vertrieb, Produktion, Logistik und After-Sales berücksichtigt werden müssen. Schraps und Peters [SP14] schränken in ihrer Arbeit bspw. die Satzstruktur der Anforderungen bereits bei ihrer Formulierung durch eine formale Grammatik ein. Die so formulierten Anforderungen werden mit Hilfe von semantischen Mustern in eine Ontologie überführt. In der Arbeit von Farfeleder et al. [FMK+11] werden Schablonen (engl. *boilerplates*) zur standardisierten Beschreibung von Anforderungen verwendet. Über eine graphische Schnittstelle wird vom Nutzer zunächst eine vorgegebene Schablone ausgewählt. Anschließend bekommt er für die Befüllung der Leerfelder vom Werkzeug Vorschläge, die aus einer Anforderungsontologie stammen. Solche Vorgehen erfordern grundsätzlich gut strukturierte Spezifikationsdokumente und viel Erfahrung der Nutzer im Umgang mit Anforderungsmanagement. Die frühe Phase in der Produktentstehung erfordert jedoch Lastenhefte, deren Erstellung möglichst flexibel gehandhabt werden kann, da große Teile eines Fahrzeugkonzepts durch Beschluss wieder entfallen können. Insbesondere durch die häufigen Änderungen am Fahrzeugkonzept würde eine exakte Dokumentation ohnehin nur unnötig die Arbeit der Domänenexperten behindern. Daher kann im vorliegenden Anwendungsfall von einer Struktur, wie sie bspw. bei den Spezifikationen mit hohem Detaillierungsgrad vorhanden ist, nicht ausgegangen werden. Eine Modellierung der Anforderungen in Form einer Ontologie durch Domänen-Experten muss aufgrund des zu hohen Aufwands ebenfalls ausgeschlossen werden.

Aus diesem Grund schlagen wir für die Abhängigkeitsanalyse im Projektlastenheft eine Kombination aus einer Ontologie in Verbindung mit einem Reasoner und Natural Language Processing (NLP) vor. Nach unserem Konzept wird ein Teil der Ontologie, das sogenannte Domänenmodell, welches das für die Produktentstehung erforderliche Wissen enthält, von Mitarbeitern aus einem zentralen Projektmanagement Office manuell erstellt. Dieses Domänenmodell beschreibt vor allem die Klassenebene der Ontologie. Die Instanzebene des Domänenmodells soll unter Zuhilfenahme von NLP vollautomatisiert erstellt werden. Damit soll es möglich sein, in einem gegebenen Projektlastenheft eine volltautomatisierte Abhängigkeitsanalyse durchzuführen. Diese Analyse wird von einem zentralen Bereich durchgeführt. Das Projektteam bekommt eine Auswertung über die ermittelten Abhängigkeiten und kann diese bei der Konzeption des Fahrzeugs berücksichtigen. Da es für diesen Anwendungsfall noch keine dedizierte Werkzeugunterstützung gibt, nutzen wir Plug-ins innerhalb der bekannten Architekturen Protégé [Mu15] und GATE [CTR+13], um die grundsätzliche Machbarkeit einer vollautomatischen Abhängigkeitsanalyse im Projektlastenheft nachzuweisen.

Das Papier gliedert sich wie folgt. In Abschnitt 2 werden die Grundlagen zu Ontologien und NLP erläutert. In Abschnitt 3 stellen wir unseren Ansatz für eine ontologiebasierte Abhängigkeitsanalyse anhand einer prototypischen Werkzeugkette vor. In Abschnitt 4 beschreiben wir erste Experimente mit der prototypischen Werkzeugkette und diskutieren die Ergebnisse. Dabei weisen wir die grundsätzliche Machbarkeit einer ontologiebasierten Abhängigkeitsanalyse im Projektlastenheft nach. Abschnitt 5 befasst sich mit verwandten Arbeiten. Unsere Ergebnisse fassen wir in Abschnitt 6 zusammen und geben darüber hinaus einen Ausblick über unser weiteres Vorgehen.

2 Grundlagen

Ziel dieses Kapitels ist es, Grundlagen über Aufbau, Erstellung und Anwendung von Ontologien zu vermitteln, sowie bekannte Methoden und Werkzeuge aus dem Natural Language Processing (NLP) vorzustellen.

2.1 Erstellung und Anwendung von Ontologien

Der größte Mehrwert von Ontologien in Verbindung mit Methoden zur Schlussfolgerung ist die Ableitbarkeit von neuem Wissen. Die Grundlage dafür ist eine Wissensbasis – die Ontologie. Eine Ontologie beschreibt das Wissen einer Anwendungsdomäne durch die Begriffe (Konzepte), die innerhalb dieser Domäne genutzt werden und ihre Beziehungen (Relationen) zueinander. In Abb. 1 ist eine schematische Darstellung einer Ontologie zu sehen. Dabei sei zunächst auf die Klassen der Ontologie verwiesen, die hier mit schwarzen Kreisen markiert sind. Klassen beschreiben die übergeordneten Begriffe einer Domäne. Klassen können Unterklassen besitzen. Im abgebildeten Beispiel hat die Klasse *Vehicle* eine Unterklasse *Distribution*. Diese Unterklasse spezifiziert eine bestimmte Art eines Fahrzeugs, nämlich die eines Fahrzeugs für den Verteilerverkehr. Klassen können

über Relationen miteinander verknüpft werden. Zwei Klassen, und die sie verbindende Relation bilden ein sogenanntes Tripel. Das Beispiel in Abb. 1 zeigt bspw. das Tripel (*CANBus, isPartOfVehicle, Distribution*) und drückt damit aus, dass ein CAN-Bus immer in der *Distribution* eines Fahrzeugs enthalten sein muss. Eine konkrete Ausprägung einer Klasse wird als Instanz oder Individuum bezeichnet. Instanzen sind in Abb. 1 durch schwarze Rauten gekennzeichnet. *EBus2X* ist bspw. ein fiktiver Eigenname für eine Instanz der Klasse *CAN-Bus*.

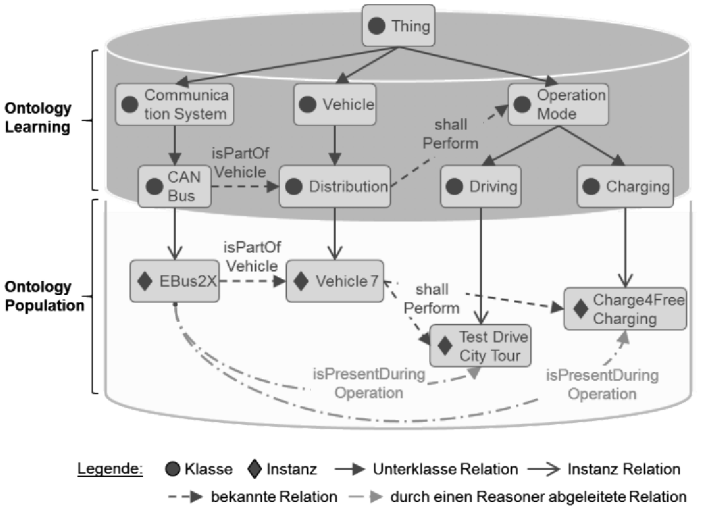

Abb. 1: Schematische Darstellung einer Ontologie

Ontologien werden in Ontologiesprachen beschrieben, wie bspw. OWL 2 (Web Ontology Language). OWL basiert auf der Prädikatenlogik erster Stufe [HKR+08].

Die Erstellung von Ontologien gliedert sich in zwei Teile, das *Ontology Learning* und das *Ontology Population* (siehe Markierungen in Abb. 1). *Ontology Learning* beschreibt die Erzeugung neuer Klassen (Konzepte) und Relationen in der Ontologie. Dadurch wird die innere Struktur der Ontologie erweitert. Demgegenüber zielt *Ontology Population* auf die Instanziierung dieser Klassen und Relationen [RMC+11].

Für gewöhnlich werden Ontologien manuell erstellt, was durchaus aufwändig sein kann. Für die Erstellung werden Ontologie-Editoren genutzt. Ein bekanntes Beispiel ist der Ontologie-Editor der Stanford University Protégé [Mu15]. Weiterhin gibt es Möglichkeiten, Ontologien oder Teile davon automatisiert zu erstellen. In der Vergangenheit wurden dazu NLP-Techniken innerhalb der General Architecture for Text Engineering (GATE) genutzt, vgl. dazu [WKR10] und [MFP09]. Dabei ist bisher vor allem die automatisierte Erstellung von flachen Klassenhierarchien mit nicht immer ausreichender Genauigkeit möglich. Im Gegensatz zum *Ontology Learning*, lässt sich *Ontology Population*, also die Anreicherung einer Ontologie mit Instanzen und den

Beziehungen zwischen ihnen, weitestgehend automatisieren.

In einer Ontologie liegt Wissen in einer formalisierten Form vor. Die Maschinenlesbarkeit dieses Wissen ermöglicht es, Schlussfolgerungen über die Inhalte der Ontologie automatisiert zu ziehen. Zu diesem Zweck werden die bereits erwähnten Reasoner genutzt. Sie prüfen die Wissensbasis auf Konsistenz und leiten neue Informationen aus dem bereits spezifizierten Wissen ab. Zu diesem Zweck werden Axiome und Ausdrücke definiert. Auch Relationen können genauer spezifiziert werden, um bspw. auszudrücken, dass zwei Klassen Synonyme sind, wenn diese in der betrachten Domäne dieselbe Semantik besitzen. In Abb. 1 kommt die Relation *isPresentDuringOperation* zweimal vor und ist im Gegensatz zu den anderen Relationen durch eine Strichpunktlinie dargestellt. Diese besondere Notation zeigt an, dass diese Relation auf der Grundlage des nachfolgenden Object-Subproperty-Axioms mit Hilfe eines Reasoners abgeleitet worden ist:

```
SubObjectPropertyOf(ObjectPropertyChain a:isPartOfVehicle
a:shallPerform)a:isPresentDuringOperation)
```

Dieses Axiom wird auch als *Property Chain* bezeichnet und sagt aus, dass beim direkten Aufeinanderfolgen von zwei vorgegebenen Relationen (im Beispiel *isPartOfVehicle* und *shallPerform*) die Endpunkte der entstehenden Kette mit einer dritten Relation (im Beispiel *isPresentDuringOperation*) zu verknüpfen sind.

2.2 Natural Language Processing

Die Automatisierung von *Ontology Population* erfolgt in der vorliegenden Arbeit mit NLP innerhalb von GATE. Die General Architecture for Text Engineering (GATE) ist eine Open Source Software, deren Hauptzweck darin besteht, Dokumente zu annotieren. Diese Annotationen können in GATE automatisch oder manuell erstellt werden. Die automatische Erstellung von Annotationen im Rahmen unseres Ansatzes erfordert unter anderem die folgenden Anwendungen, die in GATE als *Processing Resources* bezeichnet werden: *Tokeniser*, *Sentence Splitter*, *Part of Speech (POS) Tagger*, *Gazetteer*, *JAPE Transducer*. In [Cu14] werden diese wie folgt erläutert:

- *Tokeniser* spalten Text in sehr kleine Token, wie Nummern, Satzzeichen und Worte verschiedener Art.
- *Gazetteer* bestehen aus Listen, die Namen von Entitäten, wie bspw. Namen von Städten, Organisationen oder Wochentagen enthalten. Diese Listen werden dazu genutzt Begriffe wie Eigennamen im Text zu suchen (Named Entity Recognition). Alle Zeichenketten im Text, die mit dem Eigennamen einer der genutzten *Gazetteer*-Liste übereinstimmen, werden mit der Annotation Lookup markiert.
- *Sentence Splitter* segmentieren, wie der Name schon sagt, Text in Sätze. Dabei wird eine *Gazetteer*-Liste mit Abkürzungen verwendet, um diese von Satzzeichen zu unterscheiden, die ein Satzende markieren.

- *Part-of-Speech Tagger* erstellen Annotationen zu jedem Wort oder Symbol im Text, die angeben, um welche Wortart es sich bei dem jeweiligen Wort oder Symbol handelt.

Die so erstellten Annotationen können anschließend mit dem *JAPE Transducer* verändert werden. JAPE (Java Annotation Patterns Engine) erlaubt die Erkennung von Regulären Ausdrücken in Annotation, die auf einem Text erstellt wurden. Für die Anwendung von JAPE werden im ersten Schritt Grammatiken erstellt. Eine JAPE Grammatik besteht aus Phasen, von welchen jede aus Muster-Aktion-Regeln besteht. Die Phasen laufen sequentiell durch und stellen eine Kaskade von Transduktoren über Annotationen dar. Eine Regel hat eine linke und eine rechte Seite. Die linke Seite der Regeln beschreibt jeweils Muster von Annotationen, die gesucht werden sollen. Die rechte Seite besteht aus einer Anweisung für die Manipulation dieser Annotationen. Annotationen, die in das Muster der linken Seite der Regel passen, werden mit einem Label versehen. Dadurch kann auf der rechten Seite der Regel auf dieses Label Bezug genommen werden [Cu14]. Mit dem *JAPE Transducer* ist es möglich, bestehende Annotationen zu verändern oder nach Textbestandteilen zu suchen und diese zu annotieren. Nachfolgend ist eine Regel dargestellt, die im Text nach einem Token mit der Zeichenkette *Vehicle* gefolgt von einer Zahl sucht und diesen als *DistributionTruck* annotiert:

```
Rule: DistributionTruck
(
     {Token.string == Vehicle} {Token.kind == number}
):tag
-->
:tag.DistributionTruck = {rule = "DistributionTruck"}
```

Mit dieser Regel könnte bspw. nach Fahrzeugmodellen im Text gesucht werden. Alle *Processing Resources* werden in einer *Processing Pipeline* angeordnet und von GATE sequentiell ausgeführt. Am Ende liegt ein annotierter Text vor, der Aufschluss über die Semantik der Textbestandteile gibt. Diese Informationen können dazu verwendet werden, die Inhalte des Texts gezielt auszuwählen und weiterzuverarbeiten.

3 Ontologiebasierte Abhängigkeitsanalyse im Projektlastenheft

In diesem Kapitel präsentieren wir eine prototypische Umsetzung für die Abhängigkeitsanalyse im Projektlastenheft. Weiterhin stellen wir Überlegungen über die Einbindung des Lösungsansatzes in die organisatorischen Abläufe eines Industrieunternehmens an.

3.1 Prototypische Werkzeugkette

Für die Lösung der Aufgabenstellung verwenden wir eine prototypische Werkzeugkette,

die im Wesentlichen aus Protégé [Mu15], GATE [CTR+13] und dem OwlExporter [WKR10] besteht. Die Abhängigkeitsanalyse mit dieser Werkzeugkette erfordert ein bereits bestehendes Domänen-Modell in Form einer Ontologie. Dieses Domänen-Modell wird manuell erstellt (*Ontology Learning*) und besteht aus Klassen, Relationen, Axiomen und Ausdrücken. Die Begriffe (Konzepte) für die Ontologie stammen aus den an der Projektabwicklung beteiligten Domänen. Sie werden aus bereits vorhandenen Projektlastenheften und anderen Projektdokumenten gesammelt, in eine hierarchische Beziehung gebracht und über Relationen miteinander in Beziehung gesetzt. Grundsätzliche Zusammenhänge, wie bspw., dass alle Fahrzeuge auf Verkehrswegen fahren, werden über Axiome und Ausdrücke in OWL 2 formuliert und in der Ontologie hinterlegt. Die ontologiebasierte Abhängigkeitsanalyse unter Verwendung von NLP ist in Abb. 2 dargestellt. Nachfolgend werden die Schritte der Analyse näher beschrieben:

1. Das zuvor manuell erstellte Domänen-Modell wird als RDF/XML-Dokument exportiert und in einem Ordner gespeichert. Dieses Modell ist die Grundlage für die spätere *Ontology Population*.

2. Projektlastenhefte werden in GATE importiert. Es können gleichzeitig mehrere Dokumente importiert werden. Alle Dokumente werden automatisch in sogenannte GATE-Dokumente konvertiert.

3. Aus den GATE-Dokumenten wird ein Korpus erstellt. An dem Korpus erfolgt die anschließende sprachliche Analyse.

4. Der Korpus durchläuft die Komponenten *Document Reset PR*, *English Tokeniser*, *Gazetteer*, *Sentence Splitter*, *POS Tagger*, *NE Transducer* und OrthoMatcher aus der Anwendung ANNIE [CMB+02]. Die einzelnen Bestandteile des Textes werden entsprechend ihrer Zugehörigkeit von ANNIE annotiert. Als Ergebnis liegt ein annotierter Korpus vor. Zur besseren Übersichtlichkeit wurden die Komponenten für die sprachliche Analyse in Abb. 2 zu einem Schritt zusammengefasst.

5. Ausgehend von zuvor definierten JAPE-Regeln, sucht der *JAPE Transducer* nach Annotationen im Korpus. Die JAPE-Regeln sind Teil der OwlExporter Demo [WKR10] und wurden für den vorliegenden Anwendungsfall angepasst. Das Ziel dieses Schrittes ist es, Instanzen für die *Ontology Population* zu identifizieren. Der *JAPE Transducer* erstellt zusätzliche Annotationen, die die Begriffe für den anschließenden Export markieren.

6. Der OwlExporter, ein Java Plug-in, identifiziert die Instanzen und Relationen, die exportiert werden sollen, anhand der Annotationen im Korpus. Danach exportiert der OwlExporter Instanzen und Relationen in die Ontologie. Für diesen Schritt wird die zuvor erstellte Ontologie im RDF/XML-Format benötigt. Sie dient als Ausgangsdokument für die *Ontology Population*. Der OwlExporter positioniert die im Korpus gefundenen Instanzen und Relationen automatisch an die richtige Stelle in der Ontologie. Das Ergebnis ist eine Ontologie im RDF/XML-Format.

7. Die Ontologie wird mit Protégé geöffnet.
8. Der Reasoner Fact++ [TH06] wird nun verwendet, um die Konsistenz der Ontologie zu prüfen und um neue (implizite) Abhängigkeiten in der Ontologie abzuleiten.
9. Mit dem SPARQL Query Panel werden die Abhängigkeiten innerhalb der Ontologie abgefragt. Als Ausgabe werden sogenannte Tripel, also Instanz-Paare mit jeweils einer Relation abgefragt.

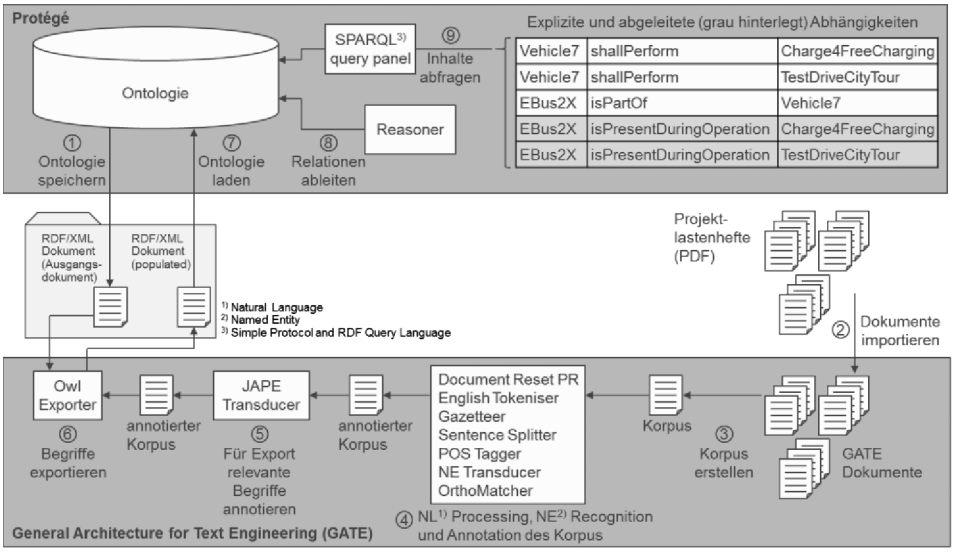

Abb. 2: Prototypische Werkzeugkette

3.2 Organisatorische Umsetzung

Die Randbedingungen in einem Industrieunternehmen lassen es nicht zu, dass Domänen-Experten eigenständig Wissen in Form einer Ontologie modellieren und sprachliche Analysen mit GATE durchführen. Neben dem Faktor Zeit spielen dabei auch noch die fehlenden Kenntnisse der Domänen-Experten im Bereich von Ontologien und NLP eine entscheidende Rolle. Aus diesem Grund schlagen wir vor, einen zentralen Bereich für Projektmanagementunterstützung, ein sogenanntes Project Management Office (PMO), mit der Durchführung der ontologiebasierten Abhängigkeitsanalyse zu beauftragen. In einigen Projekten der Fahrzeugentstehung übernimmt der Projektmanagementunterstützer (PMU) eine dem Requirements Engineer ähnliche Rolle. Er leitet das Projektteam bei der Erstellung des Projektlastenhefts an und hilft bei ausgewählten Tätigkeiten des Anforderungsmanagements. In seiner Querschnittsrolle hat er außerdem einen guten domänenübergreifenden Überblick und Zugriff auf erforderliche Projektunterlagen. Außerdem ist er als neutrale Person geeignet, um Schwachstellen im Konzept

aufzuzeigen, da er keine der am Projekt beteiligten Abteilungen vertritt und damit mit mehr Akzeptanz beim Aufzeigen von Fehlern rechnen kann. Eine organisatorische Umsetzung unseres Vorgehens ist in Abb. 3 dargestellt.

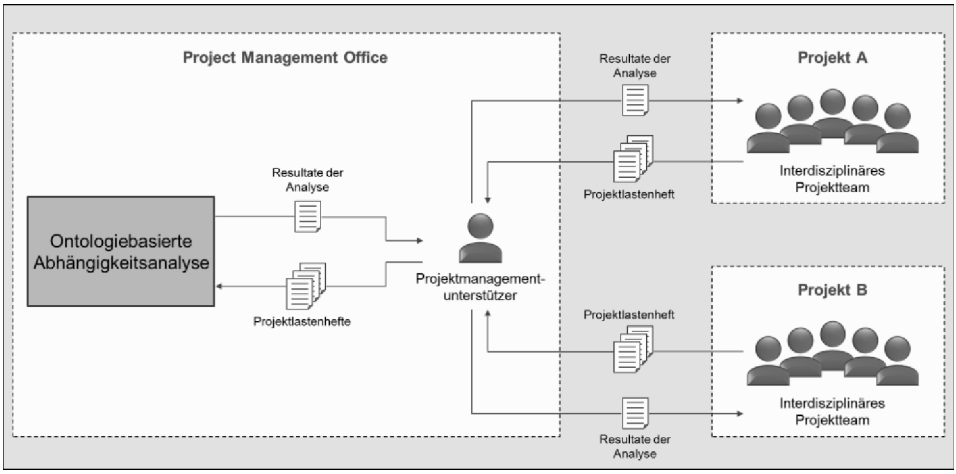

Abb. 3: Organisatorische Umsetzung der ontologiebasierten Abhängigkeitsanalyse

Dabei ist ein Projektmanagementunterstützer derjenige, der das Werkzeug für die ontologiebasierte Abhängigkeitsanalyse verwaltet. Er erstellt die Wissensbasis (Ontologie) und versorgt die Projektteams mit den Ergebnissen der ontologiebasierten Abhängigkeitsanalyse. Der Ablauf gliedert sich wie folgt:

1. Projektteams übergeben ihre Projektlastenhefte an den PMU.
2. Der PMU führt die Abhängigkeitsanalyse durch. Dabei filtert er die ermittelten Abhängigkeiten, um die Projektteams nicht mit unnötig vielen Informationen zu belasten.
3. Der PMU leitet die relevanten Ergebnisse an das jeweilige Projektteam weiter. Die Domänen-Experten werden über die Abhängigkeiten informiert und können Anforderungen ergänzen oder ändern und Anpassungen am Produktkonzept vornehmen.

4 Praktische Experimente und Diskussion der Ergebnisse

Zum Nachweis der prinzipiellen Machbarkeit unseres Ansatzes, haben wir einen Text aus fiktiven Anforderungen zusammengestellt. In Abb. 4 ist dieser Text bereits in GATE importiert. Dabei ist zu sehen, dass die für den Export relevanten Begriffe mit Hilfe der adaptierten OwlExporter Demo im Text identifiziert und für den Export mit der Annotation *OwlExportClassDomain* und *OwlExportRelationDomain* annotiert wurden.

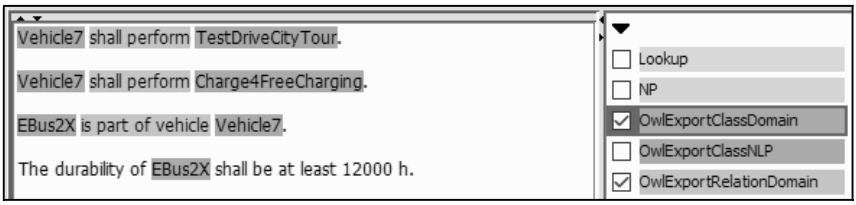

Abb. 4: Annotierte Begriffe für den Export mit OwlExporter Screenshot GATE

Im nächsten Schritt werden diese Begriffe von dem OwlExporter in das RDF/XML-Dokument exportiert (*Ontology Population*). Dieses Dokument enthält die angereicherte Ontologie, die im nächsten Schritt mit Protégé geladen wird.

In Abb. 5 ist das Ergebnis des *Ontology Population* und der Schlussfolgerung durch den Reasoner zu sehen. Die beiden Instanzen *EBus2X* und *Vehicle7* wurden zusammen mit der Relation *isPartOfVehicle* aus dem Text exportiert. Daneben wurden auch die Instanzen *TestDriveCityTour* und *Charge4FreeCharging* sowie die Relation *shallPerform* aus dem Text extrahiert, die hier aber aus Gründen der Übersichtlichkeit nicht dargestellt sind.

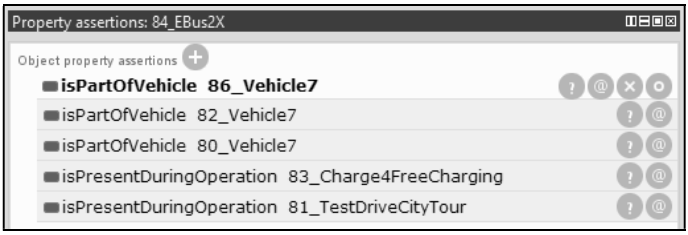

Abb. 5: Darstellung der Inferenz in Protégé

Aufgrund des zuvor definierten Axioms leitet der Reasoner aus dem vorhandenen Wissen, zusätzlich zwei neue Abhängigkeiten her:

```
(EBus2X, isPresentDuringOperation, TestDriveCityTour)
(EBus2X, isPresentDuringOperation, Charge4FreeCharging).
```

Diese zusätzlichen Relationen sind nun explizit und werden den Domänen-Experten angezeigt. Damit ist für das gesamte Projektteam klar, dass *EBus2X* durch diese beiden Operationen ebenfalls betroffen ist. Als Folge können die Domänen-Experten die Anforderung an die Lebensdauer des Bordnetzes und anschließend das Produktkonzept bereits in der frühen Projektphase anpassen.

Dieses Beispiel zeigt nur einen kleinen Ausblick dessen, was möglich ist. Die Instanzen *TestDriveCityTour* und *Charge4FreeCharging* könnten ihrerseits weitere Verknüpfungen zu anderen Einflussfaktoren aufweisen. Diese könnten ebenfalls automatisiert mit *EBus2X* und anderen Konzepten der Ontologie verknüpft werden, so dass sich noch mehr Abhängigkeiten ergeben. Durch diese Art der Analyse kann die

Qualität eines Produktkonzepts in der frühen Phase der Produktentstehung zusätzlich gesteigert werden. Der manuelle Aufwand, der für die Modellierung der Klassen und Relationen anfällt, ist dabei verhältnismäßig gering. Außerdem kann das manuell erstellte Domänen-Modell mehrfach wiederverwendet werden. Der größte Teil der Ontologie, wird mit Hilfe des *Ontology Population* automatisiert erstellt. Speziell die laufenden Änderungen auf Instanzebene, können mit Hilfe des *Ontology Population* ebenfalls automatisiert werden. Dieses Vorgehen erlaubt die Analyse von sehr umfangreichen Dokumenten in kürzester Zeit. Da Teile eines Nutzfahrzeugs häufig parallel in unterschiedlichen Projekten entwickelt werden, können so die Projektlastenhefte von mehreren Projekten gleichzeitig auf Abhängigkeiten analysiert werden. Damit könnten nicht nur domänenübergreifende, sondern auch projektübergreifende Abhängigkeiten gefunden werden.

Bevor jedoch diese Werkzeugkette genutzt werden kann, muss einige Vorarbeit geleistet werden. Das Domänen-Modell muss erstellt und validiert werden. Weiterhin müssen die NLP-Komponenten angepasst werden. Es müssen bspw. JAPE-Regeln formuliert und Gazetteer-Listen erstellt werden. Von der Qualität dieser Vorarbeit ist folglich auch die Trefferquote bei der sprachlichen Analyse im Text abhängig. Daneben bleibt der Aufwand für die Pflege und Wartung der Wissensbasis und der Werkzeuge für die sprachliche Analyse.

5 Verwandte Arbeiten

Schraps und Bosler stellen einen Ansatz vor, mit dem sie das Wissen aus Software-anforderungen in eine Anforderungsontologie überführen. Dazu werden die Anforderungen im ersten Schritt mit Hilfe von NLP-Techniken annotiert. Anschließend sucht eine Mustererkennung nach vordefinierten Mustern in der Grammatik der Anforderungen. Teile der Anforderungen, die in diese Muster fallen, werden entsprechend ihrer Semantik in die Anforderungsontologie überführt [SB16]. In ihrer Arbeit präsentiert Siegemund eine Methode für eine automatisierte Validierung und Messung des Anforderungswissens. Mit der prototypischen Implementierung OntoReq demonstriert Siegemund, wie unvollständige und inkonsistente Anforderungen und Qualitätsfehler mit Hilfe einer Anforderungsontologie automatisch identifiziert werden. Außerdem wird der Requirements Engineer bei der Lösung dieser Fehler durch wissensbasierte Vorschläge angeleitet [Si14]. Schraps und Peters haben eine Methode entwickelt, bei der Anforderungen mit Hilfe von semantischen Mustern in eine Ontologie überführt werden. Diese Ontologie nutzen sie anschließend für die Prüfung der Konsistenz innerhalb des Spezifikationsdokuments. Ihre Methode basiert nicht auf NLP. Stattdessen schränken sie die Satzstruktur der Anforderungen bereits bei der Formulierung durch eine formale Grammatik ein [SP14]. Soomro et al. präsentieren eine ontologiebasierte Visualisierung von unterschiedlichen Abhängigkeitsbeziehungen zwischen Anforderungen. In ihrem Ansatz behandeln sie speziell die Auswirkungen von Änderungen der Anwenderanforderungen auf andere Anforderungen. Soomro et al.

gehen davon aus, dass Software-Teams Abhängigkeiten zwischen Anforderungen übersehen, weil diese nicht deutlich genug dargestellt werden [SHS+14]. Farfelder et al. nutzen eine Ontologie, um qualitativ hochwertige Anforderungen bereits bei ihrer Erhebung zu formulieren. In ihrem System kann der Requirements Engineer über eine graphische Benutzerschnittstelle aus einer Menge von vorgegebenen Boilerplates wählen. Das System nutzt eine Domänenontologie, um eine Liste mit Vorschlägen für die Einzelteile einer Anforderung zur Verfügung zu stellen, die der Requirements Engineer für die Definition der Anforderungen nutzen kann [FMK+11].

Im Vergleich zu den hier vorgestellten Ansätzen bietet unsere Vorgehensweise die Möglichkeit, weitgehend vollautomatisch die Abhängigkeiten in Projektlastenheften zu finden. Werden mehrere Projektlastenhefte analysiert, können sogar Abhängigkeiten zwischen verschiedenen Projekten identifiziert werden. Unser Ansatz bietet den Vorteil, dass die Ersteller von Projektlastenheften kein spezielles Know-how zu *Natural Language Processing* oder *Ontology Engineering* benötigen und auch keine formale Grammatik erlernen müssen. Die Analyse wird in ein zentrales Project Management Office ausgelagert, wodurch eine Entlastung der Projektmannschaft erreicht wird. Eine solide sprachliche Qualität der Projektlastenhefte steigert die Leistungsfähigkeit unserer Methode. Wir arbeiten aktuell an Erweiterungen, die auch bei Rechtschreib- oder Grammatikfehlern akzeptable Ergebnisse liefern.

6 Zusammenfassung und Ausblick

In dem vorliegenden Papier wurde ein neuartiges Verfahren für die Abhängigkeitsanalyse im Projektlastenheft vorgestellt. Das für die Abhängigkeitsanalyse erforderliche, fachspezifische Wissen ist dabei zu einer gemeinsamen Wissensbasis in Form einer Ontologie aggregiert. Zusammen mit Axiomen, einem Reasoner und Werkzeugen aus dem Natural Language Processing wurde von uns eine automatisierte Abhängigkeitsanalyse im Projektlastenheft realisiert, mit der es möglich ist, bisher nicht berücksichtigte Abhängigkeiten zwischen Anforderungen zu identifizieren. Mit unseren Ergebnissen aus ersten Tests haben wir die grundsätzliche Machbarkeit des Verfahrens nachgewiesen. Zukünftig gilt es sicherzustellen, dass das im Domänen-Modell hinterlegte fachspezifische Wissen korrekt ist. Aus diesem Grund arbeiten wir derzeit an einem Verfahren, mit dem es möglich ist, eine Ontologie zu validieren. Dabei wollen wir die spezifischen Rahmenbedingungen innerhalb der Produktentstehung von Nutzfahrzeugen und die Fähigkeiten der interdisziplinären Domänen-Experten berücksichtigen. Weiterhin wollen wir quantitative Aussagen über im Projektlastenheft tatsächlich vorhandene und durch das vorgestellte Verfahren gefundene Abhängigkeiten treffen können. Zu diesem Zweck werden ausführliche Tests mit der vorgestellten Werkzeugkette durchgeführt.

Literaturverzeichnis

[CMB+02] Cunningham, H.; Maynard, D.; Bontcheva, K.; Tablan, V.: GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. Proc. of the 40th Annual Meeting of the Association for Computational Linguistics. 2002.

[CTR+13] Cunningham, H.; Tablan, V.; Roberts, A.; Bontcheva, K.: Getting More Out of Biomedical Documents with GATE's Full Lifecycle Open Source Text Analytics. PLoS Computational Biology 9(2), 2013.

[Cu14] Cunningham, et al.: Developing Language Processing Components with GATE Version 8. University of Sheffield, Department of Computer Science. 17 Nov. 2014.

[FMK+11] Farfeleder, S.; Moser, T.; Krall, A.; St˚alhane, T.; Omoronyia, I.; Zojer, H.: Ontology-Driven Guidance for Requirements Elicitation, Springer-Verlag, Berlin u.a., 2011.

[HKR+08] Hitzler, P.; Krötzsch, M.; Rudolph, S.; Sure, Y.: Semantic Web: Grundlagen. Springer-Verlag, 2008.

[MFP09] Maynard, D.; Funk, A.; Peters, W.: SPRAT: a tool for automatic semantic pattern-based ontology population. Proc. of the Int. Conf. for Digital Libraries and the Semantic Web, Trento, Italy, 2009.

[Mu15] Musen, M., A.: The Protégé project: A look back and a look forward. AI Matters. ACM 1(4), June 2015.

[PR11] Pohl, K.; Rupp, C.: Basiswissen Requirements Engineering – Aus- und Weiterbildung zum Certified Professional for Requirements Engineering. dpunkt-Verlag, 2011.

[RMC+11] Ruiz-Martínez, J., M.; Minarro-Giménez, J., A.; Castellanos-Nieves, D.; García-Sanches, F.; Valencia-García, R.: Ontology Population: An Application for the E-Tourism Domain. Int. Journal of Innovative Computing, Information and Control, 7(11), 2011.

[SB16] Schraps, M.; Bosler, A.: Knowledge Extraction from German Automotive Software Requirements using NLP-Techniques and a Grammar-based Pattern Detection. The 8th Int. Conf. on Pervasive Patterns and Applications, pp. 17-21, 2016.

[SBF98] Studer, R., V.; Benjamins, R.; Fensel, D.: Knowledge engineering: Principles and methods. Data & Knowledge Engineering, 25(1-2):161–197, 1998.

[Si14] Siegemund, K.: Contributions to Ontology-Driven Requirements Engineering, 2014.

[SHS+14] Soomro, S.; Hafeez, A.; Shaikh, A.¸Musavi, S.: Ontology Based Requirement Interdependency Representation and Visualisation. In Communication Technologies, Information Security and Sustainable Development, Springer-Verlag, pp. 259-270, 2014.

[SP14] Schraps, M.; Peters, M.: Semantic Annotation of a Formal Grammar by Semantic-Patterns, IEEE 4th Int. Workshop on Requirements Patterns, 2014.

[TH06] Tsarkov, D.; Horrocks, I.: FaCT++ description logic reasoner: System description. In Proc. of the Int. Joint Conf. on Automated Reasoning, 2006.

[WKR10] Witte, R.; Khamis, N.; Rilling, J.: Flexible Ontology Population from Text: The Owl-Exporter, Int. Conf. on Language Resources and Evaluation, pp. 3845--3850, 2010.