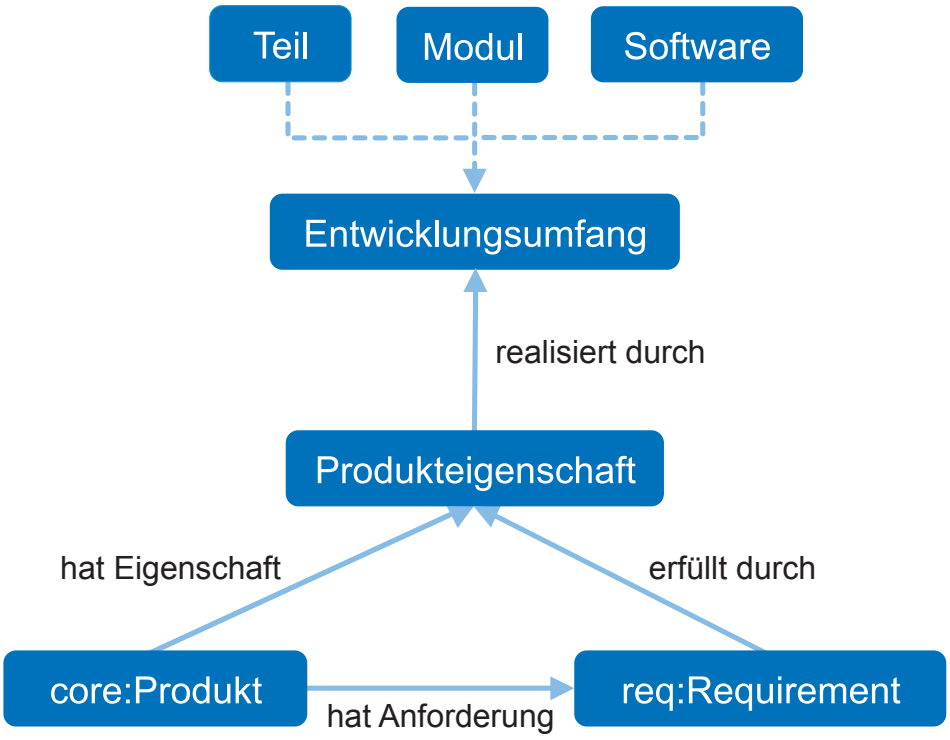
Tim Gülke

Erweiterung des Anforderungsmanagement-Fokus

Von Produkten zu Prozessen



Aachener Informatik-Berichte,
Software Engineering

Band 18

Hrsg: Prof. Dr. rer. nat. Bernhard Rumpe

Erweiterung des Anforderungsmanagement-Fokus: Von Produkten zu Prozessen

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der
RWTH Aachen University zur Erlangung des
akademischen Grades eines Doktors der Naturwissenschaften
genehmigte Dissertation

vorgelegt von

Dipl.-Wirt.-Inf. Tim Gülke
aus Hildesheim

Berichter: Prof. Dr. rer. nat. Bernhard Rumpe
Prof. Dr.-Ing. habil. Joachim Axmann

Tag der mündlichen Prüfung: 04.07.2014

[Gue14] T. Gülke:
Erweiterung des Anforderungsmanagement-Fokus. Von Produkten zu Prozessen.
Shaker Verlag, ISBN 978-3-8440-3118-8. Aachener Informatik-Berichte, Software Engineering, Band 18. 2014.
www.se-rwth.de/publications/

Kurzfassung

Unternehmen der Automobilindustrie sind mit einer zunehmenden Komplexität innerhalb ihrer Produkte sowie ihrer Organisation konfrontiert. Finanziell spürbar wird dies unter anderem in einem Anstieg der Gemeinkosten, an dem Komplexitätskosten einen signifikanten Anteil haben. Entscheidungen hinsichtlich der Produkteigenschaften, die in Fahrzeugentstehungsprojekten getroffen werden, wirken sich unmittelbar auf die Produktdaten des jeweiligen Fahrzeugs und mittelbar auf die organisatorischen Strukturen des Unternehmens aus. Als Basis für die Entscheidungsfindung auf Produktebene dient hierbei das Anforderungsmanagement, welches mithilfe von softwaregestützten Werkzeugen auf der Verknüpfung diverser Elemente im Kontext der Produktentwicklung beruht. Hiermit können die Auswirkungen auf diese Elemente – wie bspw. andere Anforderungen oder Tests – im Vorfeld berücksichtigt werden. Bei der Entscheidungsfindung wird allerdings in den meisten Fällen nur die Auswirkung auf das Produkt zugrunde gelegt – die Beeinflussung der Organisation findet zumeist keine Berücksichtigung. So kann eine Zunahme der Komplexität, bewirkt durch eine zusätzliche Belastung eines spezifischen organisatorischen Ablaufs, die eventuellen positiven Nutzeneffekte im Fahrzeugentstehungsprojekt durch einen Anstieg der Komplexitätskosten wieder aufheben. Es stehen bisher keine Werkzeuge zur Verfügung, um die von einer produktspezifischen Entscheidung in einem Projekt indirekt betroffenen Strukturen einer Organisation überhaupt zu erfassen.

Ziel dieser Arbeit ist demnach die Herstellung einer Grundlage für die Verknüpfung zwischen Produkt- und Prozessinformationen durch ein durchgängiges und semantisch unterstütztes Anforderungsmanagement. Ermöglicht wird dies durch eine zentrale Wissensbasis, welche maschineninterpretierbare, semantisch angereicherte Informationen in Form von Metamodellen und davon instantiierten konkreten Modellen als Ontologien zur Verfügung stellt.

Aus dieser Zielformulierung abgeleitet sind die Eckpunkte der Arbeit durch die Prägung einer gemeinsamen Terminologie und der Formalisierung derselben als Ausgangspunkt der Wissensbasis bestimmt. In einem dreijährigem Forschungsprojekt in Kooperation mit dem Produktmanagement der Marke Volkswagen der Volkswagen AG wurde mithilfe der Szenariotechnik die zukünftige Entwicklung des Anforderungsmanagements auf Gesamtfahrzeugebene für das Unternehmen analysiert. Basierend auf diesem Projekt führt die Arbeit das abstrakte Konstrukt des „Artefakts“ ein, um über einen Oberbegriff für Elemente aus der Produkt- und Prozesswelt zu verfügen. Er bildet somit den Ausgangspunkt der Ontologien „Core“ und „RMNet“, welche die Formalisierung als besagte Grundlage einer unternehmensspezifischen Wissensbasis vornehmen.

Weitere Eckpunkte der Arbeit stellen das Aufzeigen von Funktionalität und Nutzen eines solch umfassenden Anforderungsmanagements sowie eine Methodik zur Erstellung, initialen Befüllung sowie Pflege der Wissensbasis dar. Hierzu werden Erkenntnisse aus der theoretischen und praktischen Forschung im Anforderungsmanagement und angrenzenden Disziplinen verwendet. Mit einem Entwurf für eine technische Umsetzung und einer Darstellung der Verwendung in einem fiktiven Fahrzeugentstehungsprojekt wird die Operationalisierbarkeit der Ergebnisse demonstriert und gleichzeitig ein Eindruck der Potentiale für den Nutzen der neuen technischen und prozessualen Möglichkeiten in anderen Fachdisziplinen vermittelt.

Die Arbeit zeigt, dass durch Einsatz von Ideen und Methoden aus dem Anforderungsmanagement in Kombination mit semantischen Technologien eine Verknüpfung von Produkt- und

Prozesswelt gelingen kann. Im besonderen Hinblick auf eine weitere Zunahme der Komplexität in Produkt- und Organisationsstrukturen stellt sie somit einen signifikanten Baustein für die Handhabung der Herausforderungen dar. Ein Ausblick auf Möglichkeiten zur automatisierten Analyse von finanziellen Auswirkungen gibt die weitere Richtung für zukünftige Forschungsarbeiten vor.

Abstract

Enterprises in the automotive sectors are affected by an increasing complexity within their products and organizations. This becomes noticeable especially in raising overhead costs, which consist of a significant amount of complexity costs. Decisions regarding specific issues in vehicle development projects affect the vehicle's product data directly and an organization's structure indirectly. The decision making process is based on requirements management, that itself relies on computer aided tools building upon the relationships between product development element information. Implications on other elements like tests or different requirements can thereby be considered up front. An estimated impact on the organization though is rarely incorporated into the decision-making process. Therefore, an increase in complexity caused by additional strain on a specific business process by a decision might annihilate potential benefits in a vehicle development project. So far, no tools are available to even detect the organizational structures affected by a decision in a project.

The goal of this thesis is the development of a foundation for the combination of product and process information, using a continuous and semantically enhanced requirements management. This is achieved by the construction of a central knowledge base, consisting of meta-models and instantiated concrete models as ontologies.

Basic elements of the thesis are the definition and formalization of a terminology, providing a foundation for the knowledge base. In a three-year project with the product management of the Volkswagen brand of Volkswagen AG, the prospective evolution of requirements management in vehicle development projects was analyzed using scenario development methods. Based on this project, the thesis introduces the abstract construct of „artifacts“ as a generic term for elements from the product and process world. The term represents a point of origin for the ontologies „Core“ and „RMNet“ which conduct the aforementioned formalization of an enterprise-specific knowledge base.

Additional elements of the thesis are the depiction of functionality and benefits of such a comprehensive requirements management and a method for creating, initially filling and maintaining the knowledge base. To achieve this, findings from theoretical and practical scientific research are evaluated and applied. A technical design and a concept for the application in a fictitious vehicle development project demonstrate the operational feasibility of the results and simultaneously give an impression of the new technical and processual options.

The thesis shows that by applying ideas and methods from requirements management in combination with semantic technologies, a connection between product and process world is possible. Particularly with regard to increasing complexity in product and organizational structures, it provides a significant step in handling upcoming challenges. An outlook on automated financial analysis of decisions and other potential obstacles of the approach points into the direction of possible future research.

Für Nadine.

Danksagung

Wenn man glaubt, dass das eigentliche Schreiben einer Dissertation schwierig ist, dann kommt man zur Danksagung und stellt fest, dass man sich gründlich geirrt hat. Viele Kollegen, Freunde und Familienmitglieder haben zum Gelingen dieser Arbeit beigetragen, mir mit Rat, Tat und Unterstützung zur Seite gestanden und die typischen Gemütszustände eines Doktoranden („Hört auf zu fragen, wie viele Seiten ich schon habe!“) ertragen. Ihnen allen gebührt ein Platz an dieser Stelle und ich kann nur hoffen, dass die hier genannten sich angemessen repräsentiert finden und alle anderen mir vergeben. Letzteren sei daher insbesondere nochmal gesagt: Danke für Alles!

Ohne Prof. Bernhard Rumpe hätte es diese Arbeit nicht gegeben. Von meinen ersten Berührungspunkten mit dem SSE in Braunschweig (EOJ!) und meiner Studienarbeit an, über die unglaubliche Zeit des CarOLO-Projekts zu den Projekten in Aachen und Wolfsburg, hat er mich stets vorbehaltlos und umfassend unterstützt und gefördert. Danke Bernhard, für eine spannende Zeit! Ich bin sicher, dass sie noch nicht zu Ende ist.

Prof. Joachim Axmann gebührt großer Dank für das Ermöglichen des dieser Arbeit zugrundeliegenden Projekts bei Volkswagen und für das Übernehmen der Position des Zweitgutachters – niemand wäre besser geeignet! Nicht nur stand und steht er stets als kritischer aber fairer Sparringspartner bei wissenschaftlichen Themen zur Verfügung, seine Kenntnisse von Prozessen, Personen und Programmen haben vital zur Erarbeitung des in dieser Arbeit behandelten Kernproblems beigetragen. Danke auch dir, Joachim!

Martin Jansen danke ich für das Nahebringen der Art und Weise der Betrachtung der Welt, wie er es tut. Ich danke ihm für das Denken in Bildern und Analogien, für die Begeisterung für elegante Lösungen, für lange Diskussionen über Prozesse, für die Anstöße zu den Lösungen in dieser Arbeit, für das Verständnis was eine gute bildliche Darstellung eines Sachverhalts ausmacht, für mein Wissen über Lautsprecher, Faszien und so vieles mehr. Danke an Katrin und dich!

Prof. Seidl und Prof. Giesl danke ich dafür, dass sie als meine Prüfer zur Verfügung standen. Insbesondere hat sich Prof. Seidl dankenswerterweise bereit erklärt, den Prüfungsvorsitz zu übernehmen.

Christian Berger habe ich kennen gelernt, als er die Leitung des CarOLO-Projekts übernommen hat. Sein erster Satz zu mir war „Den Herrn Berger lassen wir aber mal weg“. Daraus entwickelte sich eine unglaubliche Freundschaft, die sich auch von Ländergrenzen nicht aufhalten lässt. Danke für all die Dinge, die viel zu zahlreich wären um sie hier aufzuführen. Hej Christian: Ich hoffe wir schaffen es mal wieder in der Wüste beim Umfahren des großen Waldbrands im Touareg zu Burger King zum Oreo-Cookie-Shake-Trinken!

Mit Jan Oliver Ringert verbindet mich ebenfalls eine langjährige Freundschaft. Sicher hätte es ohne meinen Teampartner bei Jugend forscht, EOJ, Kazumi und all den anderen an dieser Stelle (besser) ungenannt bleibenden Projekten und Aktionen diese Arbeit nicht gegeben.

Dank gebührt auch meinen (ehemaligen) Kollegen am SSE in Braunschweig und dem SE in Aachen - hier in völlig ungeordneter Reihenfolge, wie sie mir eingefallen sind: Hans Grönniger, Steven Völkel, Holger Krahn, Ingo Weisemöller, Holger Rendel, Claas Pinkernell, Andreas Horst, Andreas Wortmann, Theresa Körtgen, Arne Haber, Christoph Herrmann, Thomas Kurpick, Sylvia Gunder, Markus Look, Toni Perez, Andreas Donners und das legendäre CarOLO-Team. Es war

eine super Zeit!

Aber auch außerhalb der Büroräume mussten Familie und Freunde unter meiner knappen Zeit und meinen wechselhaften Launen leiden. Ich möchte mich daher zunächst entschuldigen und im gleichen Atemzug für die tolle und absolut nicht selbstverständliche Unterstützung bedanken. Durch meine Familie wurde mir nicht nur mein Ausbildungsweg ermöglicht, sondern eben auch die Chance, jederzeit spannende Dinge nebenbei zu erleben – seien es Praktika, Jugend forscht-Projekte oder auch das CarOLO-Projekt in den USA. Abgesehen davon, dass ohne die gut gemeisterte biologische Aufgabe durch meine Eltern diese Arbeit definitiv nicht existieren würde, so haben sie darüber hinaus noch einen erheblichen Anteil daran: Sie schafften es, immer wieder unterstützt durch die Familie Fraaß und auch durch meine Schwester Eileen, aus mir jemanden zu formen, der eine Promotion vollendet hat. Ich kann nur erahnen, was dazu alles nötig war. Meinen Eltern, meiner Schwester und der Familie Fraaß ist diese Arbeit daher in einem nahezu ebenso großen Maße gewidmet.

Um meinen Dank an meine Freundin Nadine Limburg hier auszusprechen, fehlt leider der Platz. Sie musste mich nicht nur in diversen „heißen Phasen“ ertragen, psychologische Meisterleistungen bei Fehlschlägen vollbringen sowie mich grundsätzlich durch das Anbieten von Nahrung am Leben halten, sondern auch für fachliche Diskussionen zu beliebigen Tages- und Nachtzeiten, Korrekturlesungen, Probevorträge und das Abfragen von Lernkarten zur Verfügung stehen. Aus diesem Grund habe ich die Arbeit ihr gewidmet. Der ganzen Familie Limburg werde ich darüber hinaus immer für die Aufnahme in ihre Mitte dankbar sein.

Friedel Ahlers danke ich für die großartige Unterstützung bei Grammatik und Rechtschreibung und das Feedback zum Probevortrag. Auch Ronald und Brigitte Führing sowie Jasper Bartels müssen auf diesen Seiten zu Beginn der Arbeit erwähnt werden.

Um ein Zitat von Gause/Weinberg anzuführen: I hope it meets your requirements.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Die Herausforderungen denen sich die Automobilindustrie stellen muss, wachsen stetig. Insbesondere geht die Verfügbarkeit preislich attraktiver Elektrik/Elektronik-Funktionen mit einer erhöhten Dynamik einher. Zusammenschlüsse von Unternehmen zu großen Multi-Marken-Konzernen, immer stärker differenzierte Märkte mit immer anspruchsvolleren Kunden mit sich veränderndem Verhalten, plötzliche Absatzkrisen durch Naturkatastrophen oder wirtschaftliche Flauten, verschärfte Regulierungen bzgl. Umweltschutz und Sicherheit sowie neue Technologien bringen teils hausgemachte und teils von extern induzierte Problem- und Fragestellungen mit sich [RSG08]. Die fahrzeugproduzierenden Industrieunternehmen (auch synonym Original Equipment Manufacturer (OEM) genannt) reagieren darauf mit einer ganzen Reihe von Ideen und Konzepten, wie etwa verschiedene Ansätze zur Erhöhung der Gleichteilverwendung, Akquisitionen von anderen Marken oder der stärkeren Integration von Zulieferern. Sie alle sind jedoch in der Geschichte dieser Industrie in solchen Dimensionen beispiellos, wodurch verlässliche Daten hinsichtlich Umsetzbarkeit und Wirksamkeit von eingesetzten Strategien oftmals fehlen. Generell sorgen sowohl die genannten Einflussfaktoren als auch die Strategien der Konzerne für eine steigende Komplexität innerhalb und außerhalb der Unternehmen. Vernetzungen werden dichter, es wird nahezu unmöglich für einzelne Personen, die gesamten direkten und indirekten Auswirkungen ihres Handelns im Vorfeld antizipieren zu können. Führungskräfte sind daher immer mehr auf Entscheidungshilfen angewiesen, seien dies Software-Systeme oder personelle (Stabs-)Stellen.

In den letzten Jahren ist ein hohes Niveau an Qualitätsproblemen in den Produkten der Automobilhersteller erkennbar. Dies spiegelt sich beispielsweise wider durch die gesteigerte Zahl von durch das Kraftfahrtbundesamt unterstützten Rückrufaktionen der Hersteller im Zeitraum von 1998 (55 Rückrufaktionen) bis 2010 (185 Rückrufaktionen) [o.V10]. Hierbei waren 2010 etwa bei 27% elektrische oder elektronische Komponenten die Ursache [o.V10]. Wird hierzu die Zunahme des Anteils der Elektronikkomponenten an den Herstellkosten in Kontext gesetzt, welche sich derzeit auf etwa 20 Prozent erhöht hat, zeigt sich eine überproportionale Quelle für Fehler [Bus06, ES09, WFO09]. 2012 geht die Zahl an Rückrufen leicht auf 162 zurück, auch der Anteil an Verursachung von Elektrik-/Elektronik-Komponenten sinkt auf etwa 20% und entspricht damit proportional dem Anteil an den Herstellkosten [o.V12]. Das Kraftfahrtbundesamt gibt in seinen Jahresberichten keine Auskunft über den finanziellen Umfang der durchgeführten Rückrufaktionen. Eine einzige Rückrufaktion kann allerdings aufgrund der Gleichteilverwendung eine große Zahl Fahrzeuge betreffen, wie bei den aktuellen Rückrufen von Fahrzeugen mit bestimmten Direktschaltgetrieben (DSG) von Volkswagen im asiatischen Raum deutlich wird [dpa13].

Die Zunahme der Komplexität durch das Zusammenwirken verschiedener Komponenten innerhalb der Fahrzeuge ist somit eine nicht unerhebliche Quelle für Fehler. Diese Thematik ist

bereits Gegenstand von verschiedenen Forschungsaktivitäten. Das Anforderungsmanagement ist eine Antwort auf die Herausforderungen, die sich aus ihr ergeben. Neben der Erhöhung der Produktkomplexität führen verschiedene Maßnahmen und Entscheidungen, wie etwa Unternehmenszusammenschlüsse, starke Erhöhung der Anzahl an Produktvarianten, Zulieferer-Strategien und eine genauere Aufteilung einzelner Märkte zu einer Erhöhung der Komplexität in den Prozessen und Strukturen der Unternehmen [WFO09, OB00, MSG07, RSG08]. Ähnlich wie eine Zunahme an Komponenten und Software innerhalb der Fahrzeuge zu einer Erhöhung des Vernetzungsgrads führt, ist auch das Zusammenwirken von Personen, Prozessen und Systemen in einem Unternehmen dieser Gesetzmäßigkeit unterworfen. Es findet sich jedoch eine weitaus höhere Varianz an unterschiedlichen Objekten die hier zusammenwirken, als es in den Softwareprodukten der Fall ist.

Diese Arbeit konzentriert sich auf den Problembereich der Komplexität innerhalb von Unternehmen, basierend auf dem Zusammenwirken verschiedener Objekte im Unternehmenskontext. Entscheidern ist heute bewusst, dass bspw. neue Funktionen im Produkt auf Sensoren und Aktoren basieren, die miteinander vernetzt geplant und dann in Kombination mit Steuergeräten im Fahrzeug verbaut werden, was den Grad der Produktkomplexität steigert. Nicht, oder nur selten wird bedacht, wie sich etwa die Erweiterung der Angebotspalette eines Fahrzeugs, z.B. durch das Anbieten von elektrischen Fensterhebern neben manuellen, auf die Komplexität der Prozesse und Strukturen im Unternehmen und seinen IT-Systemen auswirkt [Bus06, OB00]. Diese gesteigerte Komplexität hat vor allem finanzielle Auswirkungen auf die Gemeinkosten eines Unternehmens, da Prozesse und die sie ausführenden und unterstützenden personellen Ressourcen sowie IT-Systeme nicht notwendigerweise in einem linearen Verhältnis skalieren [Sys06, BBS04]. Es liegt somit der Verdacht nahe, dass mit bestimmten Entscheidungen zwar eine Reduktion der Einzelkosten eines Produkts bewirkt werden kann, diese jedoch von gesteigerten und nicht im Vorfeld antizipierten Gemeinkosten im Nachhinein wieder aufgehoben wird [GJRA12].

Für die Beherrschung der Einflussfaktoren und deren Zusammenwirken auf Produktseite spielt zunehmend das Anforderungsmanagement eine immer größere Rolle in der Automobilindustrie. Getrieben aus der Softwareentwicklung, werden die bekannten Prinzipien nun auf immer mehr Bereiche der Unternehmen angewandt. Wie Studien zeigen, verursachen Projekte, die nur 2% bis 3% des Projekt-Budgets für Anforderungsmanagement aufwenden im Schnitt 80% bis 200% an Mehrkosten. Projekte hingegen, die 8% bis 14% aufbringen, liegen lediglich 0% bis 50% über dem Projektbudget [You03]. In Fahrzeugprojekten, die in der heutigen Zeit eine entsprechende Größenordnung aufweisen, existiert hier ein deutliches Potential, welches sich die Automobilindustrie zunutze machen will. Das Verknüpfen von Anforderungen mit Komponenten, Teilen und schließlich Tests über den gesamten Lebenszyklus des Produkts hinweg bis zur späteren Implementierung erzeugt an vielen Stellen Mehrwert und unterstützt mehrere Disziplinen wie bspw. die Produktion oder das Änderungsmanagement. Anforderungsmanagement erscheint daher als ein geeignetes Mittel zur Beherrschung von Komplexität, da es beliebig komplexe Strukturen abbildet und Werkzeuge zum Umgang mit diesen anbietet.

In dieser Arbeit wird daher eine Verbindung zwischen Anforderungsmanagement und der in den Strukturen und Prozessen des Unternehmens enthaltenen Komplexität hergestellt. Hierzu muss nicht nur – im klassischen Sinne des Anforderungsmanagements – das zu entwickelnde Produkt in seinen unterschiedlichen Ebenen und Entwicklungsstufen (vom Vehicle- bis zum

Function-Level, siehe auch [BLP04, Dyc11]) abgebildet werden, sondern über das Produkt hinaus eine Verknüpfung in die Struktur- und Prozess-Welt des Unternehmens stattfinden. Ziel ist es, Grundlagen für ein Werkzeug zu schaffen welches in der Lage ist, die Komplexitätsveränderung im Ökosystem eines Unternehmens zum Zeitpunkt der Entscheidung hinsichtlich bestimmter Anforderungen an ein Produkt abschätzen zu können. Hierzu ist eine praktikable und standardisierte Möglichkeit zur Erfassung und formalen Beschreibung von Unternehmensobjekten notwendig.

Der Mensch ist über den Zeitraum seiner Evolution hinweg einer immer komplexeren Umwelt ausgesetzt. Dennoch sind wir in der Lage, uns auch in unbekannten Situationen zurecht zu finden. Möglich wird dies durch die uns angeborene Fähigkeit der Generalisierung – wir sind in der Lage, Dinge in einen Kontext zu stellen, die „Art und Gattung“ eines Objekts zu bestimmen [Sil12]. Ist dies geschehen, treffen wir Annahmen hinsichtlich des Zusammenspiels mit anderen Dingen und erstellen so unbewusst eine Voraussage, wie sich etwas oder jemand verhalten wird. Traditionelle Softwareentwicklung verfolgt einen anderen Ansatz, indem durch die Erstellung von möglichst vollständigen Modellen im Vorfeld ein statisches Bild geschaffen wird, dem die betriebswirtschaftliche Realität sich fügen muss. Erfüllt sich diese Annahme nicht, ist eine – oftmals umständliche – Änderung der Modelle innerhalb der Software nötig. Durch die Technologie der semantischen Netze wird versucht, die menschliche Fähigkeit des automatisierten Einordnens und Schlussfolgerns in Software abzubilden [AH08]. Es werden weiterhin Modelle im Vorfeld erstellt, allerdings basieren diese stark auf einem Top-Down-Ansatz: Wissen wird generiert, indem ein Objekt, über das wenige Informationen vorliegen, durch Generalisierung in einen mit mehr Informationen versehenen Kontext eingeordnet wird. Die Einordnung muss zwar immer noch durch einen Entwickler oder Nutzer geschehen, aber das Schlussfolgern auf der Basis der Metainformationen kann die Software eigenständig vornehmen. Dies steht vor allem im Gegensatz zu herkömmlichen modellbasierten Ansatz in der Beschreibung von Prozessen und deren Elementen, bei denen ein möglichst detailliertes Bild auf unterster Ebene erzeugt wird.

1.1 Automobilindustrie im Wandel

Tietze identifiziert im globalen Umfeld vier Veränderungstreiber: Kunden bzw. das sozio-ökonomische Umfeld, Technologie, Politik und Recht sowie die Ökonomie [Tie03]. Zusätzlich wandelt sich auch das brancheninterne Umfeld bedingt durch Marktsättigung und strukturelle Überkapazitäten sowie neue Wettbewerber. Ähnlich sehen dies auch Hüttenrauch und Baum, die sogar von einer „3. Revolution der Automobilindustrie“ sprechen [HB08]. Schömann kommt ebenfalls zu dem Ergebnis, dass die OEMs vor „gravierenden strukturellen Veränderungen“ stehen [Sch12].

Das Kundenverhalten, welches stark durch die Gesellschaft der jeweiligen Kundengruppe geprägt ist, wird von den OEMs sehr genau analysiert. Marketing-Abteilungen stellen detailliert fest, in welchen Bereichen der Welt welches Kundenmilieu welchen Eigenschaften eines Fahrzeugs welche Bedeutung beimisst. Der traditionelle Markt in West-Europa und den USA wird inzwischen bspw. bestimmt durch die Individualisierung der Fahrzeuge und deren Repräsentation eines bestimmten Lebensstils [Mar04, Sch12]. Auch findet ein zunehmender Gewöhnungseffekt statt, was die Verfügbarkeit von Komfort- und Assistenz-Systemen angeht. Nicht zuletzt der Erfolg der Firma Apple hat in den letzten 10 Jahren im Bewusstsein der Konsumenten das Vorhan-

densein elektronischer Hilfsmittel jedweder Art größtenteils zur Normalität werden lassen. Diese Entwicklung führt aufgrund der Gewöhnung zu einer Erwartungshaltung, was die Ausstattung von Fahrzeugen angeht. Ein weiterer Trend in den letzten Jahren ist geprägt von einem neuen ökologischen Bewusstsein der Kunden, bedingt durch hohe Kraftstoff-Preise, Fahrverbote in Innenstädten aufgrund von Feinstaubbelastung, neue emissionsorientierte Besteuerungsmethoden sowie die Diskussion der Klimaschutz-Thematik in der Presse [Die06]. Hinzu kommt eine sinkende Preisbereitschaft der Kunden, welche sich in Kombination mit den ersten genannten Punkten als problematisch erweist [Mar04]. Die Markt- und Kundenstruktur selbst ist weltweit geprägt vom demographischen Wandel in den traditionellen Absatzmärkten sowie vom Aufstieg der Schwellenländer wie bspw. China [Hei97, HW10].

Technologische Veränderungen sind im Automobilbau geprägt durch neue Materialien, Fertigungsverfahren und die Zunahme elektrischer und insbesondere elektronischer Komponenten in den Produkten, mit einer stark zunehmenden Bedeutung des letztgenannten Punktes [Bus06]. Laut deutscher Statistiken aus dem Jahr 2004 machten zu diesem Zeitpunkt Elektronikfehler bereits 40 Prozent der vom ADAC erfassten Pannen aus [Ric09]. Die Funktionen im Fahrzeug gehen inzwischen weit über Airbags oder Lenkunterstützung hinaus. Geforscht wird bspw. inzwischen an autonomen Fahrzeugen [RBL+08]. Aktuell erfahren allerdings Komfort-, Unterstützungs- und Unfallfolgenverminderungs-Assistentensysteme eine rapide Entwicklung im praktischen Einsatz.

1.2 Reaktionen der Automobilindustrie

Den genannten Herausforderungen begegnen die OEMs vor allem mit einer Vergrößerung der Angebotspalette [WFO09]. Das Fahrzeug wird gemäß dem erkannten Bedürfnis der Konsumenten immer individualisierbarer und bietet – gegen einen margenträchtigen Aufpreis – immer mehr Funktionalität bei immer mehr Varianten [Bec07]. Dies bringt allerdings weitreichende Konsequenzen mit sich: so lassen sich vor allem Kannibalisierungseffekte beobachten welche durch eine neue Variante bei vorhandenen Produkten ausgelöst werden. Diese führen wiederum zu sinkenden Losgrößen und somit zu Kostensteigerungen bspw. durch geringere Einkaufsgrößen [WFO09]. Zur Entgegnung dieser Problematik setzt die Industrie seit längerem auf erhöhte Gleichteilverwendung: Erst durch die Einführung sog. Plattformen, infolgedessen mithilfe einer gemeinsamen Basis unterschiedliche Modelle entwickelt und später hergestellt werden, später durch Modularisierung, welche die (Wieder-)Verwendung von Teilen oder Systemen in mehreren Plattformen kennzeichnet und schließlich die sog. modularen Fahrzeugbaukästen, welche Module und Teile markenübergreifend bündeln und in Kombination verschiedene Fahrzeuge hervorbringen können [Kre10, HB08, BS11]. Die Modularisierung und ihre Weiterentwicklung zu Baukästen eröffnet Zulieferern ganz neue Möglichkeiten der Kooperation mit den OEMs, geht jedoch auch einher mit der Notwendigkeit des detaillierten Austauschs zwischen beiden Partnern [HB08]. Zwar führen diese Strategien einerseits zu einer Reduzierung der Menge an zu verwaltenden Teilen, andererseits allerdings auch zu einem markanten Anstieg der Komplexität der Produktions- und Logistikprozesse im Sinne der Verwaltung ihrer Abhängigkeiten, infolgedessen die Gemeinkosten steigen [Wir08]. Die OEMs übernehmen immer stärker reine Koordinationsaufgaben und sind hierdurch gezwungen, Prozesse und Strukturen zur Kontrolle und Organisation zu implementieren, die in dieser umfassenden Form bisher nicht existierten.

Durch die Notwendigkeit der Verknüpfung verschiedener Objekte, bietet das Forschungsgebiet des Anforderungsmanagements bereits eine Sammlung von Strategien und Methoden zur Beherrschung von Komplexität. Es lassen sich hier verschiedene Ansätze beobachten, die von unterschiedlichen Ebenen ausgehen [BLP04]. Einerseits findet sich in diesem Feld bspw. die Elektrik/Elektronik-Entwicklung, die ein Anforderungsmanagement auf einer detaillierten Funktions-Ebene benötigt sowie andererseits das Produktmanagement, welches im Gegensatz dazu auf einer sehr hohen Fahrzeug-Ebene basiert. Beide Seiten beschreiben grundsätzlich dieselben Elemente, sind allerdings in der Praxis weit voneinander entfernt. Die Fahrzeug-Ebene ist hierbei von einem viel höheren Komplexitätsgrad betroffen; erstens muss sie die gesamten tiefer liegenden Schichten und deren Auswirkungen aufeinander abbilden. Zweitens ist sie stark von intangiblen Anforderungen geprägt, wie etwa Sitzkomfort oder Lenkverhalten eines Fahrzeugs, welche sich nicht ohne weiteres quantifizieren lassen. Drittens sorgen die stetig stärker differenzierten Märkte für immer mehr Anforderungen für laufend kleiner geschnittene Marktsegmente, die alle berücksichtigt werden müssen, dabei für die Kultur des jeweiligen Planers nicht unbedingt intuitiv sind und sich teilweise sogar gegenseitig widersprechen. Darüber hinaus müssen die unterschiedlichen Fahrzeugbaukästen und Plattformen alle hinsichtlich der durch sie induzierten Anforderungen in den Fahrzeuganforderungen berücksichtigt werden. Im Kontext des Anforderungsmanagements nimmt demnach auf der Gesamtfahrzeugebene nicht nur die Anzahl an Elementen, sondern stark überproportional auch deren Interaktion und Abhängigkeit zu.

Ein softwaregestütztes Anforderungsmanagement auf der Fahrzeug-Ebene kann demnach nur effektiv arbeiten, solange möglichst viele der relevanten und kostenintensiven Objekte inklusive ihrer Verknüpfungen abgebildet sind. Ist dies nicht der Fall, bleibt die Aussagekraft – etwa bei der Abschätzung der Auswirkung von Änderungen – gering. Aktuelle Werkzeuge sind durchaus in der Lage, verschiedene Typen von Objekten anhand von Modellen darzustellen und miteinander zu verknüpfen. Die gängige Methodik fokussiert diese Modelle aber fast ausschließlich auf das entstehende Produkt. Die den Lebenszyklus eines Produkts prägenden Elemente, wie bspw. Entwicklungsdokumente, bleiben außen vor, obwohl doch auch deren Komplexität in administrativen Strukturen immer auch Kosten verursacht. Insofern sollten Komplexitätsauswirkungen somit Teil etwa von Kostenbetrachtungen bei Entscheidungen sein. Anforderungsmanagementwerkzeugen ist die Einbeziehung von Prozessen und deren Elementen fremd, selten findet sich überhaupt die Möglichkeit der einfachen und nicht ausschließlich produktbezogenen Modellierung. Hier setzt diese Arbeit an.

1.3 Problemstellung der Arbeit

In Folge der Umsetzung neuer und veränderter Anforderungen an zu entwickelnde Produkte im Allgemeinen und Fahrzeuge im Speziellen werden Kosten generiert. Sie resultieren in den zwei bekannten Verrechnungsarten von Kosten:

1. **Einzelkosten:** Sie können direkt einem Kostenträger, bspw. einem produzierten Fahrzeug, zugeordnet werden. Anforderungen bewirken eine Erhöhung oder Verminderung von Einzelkosten, selten sind sie neutral. Im neutralen Fall haben stattdessen diese oftmals eine Auswirkung auf andere kostenrelevante Faktoren wie bspw. das Gewicht oder die Qualität [WD02].

2. **Gemeinkosten:** Im Gegensatz zu Einzelkosten ist keine direkte Zuordnung zu Kostenträgern möglich, sie werden vor allem durch administrative Arbeiten generiert [WD02].

Darüber hinaus findet eine zeitliche Trennung der Wirksamkeit der verursachten Kosten statt: Einzel- und Gemeinkosten fallen ab der Produktion des Fahrzeugs an. In einem Projekt unmittelbar wirksam werden Investitionskosten, welche im Rahmen der Investitionsrechnung behandelt werden [WD02]. Sie sind zur Umsetzung der Anforderung zu erbringen und beinhalten bspw. den Aufwand an benötigten Stunden zur Programmierung von Software, Konstruktion eines Teils oder den Kosten eines Werkzeugs. Verrechnet werden sie gegen das Investitionsbudget.

Gemeinkosten besitzen eine große Bedeutung, sind sie doch einer der großen Kostentreiber in Unternehmen [Sch94]. Problematisch ist, dass eine verursachungsgerechte Zuordnung der Gemeinkosten aufgrund ihrer Natur nicht möglich ist. Somit bleibt unklar, welche Produkte für welche Veränderung der Gemeinkosten verantwortlich sind. Die beschriebene Zunahme nicht nur der Komplexität der Produkte und der Angebotspalette, sondern deren Auswirkung auf die Prozesse eines Unternehmens schlägt sich vor allem als administrative Kosten in den Gemeinkosten nieder [Sys06, BBS04]. Dies kann zu Wettbewerbsnachteilen gegenüber Unternehmen führen, die geringerer Komplexität in ihren Prozessen ausgesetzt sind. Selten spielen allerdings bei der Entscheidung hinsichtlich neuer oder geänderter Anforderungen an ein Produkt die Auswirkungen auf die Komplexität eine Rolle [Wir08].

Sind Unternehmen bereits in der Lage, Änderungen in den Einzelkosten sowie in den Investitionskosten auf der Grundlage von Anforderungen zu errechnen und in Entscheidungsprozessen zu beachten, so fehlen zumeist Instrumente zur Einbeziehung der Komplexität. In der Automobilindustrie ist der Drang zu mehr Varianten unter gleichzeitiger Verwendung von Gleichteilen bspw. durch modulare Baukästen sehr ausgeprägt, zunehmend werden aber administrative Strukturen – personelle ebenso wie prozess- und informationstechnische – stark belastet [Wil90, Lew05].

Zur Verdeutlichung der Problematik soll das folgende fiktive Beispiel dienen: Luftauslässe im Vorderwagen eines Fahrzeugs dienen der Klimatisierung durch adäquate Verteilung der ausströmenden Luft. Da sie direkt im Betrachtungsfeld der Fahrzeuginsassen liegen, sind Gestaltung und Materialauswahl von höchster Wichtigkeit. Während der Definition der Eigenschaften eines neuen Fahrzeugs wird entschieden, dass die Luftauslässe – im Gegensatz zur vorherigen Modellversion des Fahrzeugs – nun in zwei Varianten zur Verfügung stehen sollen: Die erste mit einer hochwertigen, verchromten Einfassung und die zweite in einer preisgünstigen Variante ohne abgesetzte Einfassung. Im Vorgängermodell wurden in jedem Fahrzeug die verchromten Einfassungen verbaut. Mit der Einführung einer Low-Cost-Variante und der Definition der hochwertigen Variante als Sonderausstattung sollen die Einzelkosten gesenkt, bzw. die Marge erhöht werden. Die Berechnung der Investitions- und Einzelkosten kann dies bestätigen. Nicht bedacht werden allerdings die gesteigerte Komplexität und die damit verbundenen Auswirkungen auf nachgelagerte Prozesse. Die neue Variante muss in die Varianten- und Stücklisten-Systeme eingetragen, verschlüsselt und dort gepflegt werden. Sie muss in die Berechnung der Herstellzeiten mit einbezogen werden sowie in die laufende, regelmäßige und weltweite Kalkulation der Fahrzeuge. Die Beschaffungs- und Logistiksysteme müssen eventuell neue Zulieferer verwalten und die logistischen Transportpfade von diesen an die einzelnen Standorte planen. Bei Änderungen an den Luftauslässen ist jeweils eine Prüfung notwendig, ob die neue Variante auch geändert werden muss und das Marketing muss eine weitere Variante in ihren Bestell-Systemen und Prospekten aufnehmen. Darüber hinaus

müssen diverse digitale Dokumente wie Tabellen oder Textdokumente mithilfe von Bürosoftware erweitert und laufend aktuell gehalten werden.

Das Eintragen und Pflegen der zusätzlichen Information sowie die Ausführung bestimmter Prozesse (wie beispielsweise ein Änderungsprozess), die nun ein Element mehr zu verarbeiten haben, verursacht einen sehr kleinen Kostenanstieg, der fast unmöglich im Einzelnen zu erfassen und auch nicht automatisch im Vorfeld zu kalkulieren oder auch nur zu approximieren ist. An dieser Stelle führt der Anstieg der durch die komplexe Prozesswelt zu verarbeitenden Elemente zu Anspannungen in den einzelnen Prozessschritten. Er wird verursacht durch die hohe Anzahl an Schnittstellen, welche die Information, bspw. über die Existenz eine weiteren Variante, verarbeiten müssen. Es bedeutet Aufwand, jede einzelne Schnittstelle – teilweise manuell – zu versorgen. Manuelle Versorgung einer Schnittstelle heißt in vielen Fällen das Exportieren, Bearbeiten und wieder Importieren von Dokumenten. Bei der Einführung neuer Softwaresysteme oder Arbeitsdokumente im Ökosystem des Unternehmens wird oftmals nicht bedacht, dass diese die Komplexität durch eine erneute Erhöhung der Anzahl an Relationen vergrößern. Für die Kalkulation von Fahrzeugen etwa müssen diverse Preise verschiedener Arten – bspw. Einkaufs- oder Konzernverrechnungspreise – erfasst werden. Die Einführung eines neuen Systems für bestimmte Arten von Preisen in einem Teil eines Unternehmens führt daher unmittelbar zu einer neuen notwendigen Schnittstelle in einem etwaigen Kalkulationssystem. Entscheidungen hinsichtlich bestimmter Anforderungen an ein neues Fahrzeugprojekt verursachen Aufwand an sekundären Systemen, der zum Zeitpunkt der Aufwandschätzung nicht als direkte Auswirkung erkannt werden kann. Es fehlt hierzu an softwaregestützten Werkzeugen, die eine Auswirkungsanalyse zur Entscheidungsunterstützung basierend auf den Zusammenhängen dieser sehr unterschiedlichen Objekte im Unternehmen durchführen können.

Anforderungsmanagement-Werkzeuge besitzen heute mehr oder weniger umfangreiche Möglichkeiten zur Abbildung von Modellen. Ein Modell entspricht hierbei einer formalen Repräsentation eines bestimmten Sachverhalts, bestehend aus Elementen und deren Relationen. Während der Untersuchungen der Möglichkeiten aktueller Softwarewerkzeuge im Rahmen des Projekts mit dem Produktmanagement der Marke Volkswagen der Volkswagen AG zeigte sich zwar ein Schwerpunkt der Werkzeughersteller in der Modellierung. Jedoch liegt der Fokus hierbei auf dem zu entwickelnden Produkt, bzw. im näheren Umfeld. Es ist somit nicht möglich, den Zusammenhang einer Anforderung mit sämtlichen weiteren Objekten in den Prozessen des Unternehmens darzustellen oder Komplexitätsbetrachtungen auf der Basis von Anforderungen durchzuführen. Es fehlt an einer Methodik zur Erfassung der Objekte und ihrer Strukturen gleichermaßen wie die Verknüpfung derselben mit Produktinformationen und Anforderungen.

1.4 Lösungsansätze und Ergebnisse der Arbeit

Grundlage der Arbeit ist ein mit der Marke Volkswagen der Volkswagen AG durchgeführtes dreijähriges Projekt. In ihm wurde das Anforderungsmanagement in der Produktentstehung auf Gesamtfahrzeugebene untersucht. Es manifestierte sich der Eindruck, dass auf einer solch hohen Managementebene stark differenzierte Arten von Unternehmenselementen zusammentreffen, die durch diverse Schnittstellen miteinander kommunizieren. Allerdings werden Produkt- und Prozessinformationen getrennt betrachtet. Durch den Einsatz von modularen Fahrzeugbaukästen

bei Volkswagen steigt jedoch die Anzahl dieser Elemente und deren Schnittstellen untereinander überproportional an. Durch die Konzeption eines durchgängigen Anforderungsmanagements sollen Führungskräfte und Experten in die Lage versetzt werden, die komplexen Auswirkungen von Entscheidungen im Vorfeld approximieren zu können.

Es ist somit das Ziel dieser Arbeit, die Grundlage für ein solches Werkzeug durch die Definition von Begrifflichkeiten, Methodik und der Beschreibung des Werkzeugs selbst zu schaffen. Hierbei wird das Konzept der „Traceability“ verwendet, welches das Anforderungsmanagement für die Verknüpfung von Elementen zur Verfolgbarkeit kennt. Als grundlegende Technologie kommen sog. semantische Netze zum Einsatz. Sie ermöglichen eine Art der Formalisierung von Informationen, die Maschinen in die Lage versetzen, diese Informationen zueinander in einen vorgegebenen Kontext einzuordnen. Mithilfe standardisierter Methoden und Werkzeuge ermöglichen sie den Aufbau umfangreicher formaler Modelle, die sich in dieser Arbeit als Ontologien darstellen. Der Begriff “Ontologie“ wird in dieser Arbeit gemäß der Definition von Gruber bzw. in der Erweiterung durch Studer verwendet:

Definition 1.4.1 (Ontologie [Gru93, SBF98]) *„Eine Ontologie ist eine explizite und formale Spezifikation einer Konzeptionalisierung.“*

Die Grundlage für das Werkzeug besteht somit aus folgenden Eckpunkten:

1. Einführung und Prägung des Begriffs der „Artefakte“ als Synonym und Oberbegriff für verschiedene Arten von Unternehmensobjekten, mit deren Hilfe Modelle von Unternehmen entwickelt werden können.

2. Beschreibung von Aufbau, Funktionen und Nutzen eines Anforderungsmanagementwerkzeugs, welches in der Lage ist, mithilfe von formalen Artefaktmodellen zu operieren. Die Eigenschaften eines solchen Werkzeugs werden wiederum durch Anforderungen formuliert, weiterhin werden Technologien zur Umsetzung vorgeschlagen.

3. Bereitstellung von grundlegenden formalisierten Artefakten als Oberklassen für weitere Unternehmensartefakte. Unternehmensmodelle können diese nutzen, um spezifische Artefakte in einen Kontext zu stellen und sie so allgemeingültig zu definieren. Hierzu ist die einschlägige Literatur hinsichtlich der Verfügbarkeit von grundlegenden und allgemeingültigen Unternehmensmodellen untersucht worden. Es zeigte sich, dass zwar diverse Werkzeuge zur Modellierung bereitstehen, aber keine Modelle, die als allgemeine Basis für die datentechnische Erfassung von solchen Artefakten dienen können. Auf diese Weise wird eine formale Datenbasis von Unternehmensartefakten geschaffen, die als Grundlage für Entscheidungsprozesse dienen kann. Als Ergebnis dieses Punkts finden sich die beiden Ontologien **Core** und **RMNet**.

4. Definieren einer Methodik zur Erstellung und Pflege von Artefaktmodellen in Unternehmen. Sie dient als Basis für die Verfügbarkeit von Werkzeugen, die auf Basis der semantischen Netze konstruiert werden können.

5. Erprobung des Werkzeugs durch die Anwendung der Lösungsstrategien auf Probleme des Anforderungsmanagements. Es wird geprüft, ob durch die Verwendung der vorgeschlagenen Vorgehensweisen ein Nutzen in verschiedenen Disziplinen entsteht.

1.5 Aufbau der Arbeit

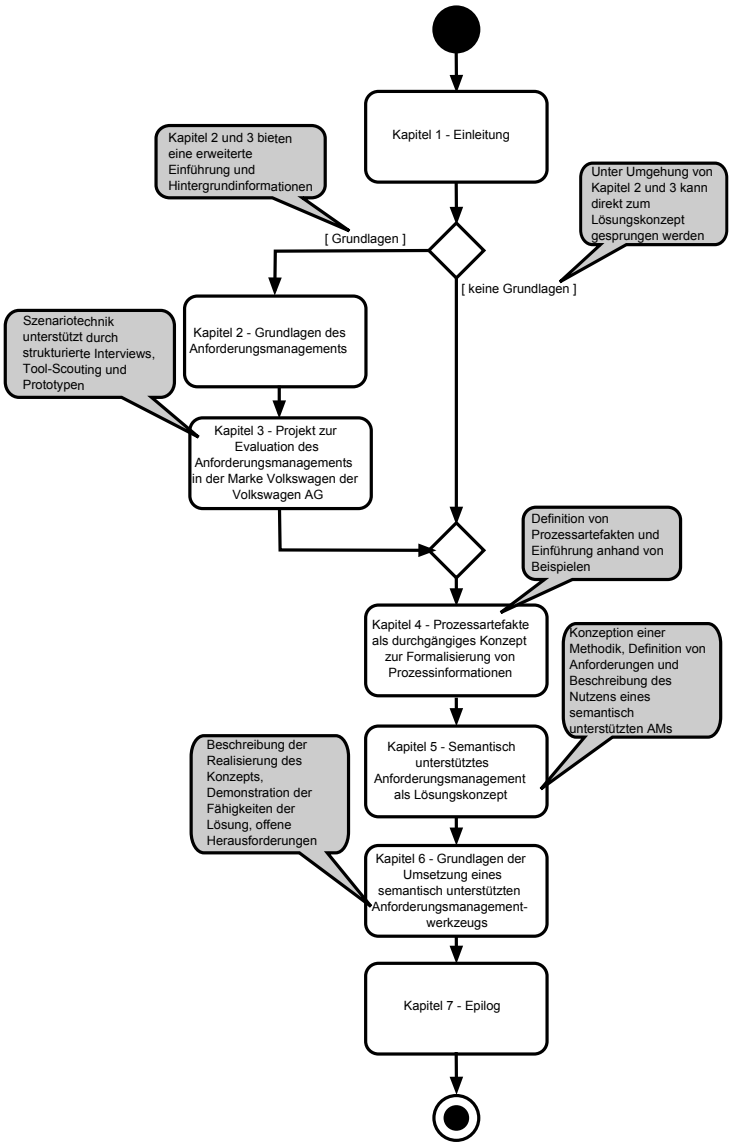


Abbildung 1.1: Aufbau der Arbeit

Das Bild 1.1 zeigt schematisch den Aufbau der Arbeit und umreißt den Inhalt der einzelnen Kapitel. Es ist möglich, Kapitel 2 und 3 zu überspringen, sofern kein Interesse an der Einführung

in das Anforderungsmanagement und an den Details des Projekts mit Volkswagen bestehen. In diesem Fall kann direkt zum Kapitel 4 gesprungen werden, welches die Einführung der Prozessartefakte vornimmt und somit den Teil mit den Kernergebnissen der Arbeit eröffnet. Die folgenden Kapitel müssen dann sequentiell gelesen werden.

Die Arbeit beginnt mit der in diesem Kapitel durchgeführten Analyse des Stands der Wissenschaft. Danach folgt mit Kapitel 2 ein Grundlagenkapitel, in dem in die Begrifflichkeiten und Methoden des Anforderungsmanagements eingeführt wird. An dieser Stelle wird insbesondere das Konzept der Traceability behandelt, welches den Ursprung der in dieser Arbeit postulierten Informationsverknüpfung zu Managementzwecken darstellt.

Es folgt in Kapitel 3 eine Beschreibung des Projekts mit der Marke Volkswagen der Volkswagen AG. Mithilfe der Szenariotechnologie wurde die Entwicklung des Anforderungsmanagements in der näheren Zukunft untersucht und Rückschlüsse auf notwendige Schritte gezogen. Ein Scouting von am Markt verfügbaren Anforderungsmanagementwerkzeugen sowie erste Prototypen zur Formalisierung von Prozessinformationen schließen das Kapitel ab.

Das folgende Kapitel 4 geht intensiver auf den Begriff der Artefakte ein und erläutert, wie Unternehmen aus verschiedenen Produkten, Prozessen, Systemen sowie weiteren Artefakten aufgebaut sind und wie diese mit Komplexität im Zusammenhang stehen. Zwei Beispiele zeigen, wie sich Komplexität auf Unternehmen auswirken kann.

Kapitel 5 geht näher auf das Konzept eines umfassenden semantisch unterstützten Anforderungsmanagementwerkzeugs ein. Es werden Anforderungen an ein solches beschrieben sowie der erwartete Nutzen auf verschiedene Disziplinen des einfachen und erweiterten Anforderungsmanagements dargestellt.

In Kapitel 6 finden sich die im vorherigen Abschnitt beschriebenen Eckpunkte. Es beschreibt zunächst, wie Artefakte in einem semantischen Netz formalisiert werden können und stellt dafür zwei grundlegende Ontologien zur Verfügung. Für den Aufbau eines umfassenden unternehmensspezifischen Artefaktmodells wird eine Methodik beschrieben. Anhand eines fiktiven und vereinfachten Fahrzeugprojekts wird dann der Einsatz von Methodik und Werkzeug erprobt. Hierzu werden typische Aufgabenstellungen gelöst, die sich aus den Anforderungen aus Kapitel 5 ergeben.

Die Arbeit schließt mit einem Fazit sowie einem Ausblick in Kapitel 7. Der Ausblick gliedert sich in zwei Teile: In Abschnitt 7.2.1 wird ein Folgeprojekt umrissen, welches spezifische Fragestellungen betrachten und eine für Mitarbeiter einsetzbare Version des Werkzeugs konstruieren muss. Der zweite Teil in Abschnitt 7.2.2 geht auf fachliche Herausforderungen ein, die sich durch den Einsatz des in dieser Arbeit beschriebenen Anforderungsmanagements ergeben und durch weitere Untersuchungen betrachtet werden müssen.

1.6 Stand der Wissenschaft

Die Literatur im Bereich des Anforderungsmanagements lässt sich nach der Praxis- oder Theoriefokussierung unterscheiden. Die erste ist getrieben aus der Praxis und erfolgreich darin, erprobte Methoden und Vorgehen zu bündeln und wiederzugeben, die andere stellt den Forschungsbereich dar. Ist die Existenz von Büchern mit jeweils entweder praktischem oder theoretischen Fokus keine Seltenheit, so ist die Diskrepanz zwischen beiden Lagern in der Literatur des Anforderungs-

managements doch ersichtlich und inzwischen Gegenstand von Diskussionen und Analysen. Ein Grund für diesen Umstand wird in der fehlenden Untersuchung der Operationalisierbarkeit von in der Theorie ausgearbeiteten Vorgehen vermutet, bzw. in der Fokussierung der Theorie auf in der Praxis nicht relevante Problemstellungen [Wie05, DH02].

Anforderungsmanagement in der Fahrzeugindustrie stellt in den letzten Jahren eine Notwendigkeit dar [WW02, ABNM06]. Bedingt durch die Zunahme an Elektronikkomponenten und Software im Fahrzeug und des damit einhergehenden Komplexitätswachstums in der Elektrik-Elektronik-Entwicklung mussten Methoden und Werkzeuge gefunden werden, diese in Entwicklungsprojekten effektiv zu managen. Es fehlt jedoch der Fokus auf das gesamte Fahrzeug und seinen Lebenszyklus mit entsprechender Literatur. Weber und Weisbrod schildern bspw. detailliert die Erfahrungen mit Anforderungsmanagement im Bereich der „Automotive Entwicklung“, beschränken sich dabei aber auf Komponenten mit Softwareinhalten [WW02]. Mit Almefelt et. al. findet sich eine der wenigen Veröffentlichungen, die mehr als nur reine Software-, Funktions- oder Elektronik-Anforderungen untersuchen [ABNM06]. Eine ihrer Empfehlungen ist, Entscheidungen auf der Basis der gesamten Kosten und des gesamten Nutzen zu treffen, was im weiteren Sinne auf die Untersuchungen der vorliegenden Arbeit einschließt.

Nuseibeh und Easterbrook haben bereits im Jahr 2000 eine Roadmap für das Requirements Engineering aufgestellt [NE00]. Diese viel zitierte Arbeit stellt die Modellierung in den Vordergrund und zwar einerseits von Anforderungen selbst und andererseits von der Domäne und explizit von der Unternehmensumgebung. Zwar fehlt eine explizite Nennung von Komplexitätseffekten oder der Kostenaspekt, jedoch sehen sie die Modellierung von Unternehmen in Anforderungsmodellen insbesondere hilfreich bei der Formulierung von Anforderungen auf einer hohen Ebene. Desweiteren plädieren sie dafür, Requirements Engineering nicht als separate Disziplin zu begreifen sondern als eine Sammlung von Methoden und Werkzeugen. Die vorliegende Arbeit greift beide Gedanken auf, indem einerseits der Modellierungsaspekt der Roadmap bearbeitet wird und andererseits die Integration von anderen Themenfeldern wie Projektmanagement und die Planung von Produkteigenschaften. Die Verknüpfung von Anforderungen an ein gesamtes Fahrzeug und damit auf einer sehr hohen Ebene steht im Vordergrund.

Die Verfügbarkeit von Literatur, welche Anforderungsmanagement in der Gesamtfahrzeugentstehung mit Komplexität und Prozessen verknüpft, ist ebenfalls gering. Schraps et al. stellen beispielsweise einen Ansatz vor, wie ein Enterprise-Domain-Network realisiert werden kann und gleichzeitig Wissen über die Struktur von Produkten in dessen Kontext gestellt werden kann [SRG11]. Auch sie verwenden hierzu semantische Netze, die Verknüpfung zum Anforderungsmanagement fehlt allerdings.

Klute et al. entwickeln im Rahmen des SFB 696 ein Anforderungsmanagementsystem, welches ähnlich dem in dieser Arbeit beschriebenen mit Hilfe einer Ontologie arbeitet [KKR11]. Ziel ist, Anforderungen maschineninterpretierbar zu machen um so strukturiert und automatisiert Zusammenhänge auswerten zu können. Sie entwickeln hierzu insbesondere eine Struktur zur Einordnung von Anforderungen. Ihre Ontologie ist jedoch ausschließlich auf ein späteres Produkt fokussiert und bietet keine Möglichkeit, Prozessartefakte mit einbeziehen zu können. Auch stellen sie ihre Arbeit grundsätzlich in den Kontext von Komplexitätsbetrachtungen von Produkten, nicht aber in den größeren Kontext der Unternehmensprozesse.

Mayer-Bachmann stellt das Anforderungsmanagement des Gesamtfahrzeugs in den Vorder-

grund [MB08]. Der Fokus wird hierbei auf die sog. Produkt-DNA und deren Evolution gelegt. Es findet jedoch keine Einbindung der das Fahrzeug umgebenden Prozessobjekte in das beschriebene Vorgehen statt. Detaillierte Informationen zu den Prozessen der Fahrzeugentwicklung finden sich bspw. bei Morgan und Liker [ML06].

Kof et al. wiederum nähern sich der integrativen Seite des Anforderungsmanagements, welche die Verknüpfung der Informationen in Dokumenten vornehmen und somit den Aufwand der Herstellung und Haltung von Konsistenz minimieren kann [KGRS10]. Sie zeigen damit zwar die Möglichkeiten, welche eine erweiterte Traceability bietet, setzen ihren Fokus jedoch nicht auf den Automobilsektor. Von Ramesh hingegen wird bereits 2002 der Zusammenhang zwischen Traceability und Prozesswissen hergestellt [Ram02]. Er ist es auch, der in Zusammenarbeit mit Jarke ein Referenzmodell für Anforderungen basierend auf empirischen Untersuchungen erstellt.

Konrad und Degen führen eine Analyse von vier Projekten durch, die mit Artefaktmodellen gearbeitet haben [KD09]. Unklar bleibt, wie exakt die verwendeten Modelle aussehen, allerdings bildet die verwendete Methodik – die Definition bestimmter Oberklassen von Anforderungsmanagement-relevanten Artefakten und die Ableitung spezifischer für ein Unternehmen und/oder Projekt – eine Grundlage für die in dieser Arbeit verwendete Vorgehensweise. Im Gegensatz zu ihrem Vorgehen wird hier allerdings ein Basismodell an die Hand gegeben, welches Unternehmen als Grundlage verwenden können. Darüber hinaus ist der Rahmen größer, in dem Artefakte erfasst werden. Die Vor- und Nachteile, die Konrad und Degen erarbeiten, gelten allerdings ebenfalls für diese Arbeit.

Grundlegend im Bereich des Komplexitätsmanagements ist die Arbeit von Wirtz [Wir08]. Er führt aus, inwiefern zunehmende Komplexität zu steigenden Gemeinkosten führt. Der Beitrag ist als Grundlagenwerk losgelöst von der erweiterten Thematik dieser Arbeit, zeigt allerdings den genannten wichtigen Zusammenhang ausführlich. Die Beiträge von Eigner und Stelzer sowie Mossmer et. al. hingegen gehen detaillierter auf die Problematik zunehmender Komplexität im Kontext der Automobilindustrie ein [ES09, MSG07]. Rishi et al. analysieren in ihrem Beitrag die aktuellen Einflüsse auf die Automobilindustrie und untersuchen sie detailliert unter anderem durch zahlreiche Interviews in diesem Umfeld [RSG08]. Als Ergebnis zeigen sie dessen nähere Zukunft bis in das Jahr 2020 auf. Die Arbeit bietet aufgrund des hohen Informationsgehalts durch die durchgeführten Interviews einen guten Einblick in die Herausforderungen, denen sich die Automobilindustrie stellen muss. Unter dem Begriff „Proaktive Flexibilität“ fassen die Autoren die Maßnahmen zusammen, die nötig sind um komplexe Entwicklungs- und Herstellungsressourcen und Prozesse schnell an veränderte Gegebenheiten anpassen zu können. Die Erkenntnisse dienen daher der vorliegenden Arbeit als Argumentation bei der Notwendigkeit der Erkennung und Erfassung von Zusammenhängen in komplexen Prozessartefakten. Nur wenn diese nutzbar sind, können die Auswirkungen von Entscheidungen auf die proaktive Flexibilität approximiert werden.

Schomann führt eine detailreiche Analyse der aktuellen Veränderungstreiber in der Automobilindustrie durch und richtet dann den Blick auf die Auswirkungen, die diese Treiber auf die Komplexität der Produktentwicklung in den Unternehmen haben [Sch12]. Er kommt zu dem Schluss, dass sich die Komplexität in den letzten Jahren im Bereich der Produktentwicklung stark erhöht hat und identifiziert diese als zentrale Herausforderung. Die Dissertation beschäftigt sich im weiteren Verlauf mit den Managementansätzen, welche die Automobilindustrie zur Beherrschung der Komplexität einsetzt und identifiziert ebenfalls die Problematik der steigen-

den Belastungen in den Unternehmensprozessen, die sich aus den Komplexitätseffekten bei Entscheidungen in der Produktentwicklung ergeben. Somit liefert der Autor eine aus der Managementperspektive geschriebene Problemstellung zur vorliegenden Arbeit, ohne jedoch auf die konkreten Kostenfaktoren einzugehen, wie es bspw. bei Wirtz der Fall ist [Wir08].

Butz verknüpft zunehmende Systemkomplexität mit den dazugehörigen Komplexitätseffekten wie etwa dem Auftreten unerwarteter Fehlerzustände von Komponenten [But11]. Er führt hierbei den Gedanken von Wirtz weiter, indem die Auswirkungen einer hohen und zunehmenden Produkt-Komplexität von einer technischen Seite betrachtet werden. Im Gegensatz zu Wirtz fokussiert er sich allerdings auf das Produkt und betrachtet die Komplexität in Unternehmensprozessen nicht weiter.

Loos beschreibt bereits 1996 die Möglichkeiten, die sich aus der Generierung bzw. Konfiguration von Software anhand eines Meta-Modells ergeben [Loo96]. Sein Fokus liegt hierbei auf der möglichst adäquaten Abbildung von Geschäftsprozessen in Software. Er identifiziert im Rahmen der kritischen Würdigung aber bereits Probleme, die sich bei der Verwendung generischer Modelle ergeben. Diese sind offensichtlich ebenfalls präsent bei der Verwendung semantischer Netze als Meta-Strukturen für artefaktzentrierte Anforderungsmanagement-Software. Insbesondere die Problematik der Interpretation der Instanzen aus den generischen Strukturen ist bei semantischen Netzen immanent, soll aber gerade durch die durchgängige Modellierung von Vererbungshierarchien nicht nur der Begriffe, sondern auch der Relationen reduziert werden.

Grundlegendes im Rahmen der Modellierung von Prozessen und deren Artefakte geleistet hat Scheer [Sch92]. Im Rahmen des ARIS-Konzepts führt er aus, auf welche Art Prozesse von Unternehmen formal zu beschreiben sind. Er definiert dazu unterschiedliche Sichten, die Prozesse aus verschiedenen Perspektiven darstellen. Ziel ist die Prozessmodellierung als Grundlage für die Generierung von Software [And06].

Kohl stellt die sog. „Objektorientierte Analyse“ eines Unternehmens vor [Koh96]. Diese bildet prinzipiell die Grundlage der Erarbeitung jedweder modelltechnischen Abbildung eines Unternehmens oder einer speziellen Domäne innerhalb eines solchen. Entsprechend ist diese Analyse auch Grundlage des in dieser Arbeit entwickelten Verfahrens zur Erarbeitung einer Artefakt-Landschaft. Es werden jedoch keine Vorschläge hinsichtlich gemeinsamer Oberklassen gemacht, außer sog. *Dokumente*, die Input- und Output-Elemente in einem Prozess darstellen. *Dokumente* finden sich ebenfalls in der Core-Ontologie in Abschnitt 6.2.2.

Das Projekt iglos des Instituts für Verkehrssicherheit und Automatisierungstechnik der TU Braunschweig erarbeitet eine technische Terminologie und untersucht darüber hinaus Methoden zur Erstellung von projektspezifischen Terminologien auf dieser Basis [Insa]. Das Projekt „iglos req“ führt den Gedanken weiter und überträgt die Möglichkeiten der umfassenden technischen Terminologiedatenbanken auf das Requirements Engineering [Insb]. Der Fokus liegt auf der Verwendung eines einheitlichen und konsistenten Fachvokabulars und damit auf der Erhöhung der Qualität der Anforderungen.

Wekse zeigt unter anderem die Geschichte der Entwicklung von Enterprise System Architekturen auf [Wes07]. Beginnend mit den Anfängen der Computertechnologie über Wiederverwendung von Komponenten und der Integration verschiedener Systeme etwa durch Datenbanken, beschreibt er das Zusammenwirken moderner Systeme durch die Verwendung einer zentralen Software für den Austausch von Informationen und Anweisungen. Interessant im Rahmen dieser Arbeit

ist hierbei die Notwendigkeit von gemeinsamen Meta-Modellen, damit ein solcher Austausch stattfinden kann. Weske fokussiert hierbei auf die verschiedenen Sprachen des „Business Process Managements“, die aktuell vorhanden sind.

Woitsch und Karagiannis verknüpfen bereits Wissensmanagement und Prozessmanagement miteinander und beschreiben einen konzeptionellen Ansatz für prozessorientiertes Wissensmanagement [WK05]. Durch semantische Technologien wird ein Webservice konstruiert, der Prozesswissen aufnehmen und wieder zur Verfügung stellen kann.

Diese Arbeit schließt eine Lücke in der Literatur, die sich zwischen Prozesskomplexität auf der einen und Produktkomplexität auf der anderen Seite aufspannt. Während Anforderungsmanagement vor allem auf letzteres fokussiert ist, ist absehbar, dass die dort entwickelten Methoden in einem größeren Kontext ebenfalls effektiv sein können. Modellierung von Prozesselementen durch semantische Netze bilden eine aussichtsreiche Erweiterung des Anforderungsmanagements, welches bereits heute auf maschineninterpretierbare Sprachen setzt. Es fehlt allerdings eine theoretische Grundlage für eine solche Erweiterung sowie eine allgemein verfügbare formale Struktur für Prozessartefakte. Auch ist kein Verfahren für die Erhebung derselben in diesem Kontext verfügbar. Methoden des Wissensmanagements und moderne Webtechnologien bilden die Grundlage für eine Erweiterung des Fokus des Anforderungsmanagements.

Kapitel 2

Grundlagen des Anforderungsmanagements

Subsumiert unter dem Begriff „Anforderungsmanagement“ werden seit einigen Jahren verschiedene Themen, die grundlegend die sinnvolle und nützliche Sammlung, Dokumentation, Einordnung und Pflege von Anforderungen an ein zu entwickelndes Produkt zum Thema haben. Anforderungen sind hierbei – allgemein formuliert – gewünschte Funktionalitäten oder Eigenschaften, die das resultierende Produkt besitzen soll. Sie formulieren also, wie Robertson und Robertson es ausdrücken, einen „business need, den das Produkt befriedigen soll“ [RR06]. Entsprechend ist das „Requirements Engineering das systematische Verstehen, was der Kunde möchte“ [HWFP08]. Ist das Vorgehen der exakten Formulierung einer Produktbeschreibung in den technischen Disziplinen bereits lange etwa in der Form von Lastenheften bekannt, so erfuhr das weite Feld des Anforderungsmanagements in der Software-Entwicklung besondere Bedeutung. Hintergrund ist die zunehmende Komplexität jeder Art von Software und die Änderungen, die aus der intangiblen Natur von Programmen oder Systemen resultieren. Mit der zunehmenden Verzahnung von Produkten wie Fahrzeugen und Software, aber auch durch die wachsende Komplexität der Produkte und ihrer virtuellen Repräsentation als Daten werden die Methoden und Werkzeuge, die das Software Engineering innerhalb der letzten zehn bis fünfzehn Jahre erarbeitet hat, für die anderen technischen Diszplinen interessant. Es schließt sich somit der Kreis zwischen der Spezifikation von komplizierten physischen Produkten und den komplexen Spezifikationen im Software Engineering.

2.1 Grundzüge und Definition des Anforderungs-Begriffs

Die Begründer des Requirements Engineering stellen fest, dass „Teams exakt das erstellen, was von ihnen verlangt wird“ und im Umkehrschluss dementsprechend nichts von dem, was ihnen im Vorfeld nicht bekannt war. Es kommt zu fehlerhaften Produkten, wenn ihnen fehlerhafte Vorgaben gemacht wurden [GW89]. Das Herstellen eines gemeinsamen Verständnisses im Vorfeld ist daher kritisch für jede Art von Produkt und somit auch für diese Arbeit. Daher werden zuerst die im Folgenden verwendeten Begriffe definiert, um im weiteren Verlauf auf eine einheitliche Wortwahl zurückgreifen zu können. Die Verwendung vieler Begriffe ist in der Literatur unterschiedlich, was unter anderem auf verschiedene Übersetzungen aus dem Englischen zurückzuführen ist, aber auch durch die unterschiedlichen Fachdisziplinen begründet ist. Beispielsweise wird die Festlegung, welcher der beiden Begriffe „Requirements Management“ und „Requirements Engineering“ der Oberbegriff des jeweils anderen ist, vor allem in der Praxis kontrovers diskutiert. Hood

et al. führen eine ganze Reihe von Definitionen an, die an dieser Stelle berücksichtigt werden sollen [HWFP08]. Allerdings wird entgegen ihrer Verwendung des Begriffs „Requirements Management and Engineering“ synonym der Begriff „Anforderungsmanagement“ als Oberbegriff genutzt. Für den Begriff „Anforderung“ kann analog die Definition von Hood et. al verwendet werden:

Definition 2.1.1 (Anforderung (nach [HWFP08])) *Eine Anforderung ist definiert als eine Aussage, die eine Fähigkeit, physische Charakteristik oder einen Qualitätsfaktor beschreibt und damit eine Produkt- oder Prozess-Notwendigkeit identifiziert, für die eine Lösung erarbeitet wird.*

Alternativ findet sich im Institute of Electrical and Electronics Engineers (IEEE) Standard Glossary of Software Engineering Terminology eine abweichende Definition für den Begriff „Anforderung“ [IEE90]. Diese sei hier nur der Vollständigkeit halber erwähnt, stammt sie doch aus dem Bereich des Software Engineering, während die Definition nach [HWFP08] ein breites Spektrum an möglichen Produkten abdeckt und damit dem Fokus dieser Arbeit eher entspricht. Wichtig ist vor allem die Abgrenzung zur Lösung, welche eine konkrete Beschreibung der angestrebten Umsetzung ist. Anforderungen und die in ihnen enthaltenen Texte müssen der Theorie nach frei von Lösungen sein, um den sie interpretierenden Entwicklern keine Vorgaben zu machen, auf welche Weise sie zu lösen sind [RR06]. In der Praxis ist diese Trennung oftmals schwer durchzuhalten [HF00].

Kamata und Tamai verknüpfen statistisch den Erfolg von Softwareprojekten mit der Qualität von Anforderungen und stellen unter anderem fest, dass einige wenige Faktoren einen hohen Einfluss auf den Projekterfolg haben [KT07]. Die Qualität und Vollständigkeit der Anforderungen korreliert direkt mit der Qualität des Produkts, allerdings werden in der Praxis kaum Maßnahmen zur Überprüfung und Anhebung der Qualität der Anforderungen eingesetzt [HP08]. Die IEEE definiert Kriterien für die Qualität von Anforderungen [IEE98]:

- **Korrektheit:** Korrektheit bedeutet in diesem Fall, dass die Anforderungen mit den Wünschen der Kunden übereinstimmen und sie weiterhin im Rahmen der gesamten Spezifikation und anderer Prozess- und Projektdokumente durchgängig stimmig ist.

- **Eindeutigkeit:** Eine Anforderung darf nur eine einzige Interpretation besitzen. Die IEEE schlägt vor diesem Hintergrund vor, nicht vermeidbare und mehrdeutige Begriffe in einem Glossar anzuführen [IEE98]. Für die Formulierung von Anforderungen in natürlichen Sprachen existieren inzwischen viele Vorschläge und Praktiken zur Reduktion der Mehrdeutigkeit durch die Verwendung immer gleicher Begriffe und Ausdrücke. Die klare und eindeutige Formulierung des Texts von Anforderungen ist eine vitale Qualitätseigenschaft, da hiervon das Verständnis der sie umsetzenden Entwickler direkt beeinflusst wird. Die automatisierte Analyse der Texte einerseits und die Unterstützung der Eingabe andererseits sind intensive Forschungsgebiete [HMD11, DBK03]. Auch das Verwenden von nur an natürliche Sprachen angelehnte Sprachen wird diskutiert [MWHN09]. Der Vorteil hiervon ist, dass eine Anforderungsbeschreibung zwar für den Menschen einfach verständlich

bleibt, allerdings durch Software interpretierbar wird.

- **Vollständigkeit:** Diese bezieht sich einerseits auf den Inhalt der Anforderung selbst und andererseits auf die Menge der Anforderungen. Ersteres kann bspw. durch die Einführung eines Templates sichergestellt werden: Eine klare Vorgabe hinsichtlich der zu erfassenden Felder minimiert das Risiko, unvollständige Anforderungen zu erhalten. Der zweite Punkt ist sicherlich schwieriger umzusetzen. Allerdings gibt es diverse Vorschläge und Methoden für das Erfassen von Anforderungen [KS98, BS12].

- **Konsistenz:** Betrifft in Ergänzung zur Korrektheit die Konsistenz der Anforderungen untereinander, also bspw. die Widerspruchsfreiheit.

- **Priorisierung nach Wichtigkeit und/oder Stabilität:** Zur Projektplanung muss festgelegt werden, welche Anforderungen in welcher Reihenfolge umgesetzt werden. Hieraus lassen sich Roadmaps, Projektstruktur- und Projektzeitpläne sowie Rollen und Verantwortlichkeiten ableiten.

- **Verifizierbarkeit:** Die Umsetzung der Anforderungen in einem Produkt muss mit verhältnismäßigem Aufwand überprüfbar sein. Insbesondere ist ein eindeutiges Kriterium in der Formulierung von Anforderungen notwendig, welches sich dazu eignet, die Umsetzung einer Anforderung im späteren Produkt positiv oder negativ evaluieren zu können.

- **Modifizierbarkeit:** Eine Anforderung muss einfach zu ändern sein, ohne das bspw. durch Redundanzen die Konsistenz der gesamten Menge an Anforderungen in Frage gestellt wird. Hierbei bezieht sich die Modifizierung auf den Text der Anforderung selbst, nicht auf den Aufwand, der sich aus den Auswirkungen der Änderung auf andere Objekte ergibt. Die Erweiterung der Aussage lässt aber durchaus die Anwendbarkeit auf die Einfachheit der Abschätzung von Auswirkungen zu.

- **Traceability:** Für diese Arbeit wird Traceability in den Definitionen 2.3.1 und 2.3.2 beschrieben. Eine Anforderung erfüllt das Kriterium der Traceability, wenn Verknüpfungen zu vor- und nachgelagerten Elementen im Rahmen der Projektarbeit vorhanden sind und diese nachverfolgt werden können. Rückwärtsgerichtete Traceability bedeutet die klare Identifizierbarkeit des Verursachers der Anforderung, vorwärtsgerichtete die Verknüpfung mit weiteren Objekten wie z.B. Tests. Es wurde bereits festgestellt, dass Traceability zu den wichtigsten Eigenschaften eines Anforderungsmanagementwerkzeugs gehört. Bereits bei der Erfassung der Anforderungen ebenso wie im späteren Verlauf ist sicherzustellen, dass alle relevanten Verknüpfungen auch tatsächlich hergestellt werden. Oftmals muss dies bei Werkzeugen noch manuell durchgeführt werden. Auf das Konzept der Traceability wird im Rahmen des Abschnitts 2.3 näher eingegangen.

Robertson und Robertson ergänzen die Kriterien der IEEE bzw. konkretisieren sie durch gezielte Anmerkungen [RR06]. Sie kommen auf insgesamt zehn Kriterien, die sich an folgenden Punkten von denen der IEEE unterscheiden:

- **Relevanz:** Während der Anforderungserhebung ist sicherzustellen, dass die Anforderungen für das Projekt relevant sind und nicht bereits durch andere Anforderungen abgedeckt werden. Wichtig ist hierfür eine frühe Definition von Umfang und Ziel des Projekts sowie die Verwendung eines Feldes „Begründung“ in der Anforderung, in dem der Autor der Anforderung klar darlegen muss, warum sie für das Projekt notwendig ist.
- **Eindeutigkeit:** Mit Hilfe eines quantifizierten „Fit Criterion“, also eines Felds welches einen einfachen Test beschreibt, an dem die korrekte Umsetzung erkannt werden kann, besteht die Möglichkeit festzulegen, wie die Anforderungen zu interpretieren sind. Es erweitert damit die Forderung nach Verifizierbarkeit um eine klare Aussage hinsichtlich der Umsetzung dieser Forderung.
- **Realisierbarkeit:** Die Anforderung muss im Rahmen des Projekts umsetzbar sein. Hierzu sind aussagekräftige Aufwandschätzungen und Auswirkungsanalysen notwendig.
- **Lösungs-Unabhängigkeit:** Die Anforderung darf keine Lösungen vorwegnehmen sondern enthält eine reine Problembeschreibung.
- **Gold Plating:** Es ist zu vermeiden Anforderungen einzubringen, die mehr Kosten als Nutzen verursachen.
- **Einschleichen:** Anforderungen, die nach Abschluss der Anforderungserhebungsphase an das Projekt gestellt werden, sind besonders hinsichtlich Relevanz, Durchführbarkeit und Gold Plating zu prüfen.

Während sich die genannten Punkte vor allem auf die Art und den Inhalt der Beschreibung von Anforderungen beziehen, stellt Young unterschiedliche Typen von Anforderungen in einen prozessualen Zusammenhang. Der Ansatz ist hierbei, das Anforderungen als abstraktes Konstrukt aus mehreren Quellen im Rahmen eines Anforderungsmanagementprozesses sortiert werden in spezifischen Beschreibungen für Systemkomponenten münden [You03]:

Young beginnt mit der Einordnung von Kundenwünschen und -erwartungen:

- Fachliche Anforderungen
- Anforderungen der Anwender
- Anforderungen an das Produkt
- Umweltanforderungen
- Unbekannte Anforderungen

Diese werden durch einen Anforderungsanalysten analysiert und in folgenden Formen beschrieben:

- Anforderungen auf einer hohen (Gesamtsystem-)Ebene
- Funktionale Anforderungen

- Nichtfunktionale Anforderungen
  - Systemeigenschaften
  - Spezielle Entwicklungsanforderungen
- Abgeleitete Anforderungen und Designvorgaben
- Leistungsanforderungen
- Schnittstellenanforderungen

Die Systemanforderungen werden getrennt in:

- Subsysteme (lokale Funktionsgruppierungen)
- Systemkomponenten (Hardware, Software, Traning, Dokumentation)

Grundlage für eine solche Betrachtung von Anforderungen im Rahmen eines Prozesses ist eine klare Nachverfolgbarkeit jeder einzelnen Anforderung während ihres Lebenszyklus sowie eine gemeinsame Terminologie basierend auf einem Glossar [You03]. Die Nachvollziehbarkeit wird durch einen eindeutigen Schlüssel hergestellt, der jeder Anforderung bei ihrer Erstellung zugeordnet wird.

2.2 Begriff und Aufgaben des Anforderungsmanagements

Das Anforderungsmanagement stellt Methoden und Werkzeuge bereit, um die während der Phase der Anforderungserhebung erstellten Anforderungen mit den anderen Disziplinen zu verknüpfen und die Einbindung zu gewährleisten, wie in Abbildung 2.1 dargestellt. Hierzu gehören Maßnahmen zur Sicherstellung der Traceability, Impact-Analyse, Steuerung, Kontrolle, Verwaltung, Planung und Umsetzung von Anforderungen. Die Anforderungserhebung wiederum besteht aus den Schritten Anforderungsaufnahme, -spezifikation, -analyse und -bewertung. Diese Schritte überlappen einander und werden oft mehrfach – iterativ – durchgeführt, bis hin zur Erstellung und Realisierung konkreter Entwicklungsaufträge [Ebe10]. Für die Definition des Begriffs „Anforderungsmanagement“ werden wiederum Hood et. al. herangezogen:

Definition 2.2.1 (Anforderungsmanagement (nach [HWFP08]))
Anforderungsmanagement ist die Summe aller Schnittstellen zwischen Anforderungserhebung und allen weiteren Teilaufgaben des Systems Engineerings zur effizienten und risikokontrollierten Entwicklung komplexer Systeme.

Als Schnittstellenfunktion hat das Anforderungsmanagement vor allem eine *Koordinationsaufgabe* mit dem Ziel der *Informationsversorgung*. Diese Informationen sind offensichtlich die Anforderungen an sich, aber auch Metainformationen wie Modelle, Glossare und Statistiken. Damit kommt dem Anforderungsmanagement eine vitale *Unterstützungsfunktion* zu, die sich über

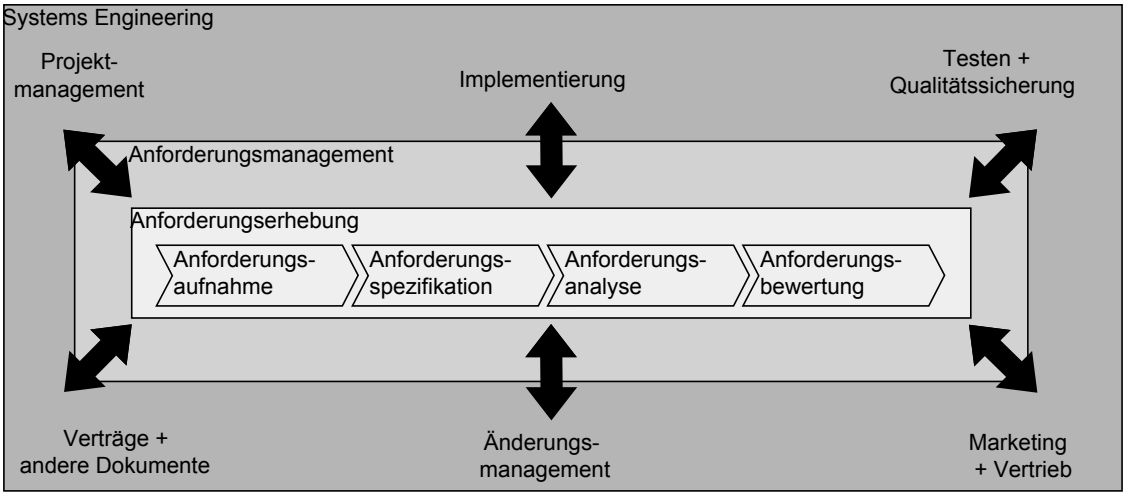


Abbildung 2.1: Die Einordnung von Anforderungsmanagement im Rahmen des Systems Engineerings (nach [HWFP08])

den gesamten Lebenszyklus des Produkts erstreckt. Diese beginnt mit dem Aufnehmen von Ideen oder Konzepten in der ersten Phase eines Entwicklungsprojekts im Rahmen der Anforderungserhebung und setzt sich beim Projektmanagement anhand des Reifegrads eines Produkts auf Basis umgesetzter Anforderungen fort. Bis hin zu späten Änderungen und der Abschätzung deren Auswirkungen, muss das Anforderungsmanagement korrekte und qualitätsgesicherte Informationen in kürzester Zeit zielgerichtet den richtigen Personen zur Verfügung stellen. Zur Sicherstellung der Informationsversorgung hat das Anforderungsmanagement geeignete Maßnahmen wie Prozesse, Methoden und Werkzeuge zu etablieren, koordinieren und zu warten.

Anforderungsmanagement unterstützt somit insbesondere folgende Tätigkeiten [HWFP08, HF00, GW89]:

- **Projektmanagement:** Vor allem in der Projektplanung und -kontrolle wird das Projektmanagement durch die Möglichkeit unterstützt, klare Zielvorgaben auf der Basis vorhandener Anforderungen einzuführen und diese zu überwachen. Das Planen der Projekt-Roadmap, die Produktkonfiguration und die Releaseplanung werden auf der Basis von Anforderungen durchgeführt. Auch im Rahmen des Risikomanagements in Projekten spielen Anforderungen und deren Status eine Rolle.

- **Implementierung:** Durch klare Sprache, eindeutige Definitionen, Vermeidung von Mehrfachverwendung von Begriffen sowie die Nachvollziehbarkeit bestimmter Entscheidungen wird es Entwicklern leichter gemacht, sich auf die Problemlösung zu konzentrieren.

- **Testen und Qualitätssicherung:** Nicht nur im Bereich der Softwareentwicklung können Tests bereits teilweise aus bestehenden Anforderungen automatisch generiert werden. Auch in der ganzheitlichen Systementwicklung führt das Testen gegenüber klaren Anforderungen zu einem sehr strukturierten Vorgehen und erleichtert das Planen von Tests.

- **Marketing und Vertrieb:** Basierend auf den bereits umgesetzten sowie den geplanten Anforderungen kann das Marketing bereits zur Projektlaufzeit auf Eigenschaften und Funktionen des zukünftigen Produkts ein fokussierte Vorgehen entwickeln.

- **Änderungsmanagement:** Das Management von Änderungen, besonders wenn Prozessobjekte betroffen sind, die viele Berührungspunkte zu anderen Projekten oder Komponenten haben, erweist sich als sehr komplex. Durch die Verknüpfung der Objekte mit Anforderungen, oben eingeführt als das Qualitätskriterium „Traceability“, können Auswirkungen im Vorfeld abgeschätzt werden.

- **Verträge und andere Dokumente:** Vor allem in der Automobilindustrie sind Verträge und Lastenhefte mit und für Zulieferer oder Partnerunternehmen essentiell. Einerseits erleichtern klare Spezifikationen von Lastenheften die Arbeiten auf Auftragnehmerseite, andererseits bieten sie dem Auftraggeber Rechtssicherheit. Das automatisierte Erstellen von Lastenheften und anderen wichtigen Dokumenten kann durch das Anforderungsmanagement unterstützt werden.

Als Erkenntnis lässt sich herausstellen, dass die Notwendigkeit der Abgrenzung eines formalen Anforderungsmanagements von einem nicht-formalen besteht. Basierend auf Definition 2.2.1 wird an dieser Stelle daher eine Konkretisierung vorgenommen.

Das Anforderungsmanagement tritt in unterschiedlichen Ausprägungen auf: In der nicht-formalen Variante werden zum – an dieser Stelle nur theoretisch vorhandenem – Zeitpunkt der Anforderungserhebung Anforderungen bspw. im Gespräch oder aber auch schriftlich in Fließtextform ausgetauscht. Das Vorgehen ist hierbei insofern nicht formal, da eine eventuelle Niederschrift der Anforderungen bzw. Wünsche nicht in einer maschinenlesbaren Form stattfindet. Hiermit ist eine durchgängige Verknüpfung der Anforderungen über verschiedene Konkretisierungs- und Detaillierungsebenen hinweg sowie zu anderen Prozessobjekten nicht möglich. Diese Form findet bspw. beim Kauf eines Fahrzeugs durch einen Kunden statt, der im Gespräch mit dem Kundenberater das gewünschte neue Auto verbal beschreibt.

Ähnliches ist auch in Unternehmen zu beobachten: Teilweise sind Anforderungen formalisiert verfügbar. Die durchgängige Verknüpfung findet aber selten statt. Es kann somit zwar automatisiert approximiert werden, welche Anforderungen oder Tests von einer bestimmten Änderungen an einer Anforderung betroffen sind, nicht jedoch was diese Änderung beinhaltet. Der Inhalt wird einerseits durch den Textinhalt der Anforderung bestimmt, andererseits durch die Einbindung von dieser in einen Kontext bestehend aus sie umgebenden Elementen. Eine automatisierte Aufwandsschätzung einer Änderung bleibt somit unmöglich, solange nur die nicht formale Form vorliegt.

Das Hinterlegen von Anforderungen mit Modellen etwa des Produkts sind ein erster Schritt in Richtung der formalen Variante des Anforderungsmanagements. Einige Werkzeuge bieten dies bereits an. Allerdings ist hierbei erstens die Gewährleistung der vollen Durchgängigkeit von High-Level Anforderungen über Teileebene hin zu Fahrzeugtests aufwändig und kostenintensiv. Zweitens findet keine Einbindung in einen prozessualen Kontext statt, es können also keine Prozesselemente außerhalb des direkten Projektkontextes modelliert und in das Anforderungsmanagement mit eingebunden werden.

2.3 Traceability als Konzept des Anforderungsmanagements

Der Begriff Traceability stammt ursprünglich aus der Softwareentwicklung. Bei der Definition für diese Arbeit wird darauf geachtet, ihn nicht ausschließlich auf die Verbindung von Anforderungen mit Programmcode zu reduzieren. In der Veröffentlichung von Ramesh und Jarke findet sich die folgende Definition [RJ01]:

Definition 2.3.1 (Traceability (nach [RJ01])) *Traceability ist die Charakteristik eines Systems Anforderungen klar ihren Quellen sowie nachgelagerten Prozessobjekten im Rahmen der Produktentstehung zuordnen zu können.*

Traditionell wurde unter Traceability die Möglichkeit verstanden, die Abhängigkeit von Anforderungen untereinander darzustellen. Als Erweiterung davon gilt das Vorgehen, Anforderungen zusätzlich noch mit vorhergehenden Informationen – z.B. Kundenwünschen – und nachfolgenden – wie bspw. Produktmodellen oder Tests – zu verknüpfen. Inzwischen wird damit grundsätzlich die Möglichkeit beschrieben, Assoziationen zwischen den verschiedenen Prozesselementen herstellen zu können und diese auch über die im Laufe des Projekts entstehenden Versionen und Varianten zu erhalten. Traceability bildet damit die Basis für sämtliche Anforderungsmanagement-Funktionalität wie bspw. die Analyse der Auswirkungen von Änderungen im Änderungsmanagement. Ihrer Bedeutung entsprechend bildet sie das Objekt intensiver Forschung [SL10, KOD09, EAG06]. Aufgrund der Bedeutung des Konzepts soll die Definition 2.3.1 als Definition der Traceability im engeren Sinne verstanden werden. Die folgende, in dieser Arbeit verwendete, somit als die im weiteren Sinne:

Definition 2.3.2 (Traceability im weiteren Sinne) *Traceability ist die Funktion eines Systems, Zusammenhänge von Prozessobjekten darstellen und verwalten zu können sowie Methoden zum Operieren hiermit zur Verfügung zu stellen.*

Dieser erweiterte Begriff löst die Traceability aus dem reinen Anforderungskontext und bietet somit die Möglichkeit, alle durch eine Produktentstehung berührten Prozessobjekte zu betrachten. Die Definition spielt in dieser Arbeit insbesondere im Kontext von Modellen eine Rolle, bei denen Objekte aus Prozessen als „Artefakte“ bezeichnet werden.

Kapitel 3

Evaluation des Anforderungsmanagements in der Marke Volkswagen

Die grundlegenden Prinzipien dieser Arbeit wurden im Rahmen eines Forschungsprojekts mit der Marke Volkswagen der Volkswagen AG erarbeitet. Für das weitere Verständnis der Arbeit sind Inhalt und Erkenntnisse desselben von Bedeutung. Vorbereitend für den Hauptteil der Arbeit folgt daher in diesem Kapitel die Beschreibung von Problemstellung, Vorgehen und Lösung des Projekts.

3.1 Problemstellung und Vorgehen

Das Projekt „durchgängiges Anforderungsmanagement für modulare Produktstrukturen“ wurde in Kooperation mit der Marke Volkswagen der Volkswagen AG von 2009 bis 2011 bearbeitet. Ziel war die Evaluation des Status quo des Anforderungsmanagements von Fahrzeugprojekten sowie das Aufzeigen innovativer Lösungen für zukünftige Herausforderungen im Rahmen der Einführung modularer Fahrzeugbaukästen. Im Fokus stand die „frühe Phase“ von Fahrzeugentwicklungsprojekten, in denen formal definierte Artefakte wie bspw. Stücklisten und informelle Artefakte wie Marktberichte und Produktkonzepte zusammentreffen. Das Anforderungsmanagement als zentraler Knotenpunkt für die verschiedenen Disziplinen des Systems Engineering nimmt hier seinen Anfang. Aufgabe ist die Sammlung und Einordnung der Informationen, Ideen und Anforderungen, die an das zu entwickelnde Produkt gestellt werden. Deutlich wird, dass das Abschätzen der Auswirkungen von Entscheidungen zu diesem Zeitpunkt von Herausforderungen geprägt ist. Mit der Einführung von verschiedenen modularen Fahrzeugbaukästen gewinnt diese Ausgangslage allerdings weiter an Dynamik: Einerseits stehen bereits zu einem frühen Zeitpunkt in der Produktentstehung formale Informationen zur Verfügung, andererseits haben Entscheidungen viel weitreichendere Auswirkungen.

Ziel des Projekts war, mithilfe der Szenariotechnik die weitere Entwicklung des Managements der Anforderungen für Fahrzeugprojekte zu analysieren. Hierbei treffen unterschiedliche Prozesse, Methoden, Systeme und Gruppen aufeinander, die an verschiedenen Zeitpunkten des Produktentstehungsprozesses und nicht unbedingt in denselben organisatorischen Einheiten tätig sind. Es kommt die von Pillkahn erarbeitete strukturierte Vorgehensweise zur Erstellung von Szenarien zum Einsatz [Pil07]. Anhand des Vergleichs mit dem Status quo wurden Handlungsempfehlungen abgeleitet sowie die Grundlagen für ein Vorgehen und ein Werkzeug skizziert, welches den durch

die modularen Fahrzeugbaukästen entstehenden Herausforderungen gewachsen ist. Anhand eines Tool-Scoutings wurde evaluiert, welche Anforderungsmanagementwerkzeuge bereits verfügbar sind und in die engere Auswahl für ein großes Industrieunternehmen kommen können. Mehrere Prototypen in unterschiedlichen Entwicklungsstufen runden das Projekt ab.

Folgende Projektergebnisse konnten gewonnen werden:

1. **Szenarien:** Vier Szenarien, welche die zukünftige Entwicklung des Anforderungsmanagement beschreiben, wurden erstellt. Ausgangspunkte sind Entwicklungen innerhalb des Unternehmens, wie z.B. die Einführung neuer Software sowie außerhalb, also z.B. neue gesetzliche Vorschriften. Es wurden 22 strukturierte Interviews durchgeführt, aus denen die Informationen für die Szenarien extrahiert sind.

2. **Tool-Scouting:** In Kombination mit den erarbeiteten Szenarien wurde eine Erhebung der am Markt verfügbaren Werkzeuge durchgeführt.

3. **Modellbasierter Prozesseditor:** Als Grundlage des in den folgenden Kapiteln vorgestellten Vorgehens und Werkzeugs zur durchgängigen formalen Verknüpfung von Prozesselementen wurde ein Editor entwickelt, der ein formales Modell eines Prozesses verarbeiten kann. Die Vorgehensweise zur Erstellung des formalen Modells, die formale Repräsentation des Modells sowie der Aufbau der Software selbst stellen Vorläufer von Methodik und technischer Umsetzung dar.

4. **Proof-of-Concept:** Mit Hilfe eines Werkzeug-Potentialkandidaten wurde für einige Prozessobjekte aus dem Entwicklungsprozess gezeigt, wie Anforderungsmanagementwerkzeuge Traceability im Sinne von Definition 2.3.2 über das reine Produkt hinaus zur Verfügung stellen können. Die hier gewonnenen Erkenntnisse flossen in das im späteren Kapitel 4 detailliert eingeführte Konzept der Prozessartefakte ein sowie in die grundlegenden Ontologien für die Abbildung von Artefaktmodellen in Kapitel 6.

5. **Konzeption Folgeprojekt:** Zur Umsetzung und Erprobung der konzipierten Ideen sowie der Beeinflussung der Entwicklung des Anforderungsmanagements im Gesamtfahrzeugbereich, wurde ein Folgeprojekt beschrieben.

Es folgt im anschließenden Abschnitt zunächst eine Einführung in die modularen Fahrzeugbaukästen, auf denen der Volkswagen-Konzern seine Strategie für kommende Fahrzeuggenerationen aufbaut. In 3.1.2 wird dargelegt, inwieweit sich die Veränderungen der Automobilindustrie konkret im Anforderungsmanagement widerspiegeln. Abschnitt 3.1.3 zeigt die detaillierte Problemstellung des Forschungsprojekts auf. Danach folgt die konkrete Anwendung der Methodik von Pillkahn in 3.2 sowie das Vorgehen des Tool-Scoutings in 3.3. Die Abschnitte 3.4 und 3.5 beschreiben zwei Prototypen zur Darstellung und Bearbeitung verknüpfter formaler Informationen. Zusammengefasst werden die Erkenntnisse des Kapitels in Abschnitt 3.6.

3.1.1 Modulare Fahrzeugbaukästen im Kontext des Anforderungsmanagements

Hüttenrauch und Baum bestimmen die modularen Fahrzeug-Baukästen als konsequente Fortsetzung der Plattform-Strategie [HB08]. Sie erweitern den Kreis der Einsetzer von Teilen und

Systemen auf den gesamten Konzern bei gleichbleibender Effizienz. Einsetzer meint an dieser Stelle die Verwendung eines Teils in einem Fahrzeug. Das Projekt mit Volkswagen hat seinen Ursprung nicht zuletzt in der Fragestellung, inwieweit das Anforderungsmanagement für die Entwicklung von Fahrzeugen durch die Strategie der Fahrzeugbaukästen beeinflusst wird. Waren Anforderungen traditionell auf ein Fahrzeugprojekt und somit die zugehörige Produktpalette beschränkt, können durch Beeinflussung von Teilen in den Fahrzeugbaukästen viele Produkte betroffen sein, die sich in unterschiedlichen Phasen des Lebenszyklus befinden.

Für diese Arbeit sind zwei Begriffe aus der Wortkonstruktion „modularer Fahrzeugbaukasten“ zu definieren: „Modul“ sowie „Baukasten“. Für den Begriff der „Module“ kann weitgehend die Definition von Hüttenrauch und Baum übernommen werden:

Definition 3.1.1 (Modul (nach [HB08])) *Ein Modul ist ein System aus Teilen oder Teil-Systemen. Eine bestimmte Menge hiervon ist als unveränderlicher Grundumfang gekennzeichnet, der Rest als an den Einsetzer anzupassender Variabilitätsumfang. Ein Modul verfügt über klar definierte Schnittstellen und Funktionen.*

Ein modularer Fahrzeugbaukasten ist somit ein spezifischer Baukasten, in dem Module für Fahrzeuge in einer Art Bibliothek gesammelt werden. Diese Module können in allen Fahrzeugen eingesetzt werden, für die der Baukasten definiert ist und deren konzeptbestimmende Maße somit in ihm berücksichtigt werden. Es wird nunmehr folgende Definition in dieser Arbeit verwendet:

Definition 3.1.2 (Modularer Fahrzeugbaukasten) *Ein modularer Fahrzeugbaukasten ist eine Sammlung von Modulen und Teilen, die bestimmten im Vorfeld definierten Kriterien genügen.*

Abbildung 3.1 zeigt, wie Projekte mit Produkten aus unterschiedlichen Fahrzeugklassen dieselben Module verwenden. Im weiteren Verlauf der Arbeit werden die Begriffe „Baukasten“ und „modularer Fahrzeugbaukasten“ synonym verwendet.

Hintergrund der Modul- und Baukasten-Strategie ist die Realisierung von Kostenvorteilen durch Skaleneffekte: Die Entwicklung bestimmter Systeme, beispielsweise von Klima-Kompressoren, muss nur einmal durchgeführt werden, statt in jeder Marke erneut. Zusätzlich wird die Marktmacht gegenüber Lieferanten gestärkt und es können bessere Konditionen aufgrund der vergrößerten Einkaufsmenge erzielt werden.

Doch Fahrzeugbaukästen bringen in der Praxis Probleme mit sich, die sich vor allem in der Komplexität der Abstimmungsarbeit finden. Der wichtigste und schwierigste Punkt ist die Definition eines gemeinsamen Vorgehens zwischen allen betroffenen Fahrzeugprojekten. Zu klären ist beispielsweise, wer die Module und Baukästen definiert, wie sie definiert sind, welche Fahrzeuge ein Modul verwenden oder in welchem Baukasten ein Modul enthalten ist. Auch muss bestimmt werden, wie und von wem die Verhandlungen mit Lieferanten geführt werden müssen, wer die Weiterentwicklung der Baukästen plant und durchführt sowie wie die Prozesse und Softwarewerkzeuge zur Koordination von Modulen und Baukästen funktionieren sollen.

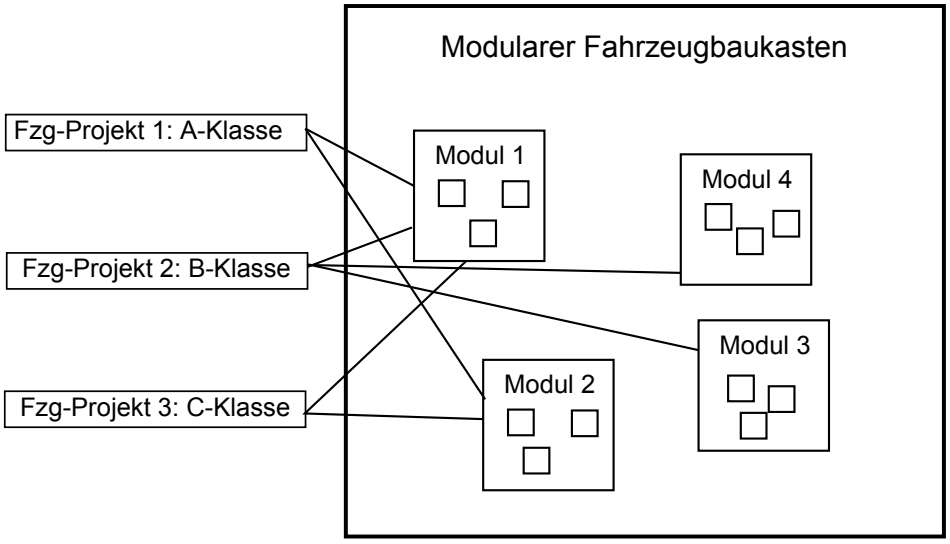


Abbildung 3.1: Modularer Fahrzeugbaukasten

Auch die Änderungsprozesse müssen angepasst werden, da von Änderungen bedeutend mehr Fahrzeuge betroffen sein können. Der hierfür benötigte Aufwand ist groß und strapaziert die Unternehmen, da Prozesse verändert, abgeschafft und kombiniert werden, die teilweise über Jahrzehnte hinweg erprobt waren.

Fahrzeugbaukästen verursachen darüber hinaus einen starken Anstieg in der Komplexität der Prozesse, da nun bspw. die Beschaffung eines Moduls über mehrere Marken hinweg koordiniert werden muss. Diese verwenden unterschiedliche Systeme für Beschaffungsaufträge, Stücklisten oder die Verwaltung von Modulen. Zusätzlich sind die zuständigen Mitarbeiter an unterschiedlichen Positionen in den jeweiligen Marken-Personal-Hierarchien verortet.

Weiterhin bringen Fahrzeugbaukästen eine bisher ungekannte Dimension von Anforderungen auf verschiedenen Ebenen hervor. Zeitliche Planungen von Fahrzeugprojekten in deren Rahmen ein Modul entwickelt wird, haben nun Auswirkungen auf in der Zukunft liegende Fahrzeugprojekte basierend auf anderen Plattformen in anderen Marken. Anforderungen z.B. hinsichtlich des Bauraums müssen bereits im Vorfeld antizipiert werden. Im Umkehrschluß entstehen durch zukünftige Fahrzeuge Anforderungen an Module, die sich aktuell in der Entwicklung befinden. Werden all diese Einflussfaktoren nicht beachtet, gehen die Kosteneffekte verloren, da die Module nicht plattformübergreifend eingesetzt werden können und daher die Entwicklungen zumindest teilweise wiederholt werden müssen.

3.1.2 Allgemeine Problemstellung des Anforderungsmanagements in der Automobilindustrie

Das Anforderungsmanagement ist von den in Abschnitt 1.1 beschriebenen Problemstellungen der Automobilindustrie in besonderem Maße betroffen. Die folgende Auflistung beschreibt, welchen Einfluss die einzelnen Herausforderungen speziell auf die Aufgabenfelder des Anforderungsma-

nagements haben:

- **Stärkere Differenzierung von Märkten:** Es ist eine zunehmende Verfeinerung der Marktsegmente zu beobachten. Dies resultiert einerseits aus der Sättigung der traditionellen Märkte in den westlichen Ländern und andererseits aus einer fortschreitenden Erschließung der (früheren) Schwellenländer [Hei97]. Die Herausforderung besteht an dieser Stelle in Form einer gesteigerten Zahl von Anforderungen, je weiter die Fragmentierung der vorhandenen Marktsegmente fortschreitet, bzw. neue hinzukommen. Diese müssen alle im Vorfeld der Entwicklung geklärt und berücksichtigt werden. Es wird zunehmend schwieriger die Auswirkungen von Änderungen in der Planung der Eigenschaften auf Märkte zu beachten.

- **Höhere Zahl an Modellen und Varianten:** Als direktes Resultat aus dem ersten Punkt gilt die erhöhte Zahl an Fahrzeugtypen und -varianten. Varianten schließen hier einerseits unterschiedliche Modelle auf derselben Plattform (bspw. VW Golf und VW Golf Variant) und andererseits Ausstattungspakete (Trend-, High-, Comfortline oder Assistenzsysteme) ein. Jede dieser Varianten muss verwaltet werden und unterliegt Änderungen, die sich wiederum auf andere Varianten auswirken.

- **Starke Klimaschutzvorgaben:** Bereits seit einigen Jahren sind für die Fahrzeughersteller strengere Klimaschutzgesetze relevant [ARSLD03]. seit dem Jahr 2012 gilt darüber hinaus in der EU ein Flottendurchschnittswert für den CO2-Ausstoß von 130 g/km [ED11]. Das Einhalten dieser Werte setzt eine hinreichend langfristige, umfassende und vernetzte Planung voraus, um Modelle mit hohem Ausstoß durch andere zu kompensieren. Darüber hinaus gelten teilweise in unterschiedlichen Ländern jeweils höhere oder niedrigere Werte, entsprechend Punkt 1. Die Planung kann an dieser Stelle demnach beliebig komplex werden, ist aber ohne Alternative wenn Strafzahlungen wegen Überschreiten der Werte vermieden werden sollen.

- **Zunahme der Komplexität an Fahrzeugteilen und -funktionen sowie deren Vernetzung:** Durch anspruchsvollere Fahrerassistenz-, Komfort- oder Diagnosesysteme steigt die Anzahl an Funktionen und Bauteilen [Hou03]. Dies bedeutet ein Umschwenken – zumindest vorerst in der Elektrik-/Elektronikentwicklung – von einer rein bauteilorientierten Sichtweise zu einer funktionsorientierten. In den traditionell sehr bauteilzentrierten Automobil-Konzernen zieht dies viel Überzeugungsarbeit nach sich [All07]. Auch mithilfe von Software-Produktlinien wird versucht, varianten-induzierte Komplexität zu reduzieren [HRRW12]. Aber auch der Umgang mit Anforderungen ändert sich, wenn eine andere Abstraktionsebene eingezogen wird. Die Funktionsorientierung betrifft demnach alle an dieser Stelle genannten Punkte.

- **Emanzipation der Konsumenten:** Sind es auf der einen Seite die Hersteller, die mit immer neuen Funktionen aufwarten, stehen auf der anderen Seite den Konsumenten eine wachsende Anzahl an Möglichkeiten zur Verfügung, den Markt detailliert zu erfassen und damit ihren Kauf genau abzuwägen. Durch verschiedene Online-Dienste können Kunden Fahrzeuge unterschiedlicher Hersteller im Detail vergleichen und das für sie passende entsprechend ihren Anforderungen heraussuchen. Über Foren, Online-Zeitschriften, Blogs

oder den Kurznachrichtendienst Twitter sind sie aktuell informiert über auftretende Probleme mit bestimmten Modellen bestimmter Hersteller oder die Service-Leistung in den Werkstätten. Kunden setzen damit zunehmend auf eigene Recherchen und glauben den Versprechen der Werbung immer weniger. Im Gegenzug hat das Marketing eines Unternehmens an dieser Stelle die Chance, zukünftige potentielle Kunden in den Gestaltungsprozess des Produkts mit einzubeziehen. Somit ist durch den Dialog mit dem Kunden auch eine weitere Quelle für Anforderungen geschaffen [Hün11].

- **Wachsende Bedeutung der Zulieferer:** OEMs vergeben ständig mehr Forschungs- und Entwicklungsarbeit sowie die Fertigung von Teilen und Komponenten in die Hände von Zulieferern. Dies bedeutet neben dem stark wachsenden Verwaltungsaufwand seitens der Auftraggeber eine Zunahme an Zahl und Bedeutung von Lasten- und Pflichtenheften [BLP04, Hou03]. Auch die Produktions- und Logistikprozesse werden komplexer, wenn Kaufteile just-in-time vom Zulieferer an die Fertigungsstraße befördert werden müssen.

- **Größe der Konzerne:** Die Volkswagen AG umfasst zu diesem Zeitpunkt (Februar 2013) 12 Marken und ist direkter Arbeitgeber für mehr als 500.000 Menschen [o.V11, VWW]. Die Planung und Kontrolle eines solch umfangreichen organisatorischen Konstrukts macht neue Methoden und Werkzeuge im Prozessmanagement notwendig, unterstützt durch eine moderne IT [MH11, HG06].

3.1.3 Spezielle Problemstellung bei der Volkswagen AG

Das Produktmanagement ist unter anderem verantwortlich für das Betreuen von Fahrzeugprojekten in einer frühen Phase, in der es hauptsächlich um die Zusammenstellung der Fahrzeugeigenschaften geht, das Definieren von produktbezogenen Prozessen und die Definition und Durchführung der Änderungssteuerung von Baukastenmodulen. Gleichzeitig werden die Prozesse und Strukturen zur operativen Umsetzung der Baukastenstrategie maßgeblich durch die Bereiche „Technik“, „Finanzen“ und „Vertrieb“ definiert und durch das Produktmanagement koordiniert zusammengeführt. Es finden sich hier somit viele der in Abschnitt 2.2 beschriebenen Schnittstellen des Anforderungsmanagements vereinigt, während gleichzeitig eine zentrale Rolle in der operativen Projektdurchführung sowie in der Langfristplanung eingenommen wird.

Das Projekt „durchgängiges Anforderungsmanagement für modulare Produktstrukturen“ hatte die Zielsetzung, eine Evaluation der aktuellen Aktivitäten des Bereichs im Kontext des Anforderungsmanagements durchzuführen. Die Fragestellung war, welche Prozesse und Aktivitäten in diese Domäne fallen und somit von aktuellen Erkenntnissen profitieren können oder es bereits tun. Auch sollten die zentralen vorhandenen Arbeitsergebnisse beschrieben und eine Einordnung in den Arbeitsprozess vorgenommen werden um prüfen zu können, ob moderne Werkzeuge für das Anforderungsmanagement eine Abbildung von diesen zulassen.

Eine spezielle Problemstellung ergibt sich aus dem Aufeinandertreffen von formalen und nicht formalen Artefakten zum Zeitpunkt der frühen Phase, welche auch als „Produktdefinition“ bezeichnet wird [BS11]. Entwicklungsprojekte sind zu diesem Zeitpunkt gekennzeichnet vom Zusammentragen von Ideen und Informationen, die zu Produktkonzepten und Designvorschlägen

führen. Aufgabe in der Produktdefinitionsphase ist es unter anderem, den sehr heterogenen Informationsfluss zu filtern, zu katalogisieren und zu strukturieren. So werden bspw. Eigenschaften eines Fahrzeugprojekts, die nicht formal vorliegen, beeinflusst durch stark formalisierte Informationen von Modulen, die sich in einem Baukasten befinden. Diese Art von Schnittstellen müssen manuell von Mitarbeitern verarbeitet werden, wenn bspw. der Einsatz bestimmter Module im Fahrzeugprojekt geprüft wird. Da die Anzahl Fahrzeuge, die Module aus den Fahrzeugbaukästen einsetzen, in den kommenden Jahren stark ansteigen werden, müssen die Prozesse entsprechend skalieren können.

3.2 Entwicklung von Szenarien für das Anforderungsmanagement

Für die Prozessanalyse musste eine Methodik gefunden werden, welche die Ergebnisse anschaulich darstellt und einen Eindruck vermittelt, wie die unterschiedlichen Einflussfaktoren die unter dem Oberbegriff „Anforderungsmanagement“ fallenden Prozesse und Arbeitsergebnisse verändern. Hierzu wurde das Mittel der Szenarioanalyse gewählt, welches eine beschriebene Methodik und Werkzeuge umfasst. Die geforderte Prozessanalyse wurde hierzu im Hinblick auf die Erstellung der Szenarien konzipiert und durch eine erprobte Vorgehensweise vorgenommen. Als Werkzeug wurden die erstellen Szenarien verwendet, um die Interpretation der Analyse mithilfe der Mitarbeiter von Volkswagen zu schärfen.

Ein Szenario stellt hierbei eine mögliche Ausprägung eines klar umrissenen Kontextes in der Zukunft dar, basierend auf der Entwicklung definierter Faktoren. Entscheidend ist, dass eine Szenarioanalyse nicht *die* Zukunft sondern, mehrere *mögliche* Zukünfte hervorbringt. Jede davon kann Wirklichkeit werden, sofern bestimmte Annahmen eintreten. Ein Szenario ist daher definiert durch diese Annahmen, den Kontext in dem es sich bewegt und den Zeitraum, der betrachtet wird. Der Vorteil der Präsentation der Ergebnisse einer Prozessanalyse in Szenarioform ist, dass erkennbar und nachvollziehbar wird, auf welche Art sich bestimmte Entscheidungen auswirken können. Dies macht die Diskussion hinsichtlich der Ergebnisse einfacher und greifbarer.

Eingesetzt wurde das Verfahren zur Szenarioanalyse nach Pillkahn, welches an dieser Stelle kurz wiedergegeben wird [Pil07]. Die Ursprünge der Szenarioanalyse liegen mit Kahn und Wiener bereits länger zurück und resultieren aus den strategischen Überlegungen des US-Militärs während des kalten Krieges [KW67]. Pillkahn erweitert die Methodik der Szenarioanalyse um wissenschaftlich geprägte Elemente. Sein Verfahren zeichnet sich aus durch ein strukturiertes Vorgehen, welches die Erstellung der Szenarien auf eine solide und nachvollziehbare Basis stellt. Der Vorteil liegt darin, das im Nachhinein durch Kritiker zwar das Ergebnis diskutiert, nicht aber der Erstellvorgang selbst in Frage gestellt werden kann. Abbildung 3.2 zeigt schematisch den durch ihn skizzierten Ablauf. Im Folgenden werden die einzelnen Phasen beschrieben.

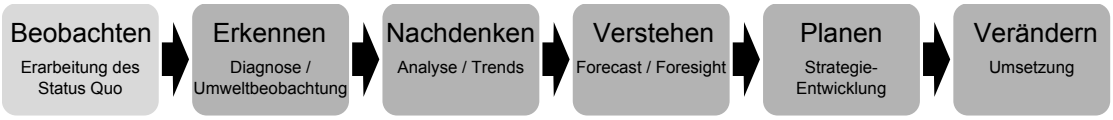


Abbildung 3.2: Phasen der Szenarioanalyse ([Pil07])

3.2.1 Phase 1: Beobachten

„Beobachten“ bedeutet das Zusammentragen der Informationen, die später als Grundlage der Szenarien dienen. Sie können aus Gesprächen, wissenschaftlichen Artikeln, Zeitungen oder auch formalen Interviews stammen. Zur Sicherung eines hohen Qualitätsstandards dieser wichtigen Arbeit wurde dieser Teil als „Knowledge Acquisition Project“ (KAC) gemäß der Arbeit von Milton definiert [Mil07]. Ziel ist das Erstellen einer sog. „Wissensbasis (engl. Knowledge Base)“, welche von Milton beschrieben wird als „eine spezielle Datenbank, die Informationen beinhaltet, welche die Expertise einer bestimmten Domäne repräsentieren“ [Mil07]. An dieser Stelle soll diese Definition folgendermaßen erweitert werden:

Definition 3.2.1 (Knowledge Base (basierend auf Milton [Mil07])) *Eine Knowledge Base ist ein System, welches relevante, strukturierte und miteinander verknüpfte Informationen einer oder mehrerer Domänen enthält und diese Nutzern und anderen Systemen zum Betrachten und Bearbeiten zur Verfügung stellt.*

Milton definiert 47 Schritte, die zu einer ausgeprägten Wissensbasis führen [Mil07]. Sie lassen sich grob gliedern in Projektplanung, initiales Erfassen und Modellieren, detailliertes Erfassen und Modellieren sowie das anschließende Verteilen von Wissen. Analog hierzu wurden in einem ersten Schritt unstrukturierte Gespräche durchgeführt, die einen ersten Überblick der gesamten Prozesssituation ermöglichten. Die Auswahl der Gesprächspartner geschah auf der Basis der Verantwortlichkeiten in den Prozessbeschreibungen sowie in bereits vorhandenem Material wie etwa Präsentationen über Projekte mit einem Bezug zum Anforderungsmanagement. Zusätzlich wurden Referenzen dieser Partner genutzt, um weitere Quellen zu erschließen.

Auf die unstrukturierten Gespräche folgten analog zum Vorgehen von Milton strukturierte Interviews. Hierzu wurden Fragen ausgearbeitet, die den ausgewählten Interviewpartnern im Vorfeld zugesandt wurden. Diese führten teilweise Korrekturen an den Inhalten durch oder verwiesen auf andere Personen. Die Interviews selbst wurden mit einem digitalen Diktiergerät aufgenommen, um die Transkription zu erleichtern. Die transkribierten Interviews wurden den Teilnehmern jeweils nach Durchführung zugesandt, um ihnen die Möglichkeit zur Korrektur zu geben. Danach wurde die jeweilige Tonaufnahme vernichtet. Auf diese Weise kamen insgesamt 22 Interviews zustande, die alle etwa eine Länge von 1,5 Stunden hatten und in der Abschrift zwischen drei und acht Seiten umfassen. Zur Transkription selbst ist anzumerken, dass keine wortwörtliche Abschrift des Gesagten – inklusive Versprecher, unvollständige Sätze, etc. – erfolgte, sondern eine Zusammenfassung der Antworten auf die jeweiligen Fragen.

Mit Hilfe der Interviews war es möglich, eine große Menge an Informationen zu katalogisieren. Vor allem Dokumente und Systeme wurden erfasst und auf einer Art „Prozess-Landkarte“ eingeordnet, um Zusammenhänge darzustellen, bzw. die Verwendungen der einzelnen Elemente zu verdeutlichen. Als Grundlage der Notation der Landkarte wurden die graphische Darstellung des Produktentstehungsprozess verwendet.

3.2.2 Phase 2: Erkennen

Die gesammelte Menge an Daten muss in einem nächsten Schritt strukturiert werden. Dies geschieht zum einen, um die Vollständigkeit der Informationen bewerten zu können. Zum anderen bilden die Ergebnisse die Voraussetzung für die nächsten Schritte.

Erkennen bedeutet, wichtige Informationen zu strukturieren und von unwichtigen zu trennen. Vorher stellte sich allerdings die Frage, welcher Art die von Milton empfohlene Knowledge Base sein sollte. An dieser Stelle wurden diverse Werkzeuge vorgeschlagen, etwa *semantische Wikis* [Kar, Lei]. Ein Wiki ist ein Sammlung von interaktiven Webseiten, die durch den Benutzer durch Text oder Medieninhalte ergänzt und untereinander verknüpft werden können. Auf diese Weise können schnell große Informationsmengen zusammengestellt werden, da die kollaborative Arbeit an verschiedenen Themen stark vereinfacht wird. Semantische Wikis verknüpfen dieses Prinzip mit einer Ontologie als Grundlage, einzelne Seiten entsprechen Instanzen von Klassen, Verknüpfungen können typisiert werden. Es wurde jedoch schnell ersichtlich, dass verfügbare semantische Wikis in der Bedienung zu umständlich sind – ohne eine recht umfangreiche Einführung sind die genannten Werkzeuge schwer zu handhaben. Darüber hinaus sind sie unübersichtlich hinsichtlich der enthaltenen Informationen und ihrer Organisation. Dies kann zu einem unstrukturierten Wachstum der Inhalte führen, was kontraproduktiv bei der Sammlung von Prozesswissen ist. Es wurde demnach auf Microsoft Office Werkzeuge zurückgegriffen, da die Inhalte den weiteren beteiligten Mitarbeitern einfach zur Verfügung gestellt werden konnten und die Bearbeitung allgemein bekannt ist.

Die Transkriptionen der Interviews lagen bereits in Textform vor, aus ihnen wurden subjektiv als bedeutend empfundene Aussagen extrahiert und in einer Tabelle abgelegt. Diese hatte nach Abschluss der Arbeit 173 Einträge. Danach wurde in Absprache mit Mitarbeitern von Volkswagen eine Priorisierung der Aussagen vorgenommen. Zusätzlich ist in einem weiteren Tabellenblatt ein Glossar erstellt worden, das die bereits identifizierten Artefakte sowie Abkürzungen erläutert.

3.2.3 Phase 3: Nachdenken

Die Grundlage von Szenarien sind sog. Zukunftselemente. Innerhalb des Schritts „Nachdenken“ geht es um die Ausarbeitung dieser konkreten Informationen, um Aussagen zueinander in Kontext zu setzen und zusätzlich ihre Bedeutung für die späteren Szenarien festlegen zu können. 33 der von den Mitarbeitern am höchsten priorisierten Aussagen sowie weitere Informationen aus Gesprächen, wissenschaftlichen Artikel und Medien wurden zu insgesamt 57 Zukunftselementen verarbeitet. Diese klar abgegrenzten Informationseinheiten unterliegen einer Struktur, welche wiederum von Pillkahn übernommen wurde und folgendermaßen aufgebaut ist [Pil07]:

- **Name:** Ein eindeutiger und aussagekräftiger Name des Elements.
- **Kontext:** Charakterisierung des Elements (Paradigma/Konstante, Trend, Widerspruch, Unsicherheit, Chaos/Wildcard) und die Einordnung in eine an STEEP (Social, Technological, Economic, Environmental/Ecological, Political) angelehnte Kategorisierung: Technologische Dimension, Soziokulturelle/Prozessuale Dimension, Politische/Rechtliche Dimension, Kunden-Dimension, Ökonomische Dimension, Strategische Dimension.

- **Beschreibung:** Ein einleitender Text, der Inhalt und Bedeutung des Elements wieder gibt.
- **Umgebung:** Eine graphische Darstellung des Elements in Bezug auf andere Elemente. Diese Einordnung unterstützt den Text in der *Beschreibung* und umgekehrt, da so Einflüsse aufgezeigt werden können. Hilfreich bei der Einordnung der einzelnen Zukunftselemente in einen gemeinsamen Kontext waren Mindmaps.
- **Treiber/Inhibitoren:** Beschreibt, welche Punkte für das Eintreffen des in der **Projektion** erwarteten weiteren Verlaufs arbeiten und welche dagegen.
- **Signale:** Welche Signale führten zu diesem Zukunftselement, z.B. die Interviews oder Artikel aus Fachzeitschriften?
- **Ausdruck:** Wie findet sich das Zukunftselement aktuell wieder, wie drückt sich das Vorhandensein aus?
- **Projektion:** Eine Einschätzung des zukünftigen Verlaufs des Elements.
- **Trigger:** Eine Liste von Artefakten, Ereignissen oder Signalen, die beobachtet werden müssen, um über eine Änderung in der Entwicklung des Elements informiert zu werden.

Die Charakterisierung hinsichtlich der Dynamik des Elements (Paradigma/Konstante, Trend, Widerspruch, Unsicherheit, Chaos/Wildcard) ist ein wichtiger Schritt zur Konzeption der Szenarien. Nach umfassender Ausarbeitung aller Zukunftselemente wurde ein Volatilitäts-/Einfluss-Diagramm erstellt und die Elemente entsprechend eingeordnet. Der obere rechte Quadrant (mittelstarker Einfluss / mittelstarke Volatilität bis sehr starker Einfluss / sehr starke Volatilität) konnte dann zur Konstruktion der eigentlichen Szenarien genutzt werden.

Die einzelnen Zukunftselemente wurden wieder mit den Projektpartnern bei Volkswagen diskutiert, um zu einer korrekten Einschätzung hinsichtlich ihrer Vollständigkeit und Bedeutung zu kommen. Nur wenn die Szenarien auf einer korrekten Ausgangsmenge an Wissen aufbauen, können sie ihrerseits wieder korrekt sein.

3.2.4 Phase 4: Verstehen

22 Interviews mit 56 extrahierten Zukunftselementen führten zu insgesamt vier Szenarien. Als Grundlage dienten die Zukunftselemente des oberen rechten Quadranten, für die jeweils vier mögliche Ausprägungen in Zusammenarbeit mit VW-Mitarbeitern erstellt wurden. Die Kombination dieser Ausprägungen ergaben dann entsprechend vier Szenariorahmen, die das jeweilige Szenario grob beschreiben. Folgende Szenarien wurden auf diese Art entwickelt:

1. German Systems Engineering: Die Kombination an Ausprägungen, die subjektiv empfunden den besten Fall darstellen. Die Komplexität wird durch verschiedene Maßnahmen beherrschbar, Anforderungsmanagement wird als wichtiger Faktor hierbei empfunden, IT-Technologien bieten ein breites Spektrum an Softwarelösungen an.

2. Complexity Burnout: In diesem Szenario wird von einer stark zunehmenden Komplexität ausgegangen, welche die administrativen Prozesse sehr stark belastet. Es existiert keine umfassende Lösung für das Anforderungsmanagement. Hierdurch können keine Synergie-Effekte aus den modularen Baukästen realisiert werden, da für diese keine strategische Planung stattfinden kann.

3. Continuous Integration: Das Szenario zeigt, wie mit der Einführung von Prozessen und Werkzeugen bereits mit vorhandenen Methoden den Herausforderungen begegnet werden kann. Es unterscheidet sich dabei von der idealen Welt des ersten Szenarios, indem keine radikale Wende vollzogen wird, sondern dass bereits durch gezielte Maßnahmen finanziell wirksame Erfolge erzielt werden können.

4. Last Minute Turnaround: Beschreibung einer Turnaround-Situation durch den Einsatz hoher finanzieller Ressourcen zur Vermeidung des „Complexity Burnouts“.

Die Szenarien beschreiben gezielt die Entwicklung des untersuchten Prozessabschnitts der frühen Phase im Produktentstehungsprozess, in dem das Fahrzeug in größten Teilen nur virtuell existiert und eine Zusammenstellung aus Baukästen, Plattformen, Vorgängern und neuen Teilen erfolgt. Die Ausarbeitung der Szenarien fand wiederum in einem Textverarbeitungsprogramm statt, da sie zum größten Teil aus Fließtext bestehen und die Distribution einfach möglich sein sollte. Sie sind das Projektergebnis Nr. 1.

3.2.5 Phase 5: Planen

Zur Planung ist im Vorfeld die Kenntnis des aktuellen Zustands notwendig, um daraus Handlungsempfehlungen ableiten zu können. Aus einem prozessualem Blickwinkel ist durch die Erarbeitung der Szenarien dieser Voraussetzung genüge getan. Bisher nicht betrachtet wurden jedoch technologische sowie strukturelle Fragen der Domäne. Um erstere behandeln zu können wurde ein Tool-Scouting zur Erhebung der Verfügbarkeit kommerzieller Lösungen für ein durchgängiges Anforderungsmanagement durchgeführt. Dieses stellt das in Abschnitt 3.1 benannte Projektergebnis 2 dar und wird gesondert in Abschnitt 3.3 behandelt.

Die Untersuchung der Strukturen der Domäne, also die Verknüpfungen von Prozessen und den darin enthaltenen Elementen, basiert auf der im Abschnitt 3.2.1 eingeführten Prozesslandkarte, welche keinen formalen Charakter hat. An dieser Stelle wurde das „Artefakt“ als Metabegriff für alle Objekte innerhalb eines Unternehmens wie bspw. Prozesse, Personen, Strukturen, etc. geprägt. Eine detaillierte Einführung und Definition in Artefakte für die weitere Verwendung in dieser Arbeit bietet Kapitel 4.

Mit Hilfe des Werkzeugs Contour der Firma Jama [Jam] wurde ein durchgängiges Metamodell der im Rahmen des Projekts erarbeiteten Prozesselemente erstellt. Hierbei wurden Anforderungen mit verschiedenen, sich im produktiven Einsatz befindlichen Elementen aus Entwicklungsartefakten – Dokumente oder Module aus Baukästen – verknüpft. Es können somit Analysen bspw. hinsichtlich der Auswirkungen der Änderungen von Anforderungen auf diese Objekte durchgeführt werden. Dies zeigt die grundlegende Operationalisierbarkeit der vorgeschlagenen Herangehensweise. Der Fokus lag hierbei auf den mit UML-Klassendiagrammen beschriebenen Dokumenten, insbesondere auf dem sog. Eigenschaftskatalog, welcher eine zentrale Rolle in

der Produktentwicklung bei Volkswagen einnimmt [Rin06]. In ihm werden die grundsätzlichen Charakteristiken von Fahrzeugen definiert und beschrieben. Aufgebaut ist er mit Hilfe einer Eigenschaftsstruktur, welche wiederum in Contour als Vorlage abgebildet werden konnte. Der Prototyp verknüpft demnach innerhalb eines Fahrzeugprojekts Eigenschaften – welche zeitlich gesehen ein sehr frühes Artefakt im Entwicklungsprozess sind – mit daraus resultierenden Anforderungen. Diese können wiederum mit Modulen aus einem Modulbaukasten oder mit neuen Entwicklungsumfängen verbunden werden.

Der Prototyp stellt das Projektergebnis Nr. 4 dar und demonstriert anschaulich die Möglichkeiten von Traceability: Werden Eigenschaften geändert, ist es sehr einfach möglich festzustellen, auf welche Teile sich Auswirkungen ergeben. Gleichzeitig lassen sich Analysen über den Verwendungsgrad von Baukasten-Modulen erstellen oder der Realisierungsgrad der Anforderungen und letztendlich der Eigenschaften in Berichten automatisch aufbereiten. Ergebnisse dieses Schritts sind somit neben der Erarbeitung einer Liste von Anforderungsmanagementwerkzeugen ein Metamodell, welches in einem Werkzeug prototypisch umgesetzt wurde.

3.2.6 Phase 6: Verändern

Innerhalb dieser Phase wird durch die gewonnenen Erkenntnisse aus den Szenarien versucht, durch Beeinflussung verschiedener Elemente im Unternehmen auf den Eintritt eines angestrebten Szenarios hinzuarbeiten. Die Projektergebnisse 1, 2 und 4 bieten in Kombination mit den Handlungsempfehlungen die Möglichkeit, in einem Pilotprojekt die theoretisch konzipierten Vorgehensweisen und Prototypen zu operationalisieren. Ein Projektstrukturplan stellt das Projektergebnis Nr. 5 dar und beschreibt das weitere Vorgehen. Er enthält folgende Kernpunkte:

- Erarbeitung eines umfassenden Metamodells sowie basierend darauf die Entwicklung diverser konkreter Modelle zur umfassenden formalen Beschreibung aller relevanten Prozesselemente im Kontext des Produktentstehungsprozesses. Diese werden als Artefakte betrachtet und verknüpft, wie in Kapitel 4 beschrieben.
- Einführung einer Methodik zur Erfassung und Pflege der Modelle.
- Einsatz eines Werkzeugs zur operativen Arbeit mit den Modellen.

Im weiteren Verlauf dieser Arbeit werden für diese Punkte die Grundlagen geschaffen. So wird in Kapitel 4 Artefakt als Oberbegriff für verschiedene Elemente in Unternehmen und dessen Prozessen eingeführt. Kapitel 5 zeigt, welche Vor- und Nachteile die Einführung eines solchen durchgängigen Anforderungsmanagements hat und definiert Anforderungen an ein Werkzeug. Kapitel 6 definiert dann die Methodik und demonstriert anhand eines prototypischen Werkzeugs die Operationalisierbarkeit.

3.3 Scouting von Anforderungsmanagementwerkzeugen

Zur Abschätzung des Stands der Technik im Bereich Anforderungsmanagemementwerkzeuge wurde ein Tool-Scouting durchgeführt. Dieses diente ebenfalls als Grundlage für die im Kapitel 5

ausgeführten Anforderungen an ein weitergehendes Anforderungsmanagement. Die Anzahl an verfügbaren Werkzeugen ist hoch, allerdings werden durch die Größe, Branche und Komplexität der Volkswagen AG spezielle Anforderungen an Werkzeuge gestellt. Daher war es nötig, die gefundenen Werkzeuge miteinander zu vergleichen um die Ergebnismenge auf einige wenige Potentialkandidaten zu reduzieren.

3.3.1 Methodik zur Auswahl von Potentialkandidaten

Das Vorgehen basierte auf der Erstellung einer Grundmenge an Werkzeugen und deren Hersteller. Herangezogen wurden hierzu sowie zur Festlegung der Methodik zwei Studien sowie ein Fachartikel [HMRV07, SG08, Sch10b]. Darüber hinaus kam generell die Internetsuche zum Einsatz. Auf diese Weise wurden insgesamt 39 Werkzeuge ermittelt, die Auflistung findet sich in Anhang C auf Seite 189. Die Menge wurde dann reduziert auf der Grundlage folgender Kriterien:

1. Oftmals war bereits anhand der Web-Auftritte erkennbar, dass der jeweilige Hersteller nicht für einen Einsatz in der Großindustrie geeignet ist: Aktualität der Inhalte, Aufmachung, Angaben von Referenzkunden, etc. spielten bereits eine wichtige Rolle beim ersten groben Filtern. Der Fokus wurde auf größere Unternehmen gelegt.

2. Einige Werkzeuge sind ausschließlich für die Entwicklung von Softwareprodukten verwendbar. Da der Fokus dort auf der Integration mit Software-Engineering-Werkzeugen und -Vorgehensmodellen liegt, wurden sie aussortiert, da der Aufwand zur Anpassung zu groß scheint.

3. Reine SaaS-Lösungen wurden im Vorfeld aussortiert, da sie mit den strengen Sicherheits-Richtlinien der Volkswagen AG nicht kompatibel sind.

3.3.2 Fragebogen zur detaillierten Evaluation für das Anforderungsmanagement bei Volkswagen

Nach der ersten Reduktion auf insgesamt 24 Werkzeuge war es notwendig, detailliertere Informationen über die jeweiligen Werkzeuge einzuholen. Nach Auswertung der Hersteller-Webseiten war ersichtlich, dass die dort vorhandenen Angaben entweder zu unvollständig oder nicht vergleichbar waren. Aus diesem Grund wurde eine Evaluation mittels Fragebogen durchgeführt. Dieser enthielt Fragen, die einerseits aus den im Projekt erarbeiteten Anforderungen abgeleitet wurden sowie weitere zur allgemeinen Information wie etwa Kosten oder Schulungsmöglichkeiten. Die Entscheidung fiel gegen einen reinen Multiple-Choice-Fragebogen, da möglichst viele Informationen über den Stand der Technik eingeholt und nicht ein einzelnes Werkzeug herausgearbeitet werden sollte. Zwar beeinträchtigt Freitext die Vergleichbarkeit, den Herstellern wird es jedoch ermöglicht, ihre Konzepte darzulegen und zu erklären. Der Fragebogen wurde mit den Mitarbeitern der Marke Volkswagen abgestimmt und dann an die Hersteller entsprechend in einer deutschen oder englischen Version verschickt. Er kann im Anhang D auf Seite 191 eingesehen werden.

Insgesamt konnten 18 ausgefüllte Fragebögen für eine Endauswertung verwendet werden. Um eine Vergleichbarkeit der Freitextantworten herzustellen, wurde eine Verschlagwortung der

jeweiligen Antworten durchgeführt. Hierzu sind die Konzepte in einer Antwort einem Begriff zugeordnet worden. Bei der Frage nach der Strukturierung von Anforderungen ergaben sich somit bspw. die Begriffe „Baumstruktur“ oder „Graph“, bei den unterstützten Import-/Export-Formaten wiederum die Begriffe „XML“, „Excel“ oder „HTML“.

Für die letzte Auswahlstufe wurden die Werkzeuge hinsichtlich der Kriterien Skalierbarkeit, Sicherheit und Schnittstellen betrachtet. Diese für den industriellen Bereich grundlegend wichtigen Eigenschaften müssen als „Hygienefaktoren“ durch die Werkzeuge mindestens erfüllt werden. Nur sieben Werkzeuge sind diesbezüglich ausreichend ausgestattet. Es fiel auf, dass vor allem bei der Verschlüsselung der gespeicherten Daten Schwachpunkte liegen. Folgende sieben Potentialkandidaten wurden identifiziert:

- Contour von Jama Software
- Reqtify von Dassault
- Rational Doors von IBM
- inteGREAT von eDev Technologies
- Rational Requirements Composer von IBM
- Teamcenter von Siemens
- TopLogic von Business Operation Systems

Über diese Produkte wurden erweiterte Informationen eingeholt, welche in aufbereiteter Form dann vor ausgesuchten Bereichen der Volkswagen AG präsentiert wurden.

3.3.3 Beurteilung traditioneller Anforderungsmanagementwerkzeuge

Die evaluierten Werkzeuge stellen bereits Möglichkeiten zur Modellierung zur Verfügung, welche sich allerdings oftmals auf einem niedrigen Niveau bewegen. Alle Werkzeuge verwenden eigene Oberflächen, Datenstrukturen und Sprachen, um ihre spezifischen lokalen Modelle abzubilden. Hierdurch werden „Insel-Lösungen“ erzeugt, die alle Modelle wiederum selber erfassen müssen, um die Traceability darauf auszudehnen. Es entstehen Inkonsistenzen, da diese Modelle nicht in jedem System gepflegt werden. Somit steht keinem System eine genügend große und aktuelle Datenbasis zur Verfügung, um geänderte oder neue Anforderungen mit relevanten Prozesselementen zu verknüpfen und diese so mit der Komplexität in den Produkten und Prozessen in Verbindung zu bringen. Weiterhin sind Verknüpfungen selten typisiert und die Möglichkeit der Einbindung konkreter Prozessinformationen ist nicht erkennbar. Außerdem fehlt ein Schema zur einheitlichen Beschreibung von Prozesselementen. Es hat sich im Rahmen des durchgeführten Projekts gezeigt, dass agile Werkzeuge benötigt werden, um Komplexität und Evolution im Rahmen des Anforderungsmanagements abbilden zu können, woraus sich umfangreiche Anforderungen an zukünftige Werkzeuge ergeben.

Publiziert im Vorfeld wurde bereits, auf welche Weise das Anforderungsmanagement mit Komplexitätskosten in Verbindung gebracht werden kann [GJRA12]. Grundsätzlich führen neue

oder veränderte Anforderungen indirekt oder direkt zur Ausführung bestimmter Prozesse. Im Nachhinein können bspw. zusätzliche Varianten die administrativen Kosten soweit erhöhen, dass eventuell errechnete Einsparpotentiale durch steigende Gemeinkosten überkompensiert werden können. Umgekehrt können bspw. auch angenommene Kosten für eine neue Anforderung aufgrund der Zunahme der Komplexitätskosten überschritten werden.

Es ist für das Anforderungsmanagement daher essentiell, eine möglichst weitreichende Analyse der Auswirkungen bspw. einer Anforderungsänderung zur Verfügung zu stellen. Für einen Entscheider muss erkennbar sein, auf welche Artefakte in seinem Unternehmen sich eine Änderung auswirkt. Die traditionellen Werkzeuge verfügen über eine unzureichende Datenbasis, um diese Informationen umfassend zur Verfügung zu stellen. Eine *„unzureichende Datenbasis“* meint an dieser Stelle demnach nicht nur die fehlende Digitalisierung aller Produktdaten, sondern eben auch die formale Verfügbarkeit von Prozess- und Artefaktinformationen, die durch das Produkt und seine Komponenten berührt werden. Eine vollständige formale Abbildung aller Prozesse und Aktivitäten ist sicherlich zu weitreichend, aber dennoch soweit als möglich anzustreben. Zukünftig kann Software auf diese Weise in die Lage versetzt werden, nicht nur Informationen zur Verfügung zu stellen, sondern diese auch zu interpretieren, um selbsttätig Komplexitätsveränderungen einzuschätzen. Darüber hinaus ist es notwendig, die Artefaktmodelle in einer geeigneten Weise allen Werkzeugen zur Verfügung zu stellen und sie nicht nur in einem zu entwickeln und ausschließlich dort zu verwenden. Nur auf diese Weise können die durch Struktur- und Schnittstelleninkompatibilitäten generierten Mehraufwände minimiert werden. Aktuell stehen hierfür allerdings keine geeigneten Werkzeuge zur Verfügung, wodurch eine Grundvoraussetzung für ein zielführendes Anforderungsmanagement heute nicht gegeben ist.

Die Feststellung der Veränderung der Komplexität von administrativen Prozessen im Nachhinein ist schwierig. Vom Prinzip her gelingt sie allerdings ähnlich wie die Abschätzung im Vorfeld. Administrative Prozesse bestehen oftmals aus Arbeiten, die für mehrere Produkte, Zusammenbauten oder Teile zusammengefasst ausgeführt werden, z.B. das Erstellen von Berichten oder Einkaufsvorgänge. Es lässt sich dann nicht mehr klar sagen, welcher exakte Zeitaufwand einem bestimmten Kostenträger oder noch genauer einer bestimmten Änderung einer Anforderung an einen Kostenträger zugeordnet werden kann. Eine vollständige Beschreibung der Prozesslandschaft ermöglicht es jedoch auch in diesen Fällen, im Voraus automatisiert zumindest festzustellen, welche Prozesse betroffen sein werden und teilweise eine Schätzung hinsichtlich der Kosten abgeben zu können. Die Datenbasis muss demnach die in Kapitel 4 eingeführten Artefakte und ihre Zusammenhänge untereinander enthalten. Je umfassender die Beschreibung ist, desto kleiner ist auch die Granularität, mit der ein eventuelles Werkzeug die Abschätzung vornehmen kann, und desto genauer sind auch die Resultate.

Hier ergibt sich demnach ein erweiterter Nutzen für formale Prozessbeschreibungen. Sehen sich diese stark mit dem Stigma des „puren Selbstzwecks“ konfrontiert [BWW10] und werden höchstens zu Prozessplanungs- und -optimierungszwecken verwendet, so entsteht mit ihrer Verwendung als Datenbasis zur Abschätzung der Auswirkungen von Änderungen in Produkten ein völlig neuer Aufgabenbereich. Da die Datenbasis viel mehr als nur reine Daten enthält, sondern ebenso Auskunft über Zusammenhänge und Strukturen von Artefakten geben kann, sollte an dieser Stelle eher von einer **Wissens**basis als von einer **Daten**basis gesprochen werden.

Broßmann und Mödinger untersuchen sehr detailliert die Herkunft und die unterschiedlichen

Bedeutungen des Begriffs „Wissen“ [BM11]. Sie benennen hierbei fünf unterschiedliche Belegungen, von denen für diese Arbeit der dritte relevant ist, da er eine Perspektive der Informatik repräsentiert. Es wird hierbei eine Unterscheidung anhand einzelner Ebenen „Daten“, „Informationen“ und „Wissen“ vorgenommen. Broßmann und Mödinger zitieren hierfür insbesondere Rehäuser und Krcmar, welche den Inhalt dieser Ebenen klar festlegen sowie als vierte Ebene auch das „Zeichen“ anführen [RK96]. Abbildung 3.3 stellt dieses Ebenenmodell dar und zeigt auch die Beziehungen zwischen den Ebenen.

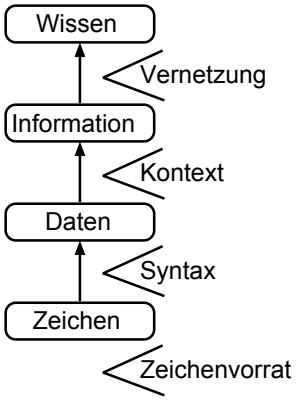


Abbildung 3.3: Ebenenmodell Wissen, Informationen, Daten, Zeichen nach [RK96]

Wichtig ist hierdurch zunächst der Schritt von „Daten“ zu „Informationen“. Er findet durch das Hinzufügen von Kontextinformationen statt: Die reine Zeichenkette „Golf“ wird erst durch das Hinzufügen bspw. einer Spaltenüberschrift in einer Datenbank zu der Information, dass es sich um das Fahrzeug „Golf“ handelt.

Der Schritt hin von „Informationen“ hin zum „Wissen“ erfolgt dann durch die Vernetzung von einzelnen Informationen. So kann die obige Information organisiert werden in die Beziehungen „Golf ist ein Automobil“ und „Automobil ist ein Fahrzeug“. Die Schlussfolgerung „Golf ist ein Fahrzeug“ ist dann eine implizite Form des Wissens, während „Golf ist ein Automobil“ eine explizite Form darstellt. Aufgrund der nicht unbedingt völlig trennscharfen Unterscheidung zwischen Information und Wissen findet sich aufbauend auf dem rein hierarchischen Modell bei Bodendorf das Konzept zweier „Pole“ von Daten und Wissen, zwischen denen sich Informationen einordnen lassen [Bod03].

Die Wissensbasis entspricht der von Weske beschriebenen „Hub-and-Spoke Integration“ von Enterprise-Systemen [Wes07]. Hierbei existiert ein zentrales System, welches den Informations- und Anweisungsaustausch zwischen verschiedenen Systemen vornimmt. Zur Implementierung einer Wissensbasis ist ein solcher zentraler „Hub“ notwendig, der allerdings nicht notwendigerweise eine aktive Rolle einnehmen muss, sondern etwa nur das Modell der Prozesslandschaft in aktueller Form zur Verfügung stellt. Die heutigen Werkzeuge im Bereich des Anforderungsmanagements verfügen zwar teilweise über formale Informationen im Sinne einer Wissensbasis, fokussieren allerdings ausschließlich die Beschreibung des zu entwickelnden Produkts. Sie können daher als Werkzeuge im *traditionellen* Sinn betrachtet werden, im Gegensatz zu Werkzeugen im *erweiterten* Sinn, die Prozesse und die darin enthaltenen Elemente mit in einer Wissensbasis

einbeziehen und diese gleichzeitig anderen zur Verfügung stellen.

3.4 Modellbasierter Prozesseditor zur Formalisierung von Prozessinformationen

Im vorherigen Abschnitt 3.3.3 wurde auf die Notwendigkeit der Verfügbarkeit formaler Informationen über Prozesse und deren Ergebnisse und Zusammenhänge hingewiesen. Zur Erprobung der Durchführbarkeit einer Formalisierung mit dem Ziel der Abbildung der Ergebnisse in Software, wurde als ein Projektergebnis ein Prototyp sowie eine Methodik zur Formalisierung von Prozessinformationen entworfen. Abschnitt 3.2.1 weist bereits darauf hin, dass eine große Anzahl an Prozesselementen während der Evaluation des Status des Anforderungsmanagements im Produktentstehungsprozess der Marke Volkswagen katalogisiert worden ist. Dieser besitzt eine graphische Repräsentation, welche die unterschiedlichen Meilensteine eines Fahrzeugentstehungsprojekts in den Vordergrund stellt. In diesen sog. Meilensteinplänen finden sich somit unterschiedliche Ergebnisse, die in einen zeitlichen Kontext gesetzt werden. Unterschiedliche Formen und Farben eines Meilensteins verleihen ihm unterschiedliche Bedeutungen. Abbildung 3.4 zeigt einen fiktiven Meilensteinplan innerhalb der Weboberfläche

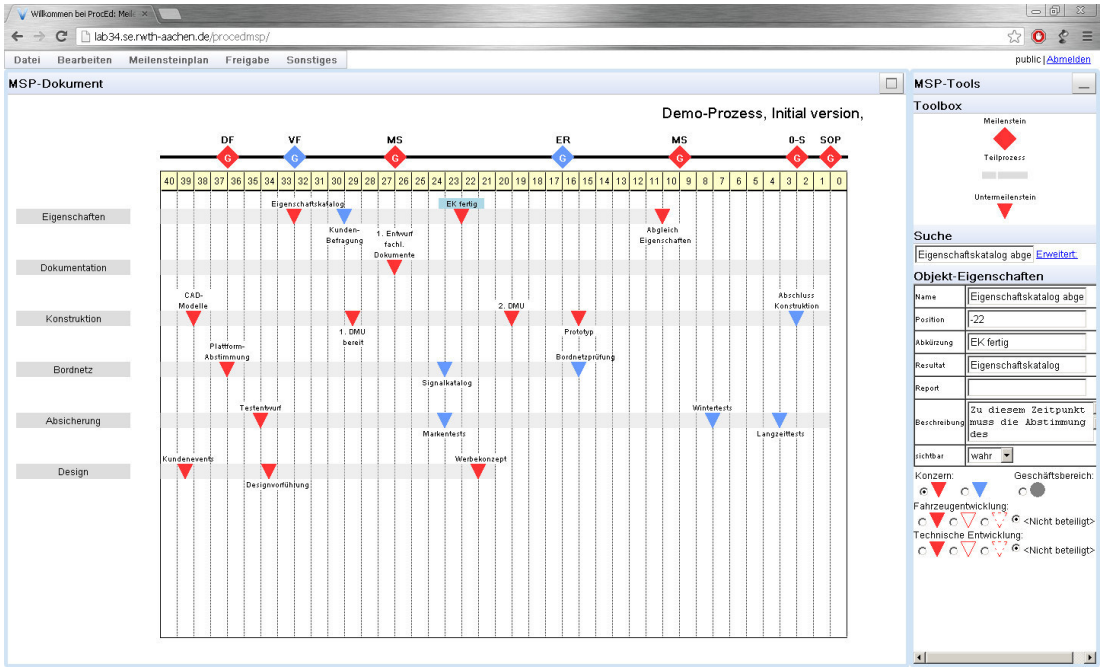


Abbildung 3.4: Ein fiktiver Meilensteinplan im webbasierten Prozesseditor ProcEd

Diese spezielle graphische Repräsentation wird derzeit noch in Microsoft PowerPoint aufbereitet und benötigt umfangreiche Informationen über den dargestellten Prozess und darin vorkommenden Objekte. Der Erstellungsprozess des Dokuments ist geprägt von hohem Aufwand vor allem bei Änderungen in den zeitlichen Informationen des PEPs, da die Objekte fast sämtlich

untereinander in Beziehung stehen. Aufgrund der Spezifizität der Darstellung und der Objekte kann kein herkömmliches Werkzeug zur Visualisierung von Prozessinformationen genutzt werden, woraus großer manueller Aufwand in der Pflege resultiert. Auch ist es derzeit nicht möglich, andere Darstellungsformen der enthaltenen Informationen zeitnah zur Verfügung zu stellen. So ist ein einfacher Auszug einer Übersichtsliste aller Meilensteine mit zugehörigen Terminen nicht möglich.

Das Projekt „modellbasierter Prozesseditor“ hatte das Forschungsziel, eine textuelle Notation für die im Rahmen des übergeordneten Projekts mit der Marke Volkswagen kartografierten Prozesselemente zu finden, in der eine Verknüpfung mit den Meilensteinen des PEPs erfolgen konnte. Losgelöst davon war eine webbasierte Editoroberfläche zu entwickeln, welche die formale Notation interpretieren und in spezifischen graphischen Darstellungen umsetzen kann. In einem ersten Schritt wurde hier das bei Volkswagen verwendete Format eines Meilensteinplans in sog. Schwimmbahnen abgebildet. Es findet somit eine Trennung zwischen formalen Modell und visueller Repräsentation statt, wodurch beliebige Darstellungen entwickelt werden können, die sich alle der gleichen Informationen bedienen. Die Bearbeitung der formalen Informationen an der Oberfläche sollte intuitiv möglich sein und wenig Berührungspunkte zwischen Benutzer und formalem Modell beinhalten.

Der in Kurzform „ProcEd“ genannte Editor ist somit ein erster Schritt hin zu einer Formalisierung von unternehmensspezifischen Prozessinformationen. Hierzu wurde in Zusammenarbeit mit den Domänenexperten eine domänenspezifische Sprache unter dem Namen „ProcDSL“ erstellt, welche in der Lage ist, die speziellen Informationen des Produktentstehungsprozesses zu repräsentieren ohne eine konkrete graphische Darstellung festzulegen. Sie entstand mithilfe der Software MontiCore, einem Werkzeug für die Entwicklung von textuellen domänenspezifischen Sprachen des Instituts für Software Engineering an der RWTH Aachen [KRV10]. Basierend auf einer gegebenen Sprachdefinition generiert es einen Lexer, einen Parser sowie Klassen für einen abstrakten Syntaxgraph. Eine Diskussion der Sprache findet sich in [BGR09]. Repräsentiert werden können Prozesselemente, die einen Meilensteinplan hauptsächlich ausmachen, folglich Meilensteine und deren Verbindungen untereinander. Die grapische Darstellungsform legt den Fokus neben der zeitlichen Einordnung eines Meilensteins vor allem auf Verantwortlichkeiten. Daher sind Informationen wie „Unternehmensbereich“ oder „organisatorische Einheit“ ebenfalls enthalten. Diese Elemente stehen in einer bestimmten Verantwortung zu den jeweiligen Meilensteinen. Konkret können dies „verantwortlich“, „mitwirkend“ oder „Kenntnis nehmend“ sein, wodurch sich die graphischen Repräsentation eines Meilensteins in einem speziellen Kontext in einem Meilensteinplan ändert. Abbildung 3.5 zeigt dies schematisch: In der Darstellung existieren zwei Organisationsbereiche, die unterschiedlich an diesem Meilenstein beteiligt sind. Während Organisationsbereich 1 für die Fertigstellung der zu diesem Meilenstein geforderten Ergebnisse verantwortlich ist, wirkt Organisationsbereich 2 bei deren Erstellung mit. In der Sprache wurden diese typisierten Relationen durch Relationsklassen abgebildet, welche später in der Software interpretiert werden konnten. Typisierte Relationen mussten darüber hinaus an verschiedenen anderen Stellen in die Sprache implementiert werden, um den vollen Informationsgehalt der ursprünglichen graphischen Darstellungen textuell formal abbilden zu können.

Zur Entwicklung der formalen Repräsentation der in den Plänen enthaltenen Informationen musste ein umfassendes Verständnis der Zusammenhänge der Elemente bei Domänenexperten

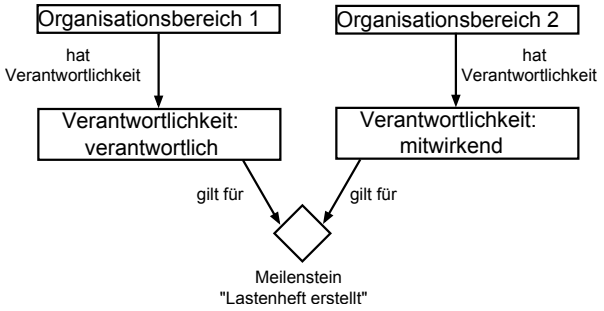


Abbildung 3.5: Konzeptionelle Darstellung des Zugriffs zweier unterschiedlicher Unternehmensbereiche über Verantwortlichkeiten auf einen Meilenstein

und Entwickler geschaffen werden. Hierzu wurden, basierend auf den in den Interviews mit den Domänenexperten im Rahmen des übergeordneten Projekts gewonnenen Erkenntnisse, zu spezifischen Prozessobjekten wiederum tiefergehende Interviews durchgeführt. Als Mittel zur Vereinfachung der Diskussion und der Dokumentation kamen UML-Klassendiagramme zum Einsatz, allerdings nicht mit ihrem vollen Funktionsumfang. Hintergrund sind folgende zwei Problempunkte:

1. Die Konzepte „Ableitung“ und „Vererbung“ waren den Experten nicht immer deutlich zu machen. Basierend auf objektorientierten Sprachparadigmen setzen sie ein Verständnis für eben diese voraus. Darüber hinaus ist der Begriff „ableiten“ mit der mathematischen Ableitung auch mehrdeutig besetzt. Es wurde daher entschieden, Vererbung nur sparsam einzusetzen und falls notwendig als „Generalisierung“ bzw. „Spezialisierung“ zu bezeichnen.

2. Die Trennung von Klasse und Objekten entstammt ebenfalls objektorientierten Programmierkonzepten. Es war während der Arbeit mit den Experten oftmals schwierig zu unterscheiden, ob sie mit dem Begriff „Fahrzeug“ die Klasse der Fahrzeuge, ein konkretes gekennzeichnetes Objekt oder ein zu definierendes Fahrzeugkonzept meinten. Da UML-Klassendiagramme als Grundlage verwendet wurden, war es unumgänglich, dieses Konzept zumindest rudimentär verständlich zu machen.

Beschränkt wurde sich daher auf folgende Elemente von Klassendiagrammen:

- Klassen
- Eigenschaften von Klassen
- Gerichtete Assoziationen ohne Kardinalitäten

Diese konnten mithilfe von Whiteboards oder Flipcharts zusammen mit den Experten konstruiert und diskutiert werden. Es war kein gesondertes Werkzeug notwendig. Die graphischen Darstellungen wurden dann in eine textuelle Repräsentation überführt, die schließlich ProcDSL bildete.

Folgende Anforderungen sollten wiederum durch eine graphische Weboberfläche bedient werden:

Nr.	Anforderung
1	Eine an Microsoft Office angelehnte Menügestaltung und -führung.
2	Drag&Drop-Funktionalität aller Gestaltungselemente der Pläne für einfaches Editieren.
3	Automatisches Ausrichten von Gestaltungselementen an anderen Objekten in Plänen mit Positionsangabe für präzises Positionieren und Gestalten.
4	Bearbeiten der Werte von Gestaltungselementen alternativ per Daten-Ansicht und -Eingabe.
5	Kontextabhängiges Ein- und Ausblenden der zum Hinzufügen durch den Benutzer verfügbaren Objekte.
6	Unterstützung des Benutzers beim Hinzufügen komplexer Objekte durch Aneinanderreihung von Dialogen („Wizard“).
7	Kontextsensitive Suche mit automatischem Vervollständigen.
8	Rückgängig- und Wiederholen-Funktion bei allen Operationen.
9	Authentifizierung und Autorisierung von Benutzern.
10	Freigeben von Plänen für andere Benutzern zum gemeinsamen Bearbeiten und Konfiguration der Schreib/Lese-Berechtigungen.
11	Export von Microsoft PowerPoint- und Excel-Dateien sowie Instanzen der textuellen Repräsentation in ProcDSL.
12	Bearbeitungshistorie mit Versionierung und Wiederherstellung alter Versionen.

Tabelle 3.6: Liste der Anforderungen an eine Webapplikation zur graphischen Modellierung formaler Prozessdarstellungen

Die technische Realisierung der graphischen Oberfläche basiert auf dem Google Web Toolkit (GWT) [GWT]. Entwickelt wurde somit eine reine Webanwendung. GWT ermöglicht eine Entwicklung in reinem Java-Quellcode und übersetzt diesen in JavaScript und generiert die zugehörigen Hypertext Markup Language (HTML)-Dateien. GWT unterstützt bereits die Implementierung von Drag&Drop-Funktionalität, wie in Anforderung 1 aus Tabelle 3.6 gefordert. Die Entwicklung in Java hat den Vorteil, dass Standardwerkzeuge wie Eclipse und automatisierte Qualitätssicherungsmethoden zum Einsatz kommen können [Ecl]. Weiterhin können die von MontiCore generierten Elemente vollständig verwendet werden, da die Generierung ebenfalls in Java erfolgt. Als Webcontainer wird Apache Tomcat verwendet, für die Datenhaltung und -persistenz MySQL sowie Hibernate [Apad, Ora, JBo]. Da die in der Oberfläche erstellten Pläne auch exportiert werden sollen, wird auf das Apache POI-Projekt zurückgegriffen, welches Bibliotheken für das Erzeugen von Microsoft Office-Dokumenten bereitstellt [Apaa].

Eine große Herausforderung bei der Entwicklung waren die komplexen Zusammenhänge zwischen graphischen Symbolen an der Oberfläche und deren semantischer Bedeutung. So werden in den Plänen unterschiedliche Ebenen aber auch Geschäftsbereiche des Unternehmens abgebildet. Ein Meilenstein kann hierbei auf einer Ebene definiert und für die darunter liegenden Ebenen in allen Geschäftsbereichen gültig sein. Allerdings ist nur ein Geschäftsbereich für das Ergebnis des Meilensteins verantwortlich. In dessen spezifischer Ansicht wird dies durch ein ausgefülltes

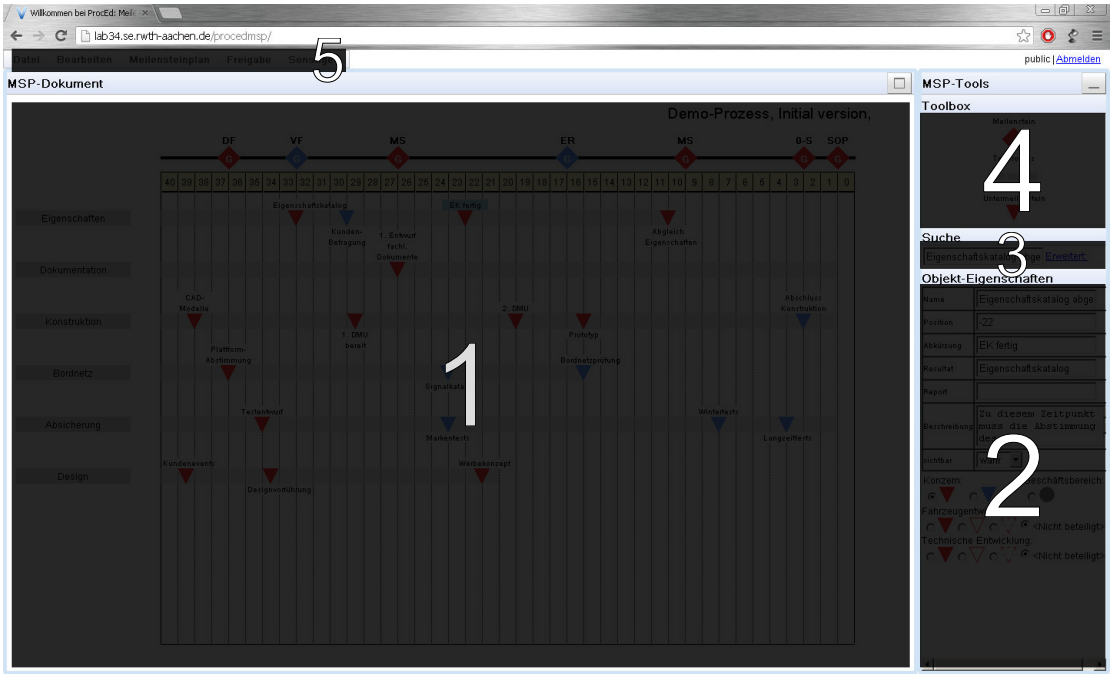


Abbildung 3.7: Schematische Darstellung der Weboberfläche des Prozesseditors ProcEd

Symbol dargestellt, in allen anderen durch ein nicht ausgefülltes. Viele Iterationsschleifen zwischen Domänenexperten und Entwicklern waren notwendig, um eine korrekte Repräsentation der Zusammenhänge in der Software zu erhalten. Mit den Domänenexperten wurde ausschließlich über die Zusammenhänge der ihnen bekannten Prozessobjekte diskutiert, nicht über technische Details. Das Datenmodell enthält somit auch nur Geschäftsobjekte. Die Entwicklung zwischen ProcDSL und graphischer Oberfläche wurde vor allem deswegen in diese zwei Teile geteilt. So konnte die Editoroberfläche entstehen, während das Datenmodell sich laufend änderte. Die Klassen des Datenmodells konnten mithilfe von MontiCore bei Bedarf neu generiert und als Bibliothek eingebunden werden. Durch den Einsatz des Programmiermusters „Command“ müssen lediglich Operationen definiert werden, die von den einzelnen Kommandos auf den Objekten der Datenklassen ausgeführt werden können. Nur die Schnittstelle zwischen Datenmodell und Persistenzschicht muss neu angepasst werden. Der Aufwand bei der Änderung des Datenmodells reduziert sich jedoch auf ein Minimum.

Anforderung 1 aus Tabelle 3.6 ist durch Designentscheidungen der Oberfläche bedient worden. Die Abbildung 3.7 zeigt die schematische Anordnung der Oberfläche, sie ist analog zu Abbildung 3.4 zu betrachten. Die mit „1“ gekennzeichnete Fläche ist die Hauptarbeitsfläche, in welcher ein Meilensteinplan erstellt und bearbeitet wird. Fläche „2“ beinhaltet die Objekteigenschaften, falls ein Objekt in der Arbeitsfläche ausgewählt wurde. Im Bereich der Fläche „3“ findet sich die Suchfunktion. Fläche „4“ stellt dem Benutzer Elemente zum Einfügen per „Drag&Drop“-Funktion in den Meilensteinplan zur Verfügung. Fläche „5“ beinhaltet eine Menüleiste. Mithilfe von Menüs wie bspw. „Datei“ oder „Bearbeiten“, die an Menüs von Microsoft Windows-Anwendungen ange-

lehnt sind, wird dem Benutzer der Umstieg auf eine Webanwendung erleichtert. Hinzu kommen Kontextmenüs, die bspw. mithilfe der rechten Maustaste aufgerufen werden können. Abbildung 3.8 zeigt die Menüleiste des Prozesseditors. Deutlich wird der konservative Sprachgebrauch der Menüeinträge.

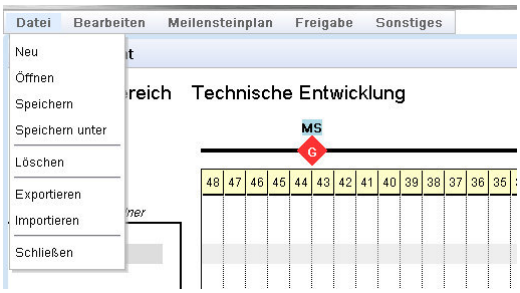


Abbildung 3.8: Das Dateimenü des webbasierten Prozesseditors ProcEd

Die Anforderungen 2 und 3 aus Tabelle 3.6 können zusammen betrachtet werden, da sie den selben Arbeitsablauf betreffen. Ein Benutzer möchte ein neues Element aus der Seitenleiste einfügen oder ein bereits vorhandenes neu platzieren. Hierzu führt er eine Drag&Drop-Operation durch: Er selektiert das gewünschte Objekt mit der linken Maustaste, hält diese gedrückt und positioniert das Objekt wie gewünscht. Da die Meilensteinpläne bei einer hohen Anzahl von Elementen unübersichtlich werden können und sie weiterhin durch gleichmäßig ausgerichtete Elemente aufgeräumt wirken müssen, wird der Benutzer bei der Positionierung unterstützt. Objekte richten sich selbstständig aneinander aus, sodass Meilensteine bspw. immer mittig in einer darunter liegenden Schwimmbahn liegen müssen. Liegen zwei Meilensteine an der selben Stelle, werden sie automatisch in einem passenden Abstand nach oben und unten verschoben. Der Benutzer wird weiterhin bei der genauen Positionierung unterstützt, indem in einem Kontextmenü neben seinem Mauszeiger während einer Drag&Drop-Operation Informationen über die Position des Objekts angezeigt werden. Anforderung 4 aus Tabelle 3.6 betrachtet den umgekehrten Fall, in dem der Benutzer etwa die Position eines Meilensteins direkt als Wert in die Objektinformationen auf der rechten Seite einträgt. Auch diese Arbeitsweise wird vom Editor unterstützt.

Die Darstellungsformen hängen im Plan stark von der jeweiligen Sicht auf die Daten ab, sie sind somit kontextabhängig. Meilensteine, die von einer übergeordneten Darstellungsebene auf einer darunter liegenden angezeigt werden, haben bspw. eine andere Farbgebung. Die Anforderung 5 aus Tabelle 3.6 trägt dem Rechnung: dem Anwender sollen zur Unterstützung während der Bearbeitung eines Plans nur solche Elemente angezeigt werden, die er auch tatsächlich verwenden kann. Eine Alternative hierzu ist das Anbieten sämtlicher Darstellungsobjekte und der Akzeptanz fehlerhafter Pläne oder auch der Ausgabe einer Fehlermeldung, wenn der Anwender Objekte „falsch“ einsetzt. Beide Varianten sind aufgrund der Komplexität der Pläne untauglich, weshalb die Unterstützung des Benutzers gefordert wird. Der Editor interpretiert daher die aktuell eingestellte Sicht und zeigt nur syntaktisch „passende“ Prozesselemente an. Die folgende Anforderung 6 aus Tabelle 3.6 ist ebenfalls durch eine auf den Kontext reagierende Hilfestellung in der Anwendung realisiert. Sie wird benötigt, wenn der Benutzer zwar ein korrektes Element zum Einfügen in den Plan gewählt hat, das Einfügen aber mehrere Schritte nach sich zieht. Dies wird vor

allem beim Einfügen von Meilensteinen benötigt. Wie in Abbildung 3.5 erkennbar, können verschiedene Organisationseinheiten durch unterschiedliche Verantwortlichkeiten mit einem Meilenstein verbunden sein. Durch einen sog. „Wizard“ kann der Benutzer Umfang und Art der verbundenen anderen Organisationseinheiten spezifizieren. Beide Anforderungen erleichtern dem Benutzer das Erstellen auch komplexer Pläne. Durch die Trennung von ProcDSL und ProcEd kann die vorhandene Menge an Prozesselementen jederzeit mit wenig Aufwand ergänzt werden.

Das Ziel von Anforderung 7 aus Tabelle 3.6 ist im Gegensatz zu den vorhergehenden Anforderungen nicht das Ergänzen des Modells, sondern das Operieren darin. ProcEd unterstützt den Anwender durch das automatische Ergänzen von Eingaben bei Suchen. Diese Funktion ist bereits heute von Internet-Suchmaschinen bekannt und kann speziell in einem komplexen Prozesskontext das Arbeiten vereinfachen. Hierbei werden die Bezeichnungen von Objekten beim Laden eines Plans in einer separaten Liste gespeichert. Operationen, die das Datenmodell verändern, führen entsprechend diese Veränderung auch auf der Liste aus. Bei Eingaben des Benutzers in vorher festgelegten Feldern, benutzt eine GWT-Funktion diese Liste um die Eingaben des Benutzers mit ihr zu vergleichen und Vorschläge zu machen.

Anforderung 8 aus Tabelle 3.6 schränkt bereits die für die programmtechnische Umsetzung von ProcEd zur Verfügung stehenden Programmiermuster ein. Eingesetzt wurde dann das Muster „Command“. Die einzelnen Kommandos führen Operationen aus, für die analog eine weitere Operation zum Umkehren der durchgeführten Änderungen vorhanden ist. Weiterhin müssen die Informationen über die veränderten Daten in den Kommandos hinterlegt werden. Die Kommando-Objekte werden auf einem Stack geschrieben und dort bei Bedarf wieder abgerufen. Das Programmiermuster ermöglicht eine sehr disziplinierte Entwicklung, da für eine neue Funktion im Editor weder Datenmodell noch Infrastruktur des Programms geändert werden müssen. Stattdessen wird ein neues Kommando eingefügt, welches die gewünschten Aktivitäten auf dem Datenmodell umsetzt. Konventionen, wie bspw. das gleichzeitige Aktualisieren der Liste für das automatische Vervollständigen, können durch automatisierte Qualitätssicherungsmaßnahmen geprüft werden.

Die Anforderungen 9 bis 12 aus Tabelle 3.6 werden an dieser Stelle nicht näher betrachtet, da sie nicht unbedingt ein formales Modell benötigen. Zwar ist dessen Vorhandensein für die Architektur der Anwendung von Vorteil, indem bspw. Freigaben auch nur für spezielle Teile eines Modells erfolgen können. Jedoch ist die technische Realisierung für die weitere Arbeit nicht relevant.

In diesem Teilprojekt zeigt sich, dass bei dem Umgang mit verschiedenen Geschäftsobjekten die durch komplexe Zusammenhänge miteinander verbunden sind, hoher Aufwand bei der Erstellung eines konsistenten Datenmodells zu erwarten ist. Auffällig ist, dass heute keinerlei Werkzeuge vorhanden sind, die das Erfassen und standardisierte Formalisieren beliebiger Geschäftsobjekte derart ermöglichen, dass diese Modelle in beliebigen anderen Programmen weiter verwendet werden können. Zwar können UML-Klassendiagramme eingesetzt werden, um die Elemente zu beschreiben. Es fehlt jedoch auch hier an Werkzeugen, die eine einfache digitale Erfassung und Formalisierung bieten. Das Abbilden von Informationen über die Art einer Relation zwischen zwei Klassen ist in UML-Klassendiagrammen nicht ohne Weiteres möglich. Eigenschaften von Relationen können nur in Relationsklassen dargestellt werden, welche wiederum von einem anderen Werkzeug, welches das UML-Modell verwendet, anders interpretiert werden können. Darüber hinaus fällt IT-fremden Domänenexperten die Arbeit mit vollständigen

Klassendiagrammen schwer.

Die programmtechnische Umsetzung von ProcDSL konnte durch den Einsatz von MontiCore stark vereinfacht werden. Die graphische Oberfläche wiederum profitierte vom Einsatz des GWT-Frameworks. Durch den Einsatz des Command-Programmiermusters konnte die Entwicklungsarbeit nach Fertigstellung der Infrastruktur der Software auf die Operationen am Datenmodell fokussiert werden. Insgesamt konnten durch die Kombination von Methodik – dem Erarbeiten eines formalen Modells mit Domänenexperten – und Werkzeugen – dem Einsatz des Command-Musters – starke Geschwindigkeitsvorteile gegenüber einer konventionellen Entwicklung erzielt werden. Wurde bspw. in einem Experteninterview klar, dass die Bedeutung eines bestimmten Prozesselements eine andere als vorher angenommen war, musste lediglich die formale Sprache angepasst, mit MontiCore die Klassen des Datenmodells neu generiert und die Kommandos, die den jeweiligen Teil des Modells betreffen verändert werden. Der Nachteil des Einsatzes einer domänenspezifischen Sprache in Erweiterte Backus-Naur-Form (EBNF)-Notation ist jedoch, dass andere Softwarewerkzeuge, welche die erstellten Modelle ebenfalls verwenden möchten, ebenfalls dieselbe Sprache verarbeiten müssen. Sie benötigen also die Spezifikation und ein Werkzeug wie MontiCore, um mit ähnlich geringem Aufwand Prozessmodelle verarbeiten zu können. Weiterhin müssen sie die Bedeutung und Funktionsweise von Konstruktionen im Modell interpretieren können. Hierzu zählen im Falle von ProcDSL hauptsächlich die Relationsklassen zur Typisierung von Relationen.

3.5 Entwicklung eines Werkzeugs zur webbasierten Visualisierung von Informationen

Ein weiterer Teil im Kontext der Zusammenarbeit mit der Volkswagen AG war das Projekt „Kundenwertnavigator“. Analog zum webbasierten Prozesseditor aus dem vorhergehenden Kapitel, war es das Ziel, eine webbasierte Oberfläche für die Datenmodellierung zu entwickeln. Allerdings lag der Fokus nicht auf der Visualisierung der Informationen in einem vorgegebenen Format, sondern auf der Modellierung grundlegender Zusammenhänge. Der Anwender sollte dazu befähigt werden, gefundene Prozesselemente einfach aufnehmen und mit anderen verknüpfen zu können. Mit Hilfe von Drag&Drop-Operationen kann somit ein Graph aufgebaut werden, dessen Kanten wiederum mit Eigenschaften versehen werden können. Eine Export-Schnittstelle stellt die Möglichkeit zur Verfügung, Formate zu definieren, in die ein Graph zur Laufzeit exportiert werden kann. Das Anwendungsgebiet des Kundenwertnavigators liegt somit im Bereich der Formalisierung der im Rahmen der Prozessanalyse gefundenen Objekte. Die entstehenden formalen Modelle können als Grundlage für die im weiteren Verlauf der Arbeit betrachteten Artefaktmodelle verwendet werden.

Der Kundenwertnavigator bietet eine Schnittstelle, welche es erlaubt, Softwaremodule zu entwickeln, die Operationen mit dem erzeugten Graph als Eingabe durchführen können. Dies befähigt Entwickler, Simulationsmodule zu entwickeln, ohne das ein gesondertes Eingabeformat notwendig ist. In Kapitel 6 wird ein Werkzeug beschrieben, welches mithilfe von Abfragen auf gegebene Modelle die Auswirkungen von Änderungen an bestimmten Stellen in einem Modell approximieren soll. Dieses kann als Modul für den Kundenwertnavigator entwickelt werden, um die vorhandenen Modelle verwenden zu können. Der Graph wird hierbei zur Laufzeit an ein vom

Nutzer ausgewähltes Modul als Datenstruktur weitergereicht, welches dann nahezu beliebige Operationen darauf ausführen kann.

Der Name der Applikation stammt von den ersten Datenstrukturen, die darin abgebildet wurden: Kunden werden in der Geschäftswelt nach sog. Milieus gruppiert, denen ganz bestimmte Eigenschaften und Werte zugeordnet werden [Asc06]. Diese Strukturen sollten in Beziehung gesetzt werden zu Fahrzeugen und deren Ausstattungsvarianten, um simulieren zu können, inwiefern eine Veränderung in den Varianten ein Produkt aus der angepeilten Zielgruppe bewegt. Hierzu war es nötig, Elemente aus verschiedenen Domänen miteinander verknüpfen zu können, um so ein Modell über Kundenmilieus und -werte, Produkteigenschaften, Anforderungen und schließlich Modulen aus Fahrzeugbaukästen aufzubauen. Die Ausprägungen der einzelnen Typen wurden in Schichten angeordnet und dann die Einflüsse zwischen zwei Schichten pro Objekt numerisch festgelegt. Es existiert somit eine Relation zwischen jedem Objekt einer Schicht zu jedem Objekt der nachfolgenden und vorhergehenden Schicht.

Die Anwendung dieses Modells findet sich somit in der Konfiguration von Fahrzeugen: Durch die klare Zuordnung der Module zur bestimmten Milieus mithilfe von Anforderungen als Bindeglied können bestimmte Module im Voraus ausgewählt werden. Kundenmilieus und deren Werte sind bereits klar definiert, ebenso Module. Durch die formale Definition im Rahmen eines Modells dienen die Anforderungen als Knotenpunkt. Es wurde somit gezeigt, wie ein Modell basierend auf verschiedenartigen Objekten mit Hilfe eines Werkzeugs konzipiert sein kann. Mithilfe der Gewichtungen der Kanten als Einflussgrößen können in einem nächsten Schritt Auswirkungen von Änderungen nachvollzogen werden: Liegt bspw. das konzipierte Fahrzeug noch innerhalb eines bestimmten Milieus, wenn ein Modul entfernt und dafür ein anderes bevorzugt wird?

Der Kundenwertnavigator demonstriert somit, wie ein einfach zu verwendender Editor für das Erstellen und Bearbeiten von Modellen entworfen werden kann. Der Einsatz von Web-Technologie ermöglicht Anwendern einen schnellen Zugang, ohne das eine Installation notwendig wird. Die Bedienung ist beschränkt auf das Anlegen von Knoten und Kanten sowie das Eingeben von Eigenschaften für diese. Durch die modularen Schnittstellen ist es möglich, die vorhandenen Modelle einer Vielzahl unterschiedlicher Programmmodule zur Verfügung zu stellen. Gleichzeitig sind Exportfunktionen als Modul realisiert, welche das Exportieren von Modellen in Formate wie die Extensible Markup Language (XML) oder eine domänenspezifische Sprache ermöglichen.

3.6 Ergebnisse des Projekts mit Volkswagen und erste Schlussfolgerungen

Das Projekt „Durchgängiges Anforderungsmanagement für modulare Produktstrukturen“ hat alle fünf im Rahmen von Abschnitt 3.1 definierten Ziele erreicht. Abschnitt 3.2 beinhaltet hierbei die Erarbeitung des Projektergebnisses Nr. 1, indem es die Anwendung der Methodik von Pillkahn zur Szenarioerstellung im konkreten Fall des Anforderungsmanagements für Fahrzeugprojekte beschreibt. Es zeigte sich, dass die notwendige umfangreiche Prozessanalyse mithilfe strukturierter Interviews diverse verschiedenartige Elemente aus den Prozessen zutage fördert, die im Rahmen des Anforderungsmanagements auf einer hohen Ebene zusammenhängend betrachtet werden müssen. Hierzu wurde der Begriff „Artefakt“ verwendet, um einen gemeinsamen Sprachgebrauch zu finden. Die Denkweise, Elemente eines Unternehmens wie bspw. Prozesse, Dokumente oder

Strukturen gemeinsam als Artefakte zu behandeln und so miteinander in Beziehung zu setzen, bestimmt den weiteren Verlauf dieser Arbeit. Sie ist grundlegend für die Idee, Modelle nicht getrennt zu betrachten sondern sie zu verbinden und in einen Kontext einzuordnen.

Das Scouting von Anforderungsmanagementwerkzeugen wurde in Abschnitt 3.3 beschrieben, welches Projektergebnis Nr. 2 aus Abschnitt 3.1 darstellt. Es zeigt sich, dass Modellierung Einzug hält in die Welt der Werkzeuge, der Fokus jedoch klar auf dem zu erstellenden Produkt liegt. Sekundäre Prozesse, in denen Gemeinkosten durch eventuelle Änderungen verursacht werden, können nicht modelliert werden. Das Werkzeug Contour von Jama, mit dessen Hilfe das Projektergebnis Nr. 4 aus der Liste des Abschnitts 3.1 erstellt wurde, stellt eine Ausnahme insofern dar, als dass dort Objekte beliebigen Typs erstellt und verbunden werden können. Zwar können keine semantischen Informationen zu den Assoziationen hinzugefügt werden, jedoch ermöglicht es die Erstellung von Modellen außerhalb des Produktkontextes.

Der webbasierte Prozesseditor ProcEd zeigt, wie Modelle durch Domänenexperten erstellt werden können, ohne dass sie über ein tiefes Verständnis für die Tätigkeit des Modellierens verfügen. Hierzu wird der Anwender mit einer ihm bekannten Darstellungsform konfrontiert, auf der er per Drag&Drop Elemente anordnen kann. Die Software sorgt dafür, dass die Vorgaben des formalen Modells eingehalten werden, indem kontextbezogen Werkzeuge und Werkzeugelemente zur Verfügung stehen. Gleichzeitig wird demonstriert, wie modellbasierte Software schnell und agil entwickelt werden kann. Der Prozesseditor aus Abschnitt 3.4 ist Projektergebnis Nr. 3.

Im Verlauf der Phase 6 der Szenarioerstellung in Abschnitt 3.2.6 wird das Projektergebnis 5 umrissen, die Konzeption eines Folgeprojekts. Der weitere Verlauf dieser Arbeit orientiert sich hierbei an den notwendigen Rahmenbedingungen für dieses Projekt. Dieses Kapitel hat gezeigt, dass die Notwendigkeit besteht, verschiedenartige Unternehmensobjekte miteinander in Beziehung setzen zu müssen, um die Zielsetzung dieser Arbeit – der Schaffung einer Grundlage für die automatische Auswirkungsanalyse von Änderungen von Anforderungen – erreichen zu können. Anforderungsmanagement bietet durch das in Definition 2.3.2 beschriebene Konzept der Traceability im weiteren Sinne ein Konzept, welches die Vernetzung von Elementen verschiedenen Typs zur Auswirkungsanalyse verwendet. Gleichzeitig wird ersichtlich, dass es zielführend ist, Domänenexperten mithilfe einfach zu verwendender Werkzeuge formale Modelle erstellen zu lassen. Der Einsatz solcher Werkzeuge wurde in unterschiedlichen Ausprägungen erprobt. Es fehlt heute jedoch eine gemeinsame Grundlage für umfassende Modelle sowie eine Methodik zur Erfassung.

Im nächsten Schritt wird daher im folgenden Kapitel der Begriff „Artefakt“ eingeführt und näher beleuchtet. Danach folgt Kapitel 5 mit einer Analyse der Möglichkeiten eines Anforderungsmanagements, welches Artefaktmodelle verarbeiten kann. Es werden Anforderungen entwickelt, die ein Werkzeug hierfür realisieren muss. Eine Methodik zur Erarbeitung der Modelle wird in Kapitel 6 entwickelt. Ebenfalls in diesem Kapitel wird mithilfe eines Prototyps gezeigt, wie die Realisierung eines Werkzeugs aussehen und über welche Fähigkeiten es verfügen kann. Hierzu wird ein Beispielprojekt erarbeitet und daran Abfragen aus dem Aufgabenspektrum des Anforderungs- und Projektmanagements gestellt.

Kapitel 4

Prozessartefakte als durchgängiges Konzept zur Formalisierung von Prozessinformationen

Dieses Kapitel führt eine Begriffsbildung durch, um die Bedeutung von Artefakten für den Fortlauf der Arbeit zu klären. Weiterhin wird auf den Zusammenhang zwischen Artefakten, Komplexität und Veränderung eingegangen. Beispiele veranschaulichen den Artefakt-Begriff und bieten eine Basis für die Lösung der Problemstellung dieser Arbeit, eine Grundlage für das Abschätzen der Auswirkung von Änderungen auf die Komplexität der Prozesse zu finden. Das Formalisieren von Informationen über Artefakte spielt hierbei eine zentrale Rolle. Hierzu muss neben dem Artefakt-Begriff ebenfalls „Komplexität“ für die weitere Verwendung in dieser Arbeit definiert werden.

4.1 Begriffsbildung

Innerhalb von Geschäftsprozessen werden Artefakte erzeugt, bearbeitet und – sobald obsolet – wieder eliminiert. Sie sind die bestimmenden Teile des inneren Ökosystems eines Unternehmens. Artefakte sind bspw. Dokumente, Software, Module, Prototypen aber auch ganze Prozesse bzw. deren Beschreibung. Zu einem Artefakt macht sie der Umstand, dass sie alle auf administrativer Ebene als Daten repräsentiert werden (müssen). Es existiert somit immer ein – teils gedankliches, teils faktisches – Metamodell, in dem die Strukturen und Inhalte der einzelnen Artefakte beschrieben sind. Notwendig ist eine einheitliche Verwendung des Begriffs „Artefakt“, daher soll folgende Definition im weiteren Verlauf der Arbeit gelten:

Definition 4.1.1 (Artefakt) *Ein Artefakt repräsentiert ein virtuelles oder physisches Geschäftsobjekt innerhalb eines Unternehmens mit definierten Eigenschaften, welches im allgemeinen Sprachgebrauch Verwendung findet und eindeutig abgrenzbar ist.*

War bisher von Geschäfts*objekten* im Zusammenhang mit Artefakten die Rede, so ist die Bezeichnung im Sinne des Software Engineerings irreführend. Wird dort der Begriff Objekt gemeinhin für eine konkrete Instanz mit ausgeprägten Werten für Eigenschaften verwendet, so ist ein Artefakt in diesem Sinne eine *Klasse*. Tatsächlich kann aus einer Modellierungsperspektive die Klasse „Artefakt“ als Oberklasse betrachtet werden. Ein später in dieser Arbeit

eingehender betrachtetes Arbeitsergebnis im Produktentstehungsprozess bei Volkswagen ist der sog. Eigenschaftskatalog, welcher in der Planung von zu entwickelnden Fahrzeugen eine entscheidende Rolle spielt [Rin06]. Es existiert somit ein Artefakt `Eigenschaftskatalog`, kenntlich gemacht im Fließtext durch eine `Typewriter-Schriftart`. Eine konkrete Instanz eines `Eigenschaftskatalogs` ist somit der *Golf XY-Eigenschaftskatalog*, zu erkennen durch eine *kursive* Schriftweise.

Artefakte können andere Artefakte enthalten, sind mit Artefakten durch Relationen verbunden und besitzen Eigenschaften. Es findet keine Eingrenzung statt, welches Element eines Unternehmens ein Artefakt sein kann und welches nicht. Diese Generalisierung ermöglicht das Erfassen aller Objekte. So können Prozesse genauso als Artefakte betrachtet werden, wie ein Prototyp, der in einem Prozess erstellt wird. Der Fokus verschiebt sich hierbei weg von der klaren Typisierung eines Elements hin zu einer Einordnung in einen Kontext. Auch der Aufbau eines Artefakts kann, muss aber nicht notwendigerweise beschrieben sein, um eine Verknüpfung mit anderen Artefakten vornehmen zu können. Gefördert wird somit eine Top-Down-Sicht, welche die Information des Vorhandenseins eines Artefakts in einem bestimmten Kontext seiner detaillierten Beschreibung überordnet. Um diesem Umstand Rechnung zu tragen, wurde der Begriff „Artefakt“ anstelle von „Geschäftsobjekt“ oder „Klasse“ gewählt, da er allgemeiner und weniger in der Literatur vorbelastet ist.

Im Rahmen der im vorherigen Kapitel 3 beschriebenen Prozessanalyse sind folgende Punkte bei der Erfassung von Artefakten bemerkenswert:

1. **Verknüpfung:** Artefakte erfahren grundsätzlich durch Verknüpfung mit anderen Artefakten Relevanz. Oftmals werden Artefakte mit unterschiedlichen Prozessen verknüpft und dort auf verschiedene Weisen genutzt und weiterverarbeitet, vor allem wenn sie digitaler Natur sind. Der Grad der Verknüpfung eines Artefakts wird als *Komplexität* beschrieben und in Abschnitt 4.2.1 näher beleuchtet. Hierin findet sich die Ursache für Gemeinkosten, die dieser Arbeit als Problemstellung dienen: Jede Änderung einer Instanz eines Artefakts beeinflusst möglicherweise die Instanzen anderer verbundener Artefakte.

2. **Veränderung:** Artefakte unterliegen Veränderungen. Hiermit ist einerseits die Änderung von Werten einer konkreten Instanz eines Artefakts über den Zeitverlauf seiner Existenz gemeint. Andererseits werden der Aufbau und auch die Verknüpfungen eines Artefakts beeinflusst, vor allem, wenn es sich um digitale Artefakte handelt. Der in dieser Arbeit verfolgte Top-Down-Ansatz trägt diesem Umstand Rechnung, da Änderungen schneller Auftreten können als eine Dokumentation des Aufbaus in Modellierungswerkzeugen. Abschnitt 4.2.2 geht näher auf diese Evolution der Artefaktwelt eines Unternehmens ein.

Bevor in den folgenden Kapiteln eine Methodik sowie ein Werkzeug zur Beschreibung und Verwaltung von Artefakten entwickelt wird, fokussiert dieses Kapitel im weiteren Verlauf auf die Eigenschaften und das Verhalten des abstrakten Artefakt-Begriffs.

4.2 Artefakte im Kontext von Komplexität und Veränderung

Die subjektiv von Individuen empfundene Darstellung der Welt, gefiltert durch verschiedene neuronale Filter, die durch Erfahrungen und Erlebnisse geprägt wurden, macht deren eigene Realität aus. Die Abbildung einer objektiven Realität gestaltet sich demnach als unmöglich. Dies gilt ebenso innerhalb von Unternehmen, da diese sich durch eine Vielzahl miteinander kommunizierender Personen und Maschinen auszeichnen. All diese Teilnehmer des Ökosystems haben eine eigene Sichtweise auf die Umwelt. Software, die versucht nur eine begrenzte Sichtweise abzubilden, diese aber auf mehrere Individuen zu übertragen, ist deshalb oft heftiger Gegenwehr ausgesetzt. Deutlich wird dies bspw. bei der Integration von Unternehmensprozessen in einer zu entwickelnden Software [And06]. Durch Stakeholderanalysen etwa wird in der Softwareentwicklung versucht, ein möglichst umfassendes Bild der Realitäten der Beteiligten zu erzeugen und später zu berücksichtigen, um die Akzeptanz einfacher zu gestalten. Auch das Modellieren von Artefakten gehört heute teilweise in Softwareentwicklungsprojekten dazu, allerdings immer nur in eng abgesteckten Kontexten. Insbesondere im Bereich der Cyber-Physical Systems wird sichtbar, dass die dortigen Produkte zunehmender Komplexität unterliegen [KRS12].

Realität kann demnach auch als das empfundene kausale Zusammenwirken von Artefakten in einem Unternehmen aus der Sicht eines Individuums oder einer Gruppe wie etwa einer Abteilung betrachtet werden. Als Beispiel sei etwa das Erstellen bestimmter Kennzahlen oder die Pflege von Daten in Systemen genannt. Je nach Umfeld fallen andere Aufgaben, andere Berichte, andere Kennzahlen und andere Systeme an. Oftmals sind die zugrundeliegenden Daten identisch, doch werden aufwändige Prozesse für den Abgleich zwischen unterschiedlichen Bereichen eingerichtet. Manuelle Arbeit fällt an, wenn zwischen einzelnen Artefakten Transformationen von Mitarbeitern durchgeführt werden müssen.

4.2.1 Komplexität von Artefakten in Unternehmensprozessen

Die Evolution nimmt Einfluss auf die Komplexität im Unternehmen, indem die Transformationsprozesse zwischen Artefakten aufwändiger oder zahlreicher werden. Wirtz unterscheidet grundsätzlich zwei Arten von Komplexität [Wir08]:

1. Elementekomplexität: Bezieht sich auf die Anzahl an Elementen innerhalb eines Systems.
2. Relationenkomplexität: Bezieht sich auf den Grad der Verknüpfung zwischen den Elementen.

Entscheidungen an einer Stelle im Unternehmen können auf zwei Arten von der Komplexität im Unternehmen beeinflusst werden:

1. Eine Entscheidung verändert die Instanz eines Artefakts, was Änderungen an anderen Artefaktinstanzen nach sich zieht. In diesem Fall entsteht Aufwand, der bestimmt ist durch die vorherrschende Elemente- und Relationenkomplexität im Unternehmen.
2. Eine Entscheidung beeinflusst die Komplexität direkt, d.h. sie verändert die Elemente- oder Relationenkomplexität. Dies ist der Fall, wenn ein gänzlich neues Artefakt eingeführt

wird, welches durch neuen Relationen mit anderen Artefakten verknüpft wird oder ein vorhandenes Artefakt abermals instantiiert wird. In diesem Fall entsteht Aufwand durch die Einführung oder Erstellung des Artefakts selbst und Folgekosten durch beeinflusste Entscheidungen vom ersten Typ.

Entscheidungen verändern demnach die Komplexität oder werden durch sie beeinflusst. Berücksichtigt wird sie in den meisten Fällen allerdings im Vorfeld nicht. Hintergrund ist die fehlende Repräsentation von Komplexitätskosten in der Kostenrechnung [Wir08]. Steigende Komplexität führt zu höheren Komplexitätskosten, ausgelöst durch vermehrte Schnittstellen zwischen den einzelnen Abläufen und Artefakten. Komplexitätskosten können in den meisten Fällen nicht einzelnen Kostenträgern zugeschrieben werden, sondern finden sich als administrative Kosten in erhöhten Gemein- und Fixkosten wieder [BBS04], wodurch sie sich oft dem traditionellen Controlling entziehen [Wir08]. Darüber hinaus sind die indirekten Folgen einer Entscheidung, die sich über mehrere Artefakte fortsetzen können, nicht unbedingt mehr der ursprünglichen Entscheidung zuzuordnen.

Abzugrenzen ist die hier beschriebene Komplexität des Unternehmens und seiner Artefakte von der im Rahmen der Entwicklung betrachteten Komplexität des Produkts [LMB08]. Die Literatur im Bereich des Anforderungsmanagements beschränkt sich zum größten Teil auf letztere, beziehungsweise betrachtet beschränkt einige die Entwicklung umgebende Prozesse [Ebe10]. Eine relativ junge Ausnahme bildet die Arbeit von Schömann [Sch12]. Wie sich auch im Rahmen des Tool-Scoutings zeigte, fehlt für ein umfassendes Anforderungsmanagement die Möglichkeit der Modellierung und Einbindung aller Prozesse und Artefakte, die das Produkt betreffen bzw. von ihm betroffen sind. Wichtig ist demnach nicht nur die Komplexität des Produkts selbst, sondern des gesamten Unternehmens.

4.2.2 Veränderungen von Artefakten in Unternehmensprozessen

Die Notwendigkeit der Veränderung von Artefakten entsteht durch eine sich ständig ändernden Realität. Veränderungen sind einerseits induziert durch externe Einflüsse, etwa durch neue Technologien, Umstrukturierungen oder veränderte Schnittstellen und andererseits durch interne Änderungen, wie bspw. Mitarbeiterfluktuation oder Maßnahmen zur Steigerung der Effizienz. Unternehmen bleibt keine andere Wahl, als diese Evolution ihres internen Ökosystems als Konsequenz aus der Veränderung der Umwelt zu akzeptieren.

Konkret bedeutet dies für Artefakte, dass sie sich nicht nur selbst verändern, sondern auch ihre Schnittstellen dem Wandel unterworfen sind, da auch ihr Kontext sich ändert. Berichte müssen angepasst werden, um neue Kennzahlen darstellen zu können. Produkte werden mit neuen Technologien ausgestattet. Prozesse werden verändert und neue Dokumente mit neuen Strukturen erzeugt. Der vorhergehende Abschnitt 4.2.1 ging bereits auf die Auswirkungen ein, die Entscheidungen an einer Stelle an einer anderen haben. Hintergrund einer solchen Entscheidung ist immer eine veränderte Realität – etwa wenn ein neues Produkt entwickelt werden soll.

Oftmals findet eine Abbildung von Artefakten und ihren Zusammenhängen in Softwarewerkzeugen statt. Veränderungen der Realität ziehen daher entsprechend eine Änderung des Werkzeugs nach sich. Problematisch ist vor diesem Hintergrund, dass ein Unternehmen auf Fremdfirmen

angewiesen sein kann, welche die Software ursprünglich hergestellt haben. Diese existieren möglicherweise nicht mehr, haben kurzfristig keine Kapazitäten zur Änderung des Systems oder auch kein Interesse daran. Kommt es doch zu einer Beauftragung, dauern die Anforderungserfassung, Umsetzung, Einführung, Schulung und Fehlerbehebung in einem Großunternehmen lange. Die Realität und die sich daraus ergebenden Softwareanforderungen sind dann möglicherweise bereits schon wieder andere.

In der Praxis kann daher oftmals der Versuch der Anpassung der Realität an eine Software beobachtet werden. Dies bedeutet konkret, dass Artefakte verändert werden, um den Abbildungsmöglichkeiten eines Systems zu genügen. Auch können Adapter geschaffen werden, um ein Artefakt „verständlich“ für ein Softwarewerkzeug zu machen, also der internen Datenrepräsentation zu genügen. Diese finden sich bspw. oftmals bei Schnittstellen zwischen Softwaresystemen, wenn Datenformate einseitig geändert wurden. Ein anderes Vorgehen ist das Erfassen kompletter Prozesse im Rahmen des „Business Process Modelling“. Studien zeigen jedoch, dass der Kosten/Nutzen-Faktor solcher Initiativen überwiegend negativ ist [BWW10].

4.3 Beispiele zur Verdeutlichung der Thematik

Artefakte und ihre Verknüpfungen definieren die Komplexität innerhalb eines Unternehmens, durch Veränderungen in der In- und Umwelt wird diese beeinflusst. Gemeinkosten resultieren aus der nicht möglichen verursachungsgerechten Zuordnung von Komplexitätskosten. Die folgenden Beispiele sollen dies anhand von fiktiven Szenarien veranschaulichen.

4.3.1 Allgemeines Beispiel für Komplexität erhöhende Evolution

Das Unternehmen „Allgemeine Radiowerke GmbH“ fertigt im Jahr 1998 genau ein Produkt: Ein Autoradio mit der Bezeichnung „Super FM“, welches in Mittelklassewagen von einigen deutschen Herstellern verbaut wird. Im Rahmen der Produktentwicklung wird von der Forschung und Entwicklung (F&E) des Unternehmens ein Projekt ins Leben gerufen, das einen Nachfolger des Produkts zur nachhaltigen Sicherung des Fortbestands des Unternehmens entwickeln soll. Eine wichtige Anforderung ist hierbei, dass das Gerät in zwei Varianten verbaut werden kann: Einmal in Verbindung mit einem Kassettenspieler sowie auch mit einem CD-Spieler. Hierzu ist ein weiterer Chip auf der Platine notwendig. Ein Entwickler stellt bei einer Besprechung eine Liste mit allen benötigten Chips und deren Verwendung zusammen und verschickt diese an seine beiden Kollegen zur Abstimmung über Hersteller und genauen Typ.

Heute, im Jahr 2013, sind die Radiowerke inzwischen ein Unternehmen mit einem Umsatz im mittleren dreistelligen Millionen-Bereich und beschäftigen weltweit etwa 10.000 Mitarbeiter. Es gibt insgesamt 7 Produktlinien, die alle Autoradios vom Low- bis High-End-Bereich abdecken. Hinzu kommen durch Zukäufe noch Fertigungen für LCD-Displays, die in den Radios der neuesten Generation sowie in Navigationsgeräten von OEMs verbaut werden. Die Radios bieten, nun neben vielen Funktionen wie bspw. Eingänge für Mobiltelefone verschiedener Hersteller oder auch WLAN-Hotspots, ebenfalls eine Plattform für die Entwicklung von kleinen Programmen, genannt „Apps“. Die F&E umfasst knapp 1.000 Mitarbeiter. In einem Projekt soll das Nachfolgemodell des Oberklasseradios „Multi-RNS 2006“ entwickelt werden. Es umfasst hunderte

von Funktionen, Schnittstellen und Bedienmöglichkeiten, kann per Mobiltelefon über WLAN gesteuert werden und hat auch ein Online-Navigationssystem integriert.

Die ursprünglich auf Papier angelegte Liste für die benötigten Chips existiert immer noch. Inzwischen heißt sie „Provisorische Bauteil-Liste (PBL)“ und enthält nicht nur die Chips, sondern auch alle anderen „wichtigen“ Bauteile, von denen angenommen wird, dass sie später einmal im Gerät verbaut werden müssen. Geführt wird sie in einer Tabelle im Format eines bekannten Tabellenkalkulationsprogramms. Da die im Unternehmen vorhandenen Produktlinien voneinander abhängen, müssen die PBL von unterschiedlichen Entwicklungsprojekten immer gegenseitig aktualisiert werden, da es ansonsten später zu Inkompatibilitäten kommt. Die PBL eines Oberklassegeräts beinhaltet in etwa 400 Teile, was aber nicht unbedingt dem Umfang der späteren Stückliste entsprechen muss. Verwendet und bearbeitet wird sie etwa ein halbes Jahr ab Beginn eines einjährigen Entwicklungsprojekts, bevor ausschließlich mit Stücklisten gearbeitet wird. In dieser Zeit stellt sie das Rückgrat des Projekts dar: die Entwicklung pflegt in ihr Teile, die noch keine Teilenummer haben, da sie noch nicht im Detail geplant oder konstruiert sind. Der Beschaffung dient sie als „Einkaufsliste“ auf deren Basis die ersten Verträge geschlossen werden. Die Software-Entwicklung muss anhand der Chip-Daten die Performance ihrer Programme auslegen. Das Marketing macht anhand der Informationen die kaufentscheidenden Punkte fest.

Alle diese Bereiche wirken auf die Liste ein und sind gleichzeitig auf die Informationen angewiesen. Andere Entwicklungsprojekte sind gezwungen, sich in ihren Spezifikationen an das „Leitprojekt“ zu halten. Bereits das Aktualisieren der Liste benötigt zwei vollständige Arbeitstage, in denen Vertreter aller Stakeholder ihre Änderungen einbringen und gegenseitig abstimmen.

Das im Beispiel skizzierte Szenario zeigt eine Situation, wie sie in allen Unternehmen angetroffen werden kann, die Produktentwicklung durchführen. Verschiedene Veränderungstreiber außerhalb und innerhalb des Unternehmens führen zu einer Änderung der Strukturen und damit der Komplexität [Sch12]. Zwar werden oftmals bestimmte Prozesse oder Artefakte durch Software ersetzt oder unterstützt, doch immer wieder ist dies auch nicht der Fall. Problematisch bei Software-Lösungen ist, dass sie nicht den gesamten Kontext betrachten, sondern das isolierte Artefakt. Ein neu geschaffenes System erhöht dann möglicherweise an bestimmten Stellen die Bearbeitungsgeschwindigkeit, verringert aber nicht die Relationen-Komplexität. Oftmals müssen neue Schnittstellen geschaffen werden und Dateneingaben manuell erfolgen.

Ursprünglich temporär und als schnelle Hilfsmittel gedachte Lösungen durchlaufen einen evolutionären Prozess und finden sich nach einem gewissen Zeitraum an Stellen wieder, für die sie vorher nie konzipiert waren. Inzwischen haben sie aber eine Reichweite und Bedeutung erlangt, die es unmöglich macht, sie einfach abzuschaffen. Oftmals sind die Auswirkungen völlig unklar.

Die Komplexitätseffekte im Beispiel liegen mindestens in den aufwändigen Abstimmungsprozessen, aber nicht ausschließlich dort; auch das Formatieren der Tabellen innerhalb der Bereiche, damit andere Systeme mit den Informationen bestückt werden können gehört ebenso dazu wie das Suchen von Fehlern in den Daten. Kosten entstehen also vor allem an den Schnittstellen zwischen den Artefakten. Weiterhin entstehen Kosten durch die Versuche der Beschreibung des Prozesses sowie durch das späte Einführen von Software zur Unterstützung.

4.3.2 Spezielles Beispiel für Relationenkomplexität im Kontext der Produktentstehung

Das vorangegangene Beispiel demonstriert zwar die Möglichkeiten der Modellierung von Artefakten in einem Entwicklungsprojekt, stellt jedoch aufgrund seiner Einfachheit nicht die Probleme heraus, die sich aus der Modellierung von Prozessartefakten im Rahmen eines komplexen Projekts ergeben. Hierzu soll daher ein näher an die Realität von komplexen Fahrzeugentwicklungsprojekten angelehntes Beispiel dienen. Es zeigt, wie die (impliziten) Beziehungen zwischen den verschiedenen Artefakten zu Mehraufwand führen. Wie bereits in Abschnitt 4.1 angekündigt, werden Artefakte nun explizit durch eine `TypeWriter-Schriftart` gekennzeichnet.

Die Entwicklung von Fahrzeugen entspricht im Grundsatz der Entwicklung beliebiger anderer Produkte. Ausgehend von Vorgaben und Ideen aus der Marktforschung, Gesetzen und der Portfolioplanung werden zunächst die groben Rahmenbedingungen eines Fahrzeugs abgesteckt, die sogenannten konzeptbestimmenden Maße. In diesem Rahmen werden Anforderungen gesammelt, die im Laufe einer Phase der Definition immer konkreter werden, bis schließlich das Fahrzeug beschrieben ist und Lastenhefte von Bauteilen vorliegen, die von Zulieferern umgesetzt werden können [Sch12]. Hierbei wird versucht, möglichst viele Tätigkeiten zur Alternativen- und Konzeptsuche in die frühe Phase der Entwicklung zu legen. Dieses Vorgehen wird gemeinhin „Frontloading“ genannt. Erste Prototypen werden gebaut und erprobt, Werkzeuge gefertigt und schließlich, mit dem Ende der Entwicklungsphase, der Serienanlauf vorbereitet. Begleitet von umfangreichen Qualitätssicherungs- und Marketing-Maßnahmen startet die Produktion, wonach dann der Verkauf beginnt.

Der zur Entwicklung benötigte Zeitraum kann mehrere Jahre betragen. Im Verlauf des Projekts wird eine große Zahl von Artefakten verwendet, die insbesondere in der frühen Phase der Fahrzeugentwicklung – während des Frontloadings – oftmals nicht formal erfasst werden. Diese frühe Phase ist gekennzeichnet von der Aufgabe der Verdichtung, Filterung und Konkretisierung von Anforderungen, die aus vielen verschiedenen Informationsströmen zusammengeführt werden müssen: Wettbewerbs-Analysen, Kosten-Ziele, Design-Wünsche, Eigenschaften des Vorgänger-Modells, verschiedene Konzepte, Forschungsergebnisse, etc. [BS11]. Auch wenn diese Informationen aus Systemen mit einem hinterlegten Modell entstammen, so macht doch ihre stark heterogene Natur es schwierig, ihre Konsolidierung in nur einem Werkzeug oder Software-System abzubilden. Es wird allerdings deutlich, dass insbesondere Anforderungen eine wichtige Rolle spielen, da sie zu einem frühen Zeitpunkt bereits mit diversen anderen Artefakten verknüpft werden. Während der Projektlaufzeit stellen sie zentrale Punkte dar, von denen ausgehend sich Änderungen an einem Fahrzeugprojekt durch die Artefaktwelt bewegen.

Das in diesem Abschnitt konstruierte Beispiel zeigt anhand fiktiver Artefakte in einem Fahrzeugentwicklungsprojekt diese komplexen Zusammenhänge auf. Es bezieht sich hierbei auf die Einführung einer neuen Variante eines Fahrzeugs für einen spezifischen Markt, z.B. eine spezielle Low-Cost-Form, die günstigere Materialien verwendet und optisch differenziert auftritt, technisch jedoch größtenteils identisch ist.

Zu Beginn werden demnach eine Reihe von `Anforderungen` entworfen, die das Vorhaben beschreiben. Eingeordnet und verknüpft werden sie in und mit den Anforderungen des übergeordneten `Fahrzeugprojekts`. Ein `Marktdossier` wird erstellt, welches die genauen Gegebenheiten des Markts untersucht und den Erfolg der Variante prognostiziert. Es dient gleich-

zeitig vielfach als Begründung für die in den Anforderungen getroffenen Entscheidungen. Für die veränderte Ausführung müssen technische Lösungen gefunden werden, die in der mit dem Hauptfahrzeugprojekt gemeinsamen Technikliste erfasst werden. Diese ist einem Tabellenkalkulationsprogramm realisiert und muss nun um eine zusätzliche Spalte ergänzt werden, anhand derer die Varianten unterschieden werden können. Weiterhin ist eine gleichzeitige Bearbeitung schwierig, da viele verschiedene Mitarbeiter aus verschiedenen Bereichen darauf zugreifen. Daher werden oftmals lokale Kopien erzeugt und diese in aufwändigen Sitzungen miteinander abgeglichen, um einen gemeinsamen Stand zu erzeugen. Die Konstruktion der einzelnen Teile findet wiederum auf Basis der Anforderungen statt und endet schließlich in verschiedenen Lastenheften, die an unterschiedliche Zulieferer gehen. Die Teile füllen schließlich eine weitere Stückliste, in die auch Teile aus dem ursprünglichen Fahrzeug übernommen werden.

Zur Kalkulation in der Phase vor der Produktion und zur späteren Abrechnung der neuen Variante in der Serienfertigung muss diese in den Abrechnungsprozess übernommen werden. Die Mitarbeiter des Controllings bereiten bereits während der Projektlaufzeit das Fahrzeug im Kalkulationssystem auf, indem die Stückliste, sowie Informationen hinsichtlich der Bezugsarten der einzelnen Teile, für Hausteile Fertigungszeiten und für Kaufteile Preise geladen werden. Dann wird eine spezielle Kalkulationskonfiguration für die neue Fahrzeugvariante in das System eingegeben. Die automatischen und regelmäßigen Kalkulationsläufe müssen somit einen weiteren Rechnungslauf durchführen, wodurch weitere CPU-Zeit auf dem Mainframe freigegeben werden muss. Die Kalkulationsergebnisse werden in einer Datenbank abgelegt, deren Speicherkapazität nun erhöht werden muss. Hierzu sind weitere physische Festplatten notwendig, die in den Datenbankcluster eingebunden werden.

Das Beispiel zeigt die verschiedenen „Schichten“ eines Unternehmens, durch die sich die Auswirkungen einer Anforderung bewegen. Einige davon betroffene Artefakte sind produktbezogen und daher eventuell in einem Anforderungsmanagementwerkzeug mit Modellierungsmöglichkeiten erfasst, wie z.B. die Technikliste. Bei anderen führt die infinitesimale Erhöhung der Leistung zu Mehrkosten, die überproportional hoch sind. Dies ist bei der Erhöhung der Datenspeicherkapazität der Fall, da eine physische Datenbankerweiterung nicht nur im Verhältnis des neu benötigten Speichers stattfindet. Ob oder wann ein solches Limit erreicht wird, ist schwer im Vorfeld einer Entscheidung zu approximieren. Auch ist dieses Beispiel immer noch reduziert, da weitaus mehr Prozesse und Systeme betroffen sind, als hier aufgeführt werden. Allerdings finden sich bereits an dieser Stelle 22 Artefakte, die teilweise untereinander stark verknüpft sind.

Die Modellierung der Artefaktlandschaft eines Unternehmens ist dabei heute beschränkt auf bestimmte fachliche Areale. So existieren Bebauungspläne, welche die Systeme eines Unternehmens erfassen und darstellen. Teilweise sind diese sogar mit den Prozessen der Serienfertigung in Beziehung gesetzt, aber losgelöst von denen der Fahrzeugentwicklung, da nur eine mittelbare, aber keine unmittelbare Beziehung zwischen ihnen besteht. Diese beispielhafte Aufzählung kann beliebig fortgesetzt werden. Diese Arbeit entwickelt im weiteren Verlauf eine Möglichkeit zur durchgängigen Erfassung aller Artefakte, um sie in einem Werkzeug für das Anforderungsmanagement von Fahrzeugentwicklungsprojekten nutzbar zu machen.

Kapitel 5

Semantisch unterstütztes Anforderungsmanagement als Lösungskonzept

Hatten die ersten Kapitel dieser Arbeit eine Erläuterung der Problemstellung zum Thema, konzentrieren sich die folgenden auf die Konzeption einer Lösung. Kapitel 3 zeigte, dass Unternehmen aus einer Vielzahl verschiedenartiger Elemente bestehen, die miteinander verknüpft sind. In Kapitel 4 wurde für diese Elemente der Begriff „Artefakte“ geprägt und anhand von Beispielen gezeigt, wie sich Komplexität, Veränderung und Artefakte zueinander verhalten. Insbesondere Anforderungen, die an Fahrzeugprojekte gestellt werden, die sich aus einem modularen Fahrzeugbaukasten bedienen, können Auswirkungen auf eine Vielzahl von Artefakten haben, welche diese Änderungen wiederum an andere Artefakte weiter geben.

In diesem Kapitel wird nun darauf aufbauend ein durchgängiges Anforderungsmanagement konzipiert, welches Modelle solcher Artefakte verarbeitet. Es bietet automatische Auswirkungsanalysen in den Artefaktmodellen und stellt somit die Lösung der Problemstellung dieser Arbeit dar. Das Konzept wird in Abschnitt 5.2 durch Anforderungen beschrieben, welche die Funktionsweise und Einsatzmöglichkeiten beschreiben. Abschnitt 5.5 zeigt den Nutzen, den es im Anforderungsmanagement und in angeschlossenen Disziplinen hat. Kapitel 6 geht dann auf die technische Umsetzung ein und erprobt das Werkzeug anhand spezifischer Problemstellungen. Hierbei wird geprüft, ob ein solches Werkzeug die gestellten Anforderungen erfüllen kann.

5.1 Semantische Netze zur Abbildung formalisierter Artefakte

Antoniou und van Harmelen sprechen in ihrem Grundlagenbuch nicht von den semantischen Netzen, sondern im Singular von dem semantischen Netz, welches eine Ergänzung des heutigen Internets darstellt. Der Begriff des semantischen Netzes kann demnach präzisiert werden als ein Kernkonzept: maschinenverarbeitbare, verknüpfte Informationen [AH08]. Ziel ist es, eine Trennung zwischen Daten und deren Repräsentation für Anwender zu erhalten, wobei es der interpretierenden Software ermöglicht wird, eine korrekte Repräsentation der Daten basierend auf einem Metamodell zu erzeugen.

Das folgende Beispiel zeigt den Unterschied zwischen einer rein textuellen Repräsentation von Daten und einer durch eine Software generierte: Es soll ein einfacher Sachverhalt einem Benutzer

angezeigt werden: *Der Volkswagen Golf basiert auf dem Modularen Querbaukasten (MQB).*

1. Die einfachste Form der Repräsentation kann durch reinen Text erfolgen. Hierzu liest die Software diesen Text etwa aus einer Datei oder einer Datenbank und gibt ihn so direkt dem Benutzer wieder. Statische Webseiten sind auf diese Weise konzipiert. Zwar ist das Ziel schnell erreicht, eine bestimmte Information wiederzugeben, jedoch ist die Pflege aufwändig: Eventuell ändert sich der Sachverhalt oder eine andere Information wird stattdessen benötigt. In beiden Fällen muss der Text angepasst werden. Eine automatisierte Abfrage der Information durch Software ist nur mit großem Aufwand möglich, etwa durch den Einsatz automatischer Texterkennungssoftware.

2. Eine andere Möglichkeit der Anzeige von Daten wird durch die Abfrage einer Datenbank gegeben. Eventuell existiert eine Tabelle, die alle Fahrzeuge enthält, die auf dem Modularen Querbaukasten aufbauen. Eine Abfrage kann alle oder auch nur bestimmte Einträge zurückgeben. Dies erleichtert die Pflege, indem der Inhalt der Tabelle angepasst werden kann. Eine andere Software ist allerdings nicht in der Lage auf die Tabellendaten zurückzugreifen, sofern dies nicht explizit durch den Betreiber erlaubt wird, weswegen eine automatisierte Abfrage ebenfalls nicht möglich ist. Falls doch, so müssen der Aufbau der Tabelle explizit bekannt sein und entsprechende Datenbankabfragen durch die fremde Software für diese spezifische Datenbank konstruiert werden. Das Einbinden anderer Datenbanken oder das Ändern des Datenbankschemas führt ebenfalls wieder zu hohem Aufwand.

3. In einem semantischen Netz werden Informationen als Relationen dargestellt, wodurch Aussagen hinsichtlich bestimmter Sachverhalte getroffen werden können. Eine Relation ist ein Tripel und beinhaltet ein Subjekt, ein Objekt und eine sie verbindende Eigenschaft. Zur Anzeige der obigen Aussage muss demnach ein Tripel existieren, welches das Subjekt „Volkswagen Golf“ mit dem Objekt „Modularer Querbaukasten“ durch die Eigenschaft „basiert auf“ verbindet. Dieses Tripel kann durch eine Anwendung angezeigt werden, indem bspw. alle Tripel abgefragt werden, die dem Muster „beinhaltet Eigenschaft“ „`basiert auf`“ und hat Objekt „`Modularer Querbaukasten`“ genügen. Die Daten selbst sind in einer Ontologie gemäß Definition 1.4.1 gespeichert, welche eine textuelle Repräsentation hat, auf der Abfragen ausgeführt werden. Diese textuelle Repräsentation steht auch anderen Anwendungen zur Verfügung und befolgt gewisse Konventionen in ihrer Struktur, wodurch sie universell verständlich ist und eine Interpretation durch Software ermöglicht.

Diese Arbeit verwendet das in Punkt 3 angerissene Konzept. Ziel ist, die im vorherigen Kapitel eingeführten Artefakte durch den Einsatz eines semantischen Netzes maschineninterpretierbar zu machen und auf diese Weise ein Anforderungsmanagementwerkzeug in die Lage zu versetzen, die Artefakte eines Unternehmens bei der Analyse von Auswirkungen einer Entscheidung zu berücksichtigen. Anforderungen stellen hierbei selbst Artefakte im Sinne von Definition 4.1.1 dar. Artefakte werden hierzu im Rahmen einer Ontologie beschrieben und durch Relationen miteinander verknüpft. Es ergibt sich auf diese Weise ein Metamodell der Artefakte im Unternehmen, welches von Werkzeugen verwendet und mit konkreten Instanzen gefüllt werden kann. Die Instanzen bilden somit das konkrete Modell, welches sich allerdings in derselben Ontologie wie das Metamodell befindet. Abschnitt 6.1 geht im folgenden Kapitel näher auf die technischen

Grundlagen einer Ontologie ein, während dieses Kapitel die Möglichkeiten aufzeigt, die sich aus der Verwendung semantischer Netze ergeben. Wichtig ist an dieser Stelle, dass der W3C-Standard OWL zur Abbildung der Ontologie verwendet wird [W3C].

5.2 Anforderungen an ein semantisch unterstütztes Anforderungsmanagementwerkzeug

Der Ansatz zur Lösung der Problemstellung der Arbeit findet sich in der Kombination von semantischen Netzen und den Methoden des Anforderungsmanagements. Daher werden nun Anforderungen an ein semantisch unterstütztes Anforderungsmanagementwerkzeug definiert. Ein solches Werkzeug ermöglicht es dem Anforderungsmanagement, über die klassisch mit dieser Disziplin assoziierten Funktionen hinaus hilfreich zu sein.

Die Applikation muss hierbei vier grundsätzliche Funktionen bieten:

1. Modellierung von Artefakten und deren Zusammenhänge.
2. Erstellung und Verwaltung von Instanzen der Artefakte.
3. Suchen und Finden von Informationen.
4. Automatisierbare Auswertung und Analyse der vorhandenen Informationen.

Aus diesen Funktionen ergibt sich eine Menge von Anforderungen, die von einem Softwarewerkzeug erfüllt werden müssen. Diese werden auf einem hohen Abstraktionslevel beschrieben. Tabelle 5.1 stellt alle Anforderungen übersichtlich als Referenz dar.

5.2.1 Modellierung, Instanzerstellung und -verwaltung

Die operative Arbeit in der Anforderungserhebung als Teil des Anforderungsmanagements gliedert sich laut Hood et. al. in zwei Phasen [HWFP08]: Die Definition des Umfangs und die Definition der eigentlichen Anforderungen.

Während für die Feststellung des Projektumfangs vor allem vorhandene Informationen – wie etwa Schnittstellen oder Stakeholder – verknüpft werden, ist die eigentliche Erhebung der Anforderungen vom Erstellen konkreter Objekte geprägt. Hierbei werden vor allem Anforderungen erhoben, genau spezifiziert und analysiert, bevor sie zueinander in Beziehung gesetzt werden [HWFP08]. Eine Kernaufgabe eines Anforderungsmanagementwerkzeugs ist daher die Bereitstellung von umfangreichen und komfortablen Editierfunktionen für Instanzen von Anforderungen und anderen Artefakten. Neben dem Anlegen neuer Objekte auf der Basis von Klassen muss auch das Verknüpfen mit vorhandenen Objekten unterstützt werden. Ebenfalls muss es möglich sein, neue Artefakte in das Netz aufzunehmen, wobei diese Art der Bearbeitung des Metamodells restriktiv gehandhabt werden können muss, um einen Qualitätsverlust zu vermeiden.

Umfangreich im Sinne von Funktionalität bedeutet, dass das Editieren von zahlreichen Assistenzfunktionen unterstützt wird. Ziel ist, bereits bei der Eingabe von Informationen etwa für Anforderungen Hilfestellungen bei Formulierungen und Begriffsverwendungen zu geben.

Nr.	Bezeichnung
1	Verwendung eines Metamodells.
2	Eingabe von Eigenschaften bei der Erstellung von Instanzen.
3	Unterstützung des Benutzers bei der Textbearbeitung unter Verwendung der Informationen des semantischen Netzes.
4	Plausibilisierung von Eingaben.
5	Qualitätssicherung von Anforderungen.
6	Zusammenstellen und Konfigurieren von Listen und Dokumenten aus Artefaktinformationen.
7	Wiederverwendung von Informationen.
8	Festlegen eines Umfangs, innerhalb dessen gesucht wird.
9	Die Freitextsuche akzeptiert als Eingabe einen beliebigen Text.
10	Es kann gezielt nach Artefakten, Instanzen von Artefakten, Eigenschaften, Werten oder auch Relationen gesucht werden.
11	Navigation innerhalb von Suchergebnissen.
12	Unterstützung des Benutzers bei der Suche durch Verwendung der Inhalte des semantischen Netzes.
13	Graphische Navigation der Suchergebnisse.
14	Konstruktion von Suchabfragen.
15	Automatisches Schlussfolgern.
16	Verwendung von Suchergebnissen als Eingabemenge.
17	Durchführung von Impact-Analysen.

Tabelle 5.1: Anforderungen an ein semantisch unterstütztes Anforderungsmanagementwerkzeug

„Komfortabel“ bedeutet, dass das Einfügen oder Editieren von Informationen einfach und angenehm möglich sein soll. Die Anzahl an notwendigen Schritten bis hin zur Editiermaske ist zu minimieren und die Bedienbarkeit des Editors soll intuitiv sein. Intuitiv meint an dieser Stelle, dass ein webanwendungs-affiner Anwender in der Lage ist, ohne vorhergehendes Lesen von Einführungen oder Durchlaufen von Schulungen den Editor mit seinem vollen Funktionsumfang zu nutzen.

1. Anforderung: Verwendung eines Metamodells. Das Werkzeug verwendet ein Metamodell, welches eine Ontologie im Ontology Web Language (OWL)-Standard Be- und Verarbeiten kann [W3C]. Es bietet die Möglichkeit, das Metamodell durch Hinzufügen, Verändern und Entfernen von Artefakten und Relationen zu bearbeiten. Hierbei gilt die Ausdrucksmächtigkeit von OWL als Beschränkung in der Modellierung. Ebenfalls können Instanzen von Artefakten und Relationen hinzugefügt, verändert und entfernt werden. Der Benutzer ordnet so die Instanzen in einen konkreten Kontext, bspw. innerhalb eines Projekts ein, indem Relationen wie `wird_detailliert_durch` verwendet werden. Durch den Einsatz von OWL werden verschiedene Konzepte in der Modellierung ermöglicht, wie bspw. Vererbung von Artefakten, Vererbung von Relationen oder auch das Hinzufügen von Annotationen zu Relationen.

2. Anforderung: Eingabe von Eigenschaften bei der Erstellung von Instanzen. Die Eingabemaske für das Erstellen bzw. Editieren von Instanzen von Artefakten stellt für einzelne Eigenschaften des typ-bestimmenden Artefakts Eingabefelder bereit. Im Werkzeug kann festgelegt werden, welcher Typ zu welchen Eigenschaften gehört. Relationen – wie etwa eine Vererbungsbeziehung – werden durch Verweise dargestellt. Es muss möglich sein, einem Verweis zu folgen, um das dort hinterlegte Objekt zu erstellen bzw. zu bearbeiten, ohne die bereits eingegebenen Werte im Ursprungsobjekt zu verlieren. Diese Kette der Bearbeitung setzt sich rekursiv fort.

3. Anforderung: Unterstützung des Benutzers bei der Textbearbeitung unter Verwendung der Informationen des semantischen Netzes. Assistenzfunktionen für die Eingabe von Text existieren heute bereits viele. Für neue Technologien wie bspw. mobile Geräte sind neue Eingabeunterstützungen notwendig geworden, um das Produkt am Markt erfolgreich zu etablieren. Unter Verwendung des semantischen Netzes im Hintergrund können diese Funktionen noch zielführender eingesetzt werden. Auch sind sie nicht beschränkt auf sog. Rich-Text-Editoren, also mehrzeiligen Eingabefeldern mit Möglichkeiten zur Formatierung des eingegebenen Texts. Ebenfalls angewendet werden können sie bei Feldern für einzelne Werte wie bspw. Auswahlen oder Zahlen. Automatisches Vervollständigen von Wörtern ist bereits von Mobiltelefonen bekannt, aber auch von integrierten Entwicklungsumgebungen (IDEs). Hierbei wird durch den Editor versucht, das vom Benutzer gewünschte Wort auf Basis der bereits eingegebenen Buchstaben zu komplettieren. Das semantische Netz enthält eine große Anzahl teils fachspezifischer Wörter, bspw. aus den Benennungen von Artefakten, die als Wörterbuch für das automatische Vervollständigen dienen können. Durch automatisches Vervollständigen wird eine Rechtschreibprüfung ergänzt, jedoch keinesfalls ersetzt. Das Projekt iglos des Instituts für Verkehrssicherheit und Automatisierungstechnik der TU Braunschweig erarbeitet bspw. technische Terminologien, welche als Grundlage für Funktionen wie das automatische Vervollständigen dienen können [Insa].

4. Anforderung: Plausibilisierung von Eingaben. Eingaben müssen überprüft und plausibilisiert werden. Wichtig ist, in einer Eingabe identifizierte Begriffe einem bekannten Begriff zuzuordnen. Das sog. „Named-Entity-Recognition“ ist eine Methode zur automatisierten Erkennung von Begriffen, auch bspw. über Synonyme [CS99, CN02]. Auf diese Weise wird sichergestellt, dass der Benutzer einheitliche und korrekte Begriffe verwendet. Unbekannte Zuordnungen in Aussagen können mit Vorschlägen zur Korrektur versehen werden. Durch die Zuordnung findet eine semantische Anreicherung der Informationen statt. Dies unterstützt einerseits automatische Analysen und andererseits den Benutzer etwa beim Suchen oder Navigieren durch die Informationen.

5. Anforderung: Qualitätssicherung von Anforderungen. Für die Erstellung oder Bearbeitung von Anforderungen findet eine zweite Überprüfungsstufe statt. Hierbei wird die Qualität der Anforderung anhand der IEEE-Vorgaben evaluiert, welche in Abschnitt 2.1 dargestellt sind. Ergebnis ist eine Einstufung in eine Qualitätsstufe „Grün“, „Gelb“ oder „Rot“. Auf diese Weise erhält der Benutzer in vereinfachter Art und Weise Feedback und kann seine Eingabe ggf. überarbeiten. Hierzu ist es notwendig, die Gründe, die zu einer Einstufung geführt haben, auch mitteilen zu können.

6. Anforderung: Zusammenstellen und Konfigurieren von Listen und Dokumenten aus Artefaktinformationen. Das Erstellen von Lastenheften und anderen Dokumenten ist in Abschnitt 2.2 als Aufgabe des Anforderungsmanagements aufgeführt. Da das Werkzeug über die Informationen von vielen verschiedenen Artefakten verfügt, muss es in der Lage sein, Listen und Dokumente in unterschiedlichen Ausprägungen mithilfe eines Konfigurators zu erzeugen. So können Lastenhefte generiert werden, indem alle Anforderungen aufgelistet werden. Ihre Titel dienen als Strukturelement und die Beschreibungen der einzelnen Anforderungen als Inhalt des Lastenhefts. Dieses kann um weitere Daten ergänzt werden, indem bspw. Realisierungsinformationen von Teilen oder Modulen extrahiert und dem Dokument hinzugefügt werden. So können Artefakte und ihre Eigenschaften als Bausteine für komplexe Dokumente und Berichte dienen.

7. Anforderung: Wiederverwendung von Informationen. Das Werkzeug verfügt über eine wachsende Menge an verknüpften Informationen. Durch die Modellierung von Artefakten werden Produkte und deren Anforderungen, Projekte, Prozesse und Bibliotheken eingebunden, aber auch Informationen über Systeme und Organisationsstrukturen. Zur Vermeidung von Redundanz muss der Benutzer Elemente wiederverwenden können. Dies ist möglich, indem Kopien bspw. von konkreten Anforderungen durchgeführt werden können. Der Benutzer kann dann entscheiden, ob verknüpfte Elemente ebenfalls kopiert oder nur referenziert werden. Die Auswahl geschieht durch eine graphische Oberfläche, in der die Verknüpfungen der Artefaktinstanz dargestellt sind und markiert werden kann, wie mit ihnen zu verfahren ist. Das System muss ebenfalls prüfen, ob das Element, welches durch einen Benutzer angelegt wird eventuell schon im System vorhanden ist und ggf. darauf hinweisen. Analog zu Anforderung 4 kann festgestellt werden, ob etwa eine ähnliche Anforderung existiert und diese dem Benutzer zur Kopie vorgeschlagen werden.

5.2.2 Suchen und Analysen

Die im vorgehenden Abschnitt aufgeführten Anforderungen 1 bis 7 dienen zur Beschreibung der Funktionen 1 und 2 eines Werkzeugs für ein durchgängiges Anforderungsmanagement. In diesem Abschnitt werden die Funktionen 3 und 4 betrachtet und hierfür die Anforderungen 8 bis 13 definiert. Die Zusammenfassung resultiert aus einer Ähnlichkeit der zugrundliegenden Funktionalität – in beiden Fällen können Anfragen semantisch unterstützt an das Werkzeug gerichtet werden, die allerdings in ihrer Komplexität und ihren Resultaten unterschiedlich sind. Es existieren hierzu zwei Such-Modi: Freitextsuche und die Konstruktion von Abfragen.

8. Anforderung: Festlegen eines Umfangs, innerhalb dessen gesucht wird. Es ist möglich, die Suche einzuschränken. Hierzu können ein oder mehrere Ausgangspunkte im semantischen Netz definiert werden, von dem aus die Suche gestartet wird. Ein Ausgangspunkt ist ein Artefakt wie bspw. ein Projekt. Der Suchalgorithmus betrachtet dann nur Elemente, die sich innerhalb einer vom Benutzer angegebenen Anzahl von Verknüpfungen befinden. Diese Anzahl kann definiert werden, indem ein Endpunkt festgelegt wird, der wiederum ein Artefakt darstellt. Die Anzahl an Verknüpfungen, die zwischen beiden Artefakten liegen, definieren den Suchradius. Auf diese Weise ist es möglich, bspw. nur innerhalb eines Projekts zu suchen.

9. Anforderung: Die Freitextsuche akzeptiert als Eingabe einen beliebigen Text. Der Suchalgorithmus versucht dann, Elemente im semantischen Netz aufgrund der in ihnen enthaltenen Informationen zu finden. Entspricht ein in die Suche eingegebenes Wort bspw. der Bezeichnung einer Relation, so wird diese Relation in einer Übersicht zurückgegeben. Artefakte werden genauso wie ihre Instanzen durchsucht. Abbildung 5.2 zeigt, wie aus dem semantischen Netz das Wissen extrahiert und in einer Übersicht zusammengestellt wird.

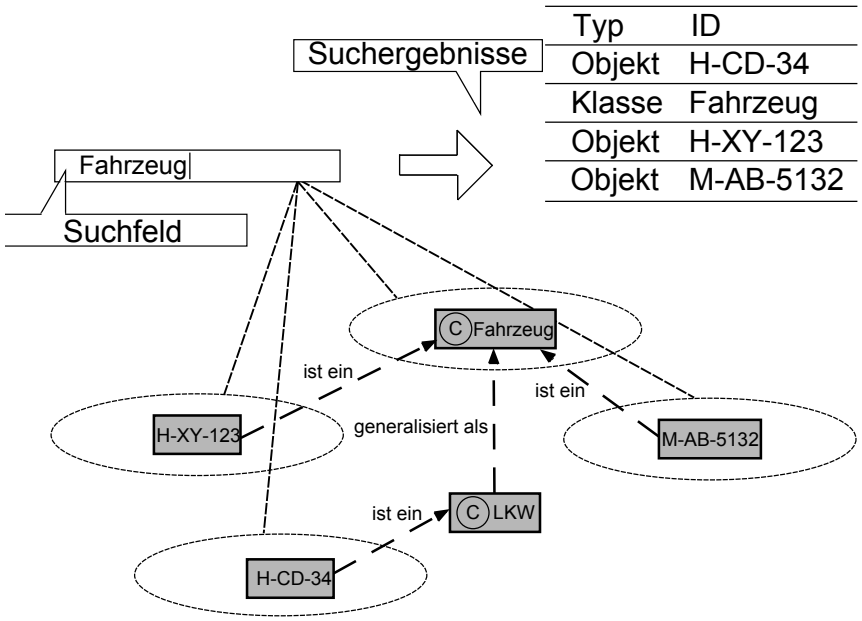


Abbildung 5.2: Suchen in Klassen und Objekten

10. Anforderung: Es kann gezielt nach Artefakten, Instanzen von Artefakten, Eigenschaften, Werten oder auch Relationen gesucht werden. Alle Elemente des semantischen Netzes und deren Eigenschaften sind demnach durchsuchbar.

11. Anforderung: Navigation innerhalb von Suchergebnissen. Die von einer durchgeführten Suche erfassten Elemente werden in einer Liste angezeigt. Von dort aus kann auf die einzelnen Einträge zugegriffen werden, Übersichtsseiten zeigen einzelne Klassen oder Objekte im Detail. Es werden die einzelnen Informationen aus dem hinterlegten Eintrag im semantischen Netz abgerufen, wie etwa die Beschreibung, welche Relationen vorhanden sind, etc. Aus der Ansicht eines Artefakts kann auf Instanzen dieses Artefakts zugegriffen werden und andersherum. Analog zu Anforderung 2 können neue Artefakte, Instanzen oder Relationen angelegt werden, während durch die Suchergebnisse navigiert wird, ohne dass die Navigation unterbrochen wird.

12. Anforderung: Unterstützung des Benutzers bei der Suche durch Verwendung der Inhalte des semantischen Netzes. Analog zu Anforderung 3 wird der Benutzer bei der Eingabe von

Schlagwörtern für die Suche unterstützt, indem Synonyme und korrigierte Begriffe angezeigt werden. Auch auf andere Weise verwandte Begriffe, wie etwa durch Generalisierungs- oder Detaillierungsbeziehungen, werden dem Benutzer angezeigt.

13. Anforderung: Graphische Navigation der Suchergebnisse. Die Ergebnisse von Suchen können schnell komplex werden. Um dem Benutzer die Navigation durch und das weitere Arbeiten mit den Ergebnissen zu erleichtern, wird eine graphische Darstellung der Ergebnisse und ihrer Zusammenhänge genereriert, die der Benutzer ebenfalls zur Navigation verwenden kann.

Während der erste Suchmodus in seiner Verwendung bereits heute bspw. durch diverse Internet-Suchmaschinen allgemein bekannt ist, so stellt der zweite eine andere Form der Informationsbeschaffung dar. Das Stellen von konkreten Fragen, die durch das Werkzeug beantwortet werden, unterscheidet sich stark von der bloßen Anzeige von Informationen, bei denen eine textliche Übereinstimmung vorliegt. Es können Elemente gesucht werden, die vom Benutzer definierten Kriterien entsprechen oder bestimmte Eigenschaften aufweisen. Ebenfalls möglich ist das Überprüfen logischer Aussagen hinsichtlich ihres Wahrheitsgehalts. Der Vorteil des Einsatzes eines semantischen Netzes wird an dieser Stelle besonders deutlich, da mit dessen Hilfe Software selbstständig Schlussfolgerungen durchführen kann. Das automatische Schlussfolgern wird auch als „Reasoning“ bezeichnet, die technischen Hintergründe werden detailliert in Abschnitt 6.1.1 besprochen. Der Abschnitt 6.4 des folgenden Kapitels demonstriert, wie umfangreich Abfragen an das semantische Netz sein können und welche Informationen auf diese Weise gefunden werden.

14. Anforderung: Konstruktion von Suchabfragen. Der zweite Suchmodus besteht aus der Konstruktion von spezifischen Fragen an das semantische Netz. Im Sinne eines „Werzeugkastens“ können einzelne Elemente zu einer komplexen Abfrage zusammengestellt werden, indem bspw. bestimmte Werte für Eigenschaften eines Artefakts bei der Suche angenommen oder spezielle Konfigurationen von Relationen gesucht werden. Das Werkzeug unterstützt hierbei den Fragesteller durch automatisches Vervollständigen der Eingaben und deren Korrektur, führt die Abfrage aus und zeigt schließlich die Ergebnisse an.

15. Anforderung: Automatisches Schlussfolgern. Bei der Ausführung von Abfragen zieht das Werkzeug selbsttätig logische Schlüsse im Rahmen der durch den OWL-Standard gegebenen Möglichkeiten. Ist eine Relation bspw. transitiver Natur, wird dies bei der Generierung der Ergebnisse berücksichtigt.

16. Anforderung: Verwendung von Suchergebnissen als Eingabemenge. Suchergebnisse aus einer vorhergehenden Suche – unabhängig vom Modus – können wiederum als Eingabemenge für eine neue Suchanfrage genutzt werden. Auf diese Weise kann der Benutzer selbstständig den Umfang der Suche steuern.

17. Anforderung: Durchführung von Impact-Analysen. Die sog. Impact-Analyse, also das Abschätzen der Auswirkungen einer geplanten Änderung an einer Stelle im semantischen Netz bildet eine zentrale Komponente des Anforderungsmanagements. Sie stellt eine dem Suchen verwandte Funktionalität dar, besteht sie schließlich auch aus der Konstruktion von

Abfragen. Unter Verwendung des semantischen Netzes wird hiermit eine Lösung für die zu Beginn dieser Arbeit formulierten Problematik dargestellt: Es werden Einflüsse nicht nur von Anforderungen auf Teile und auf Tests sichtbar gemacht, sondern die Einflüsse von Artefakten auf andere, die durch spezifische Relationen miteinander gekennzeichnet sind. Welche Relationen welchen Einfluss haben, wird durch das semantische Netz vorgegeben. Die Anforderungsmanagementsoftware interpretiert diese Zusammenhänge und kann so bspw. herausfinden, dass die Änderung einer Anforderung über mehrere Artefakte hinweg zur Änderung eines ganzen Prozesses führt. Entsprechend dieser Methodik ist das Werkzeug nicht nur für Projekt- und Anforderungsmanager ein hilfreiches Werkzeug, sondern auch für Entwickler, die etwa die Auswirkungen einer Änderung auf andere Teile eines Produkt abschätzen wollen. Die Impact-Analyse ist somit nicht zielgerichtet, sondern eine Art Breitensuche mithilfe bestimmter Kriterien.

Die Konstruktion von Fragen aus Relationen und Artefakten bzw. Instanzen ist ein wirkungsvolles Werkzeug und erlaubt sehr exakte Analysen. Fragestellungen können gespeichert werden, um bspw. regelmäßige Analysen und Berichte auf ihrer Basis zu generieren. Eine wichtige Analyse ist der Reifegrad eines Projekts, welcher misst, inwieweit ein zu entwickelndes Produkt bereits fertig gestellt wurde. Hierzu können verschiedene Kennzahlen verwendet werden, wie bspw. die Abdeckung aller Anforderungen mit technischen Lösungen oder mit Teilen, die einen speziellen Status haben. Die Durchgängigkeit und Einheitlichkeit wird insofern deutlich, dass der Projektstatus auf der Basis des ursprünglichen Lastenhefts laufend erfasst wird und so keine Brüche in der Medien-Kette von Lastenheft-Anforderung-Umsetzung-Analyse existieren.

Da der Begriff des Reifegrads im weiteren Verlauf der Arbeit mehrfach verwendet wird, gilt hierfür von nun an folgende Definition:

Definition 5.2.1 (Reifegrad) *Der Reifegrad eines Entwicklungsprojekts gibt anhand definierter Kennzahlen den Umsetzungsstatus aller Anforderungen an das zu entwickelnde Produkt an.*

5.3 Nutzen des semantischen Netzes im operativen Anforderungsmanagement

Zentrale Aufgabe des Anforderungsmanagements ist nicht die Erstellung von Produktdaten, vielmehr liegt der Fokus auf der Verwaltung, Pflege und Verteilung der erhobenen Anforderungen sowie die Herstellung und Pflege der Traceability. Das Setzen der Anforderungen in den Kontext anderer Disziplinen und das Herstellen und Pflegen von Verbindungen zu diesen ist essentiell, bildet das Anforderungsmanagement doch Schnittstellen zu vielen. Bereits in Abschnitt 2.2 wurden die einzelnen Tätigkeiten dargestellt und werden daher nun in derselben Reihenfolge in den Kontext eines semantisch unterstützten Werkzeugs gestellt. Hierdurch wird der Nutzen ersichtlich, der sich durch den Einsatz des beschriebenen Werkzeugs in den verschiedenen Disziplinen ergibt.

- **Projektmanagement:** Grundsätzlich kann das Projektmanagement stärker auf die Umsetzung eines zentralen Lastenhefts ausgerichtet werden. Dieses bildet eine hohe Aggregationsebene, die ein Steuern eines großen Projekts erleichtert. Durch die Verknüpfung aller untergeordneten Aufgaben, Sub-Anforderungen und anderer Artefakte bleibt trotzdem die Möglichkeit bestehen, detaillierte Informationen abzurufen und Zusammensetzung sowie Status des Reifegrads des Projekts nachzuvollziehen. Die Planung und Kontrolle von Aufgaben wird unterstützt, indem an zentraler Stelle der Umsetzungsstatus sowie die zugeordneten Ressourcen und betroffenen Artefakte betrachtet werden können. Durch automatisiertes Schlussfolgern können dynamische Analysen erarbeitet und zur Entscheidungsunterstützung verwendet werden.

- **Implementierung:** Implementierung ist begrifflich in der Softwareentwicklung einzuordnen. Nichtsdestotrotz findet der Vorgang der Umsetzung der Anforderungen in ein konkretes Ergebnis analog in der Konstruktion von Teilen in der Entwicklung eines gesamten Fahrzeugs statt. Unterstützt wird dieser durch die einfache Verfügbarkeit der Anforderungen auf verschiedenen Ebenen. Durch die Durchgängigkeit werden Begründungen nachvollziehbar, was zu einer verbesserten Interpretation der Anforderungen durch einen Entwickler führt. Werkzeuge für das Produktdatenmanagement sowie für die Konstruktion von Teilen oder Software können in das semantische Netz integriert werden und füllen es stetig mit Informationen. Den Entwicklern stehen somit Verwendungsnachweise hinsichtlich des Einsatzes eines Teils in einem anderen Produkt zur Verfügung, wodurch auch Beschränkungen von Änderungen an Teilen oder Modulen kommuniziert werden können. Weitere Kontextinformationen und Begrifflichkeiten werden durch die Verknüpfung der Artefakte einfach abrufbar.

- **Testen und Qualitätssicherung:** Essentiell für das Testen ist die Verfügbarkeit von Informationen hinsichtlich der Testobjekte. Unter Verwendung der Informationen aus dem semantischen Netz können Tests konstruiert werden, die den Kontext einer Anforderung oder eines Testobjekts berücksichtigen, bspw. für welche physikalischen Umgebungen ein Teil in einem anderen Produkt konzipiert werden muss. Ebenfalls ist im Rahmen der Impact-Analyse ersichtlich, welche Tests von einer Änderung betroffen sind und entsprechend neu ausgeführt, verändert oder entfernt werden müssen. Das automatisierte Generieren von Berichten hinsichtlich des Status der Tests funktioniert analog zum Punkt „Projektmanagement“.

- **Marketing und Vertrieb:** Diese Bereiche sind darauf angewiesen, ständig über den aktuellen Planungsstand eines Produkts informiert zu sein. Die Planung von Werbestrategien beruht darauf, die Eigenschaften und Funktionen eines Fahrzeugs früh zu kennen und Änderungen daran in die Strategie einfließen zu lassen [KMU11]. Durch das semantische Netz bietet sich die Chance, auch Artefakte aus dem Bereich Marketing und Vertrieb zu erfassen und im Rahmen bspw. der Änderungsanalyse zu berücksichtigen. So führt eine Werbekampagne, die nicht optimal alle Aspekte eines Produkts berücksichtigt zu geringeren Verkaufszahlen, eine Änderung der Kampagne zieht wiederum Kosten nach sich. Die Bereiche Marketing und Vertrieb stehen daher ebenfalls unter dem Einfluss von Komplexitätseffekten. Klude et al. identifizieren in ihrer Arbeit im Rahmen einer Prozessanalyse bei

verschiedenen Unternehmen, welche Anforderungen sie an eine durchgängige Lösung für ein Anforderungsmanagement haben [KMU11]. Es wird deutlich, das die Verknüpfung von Informationen über den Lebenszyklus eines Produkts eine wichtige Rolle spielt. Wichtig sind vor allem – aber nicht ausschließlich – die „Zuordnung von Lösungskonzepten zu den Anforderungen“ sowie die „Verknüpfung von Anforderungen mit Elementen der Produktstruktur“ [KMU11]. Diese Punkte werden durch das vorgestellte Modell größtenteils abgedeckt und bieten dem Produktmarketing die Möglichkeit, relativ früh im Produktentstehungsprozess Informationen und Anforderungen kontrolliert einsteuern und deren Umsetzung nachverfolgen zu können.

- **Änderungsmanagement:** Insbesondere das Änderungsmanagement profitiert von der Verfügbarkeit eines durchgängigen Anforderungsmanagements. Durch das vereinfachte Abschätzen der Auswirkungen von Änderungen und das Setzen dieser Auswirkungen in den Kontext eines Nutzens können Entscheidungen schneller und fundierter getroffen werden. Durch die Einbeziehung von Prozessartefakten in die Analyse werden erstmals auch indirekte Auswirkungen sichtbar gemacht.

- **Verträge und andere Dokumente:** Im Kontext der zunehmenden Vernetzung im Fahrzeug als auch der Fahrzeuge mit der Umwelt und untereinander sowie die Verfügbarkeit von stark in das Fahrverhalten eingreifenden Fahrerassistenzsystemen werden die genaue Einhaltung von juristischen Vorgaben sowie die Nachweisbarkeit hiervon im Rechtsfall immer bedeutender. Die Verknüpfung von Prozessartefakten vereinfacht die Nachvollziehbarkeit von Entscheidungsprozessen, bzw. ermöglicht diese Nachvollziehbarkeit überhaupt erst. Es kann einwandfrei und innerhalb kürzester Zeit dargestellt werden, welche Anforderungen erfasst und auf welche Art sie im Produkt umgesetzt wurden – inkl. der Tests zur Überprüfung. Offensichtlich steht und fällt diese Methodik mit der Qualität und Quantität der im Laufe des Projekts erhobenen Anforderungen.

  Neben akuten Rechtsfällen müssen diverse weitere Regularien beachtet und diese auch nachgewiesen werden. Das Zusammentragen der hierzu notwendigen Informationen ist zeitaufwändig, darf aber nicht fehlerhaft sein, da Zulassungen von Produkten von der Korrektheit und Vollständigkeit der erstellten Dokumente abhängen. Die Vorgaben der zu erstellenden Dokumente und einzureichenden Anträgen stellen ihrerseits Anforderungen dar und müssen dementsprechend im Anforderungsmanagementwerkzeug umgesetzt werden. So können danach Listen der notwendigen Informationen generiert werden. Sie werden bei Verfügbarkeit automatisch zusammengestellt und auch bei Bedarf formatiert. Falls notwendig kann auch für diese spezielle Teilmenge an Anforderungen ein Reifegrad ermittelt werden, der den Status der Komplettierung von Anträgen, Formularen und anderen Dokumenten widerspiegelt.

5.4 Beispiele zur Modellierung von Artefakten in Entwicklungsprojekten

Nachdem der Nutzen für das operative Anforderungsmanagement im vorhergehenden Abschnitt abstrakt demonstriert worden ist, werden nun anhand eines Beispiels die Möglichkeiten des durchgängigen Anforderungsmanagements anschaulich gemacht.

Entwicklungsprojekte – gleich welcher Domäne – sind heutzutage gekennzeichnet von der Erstellung diverser heterogener Arbeitsergebnisse, die alle mehr oder weniger stark zum Gelingen des Projekterfolgs beitragen. Sie dienen zur Organisation und Absicherung des Projektes und der dabei erstellten Arbeitsergebnisse oder stellen selbst Arbeitsergebnisse dar. Assessmentmodelle wie etwa CMMI definieren eine ganze Reihe von Artefakten, die im Laufe des Projekts erstellt werden müssen, bspw. Stakeholder-Analysen, Risikolisten, Tailoring-Dokumente und dergleichen mehr [CMM10]. Anforderungen sind Teil dieser Modelle. Allerdings müssen diese Artefakte über den Verlauf des Projekts hinweg konsistent gehalten werden: auf dem ursprünglichen Tailoring-Dokument basieren beispielsweise ein Projektplan und die genannte Stakeholder-Analyse. Beide sind wiederum mit Anforderungen verknüpft: die Stakeholder erzeugen Anforderungen, aus denen wiederum Aufgaben im Projektplan entstehen. Sind diese `Dokumente` einzelne und nicht automatisiert verknüpfte Artefakte, für die kein zentrales Werkzeug existiert, muss die Konsistenz manuell sichergestellt werden: entweder werden projektweit eindeutige Bezeichnungen für alle Artefakte geschaffen, was bedeutet, dass jeder Stakeholder eindeutig identifizierbar ist, genauso wie jede Anforderung und jede Aufgabe. Alternativ werden redundante Daten gepflegt, etwa indem die Stakeholder in einer eigenen Datei und im Anforderungsmanagementwerkzeug abgelegt werden. Muss nun aber bspw. ein Kommunikationsplan erstellt werden, wird schnell unübersichtlich, was genau wo gepflegt wird.

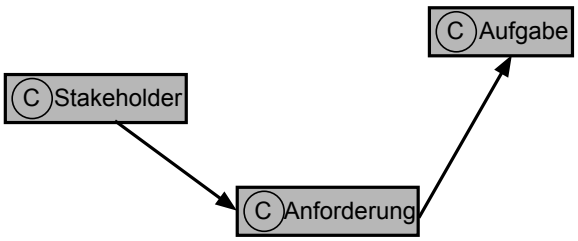


Abbildung 5.3: Ein simples Modell mit Projektartefakten

Heutige Anforderungsmanagementwerkzeuge bieten allerdings bereits diverse Möglichkeiten zur Modellierung von Artefakten im Kontext eines Projekts. Entweder stellt ein Werkzeug hierbei eine feste Menge an Typen zur Verfügung, von denen Instanzen erstellt werden können, oder es bietet ein frei konfigurierbares Typsystem. Die betrachteten Artefakte `Stakeholder`, `Anforderung` und `Aufgabe` können demnach problemlos modelliert und ihre konkreten Ausprägungen dann im Werkzeug verwendet werden. Abbildung 5.3 zeigt beispielhaft eine Repräsentation des Modells als Graph. Die Beziehungen können gerichtet zwischen den Artefakten hergestellt werden, indem bspw. die an einer Anforderung beteiligten `Stakeholder` mit dieser verknüpft werden. Durch die Richtung stehen für spätere Analysen einfache semantische Informa-

tionen zur Verfügung. Wird ein freies Typsystem verwendet, können die erzeugten Typen danach in beliebigen weiteren Projekten genutzt werden, wodurch der Aufwand in diesen reduziert wird. Im aktuell betrachteten Projekt entfällt bereits das Pflegen diverser Listen, da das Modell die Möglichkeit bietet, gewünschte Listen auch komplexerer Natur aus den Zusammenhängen zu erzeugen, bspw. welche `Arbeitspakete` indirekt welche `Stakeholder` betreffen, da sie aus „ihren“ Anforderungen resultierten.

Der Grad der Komplexität ist derselbe wie vor der Modellierung, da sich weder an der Elementenoch an der Relationen-Komplexität etwas ändert. Allerdings ist die Verwaltung der Artefakte und ihrer Beziehungen nun nicht mehr in der Verantwortung des Menschen, sondern eines Softwarewerkzeugs. In diesem können nun Instanzen der modellierten Artefakte erstellt werden, wie Abbildung 5.4 zeigt.

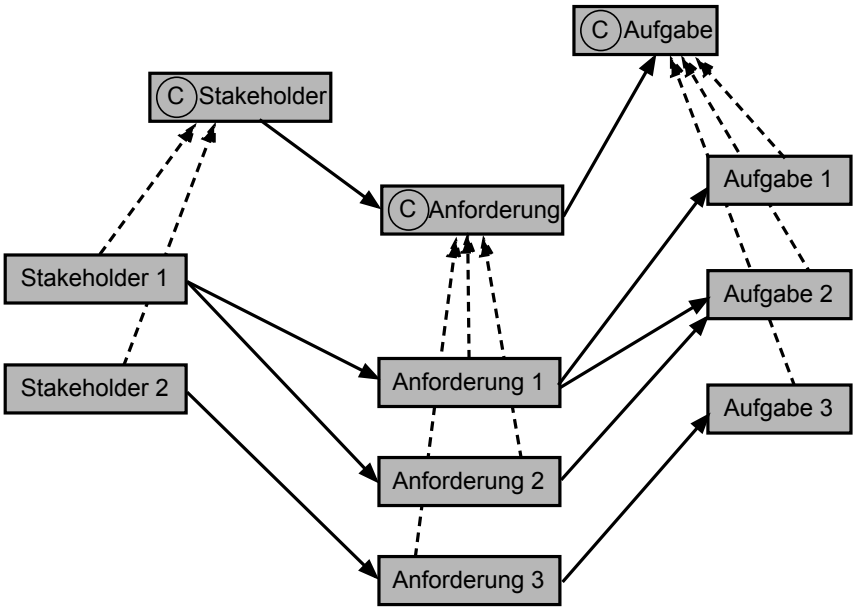


Abbildung 5.4: Ein simples Modell mit Instanzen von Projektartefakten

Gestrichelte Linien stellen Instantiierungen der Artefakt-Klassen dar, die Relationen zwischen den Objekten sind entsprechend beliebig der Modellierung hinzuzufügen. Bereits an dieser Stelle wird deutlich, dass zwar durch die Modellierung grundsätzlich eine Aufnahme der komplexen Verknüpfungen zwischen Artefakten auf einer beliebigen Ebene erfolgen kann. Jedoch ist die Aussagekraft des Modells begrenzt, da keine Informationen über den Typ der Relation vorliegen. Bspw. könnte eine Relation zwischen `Stakeholder` und `Anforderung` vom Typ „unterstützt“ und eine andere vom Typ „unterstützt nicht“. In einem semantischen Netz wird dies durch Typisierung von Relationen gelöst, die einer Relation eine Bedeutung in einem bestimmten Kontext zuweisen. Dies ermöglicht die umfassenden Analysemöglichkeiten des durchgängigen Anforderungsmanagements.

Deutlich wird der Vorteil der Verwendung eines semantischen Netzes, wenn Artefakte betrachtet werden, die nicht in Modellen wie CMMI abgebildet sind. Das Beispiel aus Kapitel

4.3.1 betrachtet das zentrale Artefakt „Provisorische Bauteilliste (PBL)“, welches einen der evolutionären Entwicklung des Unternehmens geschuldeten stark spezfischen Aufbau besitzt. Das Dokument findet sich in keiner traditionellen Vorgehensweise des Anforderungsmanagements, da es nur von dieser einen Firma für eine spezielle Aufgabe verwendet wird. Es passt auch nicht in das Muster eines klassischen Anforderungsdokuments wie etwa einem Lastenheft, da sich dort Bauteile in verschiedenen Konkretisierungsstufen miteinander vermischen. Gleichermaßen stellt es keine Stückliste dar, da nicht unbedingt jedem Teil eine Teilenummer zugeordnet werden kann, da sich manche beispielsweise noch in der Entwicklung befinden. Es muss daher vollständig neu modelliert werden, ohne dabei die Informationen über das Zusammenspiel der einzelnen Inhalte zu verlieren. Das semantische Netz als Grundlage des durchgängigen Anforderungsmanagements ermöglicht das Erstellen und Integrieren eines solchen Modells, zur anschließenden Verwendung in Abfragen, Analysen, etc.

5.5 Nutzen des semantischen Netzes im erweiterten Kontext des Anforderungsmanagements

Die Einführung einer durchgängigen Beschreibung von Artefakten durch ein semantisches Netz eröffnet diverse weitergehende Möglichkeiten, die über das reine Verwalten von Anforderungen für Fahrzeugprojekte hinausgehen. Im vorherigen Abschnitt 5.3 wurde bereits analysiert, inwiefern ein durchgängiges und werkzeugunterstütztes Anforderungsmanagement die gem. der Literatur originären Aufgaben des Anforderungsmanagements unterstützt. Klute et al. zeigen in ihrer Arbeit bereits an einem Prototyp, inwieweit die Möglichkeiten eines semantischen Netzes das Anforderungsmanagement unterstützen können [KKR11]. Das gesammelte Wissen über Artefakte und deren Zusammenhänge lässt allerdings eine Unterstützung von weitaus mehr Disziplinen zu, die bisher allenfalls im Kontext der sie anbindenden Schnittstellen betrachtet wurden. Darüber hinaus werden vorausschauende Planungen möglich, die bisher nur unter hohem manuellen Aufwand durchgeführt werden konnten. Vor allem unter Einbeziehung der Fahrzeugbaukästen, welche durch ihre hohe Relationenkomplexität nur schwierig in Softwaresystemen eingebunden werden können, ergeben sich interessante Perspektiven.

5.5.1 Erweitertes Anforderungsmanagement

Das korrekte Formulieren von Anforderungen ist seit geraumer Zeit Gegenstand intensiver Forschung [Rup11]. Es zeigt sich, dass die Qualität der Anforderungen etwa durch die durchgängige Verwendung von klar definierten Begriffen und Bezeichnungen gesteigert werden kann. Im Sinne der aus Softwareentwicklungsumgebungen (IDEs) oder auch Mobiltelefonen bekannten automatischen Vervollständigung von Wörtern kann eine auf dem semantischen Netz basierende Hilfestellung bei der Erarbeitung von Anforderungen helfen. Insbesondere in großen Unternehmen ist die Problematik der Verwendung gleicher Begriffe für unterschiedliche Konzepte bekannt und ein zunehmender Störfaktor in der digitalen und natürlichsprachlichen Kommunikation. Wird dem Benutzer eine Auswahl der verfügbaren Begriffe auf Basis des Kontext angeboten, steigt der Informationsgehalt der Anforderung. Durch automatisches Hervorheben der Begriffe

in Anforderungsbeschreibungen kann nach Erstellung durch die Nutzer der Anforderungen die korrekte Definition eingesehen werden.

Bereits in der medizinischen Forschung steht Software zur Verfügung, die in der Lage ist, basierend auf wissenschaftlichen Ausarbeitungen und einem vorhandenen semantischem Netz automatisiert Hinweise auf mögliche Zusammenhänge von Konzepten zu geben [BGG+07]. Mithilfe solcher Konzepte wird es ermöglicht, Anforderungen hinsichtlich ihrer Inhalte zu prüfen und Hinweise auf eventuelle gegenseitige Beeinflussung automatisiert zu generieren.

5.5.2 Gemeinkosten

Als mitunter ursächlich für steigende Gemeinkosten wurde in Kapitel 4 bereits die steigende Komplexität nicht nur in Produkten, sondern vor allem in der Verknüpfung von Artefakten identifiziert. Zur zunehmenden Komplexität kommt es aufgrund fehlender Möglichkeiten zur vorausschauenden Analyse von Entscheidungen, welche betroffene Elemente im Unternehmen aufzeigt. Die im Vorfeld nicht berücksichtigte Zunahme der organisatorischen Kosten als Auswirkungen einer geplanten Änderung bspw. an einem Produkt oder einem Modul als Teil eines Fahrzeugbaukastens können demnach die erwarteten Erlöse übersteigen, bzw. die approximierten Kosten übertreffen. Zwar stehen Werkzeuge zur Prozessmodellierung bereits zur Verfügung, jedoch sind diese erstens nicht für Entscheidungsunterstützung im Kontext der Produktentwicklung vorgesehen und haben zweitens ihren Fokus vor allem auf der Modellierung von Prozessen und nicht deren Artefakte.

Das in dieser Arbeit beschriebene Werkzeug löst zunächst das Problem der Verfügbarkeit einer universellen Datenbasis mit Modellen von Artefakten sowie deren Zusammenhänge und kombiniert diese mit typisierten Relationen und Metainformationen. Durch Anfragen an das semantische Netz kann erarbeitet werden, wie eine Entscheidung auf die Komplexität des Netzes wirkt.

In Abschnitt 4.2.1 sind zwei unterschiedliche Arten der Beeinflussung der Komplexität eingeführt worden: Typ 1 betrachtet die Änderungen von Artefaktinstanzen, also konkreten Ausprägungen von Artefakten. Eine Analyse der Verknüpfungen des betroffenen Artefakts kann schnell Auskunft darüber geben, welche weiteren Artefakte durch die Änderung beeinflusst werden. Hierbei ist es wichtig, die Art der Verknüpfung zu kennen und entscheiden zu können, ob eine Verknüpfung auch tatsächlich eine Veränderung in einem anderen Artefakt nach sich zieht.

Typ 2 wiederum legt den Fokus auf Änderung oder Einführung von Artefakten, wodurch die Elemente- und Relationenkomplexität betroffen ist. Es findet demnach eine strukturelle Änderung des semantischen Netzes statt. Hierbei kann der Entscheider mit Informationen über den Kontext versorgt werden, in den das neue Artefakt eingebettet wird: eventuell wird es an „Knotenpunkte“ angeschlossen, auf die bereits über viele Verbindungen eingewirkt wird und deren Artefaktinstanzen somit oft verändert werden. Ein weiteres Artefakt erhöht damit die Last, die auf den bereits vorhandenen liegt. Neben der Analyse der Komplexität kann ebenfalls festgestellt werden, ob ein ähnliches Artefakt bereits vorhanden ist. Durch unscharfe Suchen können Strukturen erfasst werden, die ähnlich der neuen oder veränderten sind. Eventuell wird damit die Strukturveränderung des Netzes obsolet.

Über das Anforderungsmanagement hinaus gehend können auf diese Weise auch besonders häufig betroffene Artefakte identifiziert und somit als Grundlage für Optimierungen verwendet

werden. Auch eine systematische Reduktion der Relationenkomplexität bestimmter Artefakte gemäß ihres Verwendungsgrades ist denkbar.

Anforderungsmanagement steht ganz am Anfang der Prozesskette der Produktentwicklung und begleitet diese fortwährend. Ist bisher die Rolle des Anforderungsmanagements eher in der Dokumentation von Entscheidungen und Strukturen zu sehen, so findet durch die Einführung einer durchgängigen semantischen Unterstützung eine Erweiterung zur Entscheidungsunterstützung statt. Entscheidern wird ein Werkzeug an die Hand gegeben, im Vorfeld die zwei Arten der Auswirkung auf die Komplexität des Unternehmens evaluieren zu können. Ein konsequenter Einsatz führt demnach nicht notwendigerweise zu einer Reduktion der Komplexität, aber zu einer Möglichkeit der Steuerung der Veränderung derselben.

5.5.3 Produktdefinition und Eigenschaftsplanung

Die Planung von Produkteigenschaften ist ein essentieller Punkt in der Entstehung von Fahrzeugen. An dieser Stelle wird der sog. Eigenschaftskatalog aufgebaut, der für den gesamten Verlauf der Entstehung die Maßstäbe des Produkts vorgibt [Rin06].

Die Planung der Eigenschaften ist bestimmt von unterschiedlichen Eingaben hinsichtlich Positionierung und Klasse des Produkts. Wichtig ist bereits an dieser Stelle die Verwendung von Eigenschaften, die durch Elemente eines Fahrzeugbaukastens beeinflusst werden, bspw. ein bestimmter Radstand. Die Berücksichtigung aller dieser Eigenschaften stellt ein komplexes Problem dar, einerseits aufgrund der hohen Anzahl an Eigenschaften, die durch Module eines Baukastens beeinflusst werden, andererseits aufgrund der hohen Anzahl an Relationen zwischen diesen. Zusätzlich schließen sich Eigenschaften gegenseitig aus oder bedingen sich.

Ein durchgängiges Anforderungsmanagement, in dem das Wissen über Abhängigkeiten automatisch zur Verfügung steht, hilft bei der Ausplanung der Eigenschaften basierend auf der Wissensbasis. Es kann bspw. softwaregestützt abgeschätzt werden, welche kostenrelevanten Auswirkungen eine Abweichung in einer Eigenschaft von der Vorgabe eines Baukastens hat. Ebenfalls möglich wird eine Priorisierung [KKR11].

Weiterhin möglich ist das Einbeziehen von Informationen über die Bedeutung und Bewertung bestimmter Eigenschaften für Kunden. Das Feedback von Kunden, etwa durch Magazine im automobilen Kontext, kann direkt bei der Ausplanung von Eigenschaften berücksichtigt werden. Im Gegenzug ist die exakte Positionierung eines neuen Produkts im gewünschten Kundenmilieu durch Setzen bestimmter Eigenschaften einfacher. Durch die Verknüpfung der Bedeutung der Eigenschaften mit den Bedürfnissen eine Kundengruppe entsteht die Möglichkeit, automatisiert Vorschläge für die Ausprägungen zu erhalten. Im Sinne des folgenden Änderungsmanagements kann sichergestellt werden, dass vorgeschlagene Änderungen keine Verschiebung aus dem angepeilten Kundenmilieu bewirken.

5.5.4 Änderungsmanagement

Das Abschätzen der technischen und finanziellen Auswirkungen einer vorgeschlagenen Änderung ist zentrale Aufgabe des Änderungsmanagements. Das durch das semantische Netz des Anforderungsmanagements mögliche maschinelle Verstehen von Zusammenhängen ermöglicht es Software, besagte Änderungen automatisiert aufzuzeigen.

In einer einfachen Form besteht das Ergebnis einer Analyse der Auswirkungen in einer Liste von betroffenen Teilen bzw. Modulen. Eine erweiterte Software kann die indirekt betroffenen Prozessartefakte mit einbeziehen, wie es bereits im obigen Abschnitt 5.5.2 beschrieben wurde. Das Änderungsmanagement ist demnach stark mit den Gemeinkosten verbunden, entsprechend besteht hier eine starke Möglichkeit der Beeinflussung der indirekten Kosten. Neben der Möglichkeit der Betrachtung der Komplexität besteht für die operative Arbeit des Änderungsmanagements ein großer Vorteil in der Möglichkeit der Einbeziehung verschiedener Elemente des semantischen Netzes und eventueller automatisierter Kostenbetrachtungen.

Das Änderungsmanagement ist hierbei insbesondere auf eine hohe Aussagekraft des Netzes angewiesen, was einen hohen Füllstand einerseits an Artefakten und andererseits an Daten bedingt. Es ist davon auszugehen, dass der Grad an Akzeptanz direkt mit der Verlässlichkeit der Ergebnisse von Abfragen korreliert.

5.5.5 Baukastenplanung und -management

Idealerweise sind Baukästen ihrem Inhalt nach konstant über ihren Lebenszyklus. Diese Konstanz ist allerdings nur möglich, wenn eine umfangreiche vorausschauende Planung der Inhalte erfolgt, basierend auf den Bedürfnissen von Fahrzeugprojekten, die noch nicht die aktive Entwicklungsphase begonnen haben, bzw. sich erst in der Planung befinden. Eine solche Planung bedingt das Vorhandensein umfangreicher Informationen über die zukünftigen Fahrzeugprojekte, um nicht eine laufende Anpassung nötig zu machen. Primär ist zwar die konzeptionelle Festlegung gewisser Maße wie bspw. der Radstand wichtig, allerdings spielen auch Funktionen in den Fahrzeugen eine immer größere Rolle. Die vorausschauende Planung von Baukasteninhalten steht demnach vor denselben Problemen wie die Planung von Fahrzeugen. Allerdings potenzieren sich hierbei die relevanten Informationen, da ein Fahrzeugbaukasten die Grundlage für eine ganze Reihe von Fahrzeugen über viele Jahre in vielen unterschiedlichen Märkten ist. Entsprechend sind Fehler etwa in der Berücksichtigung von gesetzlichen Anforderungen eines bestimmten Marktes ungleich gravierender, da die Auswirkungen einer Korrektur signifikant mehr Fahrzeuge betreffen, als eine Korrektur innerhalb einer Fahrzeugpalette.

Änderungen in einem Fahrzeugbaukasten unterliegen demnach der gleichen Problematik wie bereits im vorhergehenden Abschnitt ausgeführt, nur ist die Elemente- und Relationenkomplexität um ein Vielfaches höher. Umso wichtiger ist bereits für die Erfassung der betroffenen Fahrzeuge und Teile zur Ermittlung der Investitions- und Einzelkosten eine Unterstützung durch geeignete Software. Das auch nur näherungsweise Abschätzen der Veränderung der Gemeinkosten ist ohne Software faktisch unmöglich, erhält aber durch die Vielzahl an betroffenen Artefakten im Unternehmen eine sehr hohe Relevanz.

Problematisch mit herkömmlichen Ansätzen zu vereinbaren ist die Tatsache, dass von Änderungen in einem Fahrzeugbaukasten Fahrzeugentstehungsprojekte in unterschiedlichen Entwicklungsstadien betroffen sind. Fahrzeugentstehungsprojekte sind in der frühen Phase von einer fehlenden durchgängigen Formalisierung der Informationen geprägt. Dies ist kritisch bei Änderungen, die Baukastenmodule betreffen, die in diesen Fahrzeugen eingesetzt werden sollen, da diese auf formale Weise, bspw. in Stücklisten, verfügbar sind. Der in dieser Arbeit vorgestellte Ansatz setzt dagegen auf die Erfassung informeller Informationen ausgehend von einer hohen Ebene, was einfacher zu bewerkstelligen ist als die durchgängige detaillierte Beschreibung aller

Artefakte. Artefakte wie bspw. erzeugte Listen oder kleinere Prozesse stellen oftmals informelle Informationen dar, die nicht in traditionellen Prozessbeschreibungswerkzeugen erfasst werden. Durch die Aufnahme in das semantische Netz füllt sich dieses zunehmend. Zwar ist das Anforderungsmanagement bereits mit wenigen Informationen effektiv, die Aussagekraft bleibt aber nichtsdestotrotz abhängig vom Füllungsgrad des Netzes.

5.5.6 Generierung von juristisch relevanten Dokumenten

Vertragliche Vereinbarungen mit Fremdfirmen, aber teilweise auch zwischen einzelnen Organisationseinheiten innerhalb eines Unternehmens, erfolgen mithilfe juristischer Dokumente. Diese müssen eindeutig formuliert und inhaltlich korrekt sein, um ihrem Zweck des exakten Ausdrucks eines juristischen Sachverhalts gerecht zu werden. Nur so stellen sie für Auftraggeber und -nehmer jeweils ein probates Mittel zum Verständnis eines etwaigen Auftrags und zur Anwendung von Konsequenzen bei Nichterfüllung dar.

Die adäquate Formulierung von Rechtstexten ist dementsprechend wichtig. Aus mangelhaft formulierten Lastenheften entstehen Unternehmen hohe finanzielle Belastungen durch nicht oder unpassend erbrachte Leistungen und der Nicht-Anwendbarkeit von Konsequenzen. Eine exakte Beschreibung eines zu konstruierenden Gegenstands oder einer zu entwickelnden Software ist mit freiem Text aufwändig und fehleranfällig. Dementsprechend können das Anforderungsmanagement und die in dieser Disziplin vorhandenen Methoden eine wichtige Rolle in der Erstellung und Kontrolle von juristischen Dokumenten spielen. Eine Auftragsbeschreibung in Form eines Lastenheftes ist demnach nur eine Sammlung von Anforderungen, eingebettet in einen umgebenden Formtext, der an die jeweiligen Gegebenheiten angepasst werden muss. Diese einzelnen Bausteine können – genauso wie bei der Formulierung von Anforderungen für interne Projekte – im Werkzeug erarbeitet und dann generiert werden um bspw. einen Vertrag zu erstellen.

In einem nächsten Schritt kann auf die Erstellung eines durchgängig formulierten Schriftsatzes fast vollständig verzichtet werden. Hierzu wird das zu unterzeichnende Rahmenvertragswerk vom eigentlichen Inhalt getrennt. Letzterer wird digital signiert als Datensatz zur Verfügung gestellt. Es existiert hierzu bereits das *Requirements Interchange Format (ReqIF)* [Obj], welches es ermöglicht, Anforderungen strukturiert in einem standardisierten Format zu übermitteln.

Kapitel 6

Grundlagen der Umsetzung eines semantisch unterstützten Anforderungsmanagementwerkzeugs

Im vorherigen Kapitel wurde ein durchgängiges und semantisch unterstütztes Anforderungsmanagement konzipiert sowie der Nutzen für die einzelnen Aufgabenbereiche des Anforderungsmanagements im engeren und im weiteren Sinne beschrieben. Es basiert auf Modellen von Unternehmensartefakten und ihren Verknüpfungen im Rahmen eines semantischen Netzes. Dieses Kapitel beschreibt nun die Realisierung des semantischen Netzes, damit es in einem Anforderungsmanagementwerkzeug verwendet werden kann. Hierzu findet zuerst eine Einführung in die technische Umsetzung von semantischen Netzen mit Hilfe des Resource Description Framework (RDF) und sowie der Ontology Web Language (OWL) statt. Auch die technische Realisierung von Software wird vorgestellt, welche die erstellten Netze verarbeitet. Danach folgt der Aufbau von zwei grundlegenden Modellen zur Beschreibung von Unternehmensartefakten, um für ein unternehmensweites Netz eine eindeutige Verwendung von Begriffen anbieten zu können. Mit der Einführung einer Methodik für den Umgang mit Artefaktnetzen steht dann eine Verfahrensweise für das Erstellen und die Verwendung eines semantischen Netzes für Unternehmensartefakte zur Verfügung. Nach der Klärung der technischen Grundlagen, der Definition einer einheitlichen Begriffswelt und dem semantischen Netz als Ausgangsbasis sowie der Festlegung einer Methodik, wird eine exemplarische Artefaktlandschaft für die Produktentwicklung erstellt. Diese demonstriert die Umsetzbarkeit der erarbeiteten Anforderungen aus Kapitel 5 und den Nutzen für die erwähnten Disziplinen des Anforderungsmanagements. Zuletzt werden auf die Herausforderungen und eventuelle Lösungen eingegangen, Ergebnisse und Erkenntnisse vorgestellt und ein Fazit gezogen.

6.1 Entwicklung der Wissensbasis als Grundlage für das Anforderungsmanagement

Grundlage eines semantisch unterstützten Anforderungsmanagements ist eine Wissensbasis, in der Artefakte und ihre Zusammenhänge formal beschrieben und maschinenlesbar verfügbar sind. Technisch gelöst wird dies durch den Einsatz einer Ontologie gemäß Definition 1.4.1. Diese kann an einer geeigneten Stelle im Unternehmen als Unternehmenswissensbasis zur Verfügung stehen und bietet somit eine zentrale und ständig aktuelle Plattform für semantische Netze, welche

dann von einem Anforderungsmanagementwerkzeug und anderen Systemen genutzt werden können. Somit steht jedem Werkzeug, welches die Wissensbasis verwendet, dieselbe Menge an Informationen mit einheitlicher Struktur und demselben Stand zur Verfügung. Abbildung 6.1 zeigt schematisch die Verwendung einer gemeinsamen Wissensbasis durch zwei verschiedene Systeme.

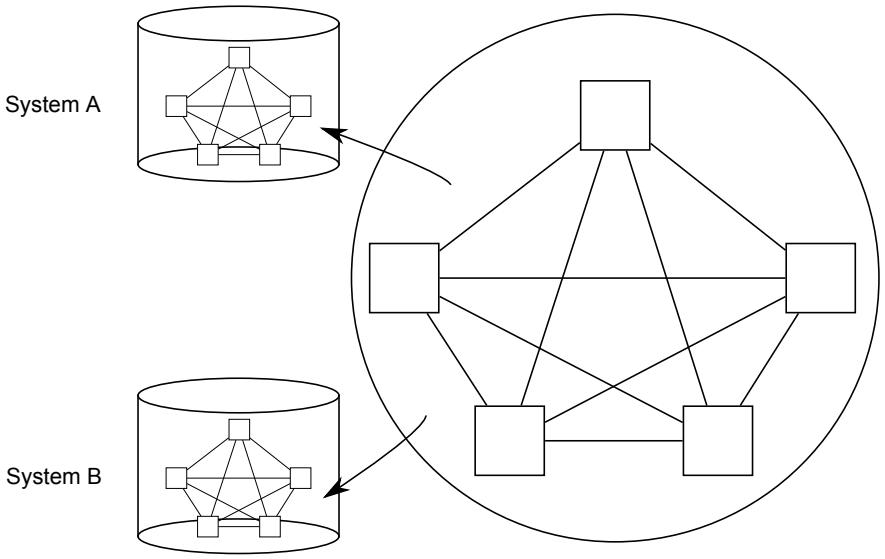


Abbildung 6.1: Zugriff von unterschiedlichen Systemen auf eine zentrale Wissensbasis

Eine Wissenbasis stellt nicht nur Daten zur Verfügung, sondern auch die Informationen über ihren Aufbau und ihre Bedeutung, daher die Verwendung des Begriffs „Wissens“ im Sinne von Kapitel 3.3.3. Softwaretechnisch gesehen befinden sich demnach Daten und Modell in einem System. Der eingesetzte Standard definiert die Syntax der Informationen und gibt so Auskunft über Verwendung und Interpretation. Hierdurch können unterschiedliche Systeme sämtliche Artefakte in einen Kontext einordnen, selbst wenn sie nur ausgesuchte verwenden.

Eine gängige Sprache zur Modellierung von semantischen Netzen als Ontologien ist die Sprache OWL [W3C]. Zwar muss die Menge an redundanter Notation durch das zugrunde liegende XML akzeptiert werden, sie ist jedoch als Standard vorgeschlagen [W3C] und daher einer eigenen Domänen-spezifischen Sprache und Notation vorzuziehen. Desweiteren stehen bereits Werkzeuge zur Verfügung, mit deren Hilfe Modelle in OWL entwickelt werden können, wie bspw. Protegè [Sta] oder OntoWiki [Lei]. Diese sind allerdings absolut rudimentär und nicht zu einer Modellierung im vorgeschlagenen Stil geeignet. Sie zeigen jedoch, welchen Funktionsumfang semantische Netze heute bereits bieten. Zur Verwendung der in OWL formulierten Ontologien in einer Software bietet sich z.B. die Jena Ontology API an [Apab]. Darüber hinaus steht die vereinfachte Notation „Turtle“ im Notation3 (N3)-Format zur Verfügung, welche eine einfacher lesbare Alternative zur XML-Notation darstellt [BLC, RDFa].

6.1.1 Modellbasierte Systementwicklung mit RDF, RDFS, OWL und SPARQL

Softwareentwicklung ist zunehmend bestimmt durch das Denken in Modellen und das Generieren von ganzen Programmteilen aus formalen Beschreibungen. Mit der Unified Modeling Language (UML) steht erstmals eine einheitliche Sprache zur Verfügung, anhand derer Systeme aus verschiedenen Perspektiven beschrieben werden können [Obj11, Rum11, Rum12]. Zur Modellierung von Daten- gegenüber Anwendungsmodellen als Grundlage von Systemen, kann ebenfalls die UML verwendet werden. Tatsächlich bieten sich aber die Sprache RDF [RDFb] und alle in diesem Kontext befindlichen Erweiterungen an, da sie ausschließlich die für semantische Technologien notwendigen Sprachumfänge enthalten. Eine Repräsentation zur graphischen Darstellung und Einbindung in weitergehende Systemplanungen in UML-Klassendiagrammen ist ohne großen Aufwand möglich.

Hintergrund von semantischen Technologien ist das Anreichern von Informationen mit kontextbezogenen Daten. Software wird somit in die Lage versetzt, Begriffe mit einer Bedeutung zu versehen, je nachdem wie sich der Kontext verhält. Es stehen diverse Möglichkeiten zur Modellierung zur Verfügung, um Inhalte auf diese Weise maschinenlesbar zu machen. Hierzu existieren im Rahmen des „Semantic Web“ eine Reihe von standardisierten Modellierungssprachen, die oftmals über eine Repräsentation in XML verfügen. Dies führt zwar zu einer hohen Redundanz an verwendeten Sprachelementen, aber gleichzeitig zu einer eingeschränkt durch Menschen lesbaren Form sowie einem großen Spektrum an vorhandenen Werkzeugen zur automatisierten Verarbeitung von in der jeweiligen Sprache erstellten Dokumenten. Aufgrund der Sprachredundanzen wurden weiterhin formale Repräsentationen in sog. Tripeln entwickelt, allgemein N3-Notation genannt. Hierbei steht der reine Dateninhalt einzelner Tripel (Subjekt, Prädikat, Objekt) im Vordergrund, was die Lesbarkeit deutlich erhöht.

Eine weit verbreitete und standardisierte Möglichkeit zur Modellierung der Beziehungen von Ressourcen ist die Sprache RDF, welche unter anderem eine Repräsentation in XML und in der N3-Notation „Turtle“ besitzt. Konzeptionell besteht sie aus Ressourcen, Eigenschaften und Aussagen, welche Tripel der ersten beiden Elemente sind. Alle verwendeten Elemente der Sprache müssen durch eine URI (Uniform Resource Identifier) definiert werden. RDF verwendet vor den eigenen Tags das `rdf:`-Präfix [W3C].

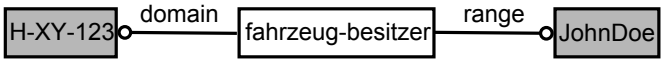


Abbildung 6.2: Modell mit zwei Ressourcen und einer Assoziation

Abbildung 6.2 zeigt beispielhaft eine einfache Relation, bestehend aus zwei Ressourcen, einer Eigenschaft und der Aussage darüber, in diesem Fall, dass das Fahrzeug `H-XY-123` einer Person `John Doe` gehört. Die Umsetzung des Modells in der XML-Repräsentation der RDF-Notation findet sich in Listing 6.1. Eine Einführung in die Notationen bieten Antoniou und van Harmelen [AH08].

Listing 6.1: Ein simples Beispiel in RDF

```
1  <?xml version="1.0"?>
2
3  <!DOCTYPE rdf:RDF [
4  <!ENTITY location "http://timguelke.de/auto.rdf" >
5  <!ENTITY auto "http://timguelke.de/auto.rdfs#" >
6  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
7  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
8  ]>
9
10 <rdf:RDF
11   xml:base ="&location;"
12   xmlns = "&location;"
13   xmlns:auto="&auto;"
14   xmlns:rdf ="&rdf;"
15    >
16
17   <auto:Fahrzeug rdf:ID="H-XY-123">
18     <auto:fahrzeug-besitzer rdf:resource="#JohnDoe"/>
19   </auto:Fahrzeug>
20
21   <auto:Person rdf:ID="JohnDoe"/>
22
23 </rdf:RDF>
```

Die Zeile 13 bindet die für dieses Beispiel konstruierte Schema-Datei ein und definiert für dort vorhandene Elemente das `auto:`-Prefix, Zeile 18 demonstriert die Verwendung der Relation `fahrzeug-besitzer`.

Wie im Beispiel bereits deutlich wird, müssen für verschiedene Domänen jeweils Schema-Definitionen in Resource Description Framework Schema (RDFS) vorhanden sein, wie sie in [RDFc] beschrieben werden. Die RDFS-Datei für das Fahrzeug-Beispiel ist in Listing 6.2 abgebildet.

Listing 6.2: Die Schemadefinition des Auto-Beispiels in RDFS

```
1  <?xml version="1.0"?>
2
3  <!DOCTYPE rdf:RDF [
4
5     <!ENTITY location "http://timguelke.de/auto.rdfs" >
6     <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          >
7     <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
8     <!ENTITY auto "&location;#" >
9
```

```
10 ]>
11
12 <rdf:RDF
13      xml:base ="&location;"
14      xmlns = "&location;"
15      xmlns:rdf ="&rdf;"
16      xmlns:rdfs="&rdfs;"
17      xmlns:auto = "&auto;"
18      >
19
20   <rdfs:Class rdf:ID="Fahrzeug">
21     <rdfs:comment>Die Klasse aller Fahrzeuge.</rdfs:comment>
22   </rdfs:Class>
23
24   <rdfs:Class rdf:ID="Person">
25     <rdfs:comment>Die Klasse aller Personen.</rdfs:comment>
26   </rdfs:Class>
27
28   <rdf:Property rdf:ID="fahrzeug-besitzer">
29     <rdfs:comment>
30       Setzt Fahrzeuge und ihre Besitzer in Beziehung.
31     </rdfs:comment>
32     <rdfs:domain rdf:resource="#Fahrzeug" />
33     <rdfs:range rdf:resource="#Person" />
34   </rdf:Property>
35
36 </rdf:RDF>
```

Entsprechend finden sich dort die Definitionen für die Klassen `Fahrzeug` (Zeile 20), `Person` (Zeile 24) sowie die Relation `fahrzeug-besitzer` (Zeile 28). Schemata, auch Namespaces genannt, können sich gegenseitig einbinden sowie direkt in einer RDF-Datei definiert sein. Auf diese Weise wird Modularität ermöglicht. RDFS unterstützt (Mehrfach-)Vererbung in Klassen und Relationen. Vor allem das Konzept der Vererbung von Relationen unterstützt den Aufbau umfassender Ontologien, in denen die Bedeutung bestimmter Zusammenhänge durch Software interpretiert werden kann.

RDF und RDFS verfügen allerdings nicht über genügend Sprachelemente, um alle im Rahmen einer Ontologie entwickelten Aussagen ausdrücken zu können. Es fehlen lokale Gültigkeiten von Eigenschaften, Disjunktheit von Klassen, boolesche Kombinatorik, Kardinalitäts-Restriktionen und die Möglichkeit des Ausdrucks spezieller Charakteristika von Eigenschaften wie etwa Transitivität oder Inversität [AH08]. An dieser Stelle kommt OWL (Ontologie Web Language) ins Spiel, welche diese Sprachkonstrukte besitzt und die RDFS-Syntax verwendet. Darüber hinaus existiert eine UML-basierte graphische Notation.

Als grundlegende Konstruktoren existieren `owl:Class` als Subklasse von `rdfs:Class` sowie `owl:ObjectProperty` und `owl:DatatypeProperty` als Subklassen von `rdf:`

`Property`. Die gesamte Dokumentation der Sprache findet sich unter [W3C]. Ein weiteres Konzept sind sog. Individuen, welche Instanzen von Klassen darstellen. Listing 6.3 stellt die bereits in Listing 6.2 definierte Ontologie dar, allerdings unter Verwendung von OWL sowie unter Einbeziehung von Individuen analog zu Listing 6.1.

Listing 6.3: Das Fahrzeug-Beispiel in OWL

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [

   <!ENTITY location "http://timguelke.de/auto.owl" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
   <!ENTITY auto "&location;#" >

]>

<rdf:RDF
     xml:base ="&location;"
     xmlns = "&location;"
     xmlns:rdf ="&rdf;"
     xmlns:rdfs="&rdfs;"
     xmlns:owl ="&owl;"
     xmlns:auto = "&auto;"
     >

<owl:Ontology rdf:about="&location;" />

<owl:Class rdf:ID="Fahrzeug">
   <rdfs:comment>Die Klasse aller Fahrzeuge.</rdfs:comment>
</owl:Class>

<owl:Class rdf:ID="Person">
   <rdfs:comment>Die Klasse aller Personen.</rdfs:comment>
</owl:Class>

<owl:ObjectProperty rdf:ID="fahrzeug_besitzer">
   <rdfs:comment>
     Setzt Fahrzeuge und ihre Besitzer in Beziehung.
   </rdfs:comment>
   <rdfs:domain rdf:resource="#Fahrzeug"/>
   <rdfs:range rdf:resource="#Person"/>
```

```
38 </owl:ObjectProperty>
39
40 <auto:Fahrzeug rdf:ID="H-XY-123">
41    <auto:fahrzeug_besitzer rdf:resource="#JohnDoe"/>
42 </auto:Fahrzeug>
43
44 <auto:Person rdf:ID="JohnDoe"/>
45
46 </rdf:RDF>
```

Die Notation „Turtle“ wurde vor dem Hintergrund der schweren Lesbarkeit der XML-Repräsentation entwickelt und stellt eine Serialisierungsform von RDF und RDFS dar, die einfacher durch Menschen zu lesen ist [RDFa]. Diese Arbeit verwendet daher für die Darstellung von RDF-Quelltext die Turtle-Notation anstelle von XML. Sämtliche Ontologien sind allerdings in ihrer XML-Originalform unter ihrem Namespace `http://timguelke.de` verfügbar. Die weiter unten eingeführten Abfragen in der SPARQL Protocol And RDF Query Language (SPARQL) werden auf diesen XML-Versionen durchgeführt. Die angeführten TURTLE-Repräsentationen sind per Transformation aus den XML-Darstellungen erzeugt worden.

Das Beispiel aus Listing 6.1 entspricht folgender Darstellung in Turtle-Notation:

Listing 6.4: Das simple Beispiel in Turtle-Notation

```
1 @prefix auto: <http://timguelke.de/auto.rdf#> .
2 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
3
4 auto:H-XY-123 rdf:type auto:Fahrzeug ,
5                        auto:Kraftfahrzeug ;
6
7               auto:fahrzeug-besitzer auto:JohnDoe .
8
9 auto:JohnDoe rdf:type auto:Person .
```

Subjekt, Prädikat und Objekt können direkt hintereinander geschrieben werden, wie in Zeile 4 ersichtlich wird: Das Subjekt `H-XY-123` ist durch die Relation `type` mit der Klasse `Fahrzeug` verbunden. Darüber hinaus bietet Turtle einige Vereinfachungen im Lesefluss: Bspw. können die Objekte von Relationen, die mehrfach in einem Subjekt verwendet werden, per Komma getrennt direkt hintereinander geschrieben werden. Zeile 5 zeigt, dass das Objekt `H-XY-123` vom Typ `Fahrzeug` und weiterhin vom Typ `Kraftfahrzeug` ist. Terminiert wird eine solche Liste von Objekten mit einem Semikolon. Eine ähnliche Methodik gilt für Subjekte: Relationen können ohne erneutes Aufschreiben des Subjekts verwendet werden, wie Zeile 7 zeigt. Terminiert wird das gesamte Subjekt schließlich durch einen Punkt, im Beispiel ebenfalls in Zeile 7. Eine vollständige Einführung in die Notation Turtle bietet der aktuelle Entwurf der Spezifikation unter [RDFa].

Die bisher vorgestellten Sprachen stellen textuelle Repräsentationen von Graphen dar. Zur Extraktion von Informationen aus in RDF formulierten Daten ist eine weitere Sprache nötig, die

das Formulieren von Abfragen in RDF-Graphen explizit unterstützt. Hierzu existiert die Sprache SPARQL, deren Name ein sog. rekursives Akronym darstellt, welches für SPARQL Protocol and RDF Query Language steht [PS, AG08]. Das zugrunde liegende Prinzip der Sprache ist der Abgleich eines Eingabe-Graphen mit dem durch das semantische Netz aufgespannten Graphen. Wird ein passender Teilgraph gefunden, wird dieser als Ergebnis zurück gegeben [AH08].

Listing 6.5: Abfrage aller Entitäten des Typs Fahrzeug im Auto-Beispiel mit SPARQL

```
1 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2 PREFIX auto: <http://timguelke.de/auto.owl#>
3
4 SELECT ?f
5 WHERE
6 {
7   ?f rdf:type auto:Fahrzeug .
8 }
```

Das Listing 6.5 zeigt eine einfache Abfrage, welche auf die in Listing 6.3 aufgebaute Ontologie angewendet werden kann. In diesem Fall würden alle Tripel mit der Verknüpfung `rdf:type` und der Klasse `Fahrzeug` zurückgegeben, also alle Entitäten des Typs `Fahrzeug`. Das Ergebnis der Abfrage lautet daher:

f
auto:H-XY-123

Tabelle 6.3: Ergebnis Abfrage aus Listing 6.5

Das folgende Listing 6.6 erweitert die Abfrage: Es gibt nun alle Entitäten aus, die durch die Relation `fahrzeug_besitzer` miteinander verbunden sind:

s	o
auto:H-XY-123	auto:JohnDoe

Tabelle 6.4: Ergebnis Abfrage aus Listing 6.6

Listing 6.6: SPARQL-Query für Relationen

```
1 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2 PREFIX auto: <http://timguelke.de/auto.owl#>
3 SELECT ?s ?o
4 WHERE
5 {
6   ?s auto:fahrzeug_besitzer ?o .
7 }
```

SPARQL ist allerdings nicht nur auf die Rückgabe von Individuen beschränkt: Das folgende Listing 6.7 findet alle Klassen vom Typ `owl:ObjectProperty` und damit alle Relationsklassen zwischen zwei Klassen.

Listing 6.7: SPARQL-Query für Relationsklassen

```
1 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2 PREFIX owl: <http://www.w3.org/2002/07/owl#>
3 SELECT ?p
4 WHERE
5 {
6   ?p rdf:type owl:ObjectProperty .
7 }
```

Die Rückgabe ist entsprechend:

p
auto:fahrzeug_besitzer

Tabelle 6.5: Ergebnis Abfrage aus Listing 6.7

Auf diese Weise können beliebig komplexe Abfragen konstruiert werden. Die vollständige Definition von SPARQL findet sich unter [PS]. Im weiteren Verlauf der Arbeit wird SPARQL verwendet, um Repräsentationen von Abfragen zu zeigen und die jeweilige Rückgabe zu demonstrieren. Zum Entwickeln und Testen von SPARQL-Abfragen bietet sich bspw. das Werkzeug „Pellet“ an [SPCG+05]. Pellet existiert als Desktop- sowie als Web-Anwendung [cla]. Aber auch der auf JAVA basierende Ontologie-Editor Protegé kann zum Entwickeln von Ontologien benutzt werden, er stellt neben vielen nützlichen Funktionen auch eine SPARQL-Abfragefunktion in der Oberfläche zur Verfügung [Sta]. Die Verwendung von SPARQL in vollständigen Anwendungen bringt allerdings einen Nachteil mit sich: Es bleibt dem Entwickler der jeweiligen Programmbibliothek oder Anwendung überlassen, wie die konkrete Implementierung von SPARQL vorgenommen wird. Deutlich wird dies vor allem bei der Verwendung von Transitivität in den Rückgabewerten von Abfragen, je nach programmtechnischer Umsetzung der Abfrage-Engine hat sie Einfluss auf die Ergebnisse oder nicht.

Listing 6.8: Erweiterung der Auto-Beispiel-Ontologie

```
1
2 @prefix auto: <http://timguelke.de/auto2.owl#>
3 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
4 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
5
6 auto:Fahrzeug rdf:type owl:Class .
7
8 auto:LKW rdf:type owl:Class ;
9         rdfs:subClassOf auto:Fahrzeug .
10
```

```
auto:Transporter rdf:type owl:Class ;
                 rdfs:subClassOf auto:LKW .

auto:VWN_Crafter rdf:type auto:Transporter .

auto:MAN_TGX rdf:type auto:LKW .

auto:Audi_A3 rdf:type auto:Fahrzeug ,
```

Das Listing 6.8 ist eine leicht abgewandelte Variante der Beispiel-Ontologie. Im Gegensatz zu dieser wurde nun eine hierarchische Kette von Vererbungen definiert. Weiterhin existieren für alle „Stufen“ der Hierarchie Individuen, also Objekte jeder der drei definierten Klassen. Die folgende Abbildung 6.6 zeigt die Zusammenhänge schematisch. Je nach Implementierung von SPARQL können nun auf eine Anfrage hinsichtlich der Objekte verschiedene Antworten gegeben werden. Listing 6.9 zeigt die – in jedem Fall identische – Abfrage.

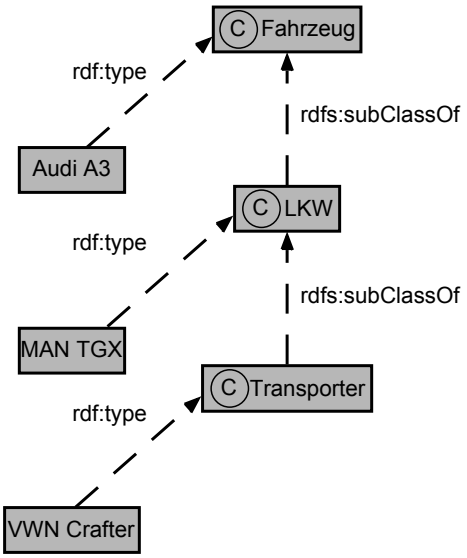


Abbildung 6.6: Erweiterung der Auto-Ontologie

Listing 6.9: SPARQL-Abfrage für Objekte des Typs Fahrzeug

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX auto: <http://timguelke.de/auto2.rdf#>

SELECT ?fzg
WHERE
 {
   ?fzg rdf:type auto:Fahrzeug .
 }
```

Der Schlüsselbegriff an dieser Stelle ist das sog. „Reasoning“ der verwendeten Abfragesoftware, also das Schlussfolgern aus gegebenen Informationen. Wird die Abfrage ohne Reasoning durchgeführt, wird ausschließlich das eingegebene SPARQL verarbeitet, welches selbst keinerlei Informationen hinsichtlch eventueller Schlussfolgerungen enthält, bzw. enthalten kann. Die Abfrage kennt den Inhalt der verwendeten Ontologie nicht und kann daher keine Annahmen über Zusammenhänge machen. Das Ergebnis enthält daher in diesem Fall nur das Objekt vom Typ Fahrzeug:

fzg
auto2:Audi_A3

Tabelle 6.7: Ergebnis Abfrage aus Listing 6.9 ohne automatischem Schlussfolgern (Reasoning) durch die Abfragesoftware

Das Verwenden des Reasoners in der Abfragesoftware führt zu einem anderen Ergebnis, da nun die `subClassOf`-Relation mit betrachtet wird. Durch die Transitivität der Relation ergibt sich folgende Ergebnismenge:

fzg
auto2:MAN_TGX
auto2:Audi_A3
auto2:VWN_Crafter

Tabelle 6.8: Ergebnis Abfrage aus Listing 6.9 mit automatischem Schlussfolgern (Reasoning) durch die Abfragesoftware

Für den weiteren Verlauf der Arbeit wird in den erstellten Abfragen davon ausgegangen, dass diese mit aktivem Reasoning ausgeführt werden.

Zur programmtechnischen Umsetzung existieren diverse APIs. Eines der aktivsten Projekte zur Entwicklung einer API ist Apache Jena [Apab]. Unter anderem enthält es eine „query engine“, also API-Funktionalität zur Ausführung von Abfragen auf einer Ontologie, die SPARQL-Abfragen mit Reasoning unterstützt.

Der Sprachgebrauch im Kontext des Semantic Web unterscheidet sich von dem in der objektorientierten Programmierung, welcher auch Einzug in die modellbasierte Softwareentwicklung gehalten hat. Um Klarheit für diese Arbeit zu erhalten, werden einige sprachliche Konventionen definiert.

1. **Klassen und Instanzen:** Klassen stellen das Meta-Konstrukt dar, welches gleiche Elemente zur Definitionszeit beschreibt. Instanzen sind konkrete Ausprägungen von Klassen, die mit Werten versehen sind und zur Laufzeit erstellt, geändert oder gelöscht werden. In der objektorientierten Programmierung wird strikt zwischen Definitions- und Laufzeit getrennt, Objekte werden zur Laufzeit von Klassen instantiiert. Im Kontext von semantischen Netzen findet ein anderer Sprachgebrauch Anwendung: es wird hier grundsätzlich von Begriffen gesprochen, eine Trennung zwischen Klasse und Objekt findet nicht statt. Begriffe können als sog. Individuen gekennzeichnet werden, sie stellen das Äquivalent zu Objekten dar.

In dieser Arbeit wird analog zur objektorientierten Programmierung von Klassen und Objekten gesprochen, wobei Objekte einen Typ besitzen.

2. **Eigenschaften:** Eigenschaften sind in der objektorientierten Programmierung fester Bestandteil von Klassen und werden von diesen gekapselt. Sie können primäre Datentypen enthalten bzw. andere Objekte einer Klasse. Definiert werden sie im Kontext der Klasse. In OWL dagegen sind Eigenschaften getrennt formuliert und somit global verfügbar. Sie sind konzeptionell durch die Definition mithilfe von `Domain` und `Range` eher als Relation zu betrachten. Primäre Datentypen sind hierbei als Subklasse von `owl:Literal` formuliert. Der Vorteil gegenüber dem Objekt-orientierten Ansatz ist die Eindeutigkeit in der Bedeutung einer bestimmten Eigenschaft - der Name wird nur einmal verwendet. Andererseits ist dieser Ansatz konträr zum etablierten Vorgehen in der Objekt-orientierten Programmierung. Im weiteren Verlauf der Arbeit wird bei Relationen zwischen Begriffen und primitiven Datentypen von Eigenschaften gesprochen, bspw. `(Auto, hatFarbe, owl:string)`. Relationen vom Typ `owl:ObjectProperty` werden auch weiterhin als Relationen bezeichnet, um den Unterschied deutlich zu machen.

6.1.2 Technologie für die Infrastruktur des Werkzeugs

Als technische Grundlage für die Realisierung des skizzierten Werkzeugs stehen bereits heute diverse Technologien als Quasi-Standard zur Verfügung. Im Folgenden werden einige APIs und Infrastrukturtechnologien vorgeschlagen, welche den Entwicklungsaufwand durch Bereitstellung benötigter Funktionalität vermindern.

Heutige Applikationen müssen hohen Anforderungen hinsichtlich Sicherheit, Wartbarkeit, Useability, Portabilität, etc. genügen. Es bietet sich daher an, Software als HTML5 Web-Applikationen zu realisieren, da die Vorteile von diesen in der Adressierung dieser Punkte liegen [Bal11]:

1. **Plattformunabhängigkeit:** Webapplikationen werden durch den Aufruf einer URL im Webbrowser eines Benutzers gestartet. Die Spannbreite der Realisierungen reicht hier von der Einbettung bestimmter Funktionen in einen Webseiten-Komplex bis hin zur Verwendung einer einzigen Seite mit dem ausschließlichen Zweck, einen Rahmen für die Applikation zu stellen. Der letzte Fall findet sich heute bspw. eindrucksvoll bei komplexen Applikationen wie Googlemail oder Google Docs. Durch die Interpretation des HTML- und JavaScript-Quelltextes in einem Webbrowser können die Applikationen losgelöst vom Betriebssystem des Benutzers verwendet werden. Da Webbrowser in den vergangenen Jahren immer bessere Standardkonformität erreicht haben, fällt auch die Präferenz des Benutzers hinsichtlich dieser Komponente kaum noch ins Gewicht. Unterschiede finden sich lediglich noch in der Performance des Webbrowsers bei der Ausführung von JavaScript. Das Nachrichtenportal Lifehacker bietet hierzu bspw. regelmäßige Vergleiche der gängigsten Browser [Lif]. Die Plattformunabhängigkeit führt weiterhin dazu, dass viele Anwendungen außer auf Desktop-PCs problemlos auch auf mobilen Geräten ausgeführt werden können.

2. **Wartbarkeit:** Web-Applikationen bieten eine verbesserte Wartbarkeit gegenüber herkömmlichen Applikationen, da keinerlei Aktualisierung von Software beim Benutzer notwendig ist. Rein technisch gesehen, findet bei jedem Start der Applikation durch einen Aufruf des Benutzers eine erneute Übermittlung der gesamten Anwendung, bzw. der benötigten Teile davon statt. Die Wartung kann daher ausschließlich an einer zentralen Stelle stattfinden. Ansonsten ist die Wartbarkeit auf Quelltext-Ebene weiterhin abhängig von einer umsichtigen Architektur der Applikation.

3. **Skalierung:** Durch die Trennung von Anzeige- und Ausführungsschicht kann die zugrundeliegende Anwendung beliebig skaliert werden. Je nach Benutzerzahl und Anzahl bzw. Komplexität von Abfragen kann die einer Anwndung zur Verfügung stehende Kapazität verändert werden, indem etwa ein weiterer Server hinzugeschaltet wird oder Jobs wie z.B. Suchen auf speziellen Ausführungsservern ausgeführt werden. Durch die Virtualisierung, die heute Firmen wie etwa Amazon oder Google bieten, können Anwendungen hochdynamisch gestaltet werden [Mur08, Ciu09].

Realisiert werden können Webanwendungen mit den beschriebenen Eigenschaften durch aktuelle Technologien wie bspw. AJAX [TK08]. Frameworks wie das Google Web Toolkit (GWT) stellen AJAX-Funktionalität bequem zur Verfügung und vereinfachen dadurch die Entwicklung von interaktiven Webanwendungen enorm [Gup08]. GWT bietet durch komfortable Werkzeuge eine umfangreiche und funktional breit aufgestellte Infrastruktur zur Entwicklung von Web-Applikationen in HTML5. Entwickelt wird in der objektorientierten Sprache Java, welche zur Laufzeit der Applikation in einem Web-Container in JavaScript und HTML übersetzt wird. Der generierte Quelltext ist hierbei angepasst an und optimiert für die jeweilige Zielplattform des Benutzers. Als Web-Container kommt bspw. Tomcat von der Apache Foundation zum Einsatz [Apad].

Ebenfalls unter dem Dach der Apache Foundation entwickelt wird das Jena-Framework [Apab], welches umfassende Möglichkeiten zur Arbeit mit semantischen Netzen bietet. Es ist ebenfalls in der Sprache Java realisiert und verfügt über definierte Schnittstellen, mit deren Hilfe Ontologien interpretiert, erstellt oder verändert werden können. Darüber hinaus bietet es eine Schnittstelle für in SPARQL beschriebene Abfragen. Eine große Zahl der Werkzeuge, die semantischen Netzen einsetzen, sind mit Hilfe des JENA-Frameworks realisiert. JENA kann direkt Ontologien aus einer Datei oder anhand einer Internet-Adresse laden und auf dieser Abfragen ausführen. Das folgende Listing 6.10 in Java demonstriert dieses Vorgehen. Der Quelltext ist dabei nicht vollständig, sondern gibt nur die verwendeten Konzepte wieder:

Listing 6.10: Einfacher und unvollständiger Java-Quelltext zum Ausführen einer SPARQL-Abfrage

```
1 public class JenaQueries {
2
3    public static void main(String[] args) {
4
5       Model data = FileManager.get().loadModel("http://
           timguelke.de/produkt-artefakte.owl");
```

```
6       String query = readFile("C:/Abfragen/query.txt");
7
8       OntModel infModel = ModelFactory.createOntologyModel(
           OntModelSpec.OWL_MEM_MICRO_RULE_INF, data);
9
10      QueryExecution qe = QueryExecutionFactory.create(query,
            infModel);
11
12      for (ResultSet rs = qe.execSelect() ; rs.hasNext() ; )
           {
13         QuerySolution sol = rs.nextSolution();
14         System.out.println(sol.get("s")  + "␣␣" + sol.get("p
              ")+ "␣␣" + sol.get("o"));
15      }
16
17     qe.close();
18
19    }
20 }
```

In Zeile 5 wird zunächst ein Modell geladen, auf welchem die Abfrage ausgeführt werden soll. Zeile 6 lädt dann besagte Abfrage aus einer lokalen Datei. Damit die Abfrage, wie in Abschnitt 6.1.1 eingeführt, mit aktiviertem Reasoning ausgeführt werden kann, muss das Modell noch durch einen sog. Reasoner interpretiert werden. JENA stellt hierzu einige Standard-Reasoner bereit. In Zeile 8 wird das Modell dann in eine interpretierte Variante transformiert. Die folgende Zeile 10 führt den als Zeichenkette aus der angegebenen Datei geladenen Query dann auf dem Modell aus, die Ergebnisse können mithilfe eines Iterators anschließend ausgelesen werden. Die Zeilen 12 und 14 schreiben die in der Abfrage spezifizierten Variablen danach in die Konsolenausgabe.

Die Nachteile des Einsatzes von Webanwendungen sowie der Verwendung von HTML5-Frameworks dürfen allerdings nicht unerwähnt bleiben. Trotz stetig steigender Verfügbarkeit von breitbandigen Internet-Zugängen, in letzter Zeit vor allem im mobilen Bereich [AS12], steht nicht unbedingt immer ein solcher zur Verfügung. Zusätzlich kann auch serverseitig ein Fehler in der Verbindung oder der Applikation vorliegen. In allen Fällen führt eine nicht vorhandene Datenverbindung zwischen Benutzer und Anbieter zu einer Nicht-Erreichbarkeit der Anwendung und somit zu einem Ausfall. Lösungen hierfür bieten die Frameworks bspw. durch das Anbieten von Offline-Modi, die bei einem Ausfall der Netzwerk-Verbindung Benutzereingaben speichern und später synchronisieren können. Nichtsdestotrotz stellt die Verfügbarkeit einer verlässlichen Datenverbindung eine Grundvoraussetzung beim Einsatz einer Web-Applikation dar.

Umfangreiche Modelle können schnell große und komplexe Strukturen aufweisen, deren Übermittlung und Verarbeitung relativ aufwändig werden kann. Dies bedeutet, dass die Geschwindigkeit der Anwendung maßgeblich von der Leistungsfähigkeit des Browsers des Benutzers abhängen kann, falls nicht die Architektur der Anwendung auf eine intelligente Lastverteilung ausgerichtet ist. Möglich wird dies beispielsweise durch das unvollständige Laden von Modellen und das Nachladen bei Bedarf oder durch das Auslagern rechenintensiver Aktionen auf den

Server. Letzteres kann bspw. durch Verwendung des Programmiermusters „Command“ erreicht werden. Der Browser des Anwenders stellt trotzdem weiterhin einen starken Flaschenhals dar, da vor allem im Bereich der mobilen Endgeräte eine starke Fragmentierung der eingesetzten Hard- und Software zu erkennen ist [AS12]. Hieraus resultieren bspw. verschiedene Bildschirm-Auflösungen oder nicht vorhandene Funktionen mit denen z.B. das im vorhergehenden Absatz beschriebene Verfahren zur Offline-Synchronisation implementiert werden kann.

Die Sicherheit von Webanwendungen basiert einerseits auf der Sicherheit des implementierten Standardverfahrens zur Verschlüsselung der Kommunikation zwischen Benutzer und Server. Andererseits bieten Webanwendungen durch die Zugriffsmöglichkeiten auf die Anwendung eine breite Angriffsfläche für verschiedenste Angriffe zur Störung des Betriebs bzw. Kompromittierung der Daten. Somit ergibt sich durchaus ein höheres Gefährdungspotential als bei traditionellen Desktop-Anwendungen [Jäg08]. Entwickler sind demnach an dieser Stelle in der Pflicht, aktuelle Maßnahmen zur Abwehr von Angriffen aller Art zu etablieren.

Webapplikationen bestehen in der Entwicklung wiederum selbst aus vielen verschiedenen Artefakten, die als Konfigurationsobjekte in der Releaseplanung berücksichtigt werden müssen. Insbesondere der Bereich der direkt und indirekt eingebundenen Bibliotheken birgt die Gefahr unübersichtlich zu werden. Die Verwendung nicht aktueller Bibliotheksfunktionen kann zu Sicherheitslücken oder Inkompatibilitäten in der Anwendung führen. Werkzeuge wie Apache Maven ermöglichen deshalb die Erstellung von eindeutigen Konfigurationen, mit definierten Versionsständen aller verwendeten Konfigurationsobjekte [Apac].

6.2 Grundlegende Ontologien für die Erarbeitung unternehmensspezifischer Artefaktnetze

Die Wissensbasis stellt durch eine Webserver-Applikation die OWL-Dateien zur Verfügung, die zur Verteilung der Modelle benötigt und durch Applikationen entsprechend importiert werden. Für jede Applikation des Anforderungsmanagements ist es zentraler Anlaufpunkt und somit einerseits eine Auskunftmöglichkeit für Applikationen, andererseits eine Steuerungsmöglichkeit für die Organisation des Wissens. Durch das Zurverfügungstellen bestimmter Modelle wird ein Fundus an vorhandenen Kernartefakten geschaffen, deren Eigenschaften und Zusammenhänge eine Grundlage für weitere, stärker spezialisierte Artefakte bilden. Zunächst müssen diese spezifiziert werden, bevor eine auf ihnen basierende Ontologie entworfen werden kann.

6.2.1 Definition von Kernartefakten als Grundlage für eine Unternehmensontologie

Zunächst findet eine Definition von bestimmten grundlegende Kategorien statt, die eine Menge von Kernartefakte bilden. Mit diesen kann ein Teil der Artefakte bereits in ihrer Art bestimmt und somit grundlegende und allgemein gültige Zusammenhänge beschrieben werden. Zusätzlich ist es durch das Konzept der Vererbung möglich Artefakte, die von diesen Kernartefakten abgeleitet werden, auf die gleiche Weise semantisch zu interpretieren wie die Kernartefakte selbst. Die meisten Artefakte sind unternehmensspezifisch und müssen daher im jeweiligen Kontext definiert und von den Kernartefakten abgeleitet werden.

Das Finden von allgemein verwendbaren Oberklassen in der Literatur gestaltet sich schwierig: Zwar finden sich Definitionen im Umfeld des *Business-Process-Modelling*, ausreichend erscheinen sie allerdings nicht. Der Rational Unified Process (RUP) definiert z.B. folgende Artefakte [Kru03]:

- Stakeholder Requests
- Vision
- Business Case
- Software Development Plan

Scherer nennt für den Bereich der IT-Service-Management-Prozesse die Artefakte „Availability Plan“, „Business Strategy“, „Budget Plan“ und viele weitere [Sch10a]. Es können demnach beliebige weitere Beispiele für weitere Domänen gefunden werden. Teilweise wird nicht der Begriff Artefakt, sondern „Objekt“ oder ähnliche Synonyme verwendet. Ihnen allen ist jedoch gemein, dass sie Elemente beschreiben, die sich durch die mit der jeweiligen Sprache beschriebenen Prozesse bewegen.

Eine domänenunabhängige Sprache ist die Business Process Modeling Notation (BPMN) zur graphischen Beschreibung von Geschäftsprozessen [All09]. In ihr finden sich folgende `Artefakte` [Fet08]:

- `Datenobjekt`: Stellen Informationen im Prozess dar, werden durch Aktivitäten erzeugt, bearbeitet oder eliminiert.
- `Gruppe`: Gruppierende Elemente im Prozess.
- `Anmerkung`: Ergänzende nicht-formale Informationen.

Unter dem Begriff *Objekte* existieren darüber hinaus `Flussobjekte` und `Verbindungsobjekte`, wobei diese Aktivitäten und die zwischen ihnen existierenden Zusammenhänge beschreiben. Auch diese Notation hilft der Beschreibung der tangiblen und intagiblen Elemente in einem Unternehmen nur bedingt weiter. Physische Objekte lassen sich auf diese Art nur schlecht modellieren.

In der Arbeit von Mayer-Bachmann findet sich eine weitergehende – produktorientierte – Abgrenzung von zentralen Begriffen im Rahmen des Anforderungsmanagements [MB08]. Diese sind z.B.

- Produkt
- Produktentwicklung
- Eigenschaft
- Merkmal
- Produktelement

- Funktion
- Anforderung
- Produktanforderung
- Qualitative Produktanforderung
- Quantitative Produktanforderung

Aufgrund Mayer-Bachmanns Fokus auf das Produkt fehlen hier weiterhin die es umgebenden Artefakte. Die an dieser Stelle notwendigen Elemente setzen sich demnach teilweise aus den obigen zusammen und werden zusätzlich noch ergänzt. Ziel ist, weniger das Produkt als die es umgebenden Entitäten im Rahmen der Produktentwicklung zu betrachten.

Bezeichnung	**Beschreibung**
`Produkt`	„Das Produkt wird als Leistung in Form eines Gegenstands verstanden. Der Gegenstand kann materieller oder immaterieller Natur sein“ [MB08]. Produkte können damit ganze Fahrzeuge, aber auch Module, Teile oder Software sein.
`Prozess`	„Ein Prozess ist die inhaltlich abgeschlossene zeitliche und sachlogische Abfolge der Funktionen, die zur Bearbeitung eines betriebswirtschaftlich relevanten Objektes notwendig sind“ [BV96].
`System`	Ein informationstechnisches System ist eine Hardware/Software-Kombination zum Zwecke der Datenverarbeitung.
`Dokument`	Ein Dokument ist ein virtuelles Datenobjekt oder eine physische Repräsentation hiervon mit dem Zweck der Datenablage oder des Datentransports. Entsprechend sind dies Text- oder Tabellendokumente, Emails oder z.B. Objekte innerhalb eines `Systems`.
`Struktur`	Eine Struktur ist eine formale Gliederung und ist wichtig z.B. für `Dokumente` sowie deren werkzeuggestützte, automatisierte Verarbeitung.
`Organisation`	Die Organisation beschreibt den formellen Aufbau aus Fachabteilungen und Hierarchien innerhalb eines Unternehmens. Eine `Organisation` ist geprägt von `Strukturen`.
`Anforderung`	Im Sinne von Definition 2.1.1.

Tabelle 6.9: Liste der Kernartefakte

Scheer betrachtet neben der Darstellung von Geschäftsprozessen im Rahmen des ARIS-Konzepts die *Daten*, *Funktionen*, *Aufbauorganisation* und *IT-Ressourcen* des Unternehmens [Sch92]. Dieser High-Level-Ansatz erscheint wenig zielführend für ein ausgeprägtes Artefaktsystem. Lediglich der Begriff *Daten* findet im eigenen Schema später eine Verwendung.

Schraps, Ronneberger und Golubski gehen mehr auf die generischen Klassen im Rahmen des Produkts und seiner organisatorischen Umgebung ein und identifizieren daher *Person*, *Kompetenz*, *Artefakt*, *Rolle* und *Projekt* [SRG11]. Der Terminus *Artefakt* ist allerdings an dieser Stelle nicht

ausreichend definiert. Als Kernartefakte gemäß Definition 4.1.1 sollen demnach die in Tabelle 6.9 gelisteten Artefakte gelten.

Die Liste resultiert einerseits aus den bereits in der Literatur gefundenen und andererseits aus den im Rahmen des Projekts mit Volkswagen erarbeiteten Artefakten. Diese mussten in grundlegenden Klassen zusammengefasst werden, um eine gemeinsame und allgemeingültige Basis zu finden, von der sie wiederum abgeleitet werden konnten.

6.2.2 Einführung der Ontologien Core und RMNet zur Formalisierung und Verknüpfung der Kernartefakte

Als Grundlage für alle Artefakte wird die Sammlung an Kernartefakten in OWL definiert. Abbildung 6.10 zeigt den Aufbau einer `Core` genannten Ontologie zur Beschreibung der eingeführten Artefakte. Die Core-Ontologie stellt sicherlich nur rudimentäre Klassen und Assoziationen zur Verfügung, dient aber als Grundlage für die Erweiterung um eine anforderungsmanagementspezifische Ontologie. Der OWL-Quelltext für die Core-Ontologie findet sich in Listing B.1 in Anhang B.

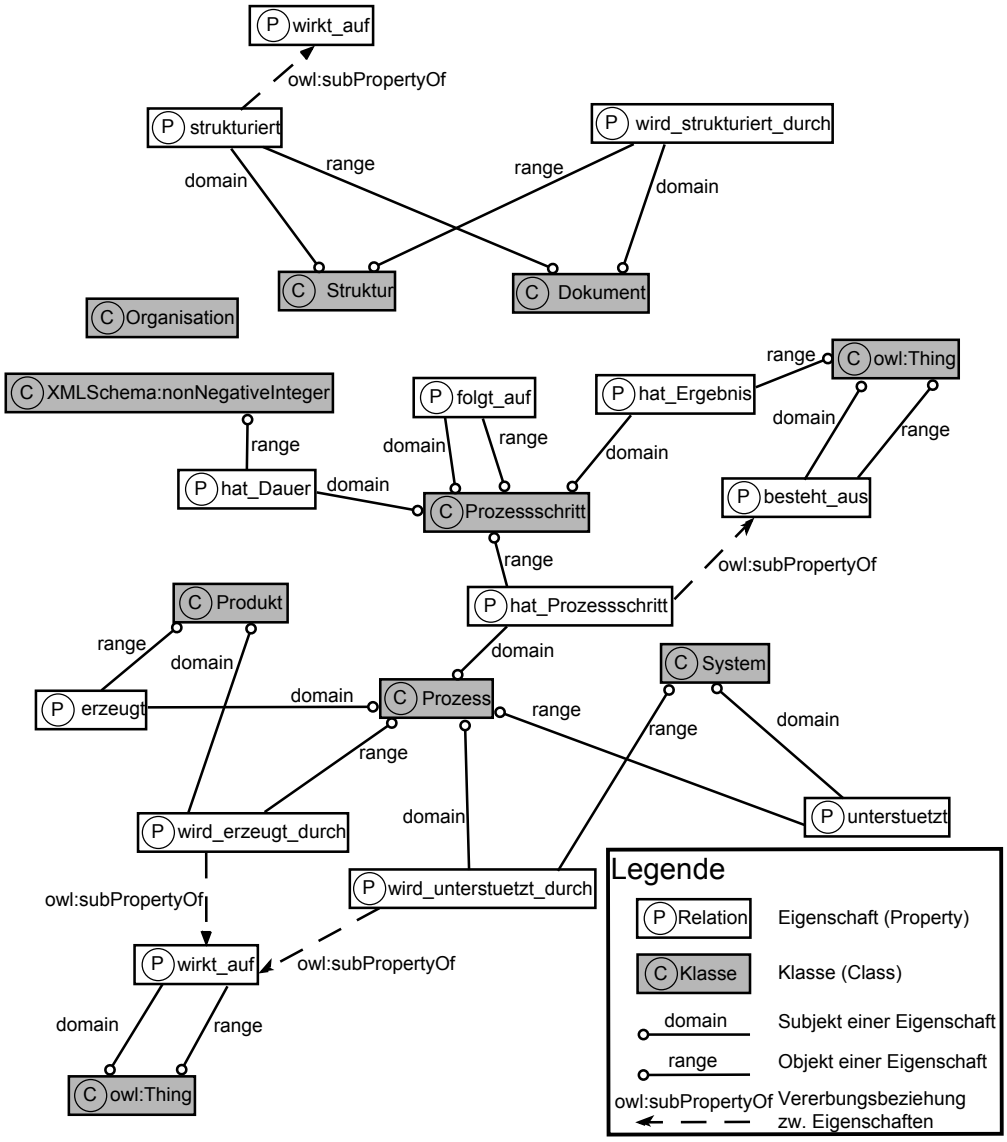


Abbildung 6.10: Eine graphische Repräsentation der Core-Ontologie

Mit Hilfe der Klassen innerhalb der Core-Ontologie kann nun grundsätzlich beschrieben werden, um welche Art Artefakt aus Tabelle 6.9 es sich handelt. `Dokumente` und `Strukturen` sind durch die Relation `wird_strukturiert_durch` und das inverse Pendant miteinander verknüpft. Hauptsächlich können hiermit Dokumente wie Listen, Lasten- und Pflichtenhefte aber auch Daten-Entitäten oder Zusammenstellungen anderer Artefakte wie bspw. Anforderungslisten repräsentiert werden. Insbesondere in der frühen Phase der Produktentwicklung sind viele informelle Artefakte vorhanden, die sich als Dokumente äußern. `Prozesse erzeugen Produkte` und werden dabei von `Systemen unterstützt`. Das `Produkt` sollte hierbei als Endprodukt verstanden werden, welches nach Durchführung des `Prozesses` zur Verfügung steht. Letzterer besteht aus einzelnen Prozessschritten, die jeweils ein `Ergebnis haben`. Diese Ergebnisse können – wie die Verwendung der OWL-Klasse `owl:Thing` als Objekt der Relation suggeriert – beliebige Artefakte sein. Ein `Prozessschritt` kann demnach bspw. ein `Dokument` erzeugen, auf dessen Erstellung die Aktivitäten des Schritts hauptsächlich ausgerichtet sind. Die Verwendung von `Prozessschritten` ist somit nicht unbedingt notwendig, ermöglicht aber eine Strukturierung des Prozesses in einzelne Phasen mit Ergebnissen. Diese Sichtweise unterstützt bspw. das Ableiten von Projektplänen aus der Beschreibung eines Referenzprozesses.

Anforderung Nr. 17 an ein semantisch unterstütztes Werkzeug für das Anforderungsmanagement aus Tabelle 5.1 beinhaltet das Durchführen von Impact-Analysen. Hiermit werden die Möglichkeiten einer ontologiebasierten Software beschrieben, automatisiert die Auswirkungen von Änderungen im Vorfeld abschätzen zu können. Wird dies ausschließlich anhand der vorhandenen Relationen durchgeführt, kann nicht unterschieden werden, ob durch eine Verknüpfung tatsächlich Einfluss ausgeübt wird. Bspw. `strukturiert` eine `Struktur` ein `Dokument`. Die inverse Relation ist `wird_strukturiert_durch`. Das gewünschte Verhalten einer Impact-Analyse ist, dass das Ändern einer `Struktur` einen Einfluss auf die `Dokumente` hat, die durch sie `strukturiert` werden. Im umgekehrten Fall dürfte das Ändern eines `Dokuments` keinerlei Einflusses auf die `Struktur` haben. Es muss also eine Möglichkeit vorhanden sein, die semantische „Wirkrichtung“ von Relationen zu hinterlegen. Die Ontologie Core stellt daher eine Oberklasse für wirkende Relationen, `wirkt_auf` zur Verfügung. Da RDF Mehrfachvererbung sowie die Vererbung von Relationen unterstützt, können auf diese Weise wirkende Relationen gekennzeichnet werden. Ein Algorithmus zur Durchführung einer Impact-Analyse kann nun dementsprechend diese Informationen verwenden oder sie ignorieren. Dem Modellierer steht allerdings ein eindeutiges Werkzeug zur Verfügung, um die Zusammenhänge in seinem Modell mit einer Wirkrichtung zur versehen.

In OWL werden Präfixe zur eindeutigen Identifizierung von Ressourcen verwendet, hierdurch können innerhalb einer Ontologie Ressourcen aus anderen Ontologien miteinander verknüpft werden. Zusätzlich kann über das Tag `<owl:imports/>`-Tag mit der zu importierenden Ontologie als `rdf:resource`-Parameter die Übernahme von Meta-Informationen aus einer entfernten Ontologie vorgenommen werden.

Eine Ontologie im Kontext des Anforderungsmanagements ist die SoftWiki Ontology for Requirements Engineering (SWORE) [RLL07], welche bereits eine Konkretisierung des Begriffs „Anforderung“ vornimmt und diesen in den Kontext von z.B. „Stakeholder“ einordnet. Allerdings liegt der Fokus in SWORE auf dem Produkt, ohne es in den größeren Kontext einer Unterneh-

mensartefaktlandschaft einzuordnen. Zur Darstellung der Entwicklung eines semantischen Netzes für ein Unternehmen wurde daher auf Basis von SWORE sowie Core eine Ontologie entwickelt, welche zum einen die Struktur eines Unternehmens und zum anderen die Anforderungen an seine Produkte im Kontext der vorhandenen Artefakte darstellen kann. Die OWL-Notation findet sich in Anhang B als Listing B.2. Die Benennung von Elementen ist in der Ontologie in Deutsch gehalten. Durch das Versehen mit `rdfs:label`-Tags können allerdings beliebig viele andere Sprachen hinzugefügt werden. Realisiert wird dies durch das Hinzufügen eines `xml:lang`-Parameters, in diesem Fall `xml:lang=„de“>`

Die RMNet-Ontologie kombiniert die in der Core-Ontologie zur Verfügung stehenden grundlegenden Unternehmensartefakte mit Anforderungen aus SWORE. Als zentrale Idee findet sich die Annahme, dass für Produkte Eigenschaften definiert werden, welche die an das Produkt gestellten Anforderungen erfüllen. Diese `Produkteigenschaften` wiederum werden durch bestimmte technische Umfänge realisiert. Bspw. kann die Anforderung „Das Fahrzeug muss dem Fahrer eine maximale Sicht bei Nacht gewährleisten“ durch die `Produkteigenschaft` „Fahrzeug erhält Nachtsicht-Technologie“ erfüllt werden. Die `Produkteigenschaft` stellt demnach eine frühe Entscheidung zugunsten bestimmter Konzepte dar, welche aber noch nicht mit konkreter Technik versehen sind. Daher wird jede `Produkteigenschaft` im Laufe des Projekts durch technische Umfänge wie etwa Teile oder Software realisiert. Durch den Einsatz der RMNet-Ontologie wird somit bereits eine gewisse Arbeitsweise, wie sie von der Anforderungsmanagement-Theorie gefordert wird, begünstigt: Beginnend bei der Definition von Anforderungen an ein Produkt werden Entscheidungen hinsichtlich der Erfüllung dieser Anforderungen getroffen und dann durch Technik realisiert.

Es findet sich daher neben der `Produkteigenschaft` eine Klasse `AbstractUmfang`, beide sind durch die Relation `wird_realisiert_durch` und der entsprechend inversen Ausführung verbunden. Die `Produkteigenschaft` selbst `erfüllt` eine Anforderung der SWORE-Ontologie. `Anforderungen` und `Produkteigenschaften` stellen somit die zentralen Punkte des Modells und entsprechend auch des gesamten Vorgehens dar. Am Ende der Kette von Artefakten gliedern sich die `AbstractUmfänge` auf in die konkreten Ausprägungen `Teile, Module` und `Software`. Durch sie lässt sich ein Produkt später vollständig beschreiben. Desweiteren finden sich in der RMNet-Ontologie viele unterstützende Relationen und Klassen: bspw. wird definiert, dass `Anforderungen` in einem `Dokument beschrieben` werden, ebenfalls wie `AbstractUmfänge`. `Produkteigenschaften` und `Anforderungen` sind wiederum `Produkten` als Ausgangspunkt zugeordnet.

Eine vereinfachte Darstellung der Ontologie ist in Abbildung 6.11 abgebildet, wobei auf stark kreuzende Relationen verzichtet wurde. Die Klassen sind allerdings vollständig dargestellt. Eine graphisch weniger ansprechende und schwerer zu lesende, jedoch vollständige Abbildung der Ontologie kann problemlos aus dem Listing B.2 im Anhang generiert werden.

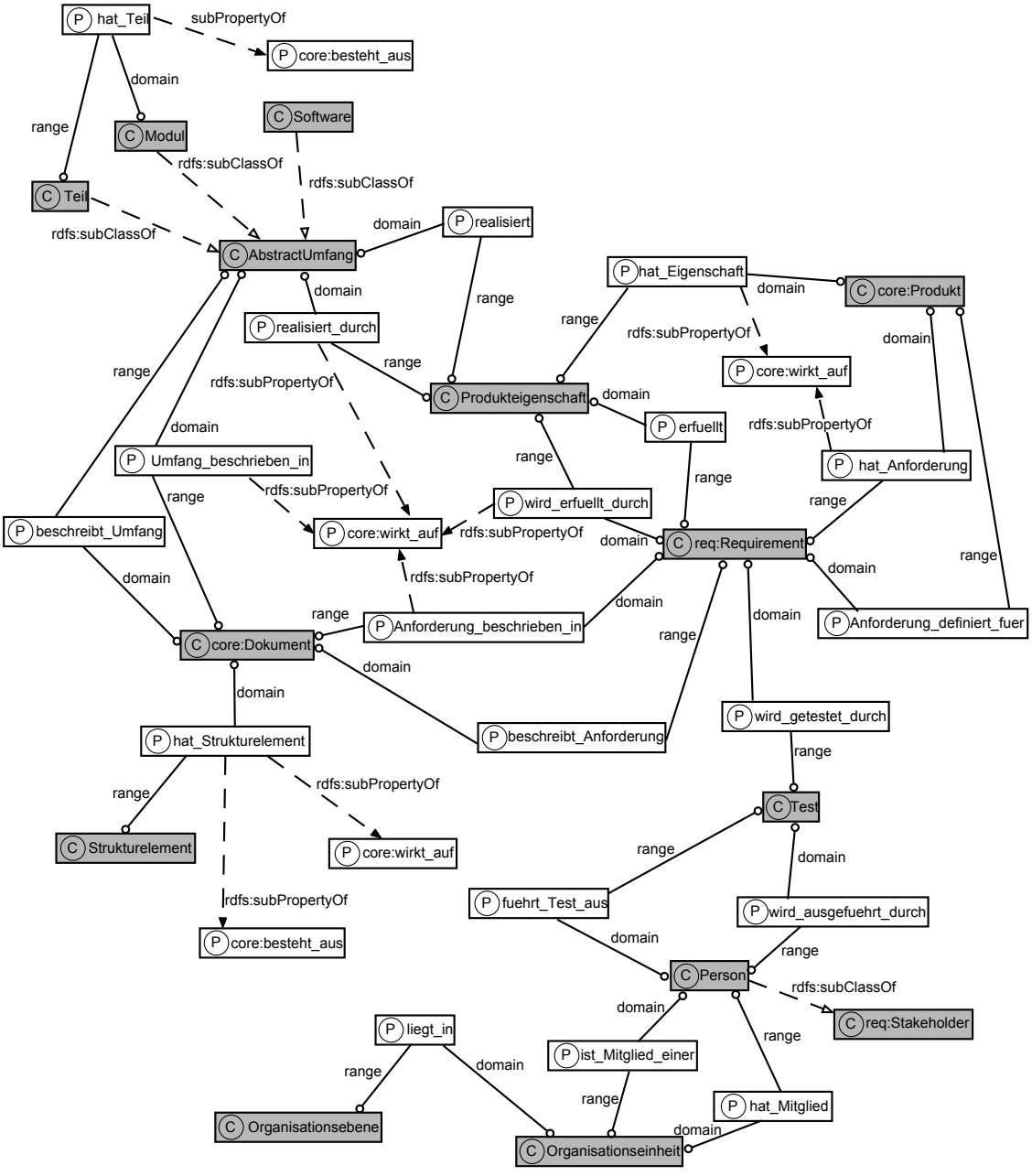


Abbildung 6.11: Eine reduzierte graphische Repräsentation der RMNet-Ontologie

6.3 Methodik zur Verwendung unternehmensspezifischer Artefaktnetze

Nachdem zunächst allgemeingültige Begriffe definiert und eingeführt worden sind, sind diese durch eine Ontologie formalisiert sowie zueinander in Beziehung gesetzt worden. Sie bilden die Grundlage des semantisch unterstützten Anforderungsmanagements. Zu Erhebung und Pflege der Informationen wird eine Methodik zur Verfügung gestellt. Hierbei können vier Phasen unterschieden werden:

1. Initiale Erstellung
2. Verwendung
3. Markierung
4. Prüfung und Überarbeitung

Sie bilden einen einfachen Regelkreis, der mit verschiedenen Methoden die adäquate Abbildung der Realität gewährleisten soll.

Es existieren drei verschiedene Rollen im Kontext der Modellierung:

- **Domänenexperte:** Für die Erfassung der Unternehmensartefakte müssen Experten ihr Wissen zur Verfügung stellen.
- **Modellierer:** Die Aufgabe des Modellierers ist die Transformation des impliziten Wissens der Experten über ihre Domäne und die darin enthaltenen Artefakte in ein formales Modell. Idealerweise ist dies eine zentrale Stelle, welche später die Hoheit über das erstellte Unternehmensartefaktmodell besitzt.
- **Anwender:** Anwender verwenden die Ontologie später im Rahmen der verschiedenen Disziplinen des Systems Engineering. Sie tragen weiterhin zur Identifikation von Fehlern oder Unzulänglichkeiten in den Modellen bei.

6.3.1 Initiale Erstellung des Artefaktnetzes

Der erste Schritt in der Erstellung der Ontologie der Unternehmensartefakte besteht im Sammeln und Bewerten relevanter Elemente. Hierzu stehen diverse Methoden aus dem Themenfeld der „Knowledge Acquisition Projects“ zur Verfügung [Mil07]. Von Experteninterviews bis hin zu komplexer Szenariotechnik bietet die Literatur viele Möglichkeiten zur Erfassung von Artefakten. Auch die Verwendung von Werkzeugen zur Erfassung von Ergebnissen ist situationsspezifisch gestaltbar. Im Hinblick auf die Modellierung als Ontologie bieten sich als Notation UML-Klassendiagramme an, da bereits einheitliche Vorgaben für Klassen und deren Beziehungen bestehen. Allerdings kann der volle Sprachumfang bei fachfremden Domänenexperten zu Verwirrung führen. In Kapitel 3 werden die Erfahrungen mit UML-Klassendiagrammen in Experteninterviews in einem Projekt in der Automobilindustrie wiedergegeben. Eine Möglichkeit zur Erarbeitung von Terminologiedatenbanken für Projekte wird im Rahmen des Projekts iglos

derzeit an der TU Braunschweig vom Institut für Verkehrssicherheit und Automatisierungstechnik untersucht [Insa]. Diese können wiederum in Ontologien überführt werden.

Mit den Ontologien Core und RMNet stehen Oberklassen für Artefakte und deren Beziehungen zur Verfügung. Es kann somit begonnen werden, die erarbeiteten Artefakte mit zu kategorisieren, um so einen Kontext zu erzeugen. Hierzu müssen die erkannten Artefakte sowie ihre Beziehungen den übergeordneten Artefakten und Relationen der beiden Ontologien zugeordnet werden. Geschehen kann dies bspw. durch Vererbung, also wiederum der Verwendung einer Relation `rdfs:subClassOf`. Dem Modellierer fällt hierbei die wichtige Aufgabe zu, die erfassten Artefakte möglichst adäquat in der Ontologie abzubilden und die Konsistenz sicher zu stellen. Weiterhin obliegt es ihm, neue Relationen vorhandenen zuzuordnen, bzw. zu prüfen, ob neue nicht durch bereits vorhandene abgebildet werden können.

Um dem in dieser Arbeit postuliertem Top-Down-Ansatz zu folgen, geht es bei der initialen Modellierung vor allem um die Erfassung möglichst vieler relevanter Artefakte, weniger um die genaue und detaillierte Modellierung jedes einzelnen. Im Gegensatz zu den meisten Prozessmodellierungsmethoden und -verfahren ist das semantische Netz bereits leistungsfähig, wenn die bloße Existenz eines Artefakts vorhanden und es in einen Kontext eingeordnet ist. Die Analyse der Auswirkungen von Änderungen kann so bereits betroffene Artefakte identifizieren. Eine Erweiterung der Ontologie mit Detailinformationen ist jederzeit möglich und erhöht offensichtlich die Aussagekraft.

Diskussionswürdig ist der Zeitpunkt der Erstellung von solch umfassenden Unternehmensmodellen. Idealerweise findet diese unabhängig von konkreten Softwareentwicklungsprojekten bspw. durch eine separate Stelle statt. Die Rollenbeschreibung der Modellierer enthält bereits einen Hinweis darauf, diese idealerweise als separates Team zu etablieren. Auf diese Weise existiert ein eigenverantwortliches Gremium, welches auch als Ansprechpartner dient und nicht Gefahr läuft, bei Ende eines Softwareentwicklungsprojekts aufgelöst zu werden. Wird dann ein Projekt durchgeführt, welches das Netz nutzen und ergänzen will, arbeiten die Entwickler mit dieser Stelle zusammen.

6.3.2 Verwendung im Unternehmenskontext

Während der Verwendungsphase existiert die Ontologie als relativ konstantes Konstrukt an einer zentralen Stelle und wird von diversen Systemen, wie z.B. einem Anforderungsmanagementwerkzeug, genutzt. Basierend auf der Ontologie wird eine Wissensbasis erarbeitet, die mit konkreten Informationen aus Projekten gefüllt wird. Die Ontologie stellt dabei ein „Rahmenwerk“ zur Verfügung, welches die Art und Gattung von Artefakten beschreibt. Die ausgeprägten Artefakte werden in der Verwendungsphase in die Wissensbasis eingefügt und mit den vorhandenen Verknüpfungen in einen Kontext gesetzt. Im Sinne der objektorientierten Programmierung kann hierzu die grundlegende Ontologie als Definition von „Klassen“ gesehen werden, die gefüllte Wissensbasis enthält dann wiederum „Objekte“, die vom Typ der Klassen sind. Die Ontologie wird in dieser Phase nicht verändert, lediglich die enthaltenen Objekte werden erstellt, entfernt oder bearbeitet.

Diese Phase ist weiterhin gekennzeichnet von an die Wissensbasis gestellte Abfragen. Es können komplexe Fragestellungen bearbeitet, wie auch simple Listen von Artefakten erstellt

werden. Abschnitt 6.4 gibt eine Einführung in die Möglichkeiten, die sich hieraus für Aufgaben innerhalb eines Projektes ergeben.

6.3.3 Markierungsprozess für Unstimmigkeiten

Eine Gefahr bei der Verwendung von umfangreichen Modellen zur Abbildung der Realität ist – aufgrund fortschreitender Veränderung im Sinne von Kapitel 4.2.2 – der Verfall der Korrektheit des Netzes. Ohne eine ausreichende Pflege der Modelle, während der diese wieder an die Realität angepasst werden, können sich Benutzer zunehmend weniger auf die aufgrund der vorhandenen Informationen getroffenen Aussagen verlassen. Vor einer Korrektur erfolgt zunächst die Erfassung von Fehlern oder Ungenauigkeiten.

Softwaregestütztes „Gardening“, also das automatisierte Finden und auch Korrigieren von Inkonsistenzen in einem semantischen Netz, kann in einem beschränkten Rahmen den Pflegeprozess unterstützen. Hierbei müssen Regeln aufgestellt werden, anhand derer bspw. eine spezielle Software oder aber auch Personen bestimmte Suchabfragen formulieren und die Ergebnisse dann bearbeiten. Diese Methodik ist gut geeignet, um z.B. Klassen ohne Relationen und Instanzen zu finden oder bestimmte kontextabhängige Fälle, wie etwa Anforderungen, die an kein Produkt gebunden sind. Je nach Ausprägung der Regel können die Ergebnisse automatisch bearbeitet, oder zum Überarbeiten vorgeschlagen werden.

Eine weitere Methodik nutzt die zuletzt immer weiter verbreitete Bereitschaft der Nutzer zur Kollaboration untereinander und mit Software. Braun et al. bieten hierzu eine Methodik zur Erstellung und Pflege von Ontologien mithilfe der Nutzer [BSW+07]. Die Autoren schlussfolgern, dass ontologiebasierte Systeme nur leistungsfähig funktionieren können, wenn Domänenexperten eine einfache Möglichkeit zur Be- und Überarbeitung der vorhandenen Informationen haben. Die hier zur Anwendung kommenden Methoden können demnach problemlos auf den Pflegeprozess eines unternehmensinternen Wissensnetzes übertragen werden. Studien zeigen, dass Menschen eher bereit sind, bereits vorhandene Dinge zu ändern bzw. als fehlerhaft zu markieren, als dass sie von Grund auf neue Informationen erstellen [BGG+07]. In der Medizinforschung werden bereits Verfahren eingesetzt, automatisierte Schlussfolgerungen durch Benutzer bewerten zu lassen, um auf diese Weise die Korrektheit zu verifizieren [BGG+07]. Benutzern, die durch verschiedene Werkzeuge auf das semantische Netz zugreifen, muss daher in den jeweiligen Oberflächen die Möglichkeit zur Markierung von fehlerhaften Elementen zur Verfügung gestellt werden.

6.3.4 Prüfung und Überarbeitung zur Aktualisierung des Netzes

Die – entweder manuell oder durch ein softwaregestütztes Verfahren – markierten Stellen müssen an zentraler Stelle gesammelt und durch das Modelliererteam regelmäßig geprüft werden. Nach der Prüfung findet eine Bewertung statt, inwieweit eine markierte Stelle überarbeitet werden muss. Die Prüfung ist offensichtlich oftmals fachlicher Natur. Kann die Inkonsistenz also nicht durch einen Modellierer bewertet werden, muss der Dialog mit dem betroffenen Fachbereich gesucht werden. Gemeinsam können Modellierer und Domänenexperte dann eine Einordnung vornehmen.

Ist eine Überarbeitung notwendig, sind zwei Fälle zu unterscheiden: Enthält eine Markierung einen konkreten Vorschlag zur Auflösung des Problems oder ist die Behebung trivial, so kann

sie direkt durch den Modellierer vorgenommen werden. Sind die notwendigen Änderungen umfangreicher, müssen eventuell die Methoden aus dem ersten Schritt erneut angewendet werden.

6.4 Anwendung auf eine exemplarische Artefaktlandschaft für die Produktentwicklung

Basierend auf den Ausführungen in diesem Kapitel wird ein exemplarisches Modell aufgebaut, welches die Umsetzung der Anforderungen an ein Werkzeug für ein durchgängiges und semantisch unterstütztes Anforderungsmanagement aus Tabelle 5.1 demonstriert. Die vorhergehenden Abschnitte in diesem Kapitel beschreiben die technischen, organisatorischen und prozessualen Grundlagen für den Einsatz eines solchen Werkzeugs. Ziel dieses Abschnitts ist nun einerseits die Demonstration der Verwendung der Core-Ontologie aus Abschnitt 6.2.2 und andererseits die Entwicklung einer generischen Artefaktlandschaft, welche in der Produktdefinition von Fahrzeugprojekten eingesetzt werden kann [BS11]. Die theroretisch beschriebenen Möglichkeiten werden somit nun mit praktischen Einsatzszenarien belegt.

6.4.1 Erarbeitung der spezifischen Artefakte

Die Entwicklung der Ontologie beginnt gemäß des in Abschnitt 6.3 definierten Vorgehens mit einer Analyse der minimal benötigten Artefakte. Es wird hierbei kein Anspruch auf Vollständigkeit erhoben, sondern eine nötige Reduktion vorgenommen. Im Gegensatz zu Bottom-Up-Verfahren, die von unterer Arbeitsebene ausgehend detaillierte Prozess-Informationen zusammentragen, hat sich dieser Top-Down-Ansatz im Rahmen des Volkswagen-Projektes als effektiv erwiesen. Grund ist, dass Mitarbeiter zwar Artefakte, von denen sie direkt betroffen sind detailliert kennen, das Wissen über vor- oder nachgelagerte Artefakte jedoch oftmals eingeschränkt ist. Beim Top-Down-Ansatz ist zunächst die Existenz eines Artefakts wichtig, nicht notwendigerweise sein Aufbau, wodurch bereits ein wirkungsvolles Netz konstruiert werden kann. Die Verwendung einer allgemein verwendbaren Sprache ermöglicht darüber hinaus die spätere Ergänzung oder Verfeinerung des Artefaktmodells, falls die Notwendigkeit besteht. Die Repräsentation der Ontologie in OWL findet sich in Listing B.3 im Anhang B.

Die Verwendung einer einzelnen und einfachen Beschreibung eines Projekts erhöht die Chancen auf Erfolg [HF00]. Zu Beginn eines Entwicklungsprojekts steht somit das Produktkonzept, welches den groben Rahmen des Produkts umreißt, um ein klares Verständnis für alle Beteiligten zu schaffen. Dieses Artefakt *Produktkonzept* ist somit ein `Dokument` und besteht aus den Strukturelementen `Beschreibung`, `Fahrzeugklasse` und `Konzept-bestimmende Maße`. Zusätzlich erhält das Projekt einen `Zieltermin`. Die einzelnen Strukturelemente werden mit dem `Produktkonzept` durch Relationen verbunden, die jeweils von der RMNet-Relation `hat_Strukturelement` abgeleitet sind, welche wiederum eine Sub-Relation von `besteht_aus` der Core-Ontologie ist. In Listing B.3 in Anhang B findet sich das Konzept des `Entwicklungskonzepts` ab Zeile 234 ff.

Durch das gesamte Projekt zieht sich die Sammlung von `Anforderungen` an das `Produkt`. Die RMNet-Ontologie stellt hierzu bereits ausreichende Möglichkeiten zur Erfassung bereit.

An dieser Stelle wird allerdings darüber hinaus der sog. *Anforderungskatalog* in das Modell aufgenommen, da er eine einfache Möglichkeit zur Sammlung der Anforderungen darstellt.

Im weiteren Verlauf des Projekts wird der bereits genannte *Eigenschaftskatalog* [Rin06], bestehend aus einzelnen *Eigenschaften* erarbeitet. Die OWL-Repräsentation findet sich in Listing B.3 ab Zeile 197 ff. Er ist ebenfalls ein `Dokument`, die *Eigenschaften* sind die bereits im RMNet definierten `Produkteigenschaften`, welche nun mit einer Relation in Zeile 88 an den Eigenschaftskatalog gekoppelt werden.

Ziel ist, jede erfasste Anforderung durch eine Produkteigenschaft zu erfüllen. Es folgt demnach die detaillierte Ausplanung der Informationen und deren Verknüpfung mit den Anforderungen, die im *Produktkonzept* nur grob skizziert wurden. Der Eigenschaftskatalog existiert hierbei als übergeordnete Klammer zur Bereitstellung von Metainformationen. Änderungen an ihm betreffen daher keine einzelnen Produkteigenschaften.

Die Struktur, welche den *Eigenschaftskatalog* gliedert, wird als *Eigenschaftsstruktur* bezeichnet, Objekte von beiden können unter Verwendung der Relationen `strukturiert` und `wird_strukturiert_durch` der Core-Ontologie verknüpft werden.

Fahrzeuge werden in Milieus eingeordnet, welche bestimmte Gruppen von Käufern zusammenfassen. Ziel ist, durch die Ausprägung bestimmter Eigenschaften eines Fahrzeugs, dieses für ein oder mehrere Milieus gezielt attraktiv zu gestalten. Dies geschieht, indem an spezielle Werte appelliert wird, die den Kunden eines ganz bestimmten Milieus zugeschrieben werden [Asc06]. Ein *Milieu* wird daher mit einem *Wert* durch die Relation *hat_Wert* verknüpft. Marktanalysen können diese beiden Artefakte sehr genau einander zuordnen. *Wert* wird wiederum mit *Produkteigenschaft* durch *bedient* bzw. *wird_bedient_durch* verknüpft. Dieser Themenkomplex wird in Listing B.3 ab Zeile 222 ff. behandelt.

Zur Erfassung von Lösungen und Konzepten der Realisierung wird eine *Technikliste* benötigt. Diese enthält `AbstractUmfaenge` der RMNet-Ontologie, welche eine abstrakte Darstellung eines Konzepts sind und in der frühen Entwickungsphase stetig zahlreicher werden. Sie repräsentieren ein Konzept für eine Lösung oder wahlweise die Lösung selbst. Ein Konzept wäre bspw. die Aussage „Lenkrad der neuesten Generation“, eine Lösung hingegen ein Teil mit einer konkrete Teilenummer. Auf diese Weise wird ein Fahrzeugkonzept zusammengestellt, welches einerseits aus noch zu entwickelnden Elementen und andererseits aus bereits vorhandenen Teilen besteht. Die RMNet-Ontologie bietet als Detaillierung von `AbstractUmfang` die Klassen `Teil`, `Modul` und `Software`, welche wieder in der `Technikliste` eingesetzt werden können und `Produkteigenschaften` realisieren.

Ein modularer Baukasten ist gemäß Definition 3.1.2 eine Zusammenfassung von Modulen. Entsprechend wird ein neues Artefakt *ModularerBaukasten* benötigt, welches eine Sammlung bestimmter Module aus verschiedenen Projekten enthält. Gemäß der Definition der Kernartefakte in Abschnitt 6.2.1 stellt auch ein solcher Baukasten wieder ein `Dokument` dar. Mit der Sammlung bestimmter entwickelter Module in einem Baukasten ist die Kette der Produktentwicklung – von der Erstellung des Projekts über die Erfassung der Anforderungen und deren Umsetzung in passende Lösungen hin zur Wiederverwendung von entwickelten Elementen – komplett.

Die folgende Abbildung 6.12 ist eine reduzierte graphische Repräsentation des semantischen Netzes. Zur Verbesserung der Lesbarkeit wurde wie in vorhergehenden Abbildungen insbesondere auf das Bild stark kreuzender Relationen verzichtet. Hauptsächlich sind dies Relationen des

Typs `rdfs:subClassOf` oder `rdfs:subPropertyOf`. Eine vollständige Darstellung kann aus Listing B.3 generiert werden. Die Abbildung macht deutlich, dass Artefakte eine Vielzahl von Verknüpfungen besitzen können. Die an dieser Stelle entwickelte Ontologie ist ein stark reduziertes Beispiel, Artefakte im realen Unternehmenskontext weisen einen deutlich höheren Verknüpfungsgrad auf.

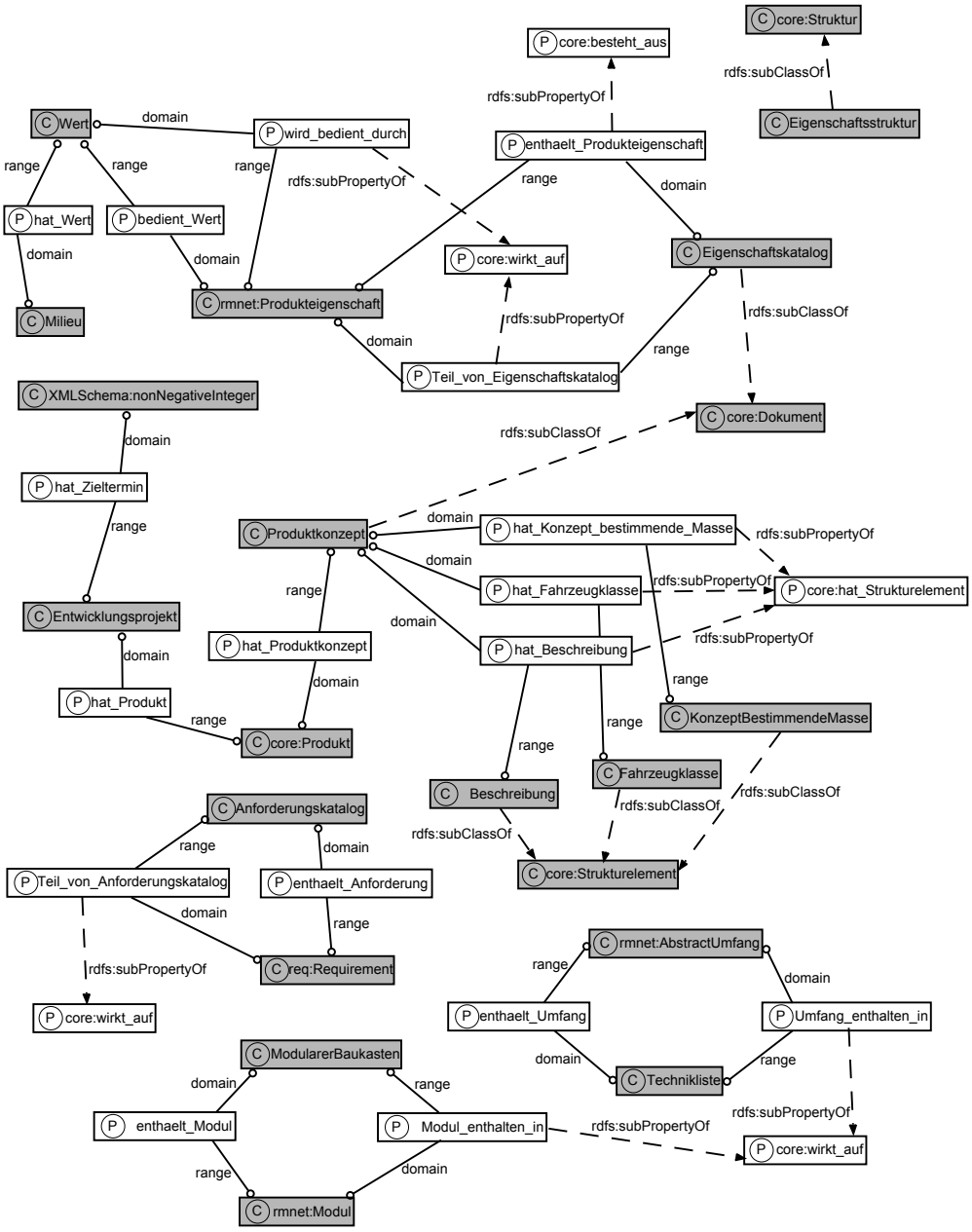


Abbildung 6.12: Eine graphische Repräsentation der Ontologie für die Produktentwicklung ohne Visualisierung der stark kreuzenden Relationen

6.4.2 Einbindung des exemplarischen Artefaktnetzes in die Produktentwicklung

Das Netz bedingt durch seine Ausgestaltung bereits eine bestimmte Arbeitsweise in der Durchführung von Projekten. Veränderungen und Erweiterungen sind aufgrund der Repräsentation in OWL möglich und im produktiven Einsatz notwendig. Denkbar ist die Verbindung der Artefakte mit Aufgaben und zugehörigen Plan- und Ist-Terminen für das Projektmanagement. Zur Demonstration der Arbeitsweise des semantischen Netzes mit Hilfe des vorgegebenen Prozesses wird mit einem fiktiven Fahrzeugprojekt die Befüllung des Netzes mit Objekten gezeigt. Die jeweils relevanten Quelltext-Beispiele werden in OWL oder SPARQL einzelnen Blöcken hinzugefügt und der Quelltext dann jeweils erweitert. Der zusammenhängende finale Quelltext ist im Anhang B zu finden. Im produktiven Einsatz ist die Verwaltung des Netzes durch ein Werkzeug zu übernehmen, wie es im Kapitel 5 beschrieben wurde. Die Darstellung der Entwicklung der einzelnen Objekte auf Quelltextebene gibt einen Einblick in die Arbeitsweise eines solchen Werkzeugs und zeigt die grundsätzliche Operationalisierbarkeit.

Im ersten Schritt wird das Fahrzeugprojekt „Golf“ als solches im Anforderungsmanagementwerkzeug angelegt. Dieses erstellt ein Objekt diesen Typs wie in Listing 6.11 dargestellt. Das Entwicklungsprojekt hat ein Produkt *Golf* zum Ergebnis, welches durch einen *Produktentstehungsprozess* entwickelt wird. Im Beispiel ist dieser Prozess bereits definiert und besteht aus den `Prozessschritten` *Erstellung Produktkonzept*, *Anforderungserhebung*, *Erstellung Eigenschaftskatalog*, *Technische Realisierung* und *Qualitätssicherung*. Die Zuordnung eines Produkts zum Prozess kann durch eine einfache Menüführung vorgenommen werden. Denkbar ist hier bspw. ein sog. Wizard, der das Anlegen eines neuen Projekts für den Benutzer übernimmt und es gleich entsprechend verknüpft.

Listing 6.11: Das Entwicklungsprojekt Golf

```
proj:EntwicklungGolf
    rdf:type
        art:Entwicklungsprojekt ,
        owl:NamedIndividual ;
    art:hat_Produkt proj:Golf .


proj:Golf
    rdf:type
        core:Produkt ,
        owl:NamedIndividual ;
    core:wird_erzeugt_durch proj:Produktentstehungsprozess .


proj:Produktentstehungsprozess
    rdf:type
        core:Prozess ,
        owl:NamedIndividual ;
```

```
19      core:erzeugt proj:Golf ;
20      core:hat_Prozessschritt
21          proj:ErstellungProduktkonzept ,
22          proj:Anforderungserhebung  ,
23          proj:TechnischeRealisierung ,
24          proj:Qualitaetssicherung .
25
26  proj:Anforderungserhebung
27      rdf:type
28          core:Prozessschritt ,
29          owl:NamedIndividual .
30
31
32  proj:ErstellungProduktkonzept
33      rdf:type
34          core:Prozessschritt ,
35          owl:NamedIndividual .
36
37
38  proj:TechnischeRealisierung
39      rdf:type
40          core:Prozessschritt ,
41          owl:NamedIndividual .
42
43
44  proj:Qualitaetssicherung
45      rdf:type
46          core:Prozessschritt ,
47          owl:NamedIndividual .
```

Nach dem Anlegen des Projekts kann mit der Erstellung des Produktkonzepts begonnen werden. Hierzu wird dieses angelegt und mit dem Produkt *Golf* verknüpft. Zusätzlich wird es dem zugehörigen Prozessschritt als Ergebnis zugeordnet, wie in Listing 6.12 gezeigt.

Listing 6.12: Produktkonzept

```
1   proj:Golf
2       rdf:type
3           core:Produkt ,
4           owl:NamedIndividual ;
5       art:hat_Produktkonzept proj:KonzeptGolf ;
6       core:wird_erzeugt_durch proj:Produktentstehungsprozess ;
7
8   proj:KonzeptGolf
9       rdf:type
```

```
        art:Produktkonzept ,
        owl:NamedIndividual .

proj:ErstellungProduktkonzept
    rdf:type
        core:Prozessschritt ,
        owl:NamedIndividual ;
    core:hat_Ergebnis proj:KonzeptGolf .
```

Nachdem der grobe Rahmen des Produkts im Produktkonzept beschrieben ist, kann mit der Sammlung konkreter Anforderungen begonnen werden. An dieser Stelle werden exemplarisch einige fiktive Anforderungen erstellt, welche als Beispiele zur Verwendung dienen sollen:

1. Das Fahrzeug soll durch den Einsatz aktiver Sicherheitssysteme einen Sicherheitsstandard bieten, der einem Mittelklassewagen angemessen ist.
2. Das Fahrzeug soll ein für seine Fahrzeugklasse durchschnittliches Fahrverhalten bieten.
3. Das Fahrzeug soll einen um 10% höheren Marktanteil im Vergleich zum Vorgänger erreichen.

Diese Anforderungen werden vom Benutzer auf die in Abschnitt 5.2.1 beschriebene Weise eingegeben. Bestimmte Textteile wie etwa „Fahrzeugklasse“ oder „Vorgänger“ können als Verknüpfung mit anderen bereits vorhandenen Objekten versehen werden, um so Informationen schnell zur Verfügung zu stellen. Die Anforderungen selbst werden einem Anforderungskatalog zugeordnet sowie mit dem Produkt verknüpft. Der Anforderungskatalog stellt das Ergebnis des Prozessschritts *Anforderungserhebung* dar. Listing 6.13 zeigt die zugehörigen Veränderungen im OWL-Quelltext.

Listing 6.13: Anforderungen und Anforderungskatalog

```
proj:Golf
    rdf:type
        core:Produkt ,
        owl:NamedIndividual ;
    rmnet:hat_Anforderung
        proj:Fahrverhalten ,
        proj:Marktanteil ,
        proj:Sicherheitssysteme ;
    art:hat_Produktkonzept proj:KonzeptGolf ;
    rmnet:hat_Anforderung  ;
    core:wird_erzeugt_durch proj:Produktentstehungsprozess .


proj:AnforderungskatalogGolf
    rdf:type
```

```
16          art:Anforderungskatalog ,
17          owl:NamedIndividual ;
18      art:enthaelt_Anforderung
19          proj:Fahrverhalten ,
20          proj:Marktanteil ,
21          proj:Sicherheitssysteme .
22
23
24  proj:Fahrverhalten
25      rdf:type
26          req:Requirement ,
27          owl:NamedIndividual ;
28      dcterms:title
29          "Durchschnittliches_Fahrverhalten" ;
30      dcterms:description
31          "Das Fahrzeug soll ein für seine Fahrzeugklasse
               durchschnittliches Fahrverhalten bieten." ;
32      rmnet:Anforderung_definiert_fuer proj:Golf .
33
34
35  proj:Marktanteil
36      rdf:type
37          req:Requirement ,
38          owl:NamedIndividual ;
39      dcterms:title
40          "Gesteigerter Marktanteil" ;
41      dcterms:description
42          "Das Fahrzeug soll einen um 10% hoeheren Marktanteil
               im Vergleich zum Vorgaenger erreichen." ;
43      rmnet:Anforderung_definiert_fuer proj:Golf .
44
45
46  proj:Sicherheitssysteme
47      rdf:type
48          req:Requirement ,
49          owl:NamedIndividual ;
50      dcterms:title
51          "Einsatz aktiver Sicherheitssysteme" ;
52      dcterms:description
53          "Das Fahrzeug soll durch den Einsatz aktiver
               Sicherheitssysteme einen Sicherheitsstandard
               bieten, der einem Mittelklassewagen angemessen ist
               ." ;
```

```
54      rmnet:Anforderung_definiert_fuer proj:Golf .
```

Nach Erhebung der Anforderungen beginnt die Suche nach Lösungen. Zur detaillierten Ausplanung des Fahrzeugs werden die `Produkteigenschaften` auf Basis der Anforderungen festgelegt. Diese sind:

1. Das Fahrzeug erhält als aktives Sicherheitssystem einen Notbremsassistenten.
2. Das Fahrzeug erhält als aktives Sicherheitssystem einen Spurhalteassistenten.
3. Das Fahrzeug erhält das Fahrwerk des Vorgängers mit einem um 5% reduzierten Bremsweg durch Neuentwicklung der Bremsanlage.
4. Das Fahrzeug erhält Voll-LED-Scheinwerfer zur Erreichung weiterer Zielgruppen.

Entsprechend findet eine Verknüpfung mit einem Eigenschaftskatalog sowie mit den Anforderungen statt, die die `Produkteigenschaften` erfüllen, wie Listing 6.14 zeigt. Die `Anforderungen` werden ergänzt um die Relation `wird_erfuellt_durch`, welche sie mit den `Produkteigenschaften` verknüpft.

Listing 6.14: Anforderungen und Eigenschaften

```
1
2  proj:EigenschaftskatalogGolf
3      rdf:type
4          art:Eigenschaftskatalog ,
5          owl:NamedIndividual ;
6      art:enthaelt_Produkteigenschaft
7          proj:EinsatzNotbremsassistent ,
8          proj:EinsatzSpurhalteassistent ,
9          proj:VerbauLEDScheinwerfer ,
10         proj:VerbesserungBremsanlage .
11
12
13 proj:EinsatzNotbremsassistent
14     rdf:type
15         rmnet:Produkteigenschaft ,
16         owl:NamedIndividual ;
17     dcterms:title
18         "Einsatz Notbremsassistent im Fahrzeug" ;
19     dcterms:description
20         "Das Fahrzeug erhaelt als aktives Sicherheitssystem
              einen Notbremsassistenten." ;
21     art:Teil_von_Eigenschaftskatalog
           proj:EigenschaftskatalogGolf ;
22     rmnet:erfuellt proj:Sicherheitssysteme .
```

```
23
24
25 proj:EinsatzSpurhalteassistent
26     rdf:type
27         rmnet:Produkteigenschaft ,
28         owl:NamedIndividual ;
29     dcterms:title
30         "Einsatz Spurhalteassistent im Fahrzeug" ;
31     dcterms:description
32         "Das Fahrzeug erhaelt als aktives Sicherheitssystem
              einen Spurhalteassistenten." ;
33     art:Teil_von_Eigenschaftskatalog
          proj:EigenschaftskatalogGolf ;
34     rmnet:erfuellt proj:Sicherheitssysteme .
35
36
37 proj:VerbauLEDScheinwerfer
38     rdf:type
39         rmnet:Produkteigenschaft ,
40         owl:NamedIndividual ;
41     dcterms:title
42         "Einsatz Voll-LED-Scheinwerfer" ;
43     dcterms:description
44         "Das Fahrzeug erhält Voll-LED-Scheinwerfer zur
              Erreichung erweiterter Zielgruppe." ;
45     art:Teil_von_Eigenschaftskatalog
          proj:EigenschaftskatalogGolf ;
46     rmnet:erfuellt proj:Marktanteil .
47
48
49 proj:VerbesserungBremsanlage
50     rdf:type
51         rmnet:Produkteigenschaft ,
52         owl:NamedIndividual ;
53     dcterms:title
54         "Vorgaenger-Fahrwerk und Verbesserung der Bremsanlage
              " ;
55     dcterms:description
56         "Das Fahrzeug erhaelt Fahrwerk des Vorgängers mit
              einem um 5% reduzierten Bremsweg durch
              Neuentwicklung der Bremsanlage." ;
57     art:Teil_von_Eigenschaftskatalog
          proj:EigenschaftskatalogGolf ;
```

```
58      rmnet:erfuellt proj:Fahrverhalten .
59
60  proj:Marktanteil
61      rdf:type
62          req:Requirement ,
63          owl:NamedIndividual ;
64      dcterms:title
65          "Gesteigerter Marktanteil" ;
66      dcterms:description
67          "Das Fahrzeug soll einen um 10% hoeheren Marktanteil
               im Vergleich zum Vorgaenger erreichen." ;
68      rmnet:Anforderung_definiert_fuer proj:Golf ;
69      rmnet:wird_erfuellt_durch proj:VerbauLEDScheinwerfer .
70
71
72  proj:Sicherheitssysteme
73      rdf:type
74          req:Requirement ,
75          owl:NamedIndividual ;
76      dcterms:title
77          "Einsatz aktiver Sicherheitssysteme" ;
78      dcterms:description
79          "Das Fahrzeug soll durch den Einsatz aktiver
               Sicherheitssysteme einen Sicherheitsstandard
               bieten, der einem Mittelklassewagen angemessen ist
               ." ;
80      rmnet:wird_erfuellt_durch
81          proj:EinsatzNotbremsassistent ,
82          proj:EinsatzSpurhalteassistent ;
83      rmnet:Anforderung_definiert_fuer proj:Golf .
84
85
86  proj:Fahrverhalten
87      rdf:type
88          req:Requirement ,
89          owl:NamedIndividual ;
90      dcterms:title
91          "Durchschnittliches_Fahrverhalten" ;
92      dcterms:description
93          "Das Fahrzeug soll ein für seine Fahrzeugklasse
               durchschnittliches Fahrverhalten bieten." ;
94      rmnet:Anforderung_definiert_fuer proj:Golf ;
95      rmnet:wird_erfuellt_durch proj:VerbesserungBremsanlage .
```

Die Produkteigenschaften können dann über Werte mit Milieus verbunden werden. Da die Redundanz des an dieser Stelle aufgeführten Quelltextes ab jetzt stark ansteigt, wird nur ein exemplarisches Beispiel in Listing 6.15 gezeigt. Das vollständige Listing des Quelltextes findet sich in Anhang B in Listing B.4. Milieus sind an dieser Stelle Gruppierungen von Personen mit bestimmten gleichen Eigenschaften [Asc06].

Listing 6.15: Werte und Milieus

```
proj:EinsatzNotbremsassistent
    rdf:type
        rmnet:Produkteigenschaft ,
        owl:NamedIndividual ;
    dcterms:title
        "Einsatz Notbremsassistent im Fahrzeug" ;
    dcterms:description
        "Das Fahrzeug erhaelt als aktives Sicherheitssystem
           einen Notbremsassistenten." ;
    art:Teil_von_Eigenschaftskatalog
       proj:EigenschaftskatalogGolf ;
    art:bedient_Wert proj:Sicherheit ;
    rmnet:erfuellt proj:Sicherheitssysteme .


proj:Sicherheit
    rdf:type
        art:Wert ,
        owl:NamedIndividual ;
    art:wird_bedient_durch proj:EinsatzNotbremsassistent ,
    proj:EinsatzSpurhalteassistent .


proj:BuergerlicheMitte
    rdf:type art:Milieu ,
    owl:NamedIndividual ;
    art:hat_Wert proj:Sicherheit .
```

Die Verknüpfung eines `Milieus` mit `Werten` und darüber wiederum mit `Produkteigenschaften` zeigt, wie mit Hilfe von semantischen Netzen eine genaue Positionierung eines Fahrzeugs möglich ist. Durch die Einbindung dieser Herangehensweise in ein semantisches Netz und somit in das Anforderungsmanagementwerkzeug können auch Auswirkungen auf die Positionierung des Fahrzeugs antizipiert werden.

Die `Produkteigenschaften` müssen schließlich durch eine technische Komponente realisiert werden. Diese wird in der RMNet-Ontologie durch die Klasse `AbstractUmfang` dargestellt, welche es ermöglicht, technische Konzepte in einem frühen Stadium anzulegen und später zu detaillieren. So kann bspw. aus einem `AbstractUmfang` erst ein `Teil` und später ein

`Modul` werden. Die `Technikliste` akzeptiert als Inhalt alle Typen von `AbstractUmfang` und kann so durchgängig im Entwicklungsprojekt verwendet werden. Listing 6.16 zeigt die beschriebenen Zusammenhänge auszughaft im Quelltext. Hierbei sind die *LEDScheinwerfer* in Zeile 29 als abstraktes Konzept beschrieben, das erst noch entwickelt werden muss. Der *Spurhalteassistent* in Zeile 36 hingegen liegt bereits als konkretes Modul vor. Beide werden dennoch durch die Relation `realisiert_durch` in den beiden `Produkteigenschaften` verknüpft, siehe Zeilen 11 und 26.

Listing 6.16: Umfänge und Eigenschaften

```
1  proj:VerbauLEDScheinwerfer
2      rdf:type
3          rmnet:Produkteigenschaft ,
4          owl:NamedIndividual ;
5      dcterms:title
6          "Einsatz Voll-LED-Scheinwerfer" ;
7      dcterms:description
8          "Das Fahrzeug erhält Voll-LED-Scheinwerfer zur
              Erreichung erweiterter Zielgruppe." ;
9      art:Teil_von_Eigenschaftskatalog
          proj:EigenschaftskatalogGolf ;
10     art:bedient_Wert proj:Exklusivitaet ;
11     rmnet:realisiert_durch proj:LEDScheinwerfer ;
12     rmnet:erfuellt proj:Marktanteil .
13
14
15 proj:EinsatzSpurhalteassistent
16     rdf:type
17         rmnet:Produkteigenschaft ,
18         owl:NamedIndividual ;
19     dcterms:title
20         "Einsatz Spurhalteassistent im Fahrzeug" ;
21     dcterms:description
22         "Das Fahrzeug erhaelt als aktives Sicherheitssystem
              einen Spurhalteassistenten." ;
23     art:Teil_von_Eigenschaftskatalog
          proj:EigenschaftskatalogGolf ;
24     art:bedient_Wert proj:Sicherheit ;
25     rmnet:erfuellt proj:Sicherheitssysteme ;
26     rmnet:realisiert_durch proj:Spurhalteassistent .
27
28
29 proj:LEDScheinwerfer
30     rdf:type
31         rmnet:AbstractUmfang ,
```

```
32          owl:NamedIndividual ;
33      rmnet:realisiert proj:VerbauLEDScheinwerfer .
34
35
36  proj:Spurhalteassistent
37      rdf:type
38          rmnet:Modul ,
39          owl:NamedIndividual ;
40      rmnet:realisiert proj:EinsatzSpurhalteassistent .
```

Die `Technikliste` in Zeile 9 des Listings 6.17 enthält alle `abstrakten Umfänge`.

Listing 6.17: Technikliste

```
1   proj:LEDScheinwerfer
2       rdf:type
3           rmnet:AbstractUmfang ,
4           owl:NamedIndividual ;
5       art:Umfang_enthalten_in proj:TechniklisteGolf ;
6       rmnet:realisiert proj:VerbauLEDScheinwerfer .
7
8
9   proj:TechniklisteGolf
10      rdf:type
11          art:Technikliste ,
12          owl:NamedIndividual ;
13      art:enthaelt_Umfang
14          proj:LEDScheinwerfer ,
15          proj:Notbremsassistent ,
16          proj:Spurhalteassistent .
```

Für jede Anforderung müssen wiederum Tests definiert werden, wie in Listing 6.18 gezeigt. Zeile 14 zeigt die Definition des Tests, Zeile 10 demonstriert die Verknüpfung des Tests mit der Anforderung:

Listing 6.18: Tests

```
1   proj:Fahrverhalten
2       rdf:type
3           req:Requirement ,
4           owl:NamedIndividual ;
5       dcterms:title
6           "Durchschnittliches_Fahrverhalten" ;
7       dcterms:description
8           "Das Fahrzeug soll ein für seine Fahrzeugklasse
               durchschnittliches Fahrverhalten bieten." ;
```

```
    rmnet:Anforderung_definiert_fuer proj:Golf ;
    rmnet:wird_getestet_durch proj:TestFahrverhalten ;
    rmnet:wird_erfuellt_durch proj:VerbesserungBremsanlage .


proj:TestFahrverhalten
    rdf:type
        rmnet:Test ,
        owl:NamedIndividual
```

Abschließend werden zwei `Module` in Listing 6.19 in einem `modularen Baukasten` zusammengefasst.

Listing 6.19: Modularer Assistentenbaukasten

```
proj:Notbremsassistent
    rdf:type
        rmnet:Modul ,
        owl:NamedIndividual ;
    art:Modul_enthalten_in proj:ModularerAssistentenBaukasten
        ;
    art:Umfang_enthalten_in proj:TechniklisteGolf .

proj:ModularerAssistentenBaukasten
    rdf:type
        art:ModularerBaukasten ,
        owl:NamedIndividual ;
    art:enthaelt_Modul proj:Notbremsassistent ;
    art:enthaelt_Umfang proj:Spurhalteassistent .
```

6.4.3 Beispielhafte SPARQL-Abfragen zur Demonstration der Möglichkeiten der Wissensbasis

In Abschnitt 6.4.1 wurde eine exemplarische Ontologie zum Einsatz in der Fahrzeugproduktentwicklung erarbeitet. Der nachfolgende Abschnitt 6.4.2 erzeugt ein semantisches Netz aus dieser Ontologie, indem Objekte auf der Basis eines fiktiven Entwicklungsprojekts generiert werden. Nun wird mithilfe dieser Wissensbasis demonstriert, wie sich einige der in Abschnitt 5.3 benannten primären und in Abschnitt 5.5 erweiterten Nutzungsmöglichkeiten äußern können.

Gezeigt wird weiterhin, wie einige der in Abschnitt 5.2 definierten Anforderungen an ein semantisch unterstütztes Anforderungsmanagement bereits durch den Einsatz von OWL und SPARQL sowie den Ontologien Core und RMNet erfüllt werden. Vier dieser Anforderungen werden allein durch den Einsatz der Wissensbasis und der Ausführung von Abfragen hierauf erfüllt. Hierbei ist die Erfüllung von Anforderung Nr. 1 (Verwendung eines Metamodells) bereits per definitionem durch den Einsatz einer Ontologie sichergestellt. Die Rückgabe von Ergebnismengen

von Abfragen in Listenform erfüllt Anforderung Nr. 6 (Zusammenstellen und Konfigurieren von Listen und Dokumenten aus Artefaktinformationen). Die Generierurung der Inhalte dieser Listen wird durch das Ausführen von Abfragen durchgeführt, auf denen der Fokus in diesem Abschnitt liegt. Sie erfüllen damit Anforderung Nr. 14 (Konstruktion von Suchabfragen). Durch das Ausführen der Abfragen mit aktiviertem Reasoning wird somit auch die Anforderung Nr. 15 (Automatisches Schlussfolgern) erfüllt.

Projektmanagement

Die Ermittlung des Reifegrads gemäß Definition 5.2.1 eines Projekts kann einerseits anhand der fertiggestellten Arbeitspakete durchgeführt werden. Andererseits muss die Menge an erfassten Aufgaben nicht unbedingt vollständig sein, weshalb eine Ermittlung anhand der bereits bearbeiteten Anforderungen ein fundierteres Ergebnis liefern kann. Dies wird erreicht, indem etwa die Menge an noch nicht durch einen `AbstractUmfang` abgedeckte `Anforderungen` ermittelt wird und sie der Menge an bereits abgedeckten gegenübergestellt wird. Die Abfrage aller nicht abgedeckten `Anforderungen` kann folgendermaßen aussehen:

Listing 6.20: SPARQL-Abfrage zur Ermittlung aller nicht abgedeckten Anforderungen

```
1  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2  PREFIX rm: <http://timguelke.de/rmnet.owl#>
3  PREFIX req: <http://ns.softwiki.de/req#>
4
5  SELECT ?anf
6  WHERE
7   {
8    {
9     ?anf rdf:type req:Requirement .
10    OPTIONAL {?anf rm:wird_erfuellt_durch ?eig }
11    FILTER ( ! bound (?eig))
12   }
13   UNION
14   {
15    ?anf rm:wird_erfuellt_durch ?eig .
16    OPTIONAL {?eig rm:realisiert_durch ?umf.}
17    FILTER ( ! bound (?umf))
18   }
19  }
```

Es werden hierbei zwei unterschiedliche Fälle untersucht: Ab Zeile 9 wird geprüft, ob `Anforderungen` existieren, die nicht durch `Produkteigenschaften` abgedeckt sind. Der zweite Fall ab Zeile 15 klärt, ob `Anforderungen` existieren, die zwar durch `Produkteigenschaften` `erfüllt`, aber nicht durch `AbstractUmfänge` `realisiert` werden. Der Reifegrad dieses Projekts setzt sich somit zusammen aus `Anforderungen`, die nicht `realisiert` sind. Das Ergebnis dieser Abfrage angewendet auf das beschriebene semantische

Netz lautet wie folgt:

anf
fahrzeug-projekt:AmbienteInnenraum
fahrzeug-projekt:Fahrverhalten
fahrzeug-projekt:Sicherheitssysteme

Tabelle 6.13: Ergebnis Abfrage aus Listing 6.20

Die folgende Abbildung 6.14 zeigt, warum drei `Anforderungen` im Abfrage-Ergebnis enthalten sind. Für das `AmbienteInnenraum` ist das Resultat trivial: Es verfügt über keine zugehörige `Produkteigenschaft`. Die `Anforderung` `Fahrverhalten` verfügt zwar über eine `Produkteigenschaft`, jedoch nicht über `AbstractUmfänge`, die diese realisieren. Bei den `Sicherheitssystemen` hingegen existiert zwar eine Realisierung durch einen `AbstractUmfang` für die `Produkteigenschaft EinsatzSpurhalteassistent`, jedoch nicht für die `Produkteigenschaft EinsatzNotbremsassistent`. Die `Anforderung` ist daher nicht vollständig umgesetzt.

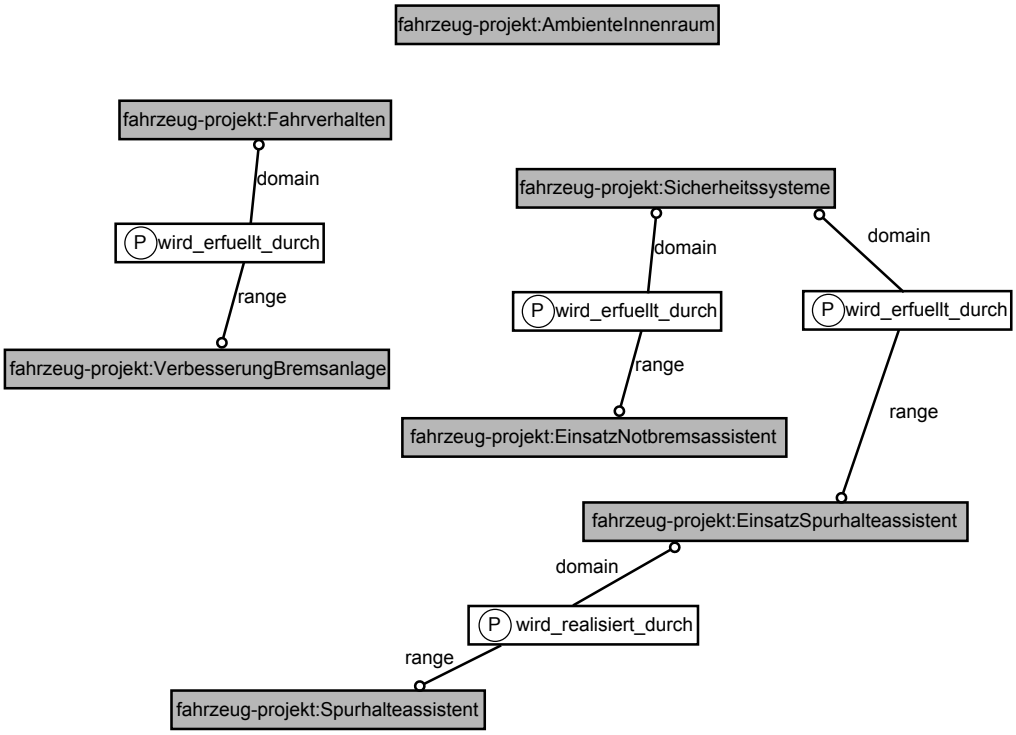


Abbildung 6.14: Eine graphische Repräsentation der SPARQL-Abfrage zum Reifegrad

Hervorzuheben ist hierbei, dass es nicht relevant ist, welchen expliziten Typs die `Abstract-Umfänge` sind. Die Realisierung kann durch `Software`, `Teile` oder auch `Module` erfolgt sein, die Abfrage ermittelt trotzdem zuverlässig den Reifegrad auf Basis der angegebenen Relationen.

Eigenschaftsplanung im Kontext des Änderungs- und Baukastenmanagements

Fahrzeuge werden für eine bestimmte Zielgruppe entwickelt. Dementsprechend verfügen sie über ganze bestimmte Eigenschaften in Design, Funktion, etc., durch die ihnen sportliche, jugendliche oder besonders alltagstaugliche Attribute zugerechnet werden. Korrespondieren die Eigenschaften gut mit den – angenommenen – Werten der jeweiligen Zielgruppe, besteht eine hohe Wahrscheinlichkeit für den Erfolg beim Vertrieb des Fahrzeugs. Eine wichtige Aufgabe ist daher, die `Produkteigenschaften` passend den jeweiligen `Werten` zuzuordnen und während des Projektverlaufs dafür Sorge zu tragen, dass die Zuordnung bestehen bleibt. Eine Änderung der `Anforderungen` kann zu anderen `Produkteigenschaften` führen, die nicht mehr den „benötigten“ `Werten` zugeordnet sind. Während der Planungsphase hingegen ist das Ziel, einen möglichst hohen Verwendungsgrad eines modularen Fahrzeugbaukastens zu erreichen und gleichzeitig keine Module einzubeziehen, die nicht mit der angestrebten Zielgruppe korrespondieren. Dies ist bspw. bei unterschiedlichen Motor-Ausstattungen der Fall: Ein Mittelklasse-Wagen für das familiäre Umfeld soll z.B. mit einem sparsamen und möglichst wartungsarmen Aggregat ausgestattet werden, während ein Sportwagen für Singles über erhöhte Leistung verfügen muss. Die Eigenschaftsplanung für ein neues Fahrzeug kann unterstützt werden, indem, basierend auf dem angestrebten Ziel-`Milieu`, unterschiedliche Vorschläge von im modularen Baukasten vorhandenen Elementen gemacht werden. Für dieses Anwendungsgebiet sind daher zwei Fälle zu betrachten: Einerseits die Untersuchung von Änderungen hinsichtlich der Auswirkungen speziell auf das `Milieu`, andererseits die Planung von `Produkteigenschaften` für ein neues Fahrzeug.

Gemäß der definierten Methodik beginnen Änderungen bei den `Anforderungen`. In einem ersten Schritt kann erfasst werden, welche `Milieus` die jeweilige `Anforderung` steuert:

Listing 6.21: SPARQL-Abfrage zur Ermittlung der Milieus, die von einer Anforderung betroffen sind

```
1
2  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3  PREFIX rm: <http://timguelke.de/rmnet.owl#>
4  PREFIX req: <http://ns.softwiki.de/req#>
5  PREFIX art: <http://timguelke.de/produkt-artefakte.owl#>
6
7  SELECT ?anf ?mil
8  WHERE
9   {
10   ?anf rdf:type req:Requirement .
11   ?anf rm:wird_erfuellt_durch ?eig .
12   ?eig art:bedient_Wert ?wert .
13   ?mil art:hat_Wert ?wert .
14  }
```

Die Variable `mil` besitzt hierbei den Typ `Milieu`, implizit dadurch festgelegt, dass sie als Subjekt der Relation `hat_Wert` eingesetzt ist. Das Ergebnis von Listing 6.21 findet sich in

Tabelle 6.15.

anf	mil
fahrzeug-projekt:Sicherheitssysteme	fahrzeug-projekt:BuergerlicheMitte
fahrzeug-projekt:Fahrverhalten	fahrzeug-projekt:ModernePerformer
fahrzeug-projekt:Sicherheitssysteme	fahrzeug-projekt:BuergerlicheMitte

Tabelle 6.15: Ergebnis Abfrage aus Listing 6.6

Das Fahrzeug ist demnach positioniert zwischen dem Milieu der *bürgerlichen Mitte* und dem der *modernen Performer*, mit einem Schwerpunkt beim ersten Milieu. Die Verschiebung hin zu diesem Schwerpunkt resultiert aus der Erfüllung der `Anforderung` `Sicherheitssysteme` durch zwei `Produkteigenschaften`, denen jeweils wieder ein `AbstractUmfang` zugeordnet ist. Die Abbildung 6.16 zeigt die Zusammenhänge im Graphen. Es wird deutlich, wie die Zuordnung der Milieus zu den Anforderungen ist und wieso das Milieu der *bürgerlichen Mitte* zweimal vertreten ist.

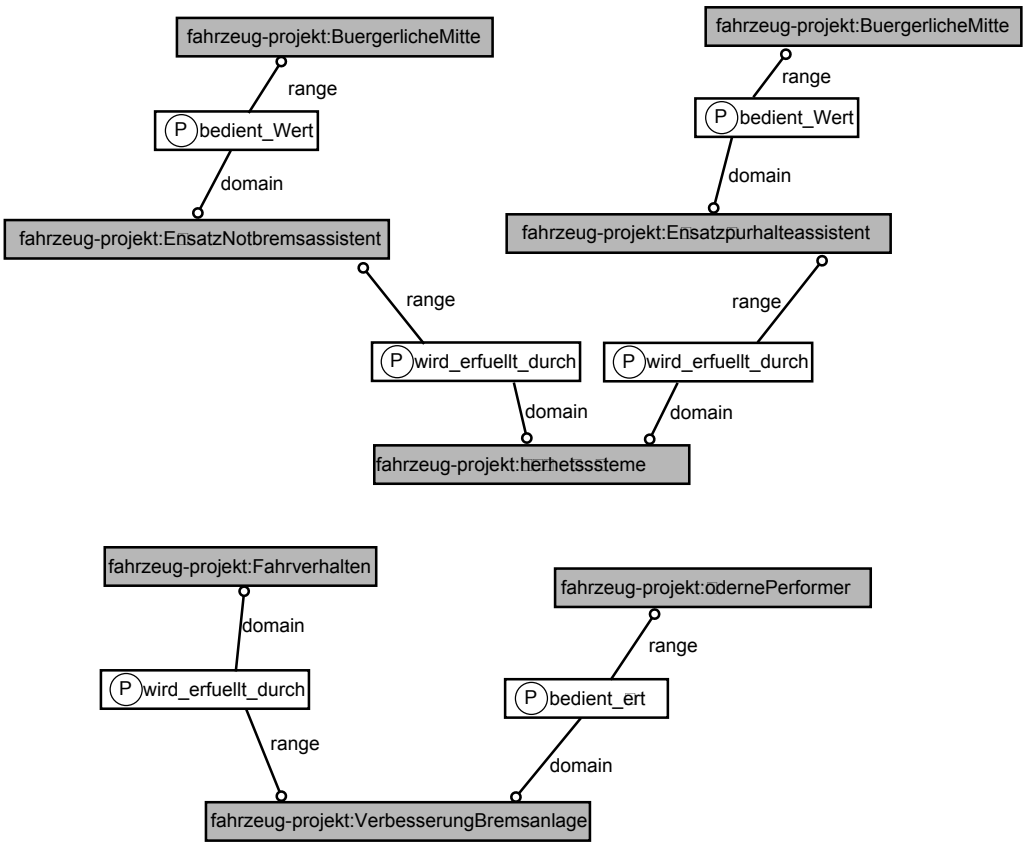


Abbildung 6.16: Darstellung der Zusammenhänge zwischen `Milieus` und `Anforderungen`.

Ein geeignetes Werkzeug kann nun mithilfe dieser Informationen bspw. eine zweidimensionale Darstellung erzeugen, eine graphische Repräsentation der Einordnung des Fahrzeugs. Entfällt

eine Anforderung nun, entfällt auch die Verbindung zu den Milieus und das Fahrzeug wird an eine andere Stelle im Koordinatensystem positioniert. Auch eine weitere Anforderung, die neue Realisierungen nach sich zieht, führt zu einer Verschiebung. Auf diese Weise wird bereits früh deutlich, welchen Einfluss eine Veränderung der Anforderungen auf die geplante Positionierung hat.

Die Konzeption eines neuen Fahrzeugprojekts muss unter Einbeziehung der in einem modularen Fahrzeugbaukasten vorhandenen Module geschehen. Nur auf diese Weise kann ein hoher Wiederverwendungsgrad erreicht und die Entwicklung von Baukasten und Fahrzeuggenerationen aufeinander abgestimmt werden. Das Beispiel, welches in dieser Arbeit verwendet wird, ist allerdings so konzipiert, dass Baukastenmodule von sich aus bereits unterschiedliche Milieus bedienen, indem sie an unterschiedliche Werte appellieren. Ein sportliches Aggregat wird so eher einem `Milieu` „moderner Performer“ zugerechnet werden als der „bürgerlichen Mitte“ und kann bereits mit diesen in der Wissensbasis verknüpft sein. In einem realen Szenario sind die technischen Module selbst allerdings wertfrei und werden – wie etwa ein Aggregat – über Konfigurationsparameter für bestimmte Baureihen eingestellt. Das Beispiel würde bei der Berücksichtigung dieser Konstruktionen zwar realitätsnäher, jedoch wiederum komplexer. Die Demonstration der Möglichkeiten gewinnt jedoch nicht, weshalb das Modell einfacher gestaltet ist.

Für die Auswahl von Modulen für ein Fahrzeug müssen die Informationen über die zugeordneten `Werte` und `Milieus` berücksichtigt werden. In der Konzeptionsphase werden `Anforderungen` gesammelt bzw. vorgegeben und diese durch `Produkteigenschaften` erfüllt. Hierbei bedeuten die `Produkteigenschaften` jeweils die Bedienung eines ganz bestimmten `Werts`. Ein `modularer Baukasten` fasst hingegen eine Gruppe von `Modulen` zusammen, die ihrerseits wiederum die `Produkteigenschaften` anderer `Fahrzeugprojekte` realisieren. Es müssen also die `Module` in einer Abfrage erfasst werden, die das avisierte `Milieu` ansprechen.

Listing 6.22: SPARQL-Abfrage für Baukastenmodule, die zu einem Milieu passen

```
1
2  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3  PREFIX rm: <http://timguelke.de/rmnet.owl#>
4  PREFIX req: <http://ns.softwiki.de/req#>
5  PREFIX art: <http://timguelke.de/produkt-artefakte.owl#>
6  PREFIX proj:<http://timguelke.de/fahrzeug-projekt.owl#>
7
8  SELECT ?mod
9  WHERE
10  {
11    ?mod rdf:type rm:Modul .
12    ?mod art:Modul_enthalten_in ?bau .
13    ?mod rm:realisiert ?eig .
14    ?eig art:bedient_Wert ?wer .
15    proj:BuergerlicheMitte art:hat_Wert ?wer .
16
```

```
17 }
```

Das obige Listing 6.22 demonstriert, wie eine Abfrage nach passenden Baukastenmodulen für ein Milieu aussehen kann. In Zeile 15 wird ein bestimmtes `Milieu` vorgegeben, in einer programmtechnischen Umsetzung könnte an dieser Stelle während der Konstruktion der Abfrage ein entsprechendes anderes eingesetzt werden. Ähnlich ist dies in Zeile 12, wo in dieser Umsetzung `Module` aus beliebigen `ModularenBaukästen` verwendet werden können – hier könnte bspw. ein spezieller Baukasten vorgegeben werden. Mit dem gegebenen semantischen Netz hat die Abfrage folgende Rückgabemenge:

mod
fahrzeug-projekt:Spurhalteassistent

Tabelle 6.17: Ergebnis Abfrage aus Listing 6.22

Es werden demnach nur die `Module` erfasst, die Teil eines `Baukastens` sind und über eine `Produkteigenschaft` und den zugehörigen `Wert` bereits ein `Milieu` `bedienen`, andere `AbstractUmfänge` bleiben unberücksichtigt. Offensichtlich funktioniert das Verfahren nur, wenn ein `Modul` bereits einmal in einem Fahrzeug eingesetzt worden ist, da sonst keine Relation zu einer `Produkteigenschaft` existiert.

Baukastenplanung

Der Umkehrschluss aus der im vorherigen Abschnitt angeführten Zuordnung von Baukastenmodulen zu Milieus, um ein Fahrzeug optimal zu positionieren, besteht darin, einen modularen Fahrzeugbaukasten optimal für die zukünftige Produktpalette auszulegen. Hierzu muss der Bedarf an Modulen im Baukasten anhand der kommenden Fahrzeugprojekte ermittelt werden. Diese langfristige Sicht ist sicherlich herausfordernd im erweiterten Kontext des Anforderungsmanagements, da fraglich ist, inwieweit die Anforderungen an zukünftige Fahrzeuge heute bereits festgelegt werden können und als wie konstant sich diese im Zeitverlauf erweisen. Eine ähnliche Abfrage wie Listing 6.22 ermöglicht jedoch die Ermittlung der benötigten Module für Fahrzeuge, die im Rahmen der Modellplanung vorgesehen sind. Wie bereits beschrieben, muss weiterhin ein Abgleich mit einer Proto-Stückliste vorgenommen werden, um festzustellen, welche `Module` noch fehlen. Im Gegensatz zur letzten Abfrage muss diese allerdings über alle vorhandenen Fahrzeugprojekte durchgeführt und dementsprechend gruppiert werden.

Die hier beispielhaft angeführte Ontologie hat allerdings einen Nachteil, der die Aussagekraft einer solchen Abfrage mindert: Die gestellten Anforderungen sind weiterhin in reiner (Frei-)Textform vorhanden und direkt per Relationen mit den `Modulen` verbunden. Dies führt dazu, dass bspw. keine Abfragen erstellt werden können, die Anforderungen automatisiert `AbstractUmfänge` mit bestimmten Eigenschaften zuordnen. Ein Beispiel hierfür ist das Finden von Motoren, die spezifischen Umwelt-Anforderungen genügen. Hierzu fehlt in der Ontologie die Möglichkeit, `AbstractUmfänge` mit formalen Eigenschaften auszustatten, sowie eine stärkere Formalisierung der Klasse `Requirement`. Eine Möglichkeit hierzu ist die Erstellung differenzierterer Typen von Anforderungen durch Spezialisierung der Klasse

`Requirement` bspw. mit `EnvironmentalRequirement`. An dieser Stelle wird der Vorteil der Verwendung einer Ontologie sichtbar, da diese beliebig erweitert werden kann, ohne die Kompatibilität gegenüber schon vorhandenen Abfragen zu verlieren. So würde eine Klasse `EnvironmentalRequirement` auch in Abfragen zurückgegeben werden, die nur nach der allgemeineren Klasse `Requirement` fragen.

Realisierung

Im Abschnitt „Projektmanagement“ wird bereits ein Reifegrad gem. Definition 5.2.1 auf Basis der umgesetzten Anforderungen ermittelt. Prozessual betrachtet werden allerdings zuerst `AbstractUmfänge` in das Netz eingesetzt, bevor konkrete Artefakte realisiert werden. Sie dienen als Platzhalter für noch zu entwickelnde Umfänge, welche dann in spezielleren Klassen wie bspw. `Teile` überführt werden. Durch die Abfrage aller `AbstractUmfänge`, die nicht in spezielle Subklassen eingeordnet sind, kann festgestellt werden, wie viele abstrakte Konzepte noch keine konkreten Ausprägungen haben. Falls nötig kann diese mit den im Rahmen der Reifegraderstellung ermittelten Werten abgeglichen werden.

Listing 6.23: SPARQL-Abfrage für alle AbstractUmfänge

```
1
2  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3  PREFIX rm: <http://timguelke.de/rmnet.owl#>
4  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
5
6  SELECT ?umf
7
8  WHERE
9   {
10    {?umf rdf:type rm:AbstractUmfang  .}
11
12     MINUS
13
14    { ?s rdfs:subClassOf rm:AbstractUmfang .
15      ?umf rdf:type ?s .
16      FILTER (?s != rm:AbstractUmfang)
17    }
18
19  }
```

Die obige Abfrage in Listing 6.23 gibt demnach nur die Objekte der Oberklasse `Abstract-Umfang` zurück und keine Objekte von Subklassen. Hintergrund ist, nur diejenigen Artefakte zu finden, die noch nicht in einer spezielleren Form vorliegen und demnach noch umgesetzt werden müssen. Die Realisierung des Beispielfahrzeugs schreitet also bereits gut voran, da nur noch ein `AbstractUmfang` vorliegt, wie Tabelle 6.18 zeigt:

req
fahrzeug-projekt:LEDScheinwerfer

Tabelle 6.18: Ergebnis Abfrage aus Listing 6.23

Eine Priorisierung der `AbstractUmfänge` kann nun bspw. anhand der Anzahl `Anforderungen` vorgenommen werden, die bei der Umsetzung des jeweiligen `AbstractUmfang realisiert` werden.

Eine Erweiterung des Modells kann sich nun bspw. darin manifestieren, die benötigten Module für ein Fahrzeug anhand einer Proto-Stückliste zu ermitteln. Diese Proto-Stückliste enthält hierbei alle benötigten Typen von Modulen, etwa ein Aggregat, ein Fahrwerk, eine bestimmte Anzahl Sitze, etc. Durch den Abgleich dieser Liste mit den vorhandenen Modulen in einem Baukasten lässt sich im Hinblick auf die Positionierung des Fahrzeugs die noch zu entwickelnde Menge an Modulen ermitteln. Weiterhin kann in Betracht gezogen werden, das Fahrzeug etwas anders zu positionieren, aber dafür schon vorhandene Module zu verwenden. Hierzu kann bspw. die Volkswagen Vehicles Ontology [Hep] in die Verabeitung einbezogen und somit Module festgelegten Typen zugeordnet werden.

Testen und Qualitätssicherung

Ist für das Projektmanagement die Erstellung eines Reifegrads auf Basis der bereits realisierten Anforderungen relevant, so besteht seitens des Fachbereichs „Entwicklung“ der Bedarf, den Abdeckungsgrad der `Anforderungen` durch `Tests` zu ermitteln. Eine entsprechende einfache Abfrage kann demnach bspw. folgende Form haben:

Listing 6.24: SPARQL-Abfrage für die Testabdeckung

```
1
2  PREFIX rm: <http://timguelke.de/rmnet.owl#>
3
4  SELECT ?req ?test
5  WHERE
6   {
7
8     ?req rm:wird_getestet_durch ?test
9
10  }
```

Folgendes Ergebnis ergibt sich aus der Beispiel-Ontologie:

req	test
fahrzeug-projekt:Fahrverhalten	fahrzeug-projekt:TestFahrverhalten

Tabelle 6.19: Ergebnis Abfrage aus Listing 6.24

Eine weitere wichtige Kennzahl ist die Anzahl ungetesteter `Anforderungen`, die dann etwa zum vorherigen Ergebnis ins Verhältnis gesetzt werden kann:

Listing 6.25: SPARQL-Abfrage für ungetestete Anforderungen

```
1
2  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3  PREFIX rm: <http://timguelke.de/rmnet.owl#>
4  PREFIX req: <http://ns.softwiki.de/req#>
5
6  SELECT ?req
7  WHERE
8   {
9     ?req rdf:type req:Requirement
10    OPTIONAL {?req rm:wird_getestet_durch ?test}
11    FILTER ( ! bound (?test))
12
13  }
```

Da im Beispiel nur die `Anforderung` *Fahrverhalten* mit einem `Test` versehen ist, werden durch diese Abfrage dementsprechend die anderen `Anforderungen` zurückgegeben:

req
fahrzeug-projekt:AmbienteInnenraum
fahrzeug-projekt:Sicherheitssysteme
fahrzeug-projekt:Marktanteil

Tabelle 6.20: Ergebnis Abfrage aus Listing 6.25

Die weiteren Konfigurationen von Abfragen sind mannigfaltig. Auch birgt eine Erweiterung der Ontologie noch Potential, etwa beim Verfolgen des aktuellen Status der Tests: Eine `DataProperty` der Form `rmnet:Test test_erfolgreich xsd:boolean` ermöglicht das einfache Aufzeichnen von Testresultaten. Auch ein komplexeres Vorgehen ist möglich, etwa das Erweitern um konkrete Testtermine, zum Erhalt von Historiendaten. So lässt sich ein Verlauf abbilden, wann Tests erfolgreich waren und wann nicht – was wiederum auf Änderungen abgebildet werden kann. Auch können verschiedene Testarten, wie z.B. Unit- oder Integrationstests unterschieden werden. Die Ontologie ermöglicht hierbei die Verknüpfung von allen Testarten mit der `Anforderung`, solange sie von `Test` abgeleitet sind. Auch die Abfragen könnten entsprechend bestehen bleiben, da sie sich über die Relation und nicht einen konkreten Datentyp definieren. Hierbei wird der Vorteil des gewählten Ansatzes der Top-Down-Detaillierung deutlich: Bereits erstellte Abfragen verlieren nicht ihre Aussagekraft, nur weil ein Artefakt ausdifferenziert wurde.

Änderungsmanagement

Das Änderungsmanagement wird unterstützt durch die automatisierte Prognose der Auswirkungen von Änderungen. Mit „Auswirkung“ ist an dieser Stelle gemeint, dass eine Änderung eines Objekts auch eine Änderung an einem anderen notwendig macht. Dieser Vorgang wird als Impact-Analyse bezeichnet und sorgt somit für die Erfüllung von Anforderung Nr. 17 an ein Anforderungsmanagementwerkzeug aus Kapitel 5.2.1.

Die Impact-Analyse prüft demnach, welche Relationen, ausgehend vom zu ändernden Artefakt, von Typ `wirkt_auf` sind. Von dieser Relations-Oberklasse abgeleitete Relationen beinhalten die semantische Information, dass im Artefakt-Tupel, welches sie verbinden, das Subjekt eine Wirkung auf das Objekt hat. Auf diese Weise können Wirkbeziehungen dargestellt werden. Das folgende Listing 6.26 demonstriert diese Abfrage beispielhaft anhand der `Produkteigenschaft proj:VerbauLEDScheinwerfer`.

Listing 6.26: SPARQL-Abfrage für die Impact-Analyse ohne Kostenbetrachtungen

```
1 PREFIX core: <http://timguelke.de/core.owl#>
2 PREFIX proj: <http://timguelke.de/fahrzeug-projekt.owl#>
3
4 SELECT ?s ?p ?o
5 WHERE
6 {
7   proj:VerbauLEDScheinwerfer core:wirkt_auf ?o
8 }
```

Tabelle 6.21 zeigt die beiden zurückgegebenen Objekte von Artefakten. Der konkrete Typ der Relation ist für die Impact-Analyse unerheblich. Ausschlaggebend ist nur, auf welche Artefakte eine Änderung dieser `Produkteigenschaft` Einfluss hat. Die SPARQL-Abfrage verwendet daher direkt die Relations-Oberklasse `wirkt_auf` ohne über die Information verfügen zu müssen, welche Relationen der `Produkteigenschaft` diese konkret implementieren.

req
fahrzeug-projekt:LEDScheinwerfer
fahrzeug-projekt:EigenschaftskatalogGolf

Tabelle 6.21: Ergebnis Abfrage aus Listing 6.26

Die Negierung der Abfrage kann ebenso interessante Ergebnisse liefern: Welche anderen Artefakte bleiben unberührt, gehen also eventuell davon aus, dass auch das geprüfte unberührt bleibt? Im gegebenen Beispiel führt zwar die Änderung der `Produkteigenschaft proj:VerbauLEDScheinwerfer` nicht zu einer veränderten `Anforderung`. Doch diese ist immer noch durch die Relation `erfuellt` verbunden, was eventuell nicht mehr der fachlichen Realität entspricht. Es besteht schließlich die Möglichkeit, dass die `Anforderung` nicht mehr durch die `Produkteigenschaft erfüllt` wird. Die Relation `wirkt_` beschreibt schließlich nur eine fachliche Wirkrichtung: Ändert sich die `Anforderung`, muss sich auch die `Produkteigenschaft` ändern. Im umgekehrten Fall ist die Relation eventuell gar nicht mehr

gegeben. Listing 6.27 zeigt eine solche Abfrage zur Ermittlung der Relationen, die eventuell auf ihre Korrektheit nach einer Änderung überprüft werden müssen.

Listing 6.27: SPARQL-Abfrage für die Gegenrichtung der Impact-Analyse

```
1  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2  PREFIX core: <http://timguelke.de/core.owl#>
3  PREFIX proj: <http://timguelke.de/fahrzeug-projekt.owl#>
4
5  SELECT ?p ?o
6  WHERE
7  {
8    {proj:VerbauLEDScheinwerfer ?p ?o}
9    MINUS { ?q core:wirkt_auf ?o }
10   MINUS { ?r rdf:type ?o }
11 }
```

Die Abfrage fragt nach allen Relationen und löscht in Zeile 9 die vom Typ `wirkt_auf`. Allerdings benötigen die Ergebnisse einiges mehr an Interpretation: Durch das reine Abfragen der Relationen werden auch solche wie bspw. `rdf:type` zurückgegeben. Entweder entfallen diese bereits in der Abfrage durch das Filtern von generischen Relationen, wie in Zeile 10 dargestellt, oder sie werden im Programmcode durch Filter entfernt. Tabelle 6.22 zeigt das Ergebnis Abfrage bei Entfall von `rdf:type` bereits in der SPARQL-Abfrage:

p	**o**
fahrzeug-projekt:bedient_Wert	fahrzeug-projekt:Exklusivitaet
dcmi-terms:description	„Das Fahrzeug erhält Voll-LED-Scheinwerfer ...“
dcmi-terms:title	„Einsatz Voll-LED-Scheinwerfer“

Tabelle 6.22: Ergebnis Abfrage aus Listing 6.27

Zusammenfassend lässt sich somit für die automatisierte Impact-Analyse konstatieren, dass gute Ergebnisse bei konsequentem Einsatz der Relations-Oberklasse `wirkt_auf` erreicht werden können.

Dokumente

Für diesen Aufgabenbereich können drei Themen unterschieden werden:

1. Das Generieren von Betriebsanleitungen (sog. Bordbücher) auf Basis der verbauten Umfänge.

2. Das Generieren von Lastenheften auf Basis der Anforderungen an einen bestimmten Umfang.

3. Die Unterstützung der Freitexteingabe durch Vorschlag von Begriffen während des Tippens.

Zur Durchführung von Punkt 1 werden alle im `Fahrzeug` eingesetzten `AbstractUmfänge` benötigt. Diese finden sich in der `Technikliste` wo sie durch `enthaelt_Umfang` assoziiert sind.

Listing 6.28: SPARQL-Abfrage für die Erzeugung eines Bordbuchs

```
PREFIX art: <http://timguelke.de/produkt-artefakte.owl#>
PREFIX proj: <http://timguelke.de/fahrzeug-projekt.owl#>

SELECT ?umf
WHERE
 {

   proj:TechniklisteGolf art:enthaelt_Umfang ?umf

 }
```

Das Ergebnis der Abfrage findet sich in Tabelle 6.23.

umf
fahrzeug-projekt:Spurhalteassistent
fahrzeug-projekt:Notbremsassistent
fahrzeug-projekt:LEDScheinwerfer

Tabelle 6.23: Ergebnis Abfrage aus Listing 6.28

An dieser Stelle verfügt die Ontologie offensichtlich nicht über genügend Inhalt zur Generierung eines umfassenden Bordbuchs. Stattdessen wird ein Rumpfdokument geschaffen, welches noch manuell gefüllt werden muss. Allerdings kann das Artefakt `AbstractUmfang` – analog zu den `Produkteigenschaften` – mithilfe der Ergänzung `<dcterms:description>` um einen beschreibenden Text erweitert werden. Durch das Erfassen dieser Erweiterung in der Abfrage kann die Erstellung eines Bordbuchs weitgehend unterstützt werden, indem neben der Struktur auch bereits viele Inhalte konsistent vorgegeben werden.

Punkt 2 kann analog behandelt werden, indem der Fokus auf den `Anforderungen` statt den `AbstractUmfängen` liegt. Es können problemlos erweiterte Informationen zu jeder Anforderung in das Dokument generiert werden, bspw. welche weiteren `Anforderungen` von einer bestimmten `Anforderung` abhängen, etc. Die Core-Ontologie bietet hierfür darüber hinaus das Artefakt `Struktur`, mit deren Hilfe `Dokumente strukturiert` werden können. In der Beispiel-Ontologie wird diese Möglichkeit nicht benutzt, denkbar ist aber durchaus die Entwicklung eines generischen Dokumentgenerators auf Basis der Artefakte `Dokument` und `Struktur`.

Prozessplanung und -optimierung

Während die operative Fahrzeugentwicklung von den Möglichkeiten der Wissensbasis profitiert, steht neben der strategischen Möglichkeit der Baukastenplanung mit der Prozessplanung eine

weitere strategische Komponente zur Verfügung. Die bisherige Methodik verwendet die erfassten Artefakte nur zur Bestimmung der Auswirkungen, die eine Entscheidung auf die Artefakte hat. Umgekehrt kann allerdings anhand der Wissensbasis eine Untersuchung durchgeführt werden, welche Artefakte stark eingebunden sind und damit eventuell „Flaschenhälse“ während der Projektdurchführung darstellen: Muss ein Artefakt bei jeder Änderung immer wieder betrachtet werden, kann untersucht werden, ob es noch notwendig ist und ob genügend Mitarbeiter bei der Bearbeitung zur Verfügung stehen. Bei Optimierungsbetrachtungen können bspw. Artefakte mit einem hohen Verknüpfungsgrad identifiziert werden, aber auch komplexere Konstrukte, wie etwa Prozesse mit Schritten, die besonders häufig stark verknüpfte Artefakte berühren. Wird ein neuer Prozess konstruiert, kann im Vorfeld evaluiert werden, ob überhaupt neue Artefakte benötigt werden oder vorhandene den gleichen Zweck erfüllen. Falls doch Artefakte konzipiert werden müssen, kann dies kontrolliert geschehen, ihr Einfluss auf andere genau definiert und somit die Anzahl betroffener Artefakte minimiert werden. Die Möglichkeiten zur Untersuchung von Artefakten sind immens, an dieser Stelle werden daher einige ausgewählte aufgezeigt. Die folgende Abfrage 6.29 ordnet die Artefakte vom Typ `Dokument` nach der Anzahl ihrer Verwendung in Relationen vom Typ `wirkt_auf`. Der Hintergrund ist zu bestimmen, welche `Dokumente` starke Knotenpunkte darstellen.

Listing 6.29: SPARQL-Abfrage um stark vernetzte Artefakte zu finden

```
1  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2  PREFIX core: <http://timguelke.de/core.owl#>
3
4  SELECT  ?p  (COUNT(?p) AS ?count )
5
6  WHERE
7  {
8     {
9        ?o core:wirkt_auf ?p  .
10       ?p rdf:type core:Dokument .
11    } UNION
12    {
13       ?p core:wirkt_auf ?o  .
14       ?p rdf:type core:Dokument .
15    }
16
17 }
18 GROUP BY ?p
19
20 ORDER BY DESC (?count)
```

Im ersten Schritt werden solche Objekte gefunden, die wahlweise Subjekt oder Objekt einer Relationsinstanz vom Typ `wirkt_auf` sind. Die Ergebnismenge wird dann gruppiert und gezählt, wie viele Relationen vorhanden sind. Schließlich wird die Rückgabemenge sortiert nach Anzahl der Relationen, welche sich in der Variable `count` befindet. Das Ergebnis ist in Tabelle

6.24 enthalten.

p	count
fahrzeug-projekt:EigenschaftskatalogGolf	4
fahrzeug-projekt:TechniklisteGolf	3
fahrzeug-projekt:ModularerAssistentenBaukasten	2

Tabelle 6.24: Ergebnis Abfrage aus Listing 6.29

Es können beliebige weitere Abfragen erzeugt werden, die keinerlei Wissen über die Struktur der Wissensbasis besitzen müssen. Der Vorteil des automatisierten Reasonings von OWL wird an dieser Stelle besonders deutlich, bspw. durch die Vererbung von Relationen wie `wirkt_auf`, die implizit bei den Abfragen berücksichtigt wird.

6.5 Herausforderungen des beschriebenen Lösungsansatzes

Nachdem in den vorangehenden Abschnitten die Möglichkeiten und das Potential der erstellten Ontologien und des zugehörigen Werkzeugs dargestellt wurden, widmet sich der nun folgende den Herausforderungen.

6.5.1 Automatisierte Kostenbetrachtungen

In Abschnitt 5.5.2 wurden die Implikationen beschrieben, die ein durchgängiges und semantisch unterstütztes Anforderungsmanagement auf die weitere Entwicklung der Komplexität im Unternehmen haben kann. Hierbei wird explizit darauf hingewiesen, dass ein erster Schritt nur die Erkennung der Auswirkungen beinhaltet, nicht deren finanzielle Bewertung. Dies stellt eine technische und fachliche Herausforderung dar. In der beschriebenen Form ist das semantische Netz nicht in der Lage, Kosten zu berücksichtigen, sondern kann nur Hinweise darauf liefern, an welcher Stelle Artefakte von einer Entscheidung betroffen sind. Wie arbeitsintensiv dies ist bleibt unklar. Ein mögliches Vorgehen ist, Artefakte selbst mit einer Bewertung zu versehen, die einen Hinweis auf den Aufwand einer etwaigen Änderung bietet. Allerdings ist eine Einschätzung der „Schwierigkeit der Änderung“ eines einzelnen Artefakts schwer objektiv zu quantifizieren, da dies von stark subjektiven Faktoren geprägt ist und grundsätzlich nur in Relation zu anderen Artefakten gesehen werden kann.

Alternativ steht durch die festgelegte Wirkbeziehung der Assoziationen in der Ontologie – unter Verwendung der Oberklasse aus der Core-Ontologie `wirkt_auf` – eine Möglichkeit zur Verfügung, Kausalitätsketten abzubilden. Bei Verwendung von Gewichten an den Assoziationen lässt sich diese Kausalitätskette bewerten. Fachlich gesehen wird hiermit der angenommene Einfluss der ausgelösten Änderung an einem Artefakt auf ein anderes durch einen Zahlenwert ausgedrückt.

Hinsichtlich der technischen Realisierung zeigt Abbildung 6.25 beispielhaft, wie Kosten von Relationen in einem Schema umgesetzt werden könnten. Hierbei wird eine `Anforderung` durch ei-

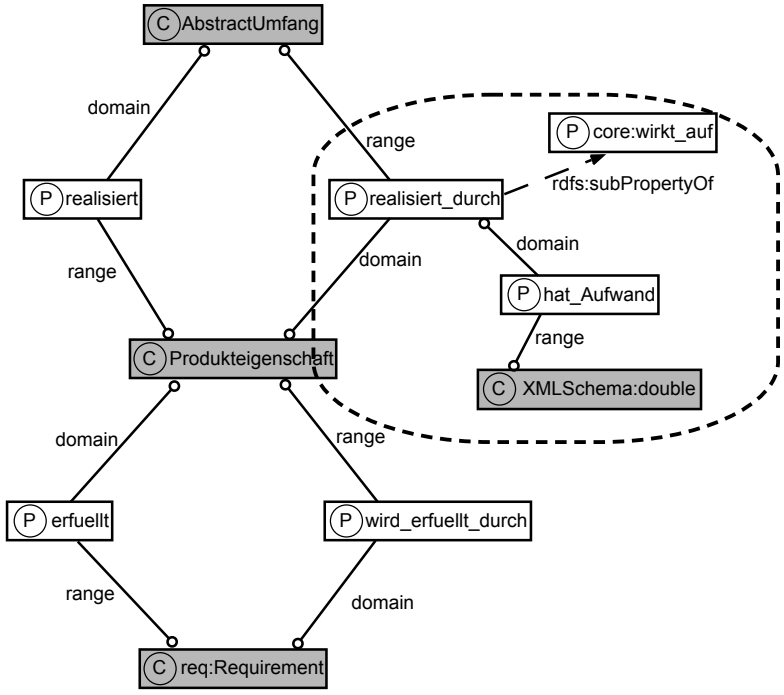


Abbildung 6.25: Gewichtungen von Eigenschaften zur automatisierten Kostenbetrachtung

ne `Eigenschaft` `erfüllt`, welche wiederum durch einen `Umfang` `realisiert` wird. Der markierte Bereich weist der Relation `wird_realisiert_durch` durch eine weitere Relation `hat_Aufwand` einen Wert vom Typ `XMLSchema:double` zu. Dieses Konstrukt ist allerdings in RDFS und OWL nicht erlaubt: Das Objekt einer Relation muss immer vom Typ `rdfs:Class` sein. Eine Möglichkeit zur Realisierung im gegebenen Sprachumfang ist bspw. die Verwendung von abgeleiteten Relationen, die jeweils einen festen Wert ausdrücken. Anstelle von der Relation `wird_realisiert_durch` wird demnach z.B. `wird_realisiert_durch_Wert_7` benutzt, wobei letztere Relation durch `rdfs:subPropertyOf` mit der ersteren verbunden ist. Dieses Hilfskonstrukt führt allerdings zu einer großen Anzahl von Relationen und erschwert somit den Pflegeprozess.

In der Ausführung von Hayes und Welty wird auf die Möglichkeit von n-ären Relationen in OWL eingegangen, welches grundsätzlich nur binäre Relationen unterstützt [HW06]. Eine dort angeführte Möglichkeit, dennoch n-äre Relationen zu verwenden, ist der Einsatz von Relationsklassen. Hierbei wird eine Klasse derart entworfen, dass sie über die benötigte Anzahl an Relationen verfügt. Die eigentliche Relation ausgehend vom Subjekt verweist dann auf eine Instanz diesen Typs, welches wiederum auf die gewünschten Objekte zeigt. Auf diese Weise kann eine n-äre Relation konstruiert werden.

Abbildung 6.26 zeigt beispielhaft für die Relation `beschrieben_in`, welche die Klassen `Anforderung` und `Dokument` verbindet, wie durch eine Relationsklasse eine Gewichtung eingeführt werden kann. Der entscheidende Nachteil bei der Realisierung auf diese Art ist, dass Metainformationen verloren gehen, bzw. implizites Wissen in die Ontologie eingefügt wird. Das

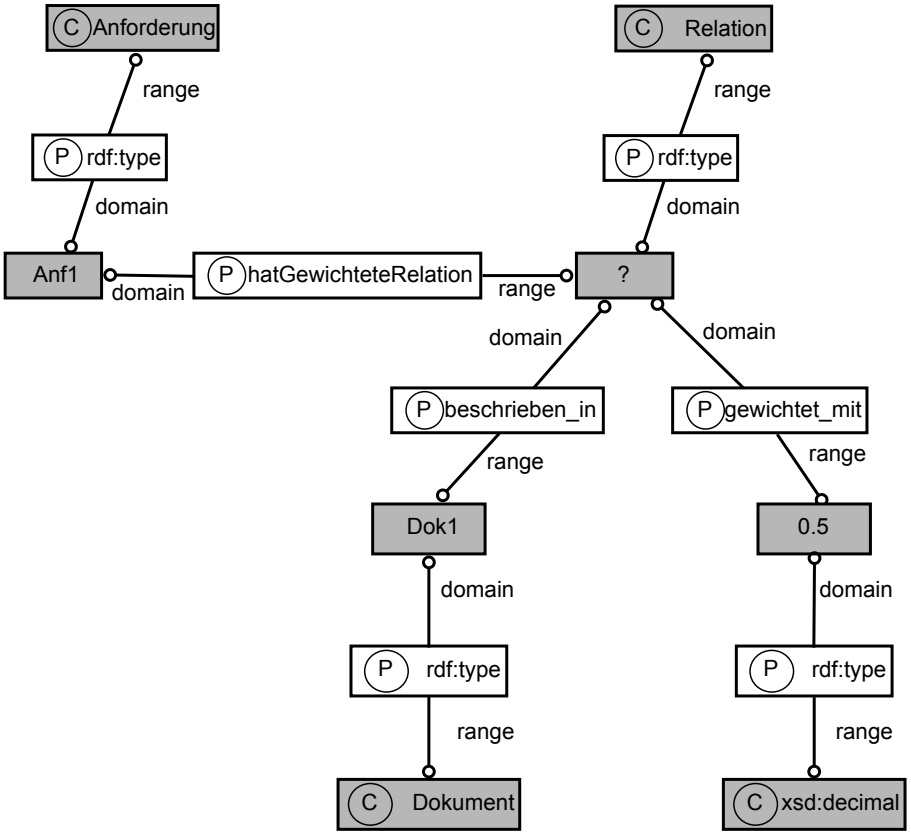


Abbildung 6.26: Verwendung einer Relationsklasse für n-äre Relationen

automatisierte Reasoning ist nicht mehr möglich, sobald ein „Zwischenschritt“ vor die eigentliche Relation gesetzt wird, da die Abfragesoftware nicht von selbst davon ausgeht, dass dieser zu ignorieren ist. Auch eine Fallunterscheidung zwischen gewichteten und ungewichteten Relationen ist aufgrund der vorhandenen Informationen nicht möglich. Das folgende Beispiel zeigt dies anhand der Transitivität:

Ist eine Relation P als transitiv markiert, führt die Abfragesoftware folgenden impliziten Schluss aus:

Existiert eine Relationsinstanz p des Typs P, welche die zwei Objekte A, B als Subjekt und Objekt beinhaltet und existiert eine weitere Relationsinstanz q des Typs P, welche die zwei Objekte B, C als Subjekt und Objekt beinhaltet, dann existiert eine Relationsinstanz r, welche die zwei Objekte A, C als Subjekt und Objekt beinhaltet.

Wird hingegen ein Relationsobjekt des Typs D mit einer passenden Relationsinstanz q des Typs Q verwendet, schlägt die Schlussfolgerung fehl:

Es existieren nun drei Relationsinstanzen, $q(A, D), p(D, B), p(B, C)$. Die Abfragesoftware führt analog den obigen Schritt aus, impliziert nun aber aufgrund der Transitivität der Relation P eine Relationsinstanz $r(D, C)$, obwohl $r(A, C)$ korrekt wäre.

Eventuell wäre auch $p(B, C)$ ebenfalls noch mit einem Relationsobjekt versehen. Die Einfüh-

rung von Relationsklassen lässt somit kein automatisiertes Schliessen auf Basis der vorhandenen Informationen zu, weshalb kein Standard-Reasoner – wie etwa im Rahmen der Apacha Jena-API – verwendet werden kann.

Zwei Möglichkeiten existieren, um dennoch mit Gewichtungen arbeiten zu können und die Ontologie trotzdem allgemein lesbar zu halten:

1. Verwendung eines Präprozessors zur Entfernung von Konstrukten mit Relationsobjekten: Die Gewichtungen sind in dieser Form nur sinnvoll und wichtig für die Verwendung im Kontext des beschriebenen Werkzeugs. Für dieses kann ein spezieller Reasoner entwickelt werden, der automatisiert Relationsobjekte in Abfragen ignoriert. Zur Verwendung der Ontologie in anderen Systemen ist ein Präprozessor denkbar, der durch eine geeignete einfache Transformation die Relationsobjekte wieder aus den Relationen entfernt und den Zustand ohne Gewichtungen wieder herstellt. Es würde somit ein „Master“ der Ontologie existieren und eine transformierte Version ohne Gewichtungen, die allgemein verwendet werden kann. Die Schwierigkeit hierbei ist allerdings die Sicherstellung der Konsistenz, wenn in beiden Ontologien gearbeitet werden kann. Eine Möglichkeit hierzu ist der Einsatz einer zentralen Stelle für das Ausführen von Kommandos in der Ontologie. So werden dann bspw. beim Einfügen einer neuen konkreten Relation zwischen zwei Objekten ein Kommando mit einem Relationsobjekt und eins ohne generiert und ausgeführt.

2. Pflege der Gewichtungen außerhalb der Ontologie: Eine simple Matrix kann die Kosten der Relationen vorhalten, indem bspw. eine Datei mit kommagetrennten Werten neben der Ontologie existiert. Software, welche die Werte verwendet, greift auf diese zu, alle anderen ignorieren sie. Der Nachteil dieser Methode ist, dass zwei Stufen in Abfragen nötig sind: Zuerst muss per SPARQL die Abfrage im semantischen Netz durchgeführt werden. Danach müssen durch eine separate Abfragemethode die passenden Werte ermittelt werden.

Unabhängig von der konkreten technischen Realisierung in OWL sind wie in Abschnitt 4.2.1 beschrieben zwei methodische Fälle bei der Kostenanalyse zu betrachten. Der erste betrifft die Änderung von bestehenden Elementen, hierbei soll mit Hilfe einer Impact-Analyse aus der Wissensbasis eine Bewertung des Aufwands der Änderung erstellt werden. Beginnend am zu ändernden Artefakt können nun rekursiv alle anderen Artefakte ermittelt werden, die durch eine Assoziation verbunden sind, die von `wirkt_auf` abgeleitet ist. Die Gewichtungen dieser „wirkenden“ Assoziationen können bspw. addiert und so ein Wert für den einmal aufzubringenden Aufwand einer Änderung ermittelt werden. Mithilfe dieses Verfahrens lassen sich weiterhin auch verschiedene Handlungsalternativen vergleichen. Abbildung 6.27 zeigt, wie Instanzen des in Abbildung 6.25 definierten Schemas aussehen könnten.

Das Schema wurde hierbei ergänzt um eine `hat_Aufwand`-Relation für die Relation `wird_erfuellt_durch`, welche in Abbildung 6.25 aus Gründen der Übersichtlichkeit nicht dargestellt ist. Wird eine Anforderung geändert, muss die zugehörige `Eigenschaft` geändert werden, weiterhin der dazu gehörige `Umfang`. Dies ergibt einen errechneten Wert von 10, den diese Änderung an Aufwand bedeuten würde.

Im zweiten Fall wird eine neue Instanz eines Artefakts hinzugefügt und somit ein neuer Faktor in eventuellen Wirkketten aufgenommen. In diesem Fall kann der umgekehrte Weg in den

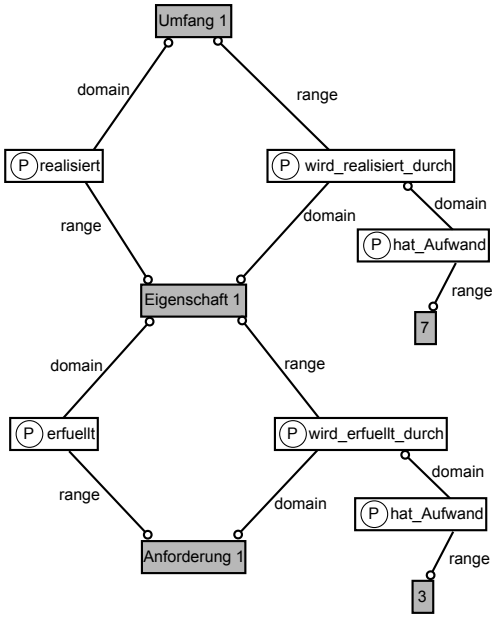


Abbildung 6.27: Beispiel von Fall 1 bei automatisierten Kostenbetrachtungen

„wirkenden“ Assoziationen gegangen und geprüft werden, welche Instanzen anderer Artefakte auf eine hinzugefügte Instanz einwirken. Danach kann errechnet werden, inwieweit der gesamte Wert der Wirkketten, deren Teil die neue Instanz ist, sich verändert. Analog gilt dies für das Entfernen von Instanzen. Aber auch das Hinzufügen oder Entfernen ganzer Artefakte kann auf diese Weise hinsichtlich der Wirkung auf die Komplexität untersucht werden.

Das gezeigte Vorgehen besitzt allerdings neben den gezeigten technischen auch einige fachliche Schwachpunkte:

1. Die zwei gezeigten Möglichkeiten basieren auf der Verfügbarkeit von fachlich festgelegten Gewichtungen für die Assoziationen zwischen Artefakten. Diese Werte können ohne strukturiertes Verfahren willkürlich sein. Lösungsansätze finden sich eventuell auch hier – ähnlich der Herangehensweise beim Pflegeprozess – in der Einbeziehung der Nutzer, etwa durch eine kontextsensitive Befragung während der Verwendung. Bspw. könnten Nutzer gebeten werden, eine Einschätzung der Gewichtung von Assoziationen durch festgelegte Textbausteine („wirkt stark“, „wirkt mittel“, „wirkt wenig“) abzugeben, wenn sie sich mit einem Kontext der betroffenen Artefakte beschäftigen. Werden Impact-Analysen durchgeführt, kann der Anwender an dieser Stelle gefragt werden, ob die Einschätzung des Systems in etwa korrekt ist. Falls nicht, kann er per Annotationen in der Wirkkette aufzeigen, an welcher Stelle ein möglicherweise dysfunktionales Gewicht vorliegt. Grundsätzlich muss allerdings ein betriebswirtschaftlich begründetes Verfahren zur Bewertung der Aufwände entwickelt werden.

2. Die Bewertung von Änderungen auf diese Weise setzt voraus, dass sich Entscheider an den beschriebenen Prozess halten. Die Änderung eines Umfangs in Abbildung 6.27 bspw.

würde zu keinem erweiterten Aufwand führen, da dieser keine wirkenden Relationen besitzt. In der praktischen Anwendung müssten die zugehörigen Artefakte nachgezogen werden. Dies widerspricht allerdings der fachlichen Interpretation der Wirksamkeit von Relationen und dem Prozess, welcher davon ausgeht, dass Änderungen an der Anforderung beginnen und nicht an der Stelle der Realisierung.

Das folgende Beispiel zeigt – separat von der erarbeiteten Ontologie für die Produktentstehung mit den bisher verwendeten Beispielwerten – wie eine solche Realisierung aussehen kann. Hierzu erfolgt eine kurze Beispiel-Ontologie in Listing 6.30, die für die folgende Abfrage verwendet wird. Es werden die zwei Klassen `Anforderung` und `Dokument` durch die Relation `beschrieben_in` verknüpft. Zeile 28 definiert eine Relationsklasse, Zeile 12 stellt somit den beschriebenen Zwischenschritt dar. Hierbei wird ein Objekt der Klasse `Anforderung` nicht direkt durch die Relation `beschrieben_in` verbunden, sondern mit `hasWeightedRelationship` zuerst mit einem anonymen Objekt der Relationklasse. Zeile `srcline:AnonymesObjekt` demonstriert die Konstruktion eines anonymen Objekts im Kontext eines anderen Objekts, in diesem Fall der `Anforderung Anf1`. Das anonyme Relationsobjekt verfügt wiederum über Relationen des Typs `weight`, wodurch die Gewichtungen repräsentiert werden. Auf diese Weise werden drei Objekte des Typs `Dokument` durch gewichtete Relationen mit der `Anforderung` verbunden.

Listing 6.30: Eine Ontologie mit gewichteten Relationen durch Relationsobjekte

```
1  @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
2  @prefix owl: <http://www.w3.org/2002/07/owl#> .
3  @prefix test: <http://timguelke.de/WeightTest.owl#> .
4  @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
5  @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
6
7  test:beschrieben_in rdf:type owl:ObjectProperty ;
8                      rdfs:range test:Dokument ;
9                      rdfs:domain owl:Thing .
10
11
12 test:hasWeightedRelationship rdf:type owl:ObjectProperty ;
13                               rdfs:domain owl:Thing ;
14                               rdfs:range test:Relation .
15
16
17 test:weight rdf:type owl:DatatypeProperty ;
18             rdfs:domain test:Relation ;
19             rdfs:range xsd:nonNegativeInteger .
20
21
22 test:Anforderung rdf:type owl:Class .
23
```

```
24
25 test:Dokument rdf:type owl:Class .
26
27
28 test:Relation rdf:type owl:Class .
29
30
31 test:Anf1 rdf:type test:Anforderung ,
32                   owl:NamedIndividual ;
33
34           test:hasWeightedRelationship
35
36                [ rdf:type test:Relation ;
37                  test:weight "20"^^xsd:nonNegativeInteger ;
38                  test:beschrieben_in test:Dok2
39                ] ,
40                [ rdf:type test:Relation ;
41                  test:weight "10"^^xsd:nonNegativeInteger ;
42                  test:beschrieben_in test:Dok1
43                ] ,
44                [ rdf:type test:Relation ;
45                  test:weight "30"^^xsd:nonNegativeInteger ;
46                  test:beschrieben_in test:Dok3
47                ] .
48
49
50 test:Dok1 rdf:type test:Dokument ,
51                   owl:NamedIndividual .
52
53
54 test:Dok2 rdf:type test:Dokument ,
55                   owl:NamedIndividual .
56
57
58 test:Dok3 rdf:type test:Dokument ,
59                   owl:NamedIndividual .
```

Die Relation `beschrieben_in` kann darüber hinaus ebenfalls von einer Oberklasse wie bspw. die in der Core-Ontologie enthaltenen Relation `wirkt_auf` abgeleitet werden, um die Impact-Analyse zu vereinfachen. Geschieht dies nicht, müssen andere Relationen wie etwa `rdf:type` aus der Abfrage gefiltert werden, wobei kein generisches Kriterium existiert, das als Filterparameter dienen könnte. Die Folge ist, dass bekannte Relationen fest benannt und die restlichen in Interaktion mit dem Benutzer entfernt werden müssen. Die folgende Abfrage 6.31 führt eine Abfrage analog zu 6.26 auf der Beispiel-Ontologie aus Listing 6.30 aus, ohne allerdings

eine Oberklasse für wirkende Relationen zu verwenden. In Zeile 9 wird daher die Ergebnismenge beispielhaft um für die Impact-Analyse irrelevante Tupel verringert.

Listing 6.31: SPARQL-Abfrage für die Änderungsanalyse inkl. Kostenbetrachtungen

```
1  PREFIX test: <http://timguelke.de/WeightTest.owl#>
2
3  SELECT ?s ?p ?o
4  WHERE
5  {
6    {?s test:hasWeightedRelationship ?rel .
7     ?rel test:weight ?weight .
8     ?rel ?p ?o .
9     } MINUS {?r rdf:type ?o .}
10 }
```

Das Ergebnis dieser Abfrage zeigt Tabelle 6.28:

s	**p**	**o**
test:Anf1	test:beschrieben_in	test:Dok3
test:Anf1	test:weight	30
test:Anf1	test:beschrieben_in	test:Dok2
test:Anf1	test:weight	20
test:Anf1	test:beschrieben_in	test:Dok1
test:Anf1	test:weight	10

Tabelle 6.28: Ergebnis Abfrage aus Listing 6.31

Für jede gewichtete Relation werden demnach im Ergebnis zwei Relationen angezeigt. An dieser Stelle kann das ausführende Programm die Rückgabe interpretieren und bspw. eine eigene Graph-Klasse mit gewichteten Kanten gefüllt werden. Es wird deutlich, dass dies mit dem impliziten Wissen um die Zusammenhänge der Relationsobjekte geschehen muss bzw. mit den weiter oben in diesem Abschnitt vorgestellten Möglichkeiten.

6.5.2 Berechtigungen

Ein unternehmensweites Netz aller relevanten Artefakte inklusive Informationen über ihre Zusammenhänge birgt ein potentielles Sicherheitsrisiko. Auch in der Arbeit von Klude, Metin und Ulbrich ist die „rollenbasierte Zugriffsregelung“ explizit als Anforderung an ein etwaiges System genannt [KMU11]. Zwar existieren im entwickelten semantischen Netz nicht notwendigerweise konkrete Instanzen der Artefakte, dennoch lassen sich Rückschlüsse auf das allgemeine Vorgehen in Projekten und bspw. auch auf den Einsatz von Technologien ziehen. Weiterhin sind innerhalb eines einzelnen Werkzeugs, welches die Wissensbasis implementiert, durchaus konkrete Objekte vorhanden, die nicht allen Projektbeteiligten zugänglich gemacht werden dürfen. Insbesondere in der Zusammenarbeit mit Zulieferern ist dies ein wichtiger Faktor. Ein Rollen- und Berechtigungskonzept ist daher notwendig. Diese kann wiederum im semantischen Netz modelliert

werden, Artefakte können dann über Relationen mit den jeweiligen Berechtigungen verknüpft werden. Dieser Ansatz verlagert allerdings die Aufgabe der tatsächlichen Prüfung der Rechte eines Benutzers auf die jeweilige Applikation und setzt demnach einerseits die Gutartigkeit aller Anwendungen und andererseits die Kompetenz der Entwickler bei der Umsetzung andererseits voraus. Beides mag sich in einem großem Unternehmen mit einer Vielzahl von Programmen und Systemen als unrealistisch erweisen. Eine Alternative ist die Hinterlegung der Berechtigungen von Benutzern in einer vorgelagerten Anwendung als eine Art „Torwächter“, die dann Anfragen von dritter Software positiv oder negativ bescheinigt.

6.5.3 Leistungsfähigkeit

Der durch Relationenkomplexität entstehende Aufwand ist hoch. Wird ein umfangreiches semantisches Netz komplett geladen, verursacht dieser Prozess rechenintensive Aufgaben. Abfragen an das Netz bedeuten teilweise das Verfolgen von vielen Relationen, was wiederum rechenintensiv ist.

Die Verwendung zu vieler unterschiedlicher Relationen ohne Bezug zu einer Relationen-Oberklasse stellt ebenfalls ein Problem dar, welches die Qualität des Netzes beeinflusst. Es tritt auf, wenn Artefakte beschrieben werden, ohne vorhandene Relationen zu verwenden. Stattdessen werden neue angelegt, die sicherlich für den jeweiligen Modellierer sinnvoll sind und in seiner Software verwendet werden. Im großen Kontext allerdings ist die Bedeutung der Relationen unklar und fremde Software, wie in diesem Fall etwa die des Anforderungsmanagements, ist nicht in der Lage, Schlüsse aus ihnen zu ziehen. Dies führt zu einer Reduktion des Nutzens des Netzes auf die jeweiligen lokalen Anwendungen in ihrem speziellen Kontext des Netzes in Kombination mit den Teilen, die korrekt von übergeordneten Relationen abgeleitet wurden. Abhilfe schaffen kann an dieser Stelle nur die konsequente Anwendung – soweit möglich – des Prinzips des Ableitens jeder Relation von den vorgegebenen Oberklassen aus der Core-Ontologie. Ist tatsächlich eine Relation separat von den vorgegebenen zu betrachten, muss mindestens eine ausführliche Beschreibung des Zwecks gegeben sein. RDFS bietet hierfür das `<rdfs:comment>`-Tag.

Insbesondere das Problem der Leistungsfähigkeit kann und muss im Rahmen der Pflege des semantischen Netzes als Grundlage des Anforderungsmanagements und darüber hinaus gehender Software behandelt werden. Durch automatisierte Werkzeuge, aber auch durch verantwortungsvolles und aufmerksames Handeln der Modellierer und Entwickler bleibt die Leistungsfähigkeit erhalten.

6.5.4 Aktualität

Das Netz repräsentiert die Situation der einzelnen Artefakte im Unternehmen. Zwar ist der Ansatz darauf ausgelegt, eine Top-Down-Sicht zu vermitteln um auf diese Weise bereits mit geringem Füllgrad des semantischen Netzes sinnvolle Schlüsse ziehen zu können. Hierbei liegt die Annahme zugrunde, dass Artefakte nicht sehr oft vollständig neu geschaffen oder wieder aus dem Ökosystem des Unternehmens entfernt werden, sondern sich nur ihr Aufbau ändert. Es kann davon ausgegangen werden, dass ein im Rahmen eines Prozesses eingeführtes Artefakt für einen längeren Zeitraum existiert. Stehen aber ausschließlich Detailinformationen zur Verfügung,

so können falsche Resultate aus einer Anfrage entstehen, ohne dass nachvollziehbar wird, an welcher Stelle aufgrund einer fehlerhaften Assoziation ein falscher Schluss gezogen wurde.

Der aktuelle Ansatz enthält im Rahmen des beschriebenen Prozesses einen Abschnitt zur Pflege des semantischen Netzes. Praktisch müssen hierzu die organisatorischen Rahmenbedingungen geschaffen und eingehalten werden. Das Pflegen des vorhandenen Wissens muss einen hohen Stellenwert erhalten. Bei der Betrachtung der aktuellen Vorgehen hinsichtlich der Modellierung von Prozessen mögen allerdings Zweifel aufkommen, ob dies konsequent eingehalten wird. Anwendungsentwickler können im Gegensatz zu Nutzern zwar die Aktualität ihres speziellen Teils des semantischen Netzes überprüfen sowie neue Modelle in Kooperation mit der jeweiligen Fachabteilung entwickeln. Es ist ihnen jedoch ebenfalls nicht möglich, die Korrektheit aller Zusammenhänge festzustellen. Der Fokus liegt daher während des Betriebs auf den Annotationen der Benutzer. Das jeweilige Werkzeug muss in seiner Oberfläche einfache Möglichkeiten bieten, Korrekturinformationen zu beliebigen Artefakten und deren Relationen hinzuzufügen. Darüber hinaus müssen diese in einem regelmäßigen Prozess in Zusammenarbeit zwischen Entwickler und Fachabteilung bearbeitet werden.

6.6 Ergebnisse und Erkenntnisse

Dieses Kapitel führt aus, wie das im vorhergehenden Kapitel konzipierte durchgängige und semantisch unterstützte Anforderungsmanagement mithilfe einer Sammlung von zentralen Begriffen, einer Methodik sowie einem Werkzeug operationalisiert werden kann. Hierzu findet zunächst in Abschnitt 6.1.1 eine Einführung in die Arbeit mit der Sprache RDF bzw. OWL sowie der Abfragesprache SPARQL statt. Gezeigt wird die technische Realisierung auf Basis der der JENA-API.

Es folgt in Abschnitt 6.2 zunächst eine Definition der grundlegenden Artefakte eines Unternehmens. Diese werden in der Ontologie Core formalisiert, um maschineninterpretierbar zur Verfügung zu stehen und somit die Ausgangspunkte einer Unternehmenswissensbasis zu bilden. Durch die Ontologie RMNet findet eine Verknüpfung mit Produktinformationen statt, wodurch automatisierte Analysen zwischen diesen beiden Welten möglich werden. Es können demnach Produkt- und Prozessartefakte derart miteinander verknüpft werden, dass die Auswirkungen der Änderungen an Produktdaten auf Prozessartefakte sichtbar werden. Auch können gezielt Prozessanalysen hinsichtlich Engpässen durchgeführt, Kennzahlen erhoben und andere Abfragen durchgeführt werden. Die Kernartefakte erwiesen sich in den Konstruktionen in Abschnitt 6.4 als ausreichend, um ein unternehmensspezifisches Artefaktmodell zu entwickeln.

Nachdem die technischen Voraussetzungen geschaffen sind und die Definition der Kernartefakte abgeschlossen ist, wird in Abschnitt 6.3 eine Vorgehensweise zur Erarbeitung eines unternehmensspezifischen Artefaktmodells vorgestellt. Der Fokus dieser Methodik liegt auf der Pflege der vorhandenen Informationen, welche durch ein Zusammenwirken von Mitarbeitern und automatisierten Prozessen durchgeführt wird. Durch die Vorgabe von bestimmten Regeln werden Auffälligkeiten zusammengetragen und Mitarbeitern zur Korrektur vorgeschlagen. Weiterhin können Bearbeiter der Informationen des Netzes auch selbstständig Markierungen vornehmen, um auf kritische Bereiche aufmerksam zu machen. Die erarbeitete Methodik hat ihren Schwerpunkt in der Pflege von einmal erarbeiteten Informationen. Durch die in Kapitel 4.2 beschriebene

Evolution findet ein ständiger Wandel im Unternehmen statt, welcher sich in den Prozessen niederschlägt. Mithilfe der Möglichkeiten der automatisierten und manuellen Markierungen von Problemstellen kann die Aktualität der Wissensbasis jedoch sichergestellt werden.

Es folgt der Abschnitt 6.4, welche die Operationalisierbarkeit des Ansatzes durch Anwendung auf eine exemplarische Artefaktlandschaft zeigt. Demonstriert wird einerseits das Vorgehen zur Erarbeitung einer solchen Artefaktlandschaft. Andererseits wird dargelegt, inwieweit die einzelnen Disziplinen des Anforderungsmanagements durch die Verfügbarkeit des beschriebenen Werkzeugs unterstützt werden. Es zeigt sich, dass diese profitieren, indem mehr Informationen auf verbesserte und schnellere Weise verfügbar gemacht werden.

Das Kapitel 5 dieser Arbeit definiert Anforderungen, die durch ein solches Werkzeug erfüllt werden müssen, damit es den vollen beschriebenen Nutzen entfalten kann. Das Kapitel 6 geht vor allem auf die Anforderungen 1 (Verwendung eines Metamodells), 14 (Konstruktion von Suchabfragen), 15 (Automatisches Schlussfolgern) und 17 (Durchführung von Impact-Analysen) ein. Hierzu ergaben sich folgende Erkenntnisse:

- Anforderung 1: Das beschriebene Anforderungsmanagementwerkzeug verwendet als Metamodell ein auf den Ontologien Core und RMNet basierendes Modell, welches mit konkreten Instanzen gefüllt wird. Basierend hierauf können Funktionen, wie in den weiteren Anforderungen beschrieben, realisiert werden.

- Anforderung 14: Die oben gezeigten Abfragen sind beispielhaft für das Konstruieren von Suchabfragen. Durch die Unterstützung der Abfragesprache SPARQL können umfassende Suchen zur Erzeugung von Berichten, Kennzahlen oder ähnlichem erstellt werden. Wird der Benutzer durch eine graphische Oberfläche unterstützt, kann die Arbeit mit SPARQL nochmals erleichtert werden.

- Anforderung 15: Durch die Reasoning-Funktion der JENA-API kann das Werkzeug automatisiert Schlüsse ziehen. Core und RMNet bieten vor allem durch die transitive Markierung und konsequente Verwendung der Relation `wirkt_auf` Unterstützung beim automatischen Schlussfolgern.

- Anforderung 17: Impact-Analysen fanden eine spezielle Behandlung im Abschnitt 6.4.3, indem sie in den Kontext des Änderungsmanagements gestellt wurden. Es konnte gezeigt werden, dass ein auf den Ontologien Core und RMNet basierendes Werkzeug unter Verwendung von SPARQL-Abfragen sehr gute Ergebnisse bei der Impact-Analyse erzielen kann.

Es zeigt sich jedoch, dass die Aussagekraft des Systems abhängig ist von der Qualität der Wissensbasis. Hiermit fällt der Methodik zu Aufbau und Pflege eine entsprechend vitale Rolle zu. Wird jedoch eine entsprechenden Disziplin eingehalten, bietet sich einem Unternehmen die Möglichkeit, langfristig eine große Menge an Wissen in strukturierter Form zusammenzutragen. Durch die Verwendung eines offenen Standards in Form von OWL steht die Wissensbasis verschiedenen Disziplinen zur Verfügung.

An Herausforderungen stechen vor allem vier Punkte heraus, welche technisch oder organisatorisch gelöst werden müssen. Insbesondere der Bereich der Berechtigungen stellt größere

Unternehmen vor Hürden, da dort umfangreiche Prozesse eingehalten werden müssen, um Mitarbeitern aber auch anderer Software Zugang zu Systemen sowie den darin enthaltenen Daten zu gewähren. Diese Problemstellung trifft kleinere Unternehmen eventuell nicht so stark, da dort möglicherweise ein anderes Verhältnis zum Unternehmenswissen vorherrscht.

Technisch stellt das Konzept der automatisierten Kostenbetrachtung die größte Herausforderung dar, da bisher keine zufriedenstellende Lösung existiert. Es werden jedoch Wege aufgezeigt, die eine solche Automatisierung zumindest ermöglichen würden. Es bleibt abzuwarten, ob das beschriebene Konzept in OWL in späteren Versionen zur Verfügung steht.

Als Ergebnis dieses Kapitels lässt sich festhalten, dass mithilfe der erstellten Ontologie Core und RMNet ein umfangreiches Werkzeug für ein semantisch unterstütztes Anforderungsmanagement aufgebaut werden kann. Durch die Einordnung von Artefakten in grundlegende Kategorien können durch das automatisierte Schlussfolgern der eingesetzten Abfragesoftware Annahmen hinsichtlich des Verhaltens von Artefakten getroffen werden. Obwohl nicht zwingend notwendig, ist eine weitergehende Spezifizierung von Artefakten im semantischen Netz möglich. Das Werkzeug ist somit in der Lage, Produkt- und Prozessinformationen zu verknüpfen.

Kapitel 7

Zusammenfassung, Fazit und Ausblick

Dieses letzte Kapitel fasst die Ergebnisse dieser Arbeit anhand der aufgestellten Eckpunkte zusammen und bewertet sie. Der daraufhin folgende Ausblick umreißt ein Folgeprojekt.

7.1 Zusammenfassung und Fazit

Die Automobilindustrie sieht sich aktuell vielfältigen Herausforderungen gegenüber, welchen sie unter anderem durch eine vergrößerte Modellpalette und individualisierte sowie individualisierbare Angebote entgegentreten will. Um das eigene Geschäft weiterhin rentabel zu halten, ist neben der Verbreiterung der Angebotspalette eine stark erhöhte Gleichteilverwendung nötig. Hierzu werden modulare Fahrzeugbaukästen eingeführt, die Module anhand bestimmter Kriterien – wie bspw. Einbaurichtung des Motors – gruppieren. Auf diese Weise werden sie für eine ganze Reihe von Fahrzeugen und Fahrzeugklassen zur Verfügung gestellt. Durch diese beiden Faktoren – Erhöhung der Varianten auf der einen und Erhöhung der Gleichteilverwendung auf der anderen Seite – steigen die Relationenkomplexität innerhalb der Unternehmen und hierdurch mittelbar die Gemeinkosten. Die Verwaltung dieser Elemente findet in verschiedenen Dokumenten, Systemen und Prozessen im Unternehmen statt, die nicht für derartig komplexe Bedingungen konzipiert wurden und daher im Zusammenspiel oftmals nicht entsprechend kosteneffizient skalieren. Auf diese Weise können Entscheidungen in Entwicklungsprojekten zu erhöhten Gemeinkosten führen, welche aufgrund der unübersichtlichen Verknüpfung von verschiedenen physischen und virtuellen Verwaltungselementen nicht angemessen erfasst werden. Auch ist eine Zuordnung von Gemeinkosten zu einem Projekt nicht unbedingt möglich. Bisher wird kostenseitig – im Sinne des Target-Costing – vor allem die Veränderung von Einzelkosten in Entscheidungen mit einbezogen. Die Literatur zeigt, dass negative Veränderungen von Gemeinkosten die erwarteten Kosteneffekte der Einzelkosten wieder aufwiegen oder sogar übersteigen lassen. Entscheidungen würden daher anders ausfallen, falls diese Auswirkungen im Vorfeld bekannt wären.

7.1.1 Zusammenfassung der Arbeit anhand der formulierten Eckpunkte

Die Formulierung der Problemstellung in Abschnitt 1.3 macht deutlich, dass es bisher keine Möglichkeit gibt, die Auswirkungen von Änderungen an Produktdaten auf die Prozesse eines Unternehmens und damit seiner Gemeinkosten präzise zu antizipieren. Komplexitätsbetrachtungen bleiben bei Entscheidungen auf Produktebene mangels Werkzeugunterstützung wage. Ursächlich ist die fehlende Verknüpfung von Produkt- und Prozessinformationen. In dieser Arbeit wird daher

eine Möglichkeit vorgeschlagen, diese Verknüpfung herzustellen und durch Software nutzbar zu machen. Dazu werden fünf Eckpunkte analysiert und diskutiert.

Eckpunkt 1 stellt die Definition und Einführung des Begriffs „Artefakte“ dar. Ein bei der Marke Volkswagen der Volkswagen AG durchgeführtes Projekt hat dazu die verschiedenen Aktivitäten des Anforderungsmanagementes während der Fahrzeugentstehung analysiert. Im Fokus standen hierbei die modularen Fahrzeugbaukästen, deren komplexe Verknüpfungen Einfluss auf die Entwickung von neuen Fahrzeugen, die strategische Planung sowie die Prozesse des Unternehmens haben. Ein Tool-Scouting zeigt, dass heutige am Markt verfügbare Werkzeuge nur begrenzt in der Lage sind Produkt- und Prozessinformationen zu verknüpfen.

Es wurden daher strukturierte Interviews eingesetzt, um Szenarien für die zukünftigen Ausprägungen des Anforderungsmanagements zu erarbeiten. Die agile Entwicklung eines webbasierten Prozesseditors demonstriert, auf welche Weise ein Werkzeug basierend auf formalisierten Prozesselementen erstellt werden kann. Es zeigte sich jedoch, dass ein übergeordneter Begriff für die verschiedenen Objekte in Prozessen und Produkten gefunden werden muss, welcher als gemeinsame Kommunikationsbasis dient und eine domänenübergreifende Verknüpfung zulässt. Der Begriff „Artefakt“ ist hierbei weitaus weniger von Mehrdeutigkeit im Modellierungskontext betroffen als bspw. „Objekt“. Kapitel 4 nimmt daher eine Einführung des Begriffs vor, welche in der Definition 4.1.1 mündet. Auch werden dort der in dieser Arbeit verwendete Komplexitätsbegriff bestimmt und die Arten der Beeinflussung auf die Komplexität im Unternehmen beschrieben. Zwei Beispiele runden das Kapitel ab. Artefakte bilden damit die Grundlage für den weiteren Aufbau der Arbeit.

Eckpunkt 2 liefert die Beschreibung von Anforderungen, Funktionen und Nutzen eines Werkzeugs für ein durchgängiges, semantisch unterstütztes Anforderungsmanagement. Zunächst werden semantische Netze als Technologie eingeführt, welche die Modellierung von Artefakten ermöglichen. Danach folgt die Definition von vier Funktionsbereichen: Modellierung von Artefakten und deren Zusammenhängen, Erstellung und Verwaltung von Instanzen der Artefakte, Suchen und Finden von Informationen sowie automatisierbare Auswertung und Analyse der vorhandenen Informationen. Aus diesen ergeben sich insgesamt 17 Anforderungen an ein Werkzeug, welche in Tabelle 5.1 zusammengefasst dargestellt sind. Gegliedert sind diese nach den Funktionsbereichen. Nach der Beschreibung der einzelnen Anforderungen wird der Nutzen für verschiedene Disziplinen betrachtet, falls die Umsetzung der Anforderungen in das beschriebene werkzeugunterstützte Anforderungsmanagement mündet. Die Aufgabenbereiche Projektmanagement, Implementierung, Testen, Marketing und Vertrieb, Änderungsmanagement sowie Verträge und andere Dokumente verfügen über direkte Schnittstellen zum Anforderungsmanagement und profitieren daher von der beschriebenen Funktionalität. Der Nutzen ergibt sich hierbei aus der Möglichkeit, mehr Informationen schneller einsehen zu können sowie automatisierbarer Impact-Analysen. Nach einem einführenden Beispiel in die Modellierung von Artefakten folgt eine Nutzenbetrachtung. Waren die in Abschnitt 5.3 genannten Disziplinen – Projektmanagement, Implementierung, Testen, Marketing und Vertrieb, Änderungsmanagement sowie das Rechtswesen – noch direkt durch die Möglichkeiten des Anforderungsmanagements betroffen, so fallen die in Abschnitt 5.5 genannten – erweitertes Anforderungsmanagment, Gemeinkosten, Produktdefinition und Eigenschaftsplanung, Änderungsmanagement, Baukastenplanung und -mangement und das Generieren juristischer Dokumente – nicht direkt in die traditionell durch Anforderungsmanagement berührte Kategorien.

Nichtsdestotrotz ergeben sich Synergieeffekte durch die gemeinsame Betrachtung. Abschnitt 5.5.2 geht hierbei explizit auf die Möglichkeiten der Beeinflussung von Gemeinkosten durch Impact-Analysen beginnend bei Anforderungen ein.

Der Eckpunkt 3 befasst sich mit der Bereitstellung von grundlegenden formalisierten Artefakten als Oberklassen für weitere Unternehmensartefakte. Nach einer Einführung in die Funktionsweise der Sprachen RDF, OWL und SPARQL sowie die technischen Grundlagen und Voraussetzungen der JENA API, wird ein semantisches Netz konstruiert und Abfragen daran gerichtet. Danach werden Kernartefakte formuliert und in der Ontologie Core formalisiert. Mithilfe der erarbeiteten Ontologie RMNet werden sie dann mit der spezifischen Anforderungsmanagement-Ontologie SWORE verknüpft. Das Konstrukt aus Core, RMNet und SWORE bildet eine Grundlage für das Entwickeln eines unternehmensspezifischen Artefaktmodells, auch Wissensbasis genannt. Abfragen in SPARQL profitieren von der Reasoning-Funktionalität, die etwa im Jena-Framework gegenüber konventionellen Datenbank-Abfragesprachen wie bspw. der Standard Query Language (SQL) integriert ist. So können Abfragen laufend aktualisierte Ergebnisse liefern. Das Erstellen von Abfragen kann unterstützt werden durch die Übersetzung von natürlicher Sprache in SPARQL-Abfrage. Diverse Forschungsarbeiten sind hierzu bereits verfügbar [WXZY07, DAC12]. In der Studie [KB07] wurde gezeigt, dass die Verwendung ganzer Sätze durch Anwender bei der Erstellung von Abfragen für semantische Netze hilfreich ist.

Eckpunkt 4 beschreibt eine Methodik zum Erstellen und Pflegen der Wissensbasis. Sie besteht aus den Schritten „Initiale Erstellung“, „Verwendung“, „Markierung“ sowie „Prüfung und Überarbeitung“. Beschrieben werden insbesondere der Einsatz von manuellen und werkzeugunterstützten Vorgehen zur Pflege der Inhalte der Wissensbasis.

Im letzten Eckpunkt wird die Operationalisierung des Werkzeugs behandelt und für die einzelnen Disziplinen im engeren und weiteren Kontext des Anforderungsmanagements geprüft, ob sich die beschriebenen Nutzenpotentiale realisieren lassen. Hierzu wird exemplarisch ein Artefaktmodell für die Produktentstehung konzipiert und somit zunächst das Vorgehen zur Entwicklung einer unternehmensspezifischen Wissensbasis anhand konkreter Artefakte gezeigt. Es werden dann typische Aufgaben aus den jeweiligen Disziplinen betrachtet und diese als Abfragen in der Sprache SPARQL formuliert. Die Ergebnisse werden aufgeführt und interpretiert. Das Kapitel schließt in Abschnitt 6.5 mit einer Analyse der Herausforderungen, die dem konzipierten Anforderungsmanagementwerkzeug bevorstehen. Hier ragt insbesondere das automatisierte Betrachten von Kostenveränderungen neben den anderen Punkten Berechtigungen, Aktualität und Leistungsfähigkeit hervor. Für alle Punkte werden Lösungsansätze angedeutet.

7.1.2 Fazit

Das Ziel dieser Arbeit ist die Schaffung einer Grundlage für ein Werkzeug zur Entscheidungsunterstützung in komplexen Strukturen von Großunternehmen. Dies sollte durch die Definition von Begrifflichkeiten, Methodik und der Beschreibung des Werkzeugs selbst geschehen.

Es kristallisieren sich die folgenden Ergebnisse heraus:

1. Einsatz von Szenariotechnik zur Betrachtung der Entwicklung der Anforderungsmanagement-Prozesslandschaft eines Industrieunternehmens.

2. Durchführung eines Tool-Scoutings am Markt für Anforderungsmanagementwerkzeuge.
3. Entwicklung eines webbasierten Prozesseditors sowie eines webbasierten Werkzeugs zur Visualisierung von verknüpften Informationen.
4. Einführung des Artefakt-Begriffs.
5. Beschreibung von Kernartefakten als Grundlage für unternehmensspezifische Artefaktlandschaften als Wissensbasen.
6. Formalisierung der Kernartefakte in der Ontologie Core.
7. Ergänzung von Core als Prozessdaten-Sicht mit der Produktdaten-Sicht durch Entwicklung der Ontologie RMNet und Verknüpfung mit der Ontologie SWORE.
8. Anforderungen an ein semantisch unterstütztes Anforderungsmanagementwerkzeug.
9. Evaluation des Nutzens eines solchen Werkzeugs für verschiedene Disziplinen im engeren und weiteren Kontext des Anforderungsmanagements.
10. Beschreibung eines Prozesses zur Erarbeitung und Pflege einer unternehmensspezifischen Wissensbasis als Grundlage für das Werkzeug.
11. Demonstration der technischen Machbarkeit des Werkzeugs.

Aus dem beschriebenen Projekt mit Volkswagen resultieren die Betrachtungen von Komplexität und deren Einfluss auf Produkt- und Prozessdaten. Der Einsatz von Szenariotechnik nicht zur Betrachtung der Entwicklung eines Geschäftsfelds oder Marktes, sondern zur Einschätzung der Veränderung einer Prozesslandschaft, ist zielführend zur Evaluation dieser Einflüsse. Es zeigte sich, dass Entscheidungen im Kontext des Produkts großen Einfluss auf die in einem Unternehmen vorhandenen Prozesse haben. Allerdings wurde deutlich, dass keine Methoden oder Werkzeuge zur Verfügung stehen, diesen Einfluss im Vorfeld der Entscheidung ausreichend antizipieren zu können.

Aufbauend auf diesen Überlegungen ergab sich als ein Ergebnis dieser Arbeit eine Definition, welche das abstrakte Konstrukt der Artefakte in ein klar umrissenes Konzept überführt. Es ermöglicht, Produkt- und Prozessdaten miteinander zu verknüpfen und stellt daher den zentralen Teil der Definition der Begrifflichkeiten dar. Alle weiteren Überlegungen, Technologien und Konzepte basieren hierauf mit dem Ziel, Prozess- und Produktdaten nicht mehr getrennt zu betrachten, sondern nur die darin vorhandenen Elemente. Die Ontologien Core und RMNet als weitere zentrale Ergebnisse formalisieren diese Begrifflichkeiten und bilden damit die Grundlage für softwaregestützte Werkzeuge. Durch die Verfügbarkeit einer Oberklasse für wirkende Relationen – die transitive Relation `wirkt_auf` – können Impact-Analysen einfach durchgeführt werden.

Der umrissene Prozess stellt vor allem die Pflege einer erstellten konkreten Wissensbasis in den Vordergrund. Hier liegt langfristig die größte fachliche Herausforderung. Die angebotene Lösung des Markierens von Problemen gleichermaßen durch Benutzer und automatisierte Prozesse maximiert die Chance, dass vorhandene Informationen korrekt bleiben.

Die Anforderungen an ein Werkzeug für ein durchgängiges und semantisch unterstütztes Anforderungsmanagement resultieren aus dem Tool-Scouting sowie den Erfahrungen des Einsatzes des webbasierten Prozesseditors. Sie dienen einer zukünftigen Anwendungsentwicklung als Grundlage bei der Erarbeitung spezifischerer Anforderungen, indem sie die mindestens vorhandene Funktionalität beschreiben. Die Betrachtung des Nutzens für andere Disziplinen ermöglicht eine entsprechend breite Formulierung eines etwaigen Entwicklungs- und Einführungsprojekts.

Kapitel 6 zeigt, dass Ontologien im Allgemeinen und die entwickelten Ontologien Core und RMNet im Speziellen eine Grundlage für ein Werkzeug sein können. Aufgrund der Fähigkeit des „Reasonings“ der Abfrage-API können Fragen ohne vollständige Kenntnis des Aufbaus des semantischen Netzes formuliert werden. Die durchgeführte exemplarische Erprobung demonstriert, auf welche Weise konkrete operative Probleme gelöst werden können.

Offen bleiben vier Herausforderungen, die sich bei der Implementierung des beschriebenen Werkzeugs in einem Unternehmen technisch sowie fachlich stellen. Das automatisierte Betrachten von Kostenveränderungen wurde bereits eingehend diskutiert. Es zeigt sich, dass Lösungen bisher nur mit Einschränkungen möglich sind. Zu untersuchen bleibt, inwieweit diese Einschränkungen in Kauf genommen werden können, um über die erweiterte Funktionalität des Werkzeugs zu verfügen. Die Leistungsfähigkeit des Werkzeugs ist ebenfalls eine technische Herausforderung, welcher aber bereits mit heutigen Technologien gut begegnet werden kann. Darüber hinaus kann das Problem auch von einer fachlichen Seite betrachtet werden, indem bei der Modellierung mit Sorgfalt und unter Verwendung von Standards vorgegangen wird.

Die Punkte Berechtigung und Aktualität haben zwar eine technische Komponente, sind allerdings zuerst fachlich zu behandeln. Zu klären ist die notwendige Tiefe der Berechtigungen der Benutzer sowie die konkrete Ausgestaltung des Prozesses zur Erhaltung der Aktualität des beschriebenen Wissens. Insbesondere für den letzten Punkt stellt die Arbeit aber das Konzept der Markierung durch Anwender zur Verfügung. Wird dieses konsequent angewendet, ist bereits eine Minimierung der nicht mehr aktuellen Informationen zu erwarten.

Hier ragt insbesondere das automatisierte Betrachten von Kostenveränderungen neben den anderen Punkten Berechtigungen, Aktualität und Leistungsfähigkeit hervor.

7.2 Ausblick

Der Fokus der vorliegenden Ausarbeitung liegt auf der Explikation von Grundlagen und dem Herstellen von disziplinübergreifenden Zusammenhängen. Gezeigt wurde das Potential der Erweiterung des Fokus des Anforderungsmanagements um Prozessartefakte bei der Bewertung von Anforderungen. Sie demonstriert weiterhin, was technisch möglich ist sowie an welchen Stellen Vorteile durch den Einsatz erreicht werden können und bietet somit das Potential für weitergehende Forschungsarbeiten. Es kann hierbei unterschieden werden zwischen einem Folgeprojekt, welches konkret die vorgeschlagene Methodik und das zugehörige Werkzeug erprobt sowie grundsätzlichen fachlichen Fragestellungen, die sich im Laufe der Arbeit ergaben.

7.2.1 Aufgaben für ein Folgeprojekt

Es bleibt zu untersuchen, inwieweit ein voll funktionsfähiges und eingesetztes Werkzeug zusammen mit einer gefüllten Wissensbasis und einer operationalisierten Methodik nachhaltig die Gemeinkosten beeinflussen kann. Die Folge wäre ein Feldversuch, in dem das beschriebene Werkzeug bis zu einer gewissen Reife entwickelt und mithilfe der vorhandenen Ontologien in einem Entwicklungsprojekt eingesetzt wird. Zu klären sind hierbei einerseits technische Aspekte der konkreten technischen Umsetzung des Werkzeugs sowie einer eventuellen Erweiterung der Ontologie Core und RMNet. Andererseits sind Managementaspekte zu klären, etwa inwieweit sich Entscheider auf das Werkzeug verlassen können und dies in operativen Abläufen auch tun. Die Arbeit hat gezeigt, dass vor allem durch den Einsatz des Reasoning und einer konsequenten Modellierung sich wertvolle und verlässliche Aussagen mithilfe des Werkzeugs treffen lassen. Ebenfalls schwierig gestaltet sich die Einschätzung der erwarteten Kosten einer Einführung sowie des Betriebs des beschriebenen Systems in einem Regelbetrieb. In einem eventuellen Projekt muss daher ein Fokus auf dem Controlling der Finanzen liegen, um eine fundierte Aussage hinsichtlich dieser Faktoren treffen zu können.

Ein weiterer zu untersuchender Punkt ist die Effektivität der Einbindung der Benutzer in die Pflege der Informationen. In Abschnitt 6.3 wurde angeregt, anstelle eines voll automatisierten Pflegeprozesses Mitarbeiter durch technische Möglichkeiten wie bspw. Annotationen dazu anzuhalten, Informationen als fehlerhaft oder verbesserungswürdig zu kennzeichen, bzw. gleich Korrekturen zur Verfügung zu stellen. Hierzu sollte eine Studie zur Benutzerfreundlichkeit solcher Vorgehen durchgeführt werden, bevor das Verfahren etwa im Rahmen des bereits erwähnten Feldversuchs erprobt werden kann. Die Ergebnisse könnten einen Beitrag bei der konkreten Ausgestaltung eines Pflegeprozesses leisten.

Darüber hinaus bieten Teilbereiche der Arbeit das Potential, nützlich bei anderen Problemstellungen im Kontext der Fahrzeugentstehung zu sein. Die automatisierte Unterstützung der langfristigen Planung von Baukästen kann durch Verwendung von typisierten und formalisierten Anforderungen im Abgleich mit Proto-Stücklisten oder Stücklisten-Strukturen ausgebaut werden. Die Volkswagen Vehicle Ontology [Hep] stellt bereits viele Informationen über die in Fahrzeugen und darüber hinaus enthaltenen Elemente zur Verfügung. Da sie in RDF vorliegt, kann sie in das bestehende Konstrukt einfach eingebunden werden.

Methodik und Werkzeug sind weiterhin nicht auf den Einsatz in der Automobilindustrie begrenzt. Der Aufbau einer Ontologie von Artefakten mithilfe der Szenariotechnik ist für jedes Unternehmen möglich, welches Entwicklungsprojekte durchführt. Es ist allerdings zu klären, ab welcher Unternehmens- und Projektgröße ein formales Anforderungsmanagement dieser Natur sinnvoll und lohnenswert erscheint. Weitere Untersuchungen können daher einerseits den Aspekt des vorauszusetzenden Größenumfangs klären sowie andererseits Branchen oder Unternehmen identifizieren, bei denen der Einsatz in Frage kommt. Danach kann eine Erprobung in einem solchen veränderten Kontext stattfinden.

7.2.2 Fachliche Herausforderungen

Steht ein auf den Ergebnissen dieser Arbeit aufsetzendes Projekt vor allem vor der Schwierigkeit der Evaluation der Effektivität von Methodik und Werkzeug, so existieren weitere Fragestellun-

gen hinsichtlich spezieller Teilaspekte. Die aufgezeigten Lösungen zur Einführung gewichteter Relationen sind nicht näher betrachtet worden. Zwar wurde grundsätzlich die Anwendbarkeit demonstriert, jedoch sind die Lösungen mithilfe von Relationsklassen oder veränderter Gültigkeit des Objekts bei Relationen nicht unbedingt praktikabel. Es geht die Allgemeingültigkeit des Netzes verloren, welches bspw. beim Reasoning eine große Rolle spielt. Angeregt wurde, etwa eine Transformation zu entwerfen, welche das Netz nach außen hin in einer allgemeingültigen Form ohne gewichtete Relationen darstellt. Eine andere Möglichkeit ist der Einsatz eines Präprozessors bei Abfragen mithilfe einer eigens entworfenen API, welche mit Abfragen nach gewichteten sowie ungewichteten Relationen umgehen kann. Der Präprozessor könnte sich an einer Wertetabelle hinsichtlich der Gewichtungen orientieren. In einer weitergehenden Arbeit können die Varianten evaluiert und auf ihre Operationalisierbarkeit geprüft werden.

Es bleibt weiterhin zu untersuchen, ob es möglich ist, ein Modul aufgrund bestimmter Anforderungen an das Fahrzeug auszuwählen. Eine erweiterte Ontologie kann bspw. einen Motor aufgrund der Umwelt-Anforderungen an ein Fahrzeug heraussuchen. Hierzu werden allerdings folgende Informationen in formaler Form benötigt: Erstens muss ein Typ `Aggregat` vorhanden sein, der über bestimmte Eigenschaften verfügt, wie bspw. `CO2-Ausstoß`. Zweitens müssen Anforderungen, die bisher nur in Textform vorliegen, in eine Form gebracht werden, die den Inhalt der Anforderung formal beschreibt. Auch hier könnte bspw. ein Punkt `CO2-Ausstoß` vorhanden sein, der dann über einen Algorithmus mit den Eigenschaften der Module abgeglichen werden kann.

Bezüglich der `Werte` und `Milieus` findet bisher eine reine Fokussierung auf einen deutschen Markt statt. In anderen Regionen, in denen ein Fahrzeug vertrieben wird, existieren andere Milieus, bzw. setzen sich die Milieus aus anderen Werten zusammen. Die Ontologie unterstützt diese differenzierende Sichtweise bisher noch nicht. Eine Ergänzung an dieser Stelle führt auch zu einer erweiterten Eigenschafts-, Baukasten- und Marketingplanung. Der damit verbundene Umfang sprengt allerdings den Rahmen dieser Arbeit. Es liegt aber die Vermutung nahe, dass die Einbeziehung der Internationalität in die Ontologie und somit in die Entscheidungsprozesse eine Verbesserung der Aussagekraft des Anforderungsmanagements mit sich bringen würde. Die Durchführung einer wissenschaftlichen Arbeit mit dem Fokus auf der Einbeziehung verschiedener Märkte und Regionen ist daher vielversprechend.

Bereits im obigen Abschnitt 7.1.1 wurde auf die Möglichkeit der Verwendung von natürlicher Sprache hingewiesen. Studien haben gezeigt, dass hierdurch die Akzeptanz eines Systems mit semantischer Unterstützung steigt [KB07]. Es stehen bereits verschiedene Technologien zur Verfügung, natürlichsprachlich formulierte Abfragen in SPARQL-Abfragen zu übersetzen, womit zu untersuchen bleibt, ob diese bei den in dieser Arbeit beschriebenen Methoden und Modellen zu Mehrwert in der Verwendung führen. Der Einsatz solcher Technologie hat einen direkten Einfluss auf das Interfacedesign der Software und damit auch auf die Akzeptanz bei den Anwendern.

Literaturverzeichnis

[ABNM06] ALMEFELT, Lars ; BERGLUND, Fredrik ; NILSSON, Patrik ; MALMQVIST, Johan: Requirements management in practice: findings from an empirical study in the automotive industry. In: *Research in Engineering Design* 17 (2006), Nr. 3, S. 113–134

[AG08] ANGLES, Renzo ; GUTIERREZ, Claudio: The Expressive Power of SPARQL. In: *Proceedings of the 7th International Semantic Web Conference (ISWC 2008)*, 2008 (Lecture Notes in Computer Science (LNCS)), S. 114–129

[AH08] ANTONIOU, Grigoris ; HARMELEN, Frank van: *A Semantic Web Primer*. second edition. Cambridge, MA, USA : MIT Press, 2008

[All07] ALLMANN, Christian: Anforderungen auf Kundenfunktionsebene in der Automobilindustrie. In: *SE 2007 – die Konferenz rund um Softwaretechnik*, 2007

[All09] ALLWEYER, Thomas: *BPMN 2.0 - Business Process Model and Notation*. Norderstedt : Books on Demand, 2009

[And06] ANDRES, Thomas: Vom Geschäftsprozess zur Anwendung: Modellgetriebene Entwicklung betriebswirtschaftlicher Software. In: *Agilität durch ARIS Geschäftsprozessmanagement*. Berlin : Springer, 2006, S. 231–242

[Apaa] APACHE SOFTWARE FOUNDATION: *Apache POI Project* `http://poi.apache.org/`. – (Stand: 01.10.2013)

[Apab] APACHE SOFTWARE FOUNDATION: *Jena* `http://jena.apache.org/`. – (Stand: 04.06.2012)

[Apac] APACHE SOFTWARE FOUNDATION: *Maven* `http://maven.apache.org/`. – (Stand: 16.06.2013)

[Apad] APACHE SOFTWARE FOUNDATION: *Tomcat* `http://tomcat.apache.org/`. – (Stand: 01.10.2013)

[ARSLD03] AUSTIN, Duncan ; ROSINSKI, Niki ; SAUER, Amanda ; LE DUC, Colin: The Impact of Climate Change on Competitiveness and Value Creation in the Automotive Industry / World Resources Institute (WRI). 2003. – Forschungsbericht

[AS12] ALBERT, Kristin ; STILLER, Michael: Der Browser als mobile Plattform der Zukunft – Die Möglichkeiten von HTML5-Apps. In: *Smart Mobile Apps. Mit Business-Apps ins Zeitalter mobiler Geschäftsprozesse.* Berlin : Springer, 2012 (Xpert.press), S. 147–160

[Asc06] ASCHEBERG, Carsten: Milieuforschung und Transnationales Zielgruppenmarketing. In: *Aus Politik und Zeitgeschichte* 56 (2006), Nr. 44/45, S. 18–25

[Bal11] BALZERT, Helmut: Verteilte Architekturen. In: *Lehrbuch der Softwaretechnik: Entwurf, Implementierung, Installation und Betrieb*. Heidelberg : Spektrum Akademischer Verlag, 2011, S. 191–204

[BBS04] BENSBERG, Frank ; BROCKE, Jan vom ; SCHULTZ, Martin B.: *Trendberichte zum Controlling: Festschrift für Heinz Lothar Grob*. Heidelberg : Physica-Verlag HD, 2004

[Bec07] BECKER, Helmut: *Auf Crashkurs: Automobilindustrie im globalen Verdrängungswettbewerb*. 2., aktualisierte Aufl. Berlin : Springer, 2007

[BGG+07] BARAL, C. ; GONZALEZ, G. ; GITTER, A. ; TEEGARDEN, C. ; ZEIGLER, A. ; JOSHI-TOPÉ, G.: CBioC: beyond a prototype for collaborative annotation of molecular interactions from the literature. In: *Computational systems bioinformatics / Life Sciences Society. Computational Systems Bioinformatics Conference* 6 (2007), S. 381–384

[BGR09] BERGER, Christian ; GÜLKE, Tim ; RUMPE, Bernhard: ProcDSL + ProcEd – a Web-based Editing Solution for Domain Specific Process-Engineering. In: *Proceedings of the 9th OOPSLA Workshop on Domain-Specific Modeling (DSM '09)*, 2009, S. 70–74

[BLC] BERNERS-LEE, Tim ; CONNOLLY, Dan: *Notation3 (N3): A readable RDF syntax* `http://www.w3.org/TeamSubmission/n3/`. – (Stand: 26.10.2013)

[BLP04] BÜHNE, Stan ; LAUENROTH, Kim ; POHL, Klaus: Anforderungsmanagement in der Automobilindustrie: Variabilität in Zielen, Szenarien und Anforderungen. In: *Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI)* Bd. 2, GI, 2004 (Lecture Notes in Informatics (LNI)), S. 23–27

[BM11] BROSSMANN, Michael ; MÖDINGER, Wilfried: *Praxisguide Wissensmanagement*. Berlin : Springer, 2011

[Bod03] BODENDORF, Freimut: *Daten-und Wissensmanagement*. Berlin : Springer, 2003

[BS11] BRAESS, Hans-Hermann ; SEIFFERT, Ulrich: *Vieweg Handbuch Kraftfahrzeugtechnik*. Wiesbaden : Springer, 2011

[BS12] BERGER, Christian ; SIEGL, Sebastian: Constructive Requirements Modeling - More Reliable Implementations in a Shorter Time. In: *Proceedings of the Conference Automotive – Safety & Security 2012*, GI, 2012 (Lecture Notes in Informatics (LNI)), S. 149–162

[BSW+07] BRAUN, Simone ; SCHMIDT, Andreas P. ; WALTER, Andreas ; NAGYPAL, Gabor ; ZACHARIAS, Valentin: Ontology maturing: a collaborative web 2.0 approach to ontology engineering. In: *Proceedings of the Workshop on Social and Collaborative*

Construction of Structured Knowledge at the 16th International World Wide Web Conference (WWW2007), Banff, Canada Bd. 273, 2007

[Bus06] BUSCH, Rainer: Der Siegeszug der Platine. In: *Automobilwoche* 23 (2006)

[But11] BUTZ, Henning: Systemkomplexität methodisch erkennen und vermeiden. In: *Anforderungsmanagement in der Produktentwicklung*. Düsseldorf : Symposion Publishing, 2011, S. 183–217

[BV96] BECKER, Jörg ; VOSSEN, Gottfried: Geschäftsprozeßmodellierung und Workflow-Management: Eine Einführung. In: *Geschäftsprozeßmodellierung und Workflow-Management. Modelle, Methoden, Werkzeuge*. Bonn, Albany : International Thomson Publishing, 1996 (Informatik Lehrbuch-Reihe), S. 17–26

[BWW10] BECKER, Jörg ; WEISS, Burkhard ; WINKELMANN, Axel: Utility vs. Efforts of Business Process Modeling An Exploratory Survey in the Financial Sector. In: *Proceedings of the Multikonferenz Wirtschaftsinformatik 2010*, 2010, S. 41–54

[Ciu09] CIURANA, Eugene: *Developing with Google App Engine*. Berkeley, CA, USA : Apress, 2009

[cla] CLARK&PARSIA: *Pellet: OWL 2 Reasoner for Java* `http://clarkparsia.com/pellet`. – (Stand: 29.07.2012)

[CMM10] CMMI PRODUCT TEAM: CMMI®for Development, Version 1.3 / Carnegie Mellon University. Version: 2010. `http://www.sei.cmu.edu/library/abstracts/reports/10tr033.cfm`. 2010. – Forschungsbericht. – (Stand: 25.05.2011)

[CN02] CHIEU, Hai L. ; NG, Hwee T.: Named entity recognition: a maximum entropy approach using global information. In: *Proceedings of the 19th international conference on Computational linguistics - Volume 1*. Stroudsburg, PA, USA : Association for Computational Linguistics, 2002 (COLING '02), S. 1–7

[CS99] COLLINS, Michael ; SINGER, Yoram: Unsupervised models for named entity classification. In: *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 1999

[DAC12] DAMLJANOVIC, Danica ; AGATONOVIC, Milan ; CUNNINGHAM, Hamish: FREyA: An Interactive Way of Querying Linked Data Using Natural Language. In: *The Semantic Web: ESWC 2011 Workshops* Bd. 7117. Berlin : Springer, 2012, S. 125–138

[DBK03] DENGER, Christian ; BERRY, Daniel M. ; KAMSTIES, Erik: Higher quality requirements specifications through natural language patterns. In: *Proceedings of the IEEE International Conference on Software: Science, Technology and Engineering 2003 (SwSTE '03)*, IEEE, 2003, S. 80–91

[DH02] DAVIS, Alan M. ; HICKEY, Ann M.: Requirements Researchers: Do We Practice What We Preach? In: *Requirements Engineering* 7 (2002), S. 107–111

[Die06] DIEZ, Willi: *Automobil-Marketing: Navigationssystem für neue Absatzstrategien*. Landsberg am Lech : mi-Fachverlag, 2006

[dpa13] DPA: Rückruf: Volkswagen hat Probleme mit dem DSG `http://www.heise.de/autos/artikel/Rueckruf-Volkswagen-hat-Probleme-mit-dem-DSG-1887208.html`. In: *heise Autos* (2013). – (Stand: 07.11.2013)

[Dyc11] DYCK, Christoph: Anforderungsmanagement für Automobil- und Avioniksysteme. In: *Anforderungsmanagement in der Produktentwicklung*. Düsseldorf : Symposion Publishing, 2011, S. 57–73

[EAG06] ESPINOZA, Angelina ; ALARCON, Pedro P. ; GARBAJOSA, Juan: Analyzing and Systematizing Current Traceability Schemas. In: *Software Engineering Workshop, 2006. SEW '06. 30th Annual IEEE/NASA*, IEEE, 2006, S. 21–32

[Ebe10] EBERT, Christof: *Systematisches Requirements Engineering: Anforderungen ermitteln, spezifizieren, analysieren und verwalten*. Heidelberg : Dpunkt Verlag, 2010

[Ecl] ECLIPSE FOUNDATION: *Eclipse* `http://www.eclipse.org/`. – (Stand: 01.10.2013)

[ED11] ECKL-DORNA, Wilfried: *CO2-Vorgaben - Autohersteller in der Klemme* `http://www.manager-magazin.de/unternehmen/autoindustrie/0,2828,767321,00.html`. 2011. – (Stand: 10.07.2011)

[ES09] EIGNER, Martin ; STELZER, Ralph: *Product Lifecycle Management: Ein Leitfaden für Product Development und Life Cycle Management*. Berlin : Springer, 2009

[Fet08] FETTKE, Peter: Business Process Modeling Notation. In: *WIRTSCHAFTSINFORMATIK* 50 (2008), Nr. 6, S. 504–507

[GJRA12] GÜLKE, Tim ; JANSEN, Martin ; RUMPE, Bernhard ; AXMANN, Joachim: High-Level Requirements Management and Complexity Costs in Automotive Development Projects: A Problem Statement. In: *Proceedings of the 18th international conference on Requirements Engineering: foundation for software quality (REFSQ 2012)*, Springer, 2012 (Lecture Notes in Computer Science (LNCS)), S. 94–100

[Gru93] GRUBER, Thomas R.: A translation approach to portable ontology specifications. In: *Knowledge acquisition* 5 (1993), Nr. 2, S. 199–220

[Gup08] GUPTA, Vipul: *Accelerated GWT*. Berkeley, CA, USA : Apress, 2008

[GW89] GAUSE, Donald C. ; WEINBERG, Gerald M.: *Exploring Requirements: Quality Before Design*. New York, NY, USA : Dorset House Publishing, 1989

[GWT] GWT PROJECT: *GWT* `http://www.gwtproject.org/`. – (Stand: 01.10.2013)

[HB08] HÜTTENRAUCH, Mathias ; BAUM, Markus: *Effiziente Vielfalt: Die dritte Revolution in der Automobilindustrie*. Berlin : Springer, 2008

[Hei97] HEISE, Gilbert: *Internationale Marktsegmentierung im Automobilmarketing*. Wiesbaden : Gabler, 1997

[Hep] HEPP, Martin: *Volkswagen Vehicles Ontology* `http://www.volkswagen.co.uk/vocabularies/vvo/ns`. – (Stand: 04.11.2012)

[HF00] HOOKS, Ivy F. ; FARRY, Kristin A.: *Customer Centered Products: Creating Successful Products Through Smart Requirements Management*. New York, NY, USA : AMACOM/American Management Association, 2000

[HG06] HÖBEL, Elke ; GRUNDEI, Jens: Organisations-Controlling im Volkswagen-Konzern. In: *Organisations-Controlling*. Wiesbaden : Gabler, 2006, S. 285–298

[HMD11] HOLTMANN, J. ; MEYER, J. ; DETTEN, M. von: Automatic Validation and Correction of Formalized, Textual Requirements. In: *Proceedings of the ICST Workshop Requirements and Validation, Verification & Testing (ReVVerT 2011)*, IEEE, 2011, S. 486–495

[HMRV07] HOOD, Colin ; MÜHLBAUER, Susanne ; RUPP, Chris ; VERSTEEGEN, Gerhard: *iX-Studie Anforderungsmanagement*. 2. Aufl. Hannover : Heise Zeitschriften Verlag, 2007

[Hou03] HOUDEK, Frank: Requirements Engineering Erfahrungen in Projekten der Automobilindustrie. In: *Softwaretechnik-Trend* 23 (2003), Nr. 1

[HP08] HECK, Petra ; PARVIAINEN, Päivi: Experiences on Analysis of Requirements Quality. In: *Proceedings of The Third International Conference on Software Engineering Advances 2008 (ICSEA '08)*, IEEE, 2008, S. 367–372

[HRRW12] HOPP, Christian ; RENDEL, Holger ; RUMPE, Bernhard ; WOLF, Fabian: Einführung eines Produktlinienansatzes in die automotive Softwareentwicklung am Beispiel von Steuergerätesoftware. In: *Software Engineering 2012: Fachtagung des GI-Fachbereichs Softwaretechnik, 27. Februar - 2. März 2012 in Berlin* Bd. 198, 2012 (LNI), S. 181–192

[Hün11] HÜNNEKENS, Wolfgang: *Die Ich-Sender: Das Social Media-Prinzip - Twitter, Facebook & Communities erfolgreich einsetzen*. 3. Aufl. Göttingen : BusinessVillage, 2011

[HW06] HAYES, Pat ; WELTY, Chris ; NOY, Natasha (Hrsg.) ; RECTOR, Alan (Hrsg.): *Defining N-ary Relations on the Semantic Web*. `http://www.w3.org/TR/swbp-n-aryRelations/`. Version: 2006 (W3C Working Group Note). – (Stand: 23.11.2012)

[HW10] HAB, Gerhard ; WAGNER, Reinhard: *Projektmanagement in der Automobilindustrie: Effizientes Management von Fahrzeugprojekten entlang der Wertschöpfungskette*. 3., überarb. u. erw. Aufl. Wiesbaden : Springer Gabler, 2010

[HWFP08] HOOD, Colin ; WIEDEMANN, Simon ; FICHTINGER, Stefan ; PAUTZ, Urte: *Requirements Management: The Interface Between Requirements Development and All Other Systems Engineering Processes*. Berlin : Springer, 2008

[IEE90] IEEE: IEEE Standard Glossary of Software Engineering Terminology / IEEE. IEEE, 1990. – Forschungsbericht

[IEE98] IEEE: IEEE Recommended Practice for Software Requirements Specifications / IEEE. IEEE, 1998. – Forschungsbericht

[Insa] INSTITUT FÜR VERKEHRSSICHERHEIT UND AUTOMATISIERUNGSTECHNIK, TU BRAUNSCHWEIG: *iglos* `http://www.iglos.de`. – (Stand: 09.01.2013)

[Insb] INSTITUT FÜR VERKEHRSSICHERHEIT UND AUTOMATISIERUNGSTECHNIK, TU BRAUNSCHWEIG: *iglos req* `http://www.iva.ing.tu-bs.de/?iT=4_595&projectId=450`. – (Stand: 09.01.2013)

[Jäg08] JÄGER, Kai: *Ajax in der Praxis*. Berlin : Springer, 2008 (Xpert.press)

[Jam] JAMA: *Jama Contour* `http://www.jamasoftware.com/contour/`. – (Stand: 26.06.2012)

[JBo] JBOSS: *Hibernate* `http://www.hibernate.org///`. – (Stand: 01.10.2013)

[Kar] KARLSRUHE, Universität: *Semantic Mediawiki* `http://semantic-mediawiki.org/`. – (Stand: 08.07.2011)

[KB07] KAUFMANN, Esther ; BERNSTEIN, Abraham: How Useful Are Natural Language Interfaces to the Semantic Web for Casual End-Users? In: *The Semantic Web* Bd. 4825. Berlin : Springer, 2007, S. 281–294

[KD09] KONRAD, Sascha ; DEGEN, Helmut: Lessons Learned from the Use of Artifact Models in Industrial Projects. In: *Proceedings of the 17th IEEE International Requirements Engineering Conference 2009 (RE '09)*, IEEE, 2009, S. 349–354

[KGRS10] KOF, Leonid ; GACITUA, Ricardo ; ROUNCEFIELD, Mark ; SAWYER, Pete: Concept mapping as a means of requirements tracing. In: *Proceedings of The Third International Workshop on Managing Requirements Knowledge 2010 (MaRK '10), held in conjunction with The 18th International IEEE Requirements Engineering Conference (RE'10)*, IEEE, 2010, S. 22–31

[KKR11] KLUTE, Sandra ; KOLBE, Constanze ; REFFLINGHAUS, Robert: Die Planung komplexer Produkte. In: *Anforderungsmanagement in der Produktentwicklung*. Düsseldorf : Symposion Publishing, 2011, S. 75–100

[KMU11] KLUDE, Bert ; METIN, Altug ; ULBRICH, Armin: Anforderungsmanagement im Produktmarketing mit PLM. In: *Anforderungsmanagement in der Produktentwicklung*. Düsseldorf : Symposion Publishing, 2011, S. 103–120

[KOD09] KASSAB, M. ; ORMANDJIEVA, O. ; DANEVA, M.: A Metamodel for Tracing Non-functional Requirements. In: *Proceedings of the WRI World Congress on Computer Science and Information Engineering 2009 (CSIE 2009)* Bd. 7, IEEE, 2009, S. 687–694

[Koh96] KOHL, Claudia: Objektorientierte Analysekonzepte in der Unternehmensmodellierung. In: *Geschäftsprozeßmodellierung und Workflow- Management. Modelle, Methoden, Werkzeuge*. Bonn, Albany : International Thomson Publishing, 1996 (Informatik Lehrbuch-Reihe), S. 63–79

[Kre10] KREUTZER, Ralf T.: *Praxisorientiertes Marketing*. Wiesbaden : Gabler, 2010

[KRS12] KOWALEWSKI, Stefan ; RUMPE, Bernhard ; STOLLENWERK, Andre: Cyber-Physical Systems - Eine Herausforderung an die Automatisierungstechnik? In: *Automation 2012*, VDI-Verlag, 2012 (VDI Berichte). – Langfassung auf CD-Rom

[Kru03] KRUCHTEN, Philippe: *The Rational Unified Process: An Introduction*. 3rd Edition. Boston, MA, USA : Addison-Wesley Professional, 2003

[KRV10] KRAHN, Holger ; RUMPE, Bernhard ; VÖLKEL, Steven: MontiCore: A Framework for Compositional Development of Domain Specific Languages. In: *International Journal on Software Tools for Technology Transfer (STTT)* 12 (2010), S. 353–372

[KS98] KOTONYA, Gerald ; SOMMERVILLE, Ian: *Requirements Engineering : Processes and Techniques*. New York, NY, USA : John Wiley & Sons, 1998

[KT07] KAMATA, Mayumi I. ; TAMAI, Tetsuo: How Does Requirements Quality Relate to Project Success or Failure? In: *Proceedings of the 15th IEEE International Requirements Engineering Conference 2007 (RE '07)*, IEEE, 2007, S. 69–78

[KW67] KAHN, Herman ; WIENER, Anthony J.: *The year 2000: A Framework for Speculation on the Next Thirty-three Years*. New York : Macmillan, 1967

[Lei] LEIPZIG, Universität: *OntoWiki* `http://ontowiki.net/Projects/OntoWiki`. – (Stand: 08.07.2011)

[Lew05] LEWIN, Tony: VW executive: Modules are too complex. In: *Automotive News* 79 (2005), August, Nr. 6159, 24+. `http://www.autonews.com/apps/pbcs.dll/article?AID=/20050801/SUB/508010729`. – (Stand: 25.11.2012)

[Lif] LIFEHACKER: *Browser Speedtests* `http://lifehacker.com/browserspeedtests`. – (Stand: 10.02.2013)

[LMB08] LINDEMANN, Udo ; MAURER, Maik ; BRAUN, Thomas: *Structural Complexity Management: An Approach for the Field of Product Design*. Berlin : Springer, 2008

[Loo96] LOOS, Peter: Geschäftsprozessadäquate Informationssystemadaption durch generische Strukturen. In: *Geschäftsprozeßmodellierung und Workflow- Management. Modelle, Methoden, Werkzeuge*. Bonn, Albany : International Thomson Publishing, 1996 (Informatik Lehrbuch-Reihe), S. 163–175

[Mar04] MARSCHNER, Karina: *Wettbewerbsanalyse in der Automobilindustrie*. Wiesbaden : Deutscher Universitätsverlag, 2004

[MB08] MAYER-BACHMANN, Roland: *Integratives Anforderungsmanagement: Konzept und Anforderungsmodell am Beispiel der Fahrzeugentwicklung*. Karlsruhe : KIT Scientific Publishing, 2008

[MH11] MÜHLECK, Klaus H. ; HACKENBERG, Wolfgang: Universelles SAP-Template als Schlüssel zur Standardfabrik. In: *Informatik-Spektrum* 34 (2011), S. 29–37

[Mil07] MILTON, Nicholas R.: *Knowledge Acquisition in Practice: A Step-by-step Guide (Decision Engineering)*. London : Springer, 2007

[ML06] MORGAN, James M. ; LIKER, Jeffrey K.: *The Toyota Product Development System: Integrating People, Process And Technology*. New York, NY, USA : Productivity Press, 2006

[MSG07] MÖSSMER, H. E. ; SCHEDLBAUER, M. ; GÜNTHNER, W. A.: Die automobile Welt im Umbruch. In: *Neue Wege in der Automobillogistik*. Berlin : Springer, 2007 (VDI-Buch), S. 3–15

[Mur08] MURTY, James: *Programming Amazon Web Services: S3, EC2, SQS, FPS, and SimpleDB*. Sebastopol, CA, USA : O'Reilly Media, 2008

[MWHN09] MAVIN, Alistair ; WILKINSON, Philip ; HARWOOD, Adrian ; NOVAK, Mark: Easy Approach to Requirements Syntax (EARS). In: *Proceedings of the 17th IEEE International Requirements Engineering Conference*, IEEE, 2009, S. 317–322

[NE00] NUSEIBEH, Bashar ; EASTERBROOK, Steve: Requirements engineering: a roadmap. In: *Proceedings of the Conference on The Future of Software Engineering*, ACM, 2000 (ICSE '00), S. 35–46

[OB00] OLBRICH, Rainer ; BATTENFELD, Dirk: Forschungsbericht Nr. 3, Komplexität aus Sicht des Marketing und der Kostenrechnung. Hagen : Lehrstuhl für Betriebswirtschaftslehre, insb. Marketing, FernUniversität Hagen, 2000. – Forschungsbericht

[Obj] OBJECT MANAGEMENT GROUP (OMG): *Requirements Interchange Format (ReqIF)* `http://www.omg.org/spec/ReqIF/`. – (Stand: 08.06.2013)

[Obj11] OBJECT MANAGEMENT GROUP (OMG): *UML 2.4.1* `http://www.omg.org/spec/UML/2.4.1/`. 2011. – (Stand: 26.10.2013)

[Ora] ORACLE CORPORATION: *MySQL* `http://www.mysql.com//`. – (Stand: 01.10.2013)

[o.V10] O.V.: *Jahresbericht 2010 Kraftfahrt-Bundesamt* `http://www.kba.de/cln_031/nn_124834/DE/Presse/Jahresberichte/jahresbericht_2010.html/`. 2010. – (Stand: 19.08.2013)

[o.V11] O.V.: *Geschäftsbericht 2011 Volkswagen AG*. 2011

[o.V12] O.V.: *Jahresbericht 2012 Kraftfahrt-Bundesamt* `http://www.kba.de/cln_031/nn_124834/DE/Presse/Jahresberichte/jahresbericht_2012.html`. 2012. – (Stand: 19.08.2013)

[Pil07] PILLKAHN, Ulf: *Trends und Szenarien als Werkzeuge zur Strategieentwicklung: Wie Sie die unternehmerische und gesellschaftliche Zukunft planen und gestalten*. Erlangen : Publicis, 2007

[PS] PRUD'HOMMEAUX, Eric ; SEABORNE, Andy: *SPARQL* `http://www.w3.org/TR/rdf-sparql-query/`. – (Stand: 02.06.2012)

[Ram02] RAMESH, Balasubramaniam: Process knowledge management with traceability. In: *Software, IEEE* 19 (2002), Nr. 3, S. 50–52

[RBL+08] RAUSKOLB, Fred W. ; BERGER, Kai ; LIPSKI, Christian ; MAGNOR, Marcus ; CORNELSEN, Karsten ; EFFERTZ, Jan ; FORM, Thomas ; GRAEFE, Fabian ; OHL, Sebastian ; SCHUMACHER, Walter ; WILLE, Jörn M. ; HECKER, Peter ; NOTHDURFT, Tobias ; DOERING, Michael ; HOMEIER, Kai ; MORGENROTH, Johannes ; WOLF, Lars ; BASARKE, Christian ; BERGER, Christian ; GÜLKE, Tim ; KLOSE, Felix ; RUMPE, Bernhard: Caroline: An autonomously driving vehicle for urban environments. In: *Journal of Field Robotics* 25 (2008), Nr. 9, S. 674–724

[RDFa] RDF WORKING GROUP, W3C: *Terse RDF Triple Language: W3C Working Draft 10 July 2012* `http://www.w3.org/TR/2012/WD-turtle-20120710/`. – (Stand: 18.11.2012)

[RDFb] RDFCORE WORKING GROUP, W3C: *RDF Primer* `http://www.w3.org/TR/2004/REC-rdf-primer-20040210/`. – (Stand: 26.07.2011)

[RDFc] RDFCORE WORKING GROUP, W3C: *RDF Vocabulary Description Language 1.0: RDF Schema* `http://www.w3.org/TR/2004/REC-rdf-schema-20040210/`. – (stand: 26.07.2011)

[Ric09] RICHTER, Harald: Elektronik und Datenkommunikation im Automobil / TU Clausthal, Department of Informatics. 2009 (IfI-09-05). – Forschungsbericht

[Rin06] RINGLE, Tanja: *Strategische identitätsorientierte Markenführung: Mit Fallstudien aus der Automobilindustrie*. Wiesbaden : Deutscher Universitätsverlag, 2006

[RJ01] RAMESH, B. ; JARKE, M.: Toward reference models for requirements traceability. In: *Software Engineering, IEEE Transactions on* 27 (2001), Nr. 1, S. 58–93

[RK96] REHÄUSER, J. ; KRCMAR, H.: Wissensmanagement im Unternehmen. In: *Managementforschung 6: Wissensmanagement*. Berlin : de Gruyter, 1996 (Managementforschung), S. 1–40

[RLL07] RIECHERT, Thomas ; LAUENROTH, Kim ; LEHMANN, Jens: Semantisch unterstütztes Requirements Engineering. In: *Proceedings of the First Conference on Social Semantic Web 2007 (CSSW 2007)* Bd. 113, GI, 2007 (Lecture Notes in Informatics (LNI))

[RR06] ROBERTSON, Suzanne ; ROBERTSON, James C.: *Mastering the Requirements Process*. 2nd edition. Boston, MA, USA : Addison-Wesley Professional, 2006

[RSG08] RISHI, Sanjay ; STANLEY, Benjamin ; GYIMESI, Kalman: Automotive 2020: Clarity Beyond The Chaos / IBM Institute for Business Valuebb. 2008. – Forschungsbericht

[Rum11] RUMPE, Bernhard: *Modellierung mit UML: Sprache, Konzepte und Methodik*. 2. Aufl. 2011. Berlin : Springer, 2011

[Rum12] RUMPE, Bernhard: *Agile Modellierung mit UML*. 2. Aufl. Berlin : Springer, 2012 (Xpert.press)

[Rup11] RUPP, Chris: Qualitätssicherung von Anforderungsspezifikationen. In: *Anforderungsmanagement in der Produktentwicklung*. Düsseldorf : Symposion Publishing, 2011, S. 37–56

[SBF98] STUDER, Rudi ; BENJAMINS, V R. ; FENSEL, Dieter: Knowledge engineering: principles and methods. In: *Data & knowledge engineering* 25 (1998), Nr. 1, S. 161–197

[Sch92] SCHEER, August-Wilhelm: *Architektur integrierter Informationssysteme: Grundlagen der Unternehmensmodellierung*. 2., verb. Aufl. Berlin : Springer, 1992

[Sch94] SCHEHL, Michael: *Die Kostenrechnung der Industrieunternehmen vor dem Hintergrund unternehmensexterner und -interner Strukturwandlungen: eine theoretische und empirische Untersuchung*. Berlin : Duncker & Humblot, 1994

[Sch10a] SCHERER, Alexander: *Entwicklung einer Methodik zur Informations- und Datenmodellierung in IT-Service-Management-Prozessen am Beispiel der ITIL-Prozesse Service Level Management und Configuration Management*. Norderstedt : GRIN Verlag, 2010

[Sch10b] SCHULZ, Joachim: Einführung eines Requirements Management Tools - Best Practice und Fallstricke von der Prozessdefinition bis zum Softwarebetrieb. In: *OBJEKTspektrum* RE/2010 (2010)

[Sch12] SCHÖMANN, S. O.: *Produktentwicklung in der Automobilindustrie: Managementkonzepte vor dem Hintergrund gewandelter Herausforderungen*. Wiesbaden : Gabler Verlag, 2012

[SG08] SCHWABER, Carey ; GERUSH, Mary: *The Forrester Wave: Requirements Management, Q2 2008*. `http://www.mks.com/resources/data/documents/reports/instances/independent-research-report-the-forrester-wave-tm-requirements-management-q2-2008-31`. Version: 2008 (Forrester Wave). – (Stand: 08.04.2011)

[Sil12] SILVER, Nate: *The Signal and the Noise: Why Most Predictions Fail but Some Don't*. New York, USA : Penguin Press, 2012

[SL10] SOONSONGTANEE, Surachet ; LIMPIYAKORN, Yachai: Enhancement of requirements traceability with state diagrams. In: *Proceedings of the 2nd International Conference on Computer Engineering and Technology 2010 (ICCET 2010)* Bd. 2, IEEE, 2010, S. 248–252

[SPCG+05] SIRIN, Evren ; PARSIA, Bijan ; CUENCA GRAU, Bernardo ; KALYANPUR, Aditya ; KATZ, Yarden: Pellet: A practical OWL-DL reasoner / University of Maryland Institute for Advanced Computer Studies (UMIACS). 2005 (2005-68). – Forschungsbericht

[SRG11] SCHRAPS, Mathias ; RONNEBERGER, Torsten ; GOLUBSKI, Wolfgang: Automotive Engineering im Enterprise-Domain-Network. In: *Informatik 2011: Informatik schafft Communities, Beiträge der 41. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 4.-7.10.2011, Berlin* Bd. 192, GI, 2011 (Lecture Notes in Informatics (LNI)), S. 211

[Sta] STANFORD CENTER FOR BIOMEDICAL INFORMATICS RESEARCH: *The Protégé Ontology Editor* `http://protege.stanford.edu/`. – (Stand: 01.12.2012)

[Sys06] SYSKA, Andreas: *Produktionsmanagement*. Wiesbaden : Gabler, 2006

[Tie03] TIETZE, Oliver: *Strategische Positionierung in der Automobilbranche*. Wiesbaden : Deutscher Universitäts-Verlag, 2003

[TK08] THIESING, Frank ; KORTEMEYER, Sebastian: Entwicklung moderner Web-Anwendungen mit Open-Source-Bausteinen. In: *Informatik-Spektrum* 31 (2008), Nr. 2, S. 115–132

[VWW] *Volkswagen Webseite, Übersicht Marken und Produkte* `http://www.volkswagenag.com/content/vwcorp/content/de/brands_and_products.html`. – (Stand: 24.02.2013)

[W3C] W3C OWL WORKING GROUP: *OWL 2* `http://www.w3.org/TR/owl2-overview/`. – (Stand: 21.07.2011)

[WD02] WÖHE, Günter ; DÖRING, Ulrich: *Einführung in die Allgemeine Betriebswirtschaftslehre.* München : Vahlen, 2002

[Wes07] WESKE, Mathias: *Business Process Management: Concepts, Languages, Architectures*. Berlin : Springer, 2007

[WFO09] WALLENTOWITZ, Henning ; FREIALDENHOVEN, Arndt ; OLSCHEWSKI, Ingo: *Strategien in der Automobilindustrie: Technologietrends und Marktentwicklungen (ATZ/MTZ-Fachbuch)*. Wiesbaden : Vieweg+Teubner, 2009

[Wie05] WIERINGA, Roel: Requirements researchers: are we really doing research? In: *Requirements Engineering* 10 (2005), S. 304–306

[Wil90] WILDEMANN, Horst: Kostengünstiges Variantenmanagement. In: *io Management Zeitschrift* 59 (1990), Nr. 11, S. 37–41

[Wir08] WIRTZ, Bernd W.: *Multi-Channel-Marketing: Grundlagen - Instrumente - Prozesse*. Wiesbaden : Gabler, 2008

[WK05] WOITSCH, R. ; KARAGIANNIS, D.: Process oriented knowledge management: a service based approach. In: *Journal of Universal Computer Science* 11 (2005), Nr. 4, S. 565–588

[WW02] WEBER, M. ; WEISBROD, J.: Requirements Engineering in Automotive Development - Experiences and Challenges. In: *Proceedings of the IEEE Joint International Conference on Requirements Engineering 2002 (RE 2002)*. Essen, Germany : IEEE, 2002, S. 331–340

[WXZY07] WANG, Chong ; XIONG, Miao ; ZHOU, Qi ; YU, Yong: PANTO: A Portable Natural Language Interface to Ontologies. In: *The Semantic Web: Research and Applications* Bd. 4519. Berlin : Springer, 2007, S. 473–487

[You03] YOUNG, Ralph R.: *The Requirements Engineering Handbook*. Norwood, MA, USA : Artech House, 2003

Anhang A

Glossar und Abkürzungsverzeichnis

Glossar

Anforderung

Nach [HWFP08] ist eine Anforderung eine Aussage, die eine Fähigkeit, physische Charakteristik oder einen Qualitätsfaktor beschreibt und damit eine Produkt- oder Prozess-Notwendigkeit identifiziert, für die eine Lösung erarbeitet wird.

Anforderungsmanagement

Anforderungsmanagement ist nach [HWFP08] die Summe aller Schnittstellen zwischen Anforderungserhebung und allen weiteren Teilaufgaben des Systems Engineerings zur effizienten und risikokontrollierten Entwicklung komplexer Systeme.

Artefakt

Ein Artefakt repräsentiert ein virtuelles oder physisches Geschäftsobjekt innerhalb eines Unternehmens mit definierten Eigenschaften, welches im allgemeinen Sprachgebrauch Verwendung findet und eindeutig abgrenzbar ist.

Extensible Markup Language

XML ist eine Beschreibungssprache für strukturierte Daten. Sie legt Regeln für den Aufbau von hierarchisch strukturierten Dokumenten fest und spielt so eine große Rolle beim Austausch elektronischer Daten.

Hypertext Markup Language

HTML ist eine Beschreibungssprache für digitale Dokumente, deren Inhalte durch einen Webbrowser interpretiert und dargestellt werden.

Klasse

Eine Klasse im Sinne des Software Engineerings ist ein abstraktes Konzept, welches die Eigenschaften und Funktionen der von ihm instantiierten Objekte beschreibt.

Modularer Fahrzeugbaukasten

Ein modularer Fahrzeugbaukasten ist eine Sammlung von Modulen und Teilen, die bestimmten im Vorfeld definierten Kriterien genügen.

Notation3

Eine Serialisierung von RDF-Modellen, die nicht auf XML beruht und damit einfacher durch Menschen lesbar ist.

Objekt

Ein Objekt stellt eine konkrete Instanz einer Klasse mit ausgeprägten Werten für Eigenschaften dar.

Ontologie

Nach [Gru93, SBF98] ist eine Ontologie eine explizite und formale Spezifikation einer Konzeptionalisierung.

Ontology Web Language

Eine logikbasierte Beschreibungssprache für Relationen und Klassen welche RDFS um Möglichkeiten bspw. zur Angabe von Kardinalitäten oder Charakteristika von Relationen erweitert.

Resource Description Framework

Ein standardisiertes konkretes Modell für die Darstellung von Tripeln bestehend aus zwei Objekten und des sie verbindenden, gerichteten Relationsobjekts. Erzeugt einen gerichteten Graphen mit beschrifteten Kanten.

Resource Description Framework Schema

Eine abstrakte Beschreibungssprache zur Definition von Klassen und Relationen sowie deren Hierarchien durch Spezialisierung und Generalisierung. RDF-Modelle verwenden RDFS als Schema.

SPARQL Protocol And RDF Query Language

Eine Abfragesprache für RDF-Graphen.

Standard Query Language

Eine Abfragesprache für Datenbanken.

Traceability

Traceability ist die Funktion eines Systems, Zusammenhänge von Prozessobjekten darstellen und verwalten zu können sowie Methoden zum Operieren hiermit zur Verfügung zu stellen.

Wissensbasis

Eine Wissensbasis oder Knowledge Base ist basierend auf der Definition von [Mil07] ein System, welches relevante, strukturierte und miteinander verknüpfte Informationen einer oder mehrerer Domänen enthält und diese Nutzern und anderen Systemen zum Betrachten und Bearbeiten zur Verfügung stellt.

Abkürzungen

EBNF	Erweiterte Backus-Naur-Form
GWT	Google Web Toolkit
HTML	Hypertext Markup Language
IEEE	Institute of Electrical and Electronics Engineers
N3	Notation3
OEM	Original Equipment Manufacturer
OWL	Ontology Web Language
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
ReqIF	Requirements Interchange Format
SPARQL	SPARQL Protocol And RDF Query Language
SQL	Standard Query Language
SWORE	SoftWiki Ontology for Requirements Engineering
UML	Unified Modeling Language
XML	Extensible Markup Language

Anhang B

Ontologien

Listing B.1: http://timguelke.de/core.owl

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix core: <http://timguelke.de/core.owl#> .

<http://timguelke.de/core.owl> rdf:type owl:Ontology .

################################################################
#
#    Object Properties
#
################################################################

core:besteht_aus
    rdf:type owl:ObjectProperty ;
    rdfs:label "besteht aus"@de ;
    rdfs:comment "Zur Abbildung von Strukturen." ;
    rdfs:range owl:Thing ;
    rdfs:domain owl:Thing .


core:erzeugt
    rdf:type owl:ObjectProperty ;
    rdfs:label "erzeugt"@de ;
    rdfs:comment "Ein Prozess kann ein Produkt erzeugen." ;
    rdfs:range core:Produkt ;
    rdfs:domain core:Prozess .


core:folgt_auf
    rdf:type owl:ObjectProperty ;
```

```
33      rdfs:label "folgt auf"@de ;
34      rdfs:comment "Mithilfe der einzelnen Prozessschritte
           lassen sich bspw. MPM-Netzpläne erstellen." ;
35      rdfs:domain core:Prozessschritt ;
36      rdfs:range core:Prozessschritt .
37
38
39  core:hat_Ergebnis
40      rdf:type owl:ObjectProperty ;
41      rdfs:label "hat Ergebnis"@de ;
42      rdfs:comment "Ein Prozessschritt kann ein beliebiges
           Ergebnis haben." ;
43      rdfs:domain core:Prozessschritt ;
44      rdfs:range owl:Thing .
45
46
47  core:hat_Prozessschritt
48      rdf:type owl:ObjectProperty ;
49      rdfs:label "hat Prozessschritt"@de ;
50      rdfs:comment "Ein Prozess besteht aus einzelnen
           Prozessschritten" ;
51      rdfs:domain core:Prozess ;
52      rdfs:range core:Prozessschritt ;
53      rdfs:subPropertyOf core:besteht_aus .
54
55
56  core:strukturiert
57      rdf:type owl:ObjectProperty ;
58      rdfs:label "strukturiert"@de ;
59      rdfs:comment "Eine Struktur strukturiert ein Dokument.
           Ändert sich eine Struktur, müssen auch zugehörige
           Dokumente neu strukturiert werden." ;
60      rdfs:range core:Dokument ;
61      rdfs:domain core:Struktur ;
62      rdfs:subPropertyOf core:wirkt_auf .
63
64
65  core:unterstuetzt
66      rdf:type owl:ObjectProperty ;
67      rdfs:label "unterstützt"@de ;
68      rdfs:comment "Ein System unterstützt einen bestimmten
           Prozess." ;
69      rdfs:range core:Prozess ;
```

```
70      rdfs:domain core:System .
71
72
73  core:wird_erzeugt_durch
74      rdf:type owl:ObjectProperty ;
75      rdfs:label "wird erzeugt durch"@de ;
76      rdfs:comment "Ein Produkt wird durch einen Prozess
            erzeugt. Ändert sich das Produkt, muss sich auch der
            Prozess ändern." ;
77      rdfs:domain core:Produkt ;
78      rdfs:range core:Prozess ;
79      rdfs:subPropertyOf core:wirkt_auf .
80
81
82  core:wird_strukturiert_durch
83      rdf:type owl:ObjectProperty ;
84      rdfs:label "wird strukturiert durch"@de ;
85      rdfs:comment "Ein Dokument wird durch eine Struktur
            strukturiert." ;
86      rdfs:domain core:Dokument ;
87      rdfs:range core:Struktur .
88
89
90  core:wird_unterstuetzt_durch
91      rdf:type owl:ObjectProperty ;
92      rdfs:label "wird unterstützt durch"@de ;
93      rdfs:comment "Ein Prozess wird durch ein System
            unterstützt. Eine Änderung am Prozess zieht auch eine
            Änderung am System nach sich." ;
94      rdfs:domain core:Prozess ;
95      rdfs:range core:System ;
96      rdfs:subPropertyOf core:wirkt_auf .
97
98
99  core:wirkt_auf
100     rdf:type owl:ObjectProperty ;
101     rdfs:label "wirkt auf"@de ;
102     rdfs:comment "Zur Abbildung von Wirkbeziehungen." ;
103     rdfs:domain owl:Thing ;
104     rdfs:range owl:Thing .
105
106 ###############################################################
107 #
```

```
#    Data properties
#
#################################################################

core:hat_Dauer
    rdf:type owl:DatatypeProperty ;
    rdfs:label "hat_Dauer"@de ;
    rdfs:comment "Jeder Prozessschritt hat eine bestimmte
       Dauer." ;
    rdfs:domain core:Prozessschritt ;
    rdfs:range xsd:nonNegativeInteger .

#################################################################
#
#    Classes
#
#################################################################

core:Dokument
    rdf:type owl:Class ;
    rdfs:comment "Ein Dokument ist ein virtuelles Datenobjekt
        oder eine physische Repräsentation hiervon mit dem
       Zweck der Datenablage oder des Datentransports.
       Dokumente können eine Struktur haben. " .


core:Organisation
    rdf:type owl:Class ;
    rdfs:label "Organisation" ;
    rdfs:comment "Organisation beschreibt den formellen
       Aufbau aus Fachabteilungen und Hierarchien innerhalb
       eines Unternehmens." .


core:Produkt
    rdf:type owl:Class ;
    rdfs:label "Produkt" ;
    rdfs:comment "Das Produkt wird als Leistung in Form eines
        Gegenstands verstanden. Der Gegenstand kann
       materieller oder immaterieller Natur sein." .


core:Prozess
```

```
143     rdf:type owl:Class ;
144     rdfs:label "Prozess" ;
145     rdfs:comment "Ein Unternehmensprozess ist ein Bündel von
           Aktivitäten, für das ein oder mehrere unterschiedliche
            Inputs benötigt werden und das für den Kunden ein
           Ergebnis von Wert erzeugt." .
146
147
148 core:Prozessschritt
149     rdf:type owl:Class ;
150     rdfs:label "Prozessschritt" ;
151     rdfs:comment "Ein Prozessschritt wird im Kontext eines
           Prozesses durchgeführt." .
152
153
154 core:Struktur
155     rdf:type owl:Class ;
156     rdfs:label "Struktur" ;
157     rdfs:comment "Eine Struktur ist eine formale Gliederung."
            .
158
159
160 core:System
161     rdf:type owl:Class ;
162     rdfs:label "System" ;
163     rdfs:comment "Ein informationstechnisches System ist eine
            Hardware/Software-Kombination zum Zwecke der
           Datenverarbeitung." .
```

Listing B.2: http://timguelke.de/rmnet.owl

```
1  @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
2  @prefix core: <http://timguelke.de/core.owl#> .
3  @prefix owl: <http://www.w3.org/2002/07/owl#> .
4  @prefix req: <http://ns.softwiki.de/req#> .
5  @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
6  @prefix rmnet: <http://timguelke.de/rmnet.owl#>.
7
8  <http://timguelke.de/rmnet.owl> rdf:type owl:Ontology ;
9                                   rdfs:comment "Die Grund-
                                       Begriffe, -Eigenschaften
                                       und -Relationen des AM-
                                       Netz." ;
10                                  owl:imports <http://timguelke
```

```
                                        .de/core.owl> .

################################################################
#
#    Object Properties
#
################################################################

rmnet:Anforderung_beschrieben_in
    rdf:type owl:ObjectProperty ;
    rdfs:label "beschrieben in" ;
    rdfs:comment "Eine Anforderung wird in einem Dokument
       beschrieben. Wird die Anforderung geaendert, aendert
       sich auch das Dokument." ;
    rdfs:domain req:Requirement ;
    rdfs:range core:Dokument ;
    rdfs:subPropertyOf core:wirkt_auf .


rmnet:Anforderung_definiert_fuer
    rdf:type owl:ObjectProperty ;
    rdfs:label "definiert fuer" ;
    rdfs:comment "Eine Anforderung ist definiert fuer ein
       Produkt." ;
    rdfs:domain req:Requirement ;
    rdfs:range core:Produkt .


rmnet:Umfang_beschrieben_in
    rdf:type owl:ObjectProperty ;
    rdfs:subPropertyOf core:wirkt_auf ;
    rdfs:label "beschrieben in" ;
    rdfs:comment "Ein Umfang wird in einem Dokument
       beschrieben. Wird der Umfang geaendert, aendert sich
       auch das Dokument." ;
    rdfs:range core:Dokument ;
    rdfs:domain rmnet:AbstractUmfang .


rmnet:beschreibt_Anforderung
    rdf:type owl:ObjectProperty ;
    rdfs:label "beschreibt Anforderung" ;
    rdfs:comment "Ein Dokument kann eine Anforderung
```

```
       beschreiben." ;
48     rdfs:range req:Requirement ;
49     rdfs:domain core:Dokument .
50
51
52 rmnet:beschreibt_Umfang
53     rdf:type owl:ObjectProperty ;
54     rdfs:label "beschreibt Umfang" ;
55     rdfs:comment "Ein Dokument kann einen Umfang beschreiben.
          " ;
56     rdfs:domain core:Dokument ;
57     rdfs:range rmnet:AbstractUmfang .
58
59
60 rmnet:erfuellt
61     rdf:type owl:ObjectProperty ;
62     rdfs:label "erfuellt" ;
63     rdfs:comment "Eine Produkteigenschaft erfuellt eine
          Anforderung." ;
64     rdfs:range req:Requirement ;
65     rdfs:domain rmnet:Produkteigenschaft .
66
67
68 rmnet:fuehrt_Test_aus
69     rdf:type owl:ObjectProperty ;
70     rdfs:label "fuehrt Test aus" ;
71     rdfs:comment "Eine Person kann einen Test ausfuehren." ;
72     rdfs:domain rmnet:Person ;
73     rdfs:range rmnet:Test .
74
75
76 rmnet:hat_Anforderung
77     rdf:type owl:ObjectProperty ;
78     rdfs:subPropertyOf core:wirkt_auf ;
79     rdfs:label "hat Anforderung" ;
80     rdfs:comment "Ein Produkt kann eine Anforderung haben.
          Eine Aenderung des Produkts fuehrt auch zu einer
          Aenderung der Anforderungen." ;
81     rdfs:range req:Requirement ;
82     rdfs:domain core:Produkt .
83
84
85 rmnet:hat_Eigenschaft
```

```
86      rdf:type owl:ObjectProperty ;
87      rdfs:label "hat Produkteigenschaft" ;
88      rdfs:comment "Ein Produkt hat eine Produkteigenschaft.
           Eine Aenderung des Produkts fuehrt auch zu einer
           Aenderung der Produkteigenschaft." ;
89      rdfs:domain core:Produkt ;
90      rdfs:subPropertyOf core:wirkt_auf ;
91      rdfs:range rmnet:Produkteigenschaft .
92
93
94  rmnet:hat_Mitglied
95      rdf:type owl:ObjectProperty ;
96      rdfs:label "hat Mitglied" ;
97      rdfs:comment "Organisationseinheiten bestehen aus
           Personen." ;
98      rdfs:domain rmnet:Organisationseinheit ;
99      rdfs:range rmnet:Person .
100
101
102 rmnet:hat_Strukturelement
103     rdf:type owl:ObjectProperty ;
104     rdfs:subPropertyOf
105         core:besteht_aus ,
106         core:wirkt_auf ;
107     rdfs:label "hat Strukturelement" ;
108     rdfs:comment "Ein Dokument besteht aus Strukturelementen.
            Strukturen verwenden besteht_aus! Eine Aenderung am
           Dokument kann auch zu Aenderungen an den
           Strukturelementen fuehren." ;
109     rdfs:domain core:Dokument ;
110     rdfs:range rmnet:Strukturelement .
111
112
113 rmnet:hat_Teil
114     rdf:type owl:ObjectProperty ;
115     rdfs:label "besteht aus" ;
116     rdfs:comment "Ein Modul besteht aus Teilen. Aenderungen
           an den Teilen aendern das Modul." ;
117     rdfs:subPropertyOf core:besteht_aus ;
118     rdfs:domain rmnet:Modul ;
119     rdfs:range rmnet:Teil .
120
121
```

```
122 rmnet:ist_Mitglied_einer
123     rdf:type owl:ObjectProperty ;
124     rdfs:label "ist Mitglied einer" ;
125     rdfs:comment "Personen sind Mitglied einer
           Organisationseinheit." ;
126     rdfs:range rmnet:Organisationseinheit ;
127     rdfs:domain rmnet:Person .
128
129
130 rmnet:liegt_in
131     rdf:type owl:ObjectProperty ;
132     rdfs:label "liegt in" ;
133     rdfs:comment "Organisationseinheiten liegen in einer
           Organisationsebene" ;
134     rdfs:range rmnet:Organisationsebene ;
135     rdfs:domain rmnet:Organisationseinheit .
136
137
138 rmnet:realisiert
139     rdf:type owl:ObjectProperty ;
140     rdfs:label "realisiert" ;
141     rdfs:comment "Ein abstrakter Umfang realisiert eine
           Produkteigenschaft." ;
142     rdfs:domain rmnet:AbstractUmfang ;
143     rdfs:range rmnet:Produkteigenschaft .
144
145
146 rmnet:realisiert_durch
147     rdf:type owl:ObjectProperty ;
148     rdfs:label "realisiert durch" ;
149     rdfs:comment "Eine Eigenschaft wird durch einen
           abstrakten Umfang realisiert. Eine Aenderung der
           Produkteigenschaft zieht eine Aenderung des Umfangs
           nach sich." ;
150     rdfs:subPropertyOf core:wirkt_auf ;
151     rdfs:range rmnet:AbstractUmfang ;
152     rdfs:domain rmnet:Produkteigenschaft .
153
154
155 rmnet:wird_ausgefuehrt_durch
156     rdf:type owl:ObjectProperty ;
157     rdfs:label "wird ausgefuehrt durch" ;
158     rdfs:comment "Ein Test wird durch eine Person ausgefuehrt
```

```
       ." ;
159    rdfs:range rmnet:Person ;
160    rdfs:domain rmnet:Test .
161
162
163 rmnet:wird_erfuellt_durch
164    rdf:type owl:ObjectProperty ;
165    rdfs:label "wird erfuellt durch" ;
166    rdfs:comment "Eine Anforderung wird durch eine
          Produkteigenschaft erfuellt. Eine Aenderung der
          Anforderung fuehrt zu einer Aenderung der
          Produkteigenschaft." ;
167    rdfs:domain req:Requirement ;
168    rdfs:subPropertyOf core:wirkt_auf ;
169    rdfs:range rmnet:Produkteigenschaft .
170
171
172 rmnet:wird_getestet_durch
173    rdf:type owl:ObjectProperty ;
174    rdfs:label "wird getestet durch" ;
175    rdfs:comment "Eine Anforderung wird durch einen Test
          getestet. Eine Aenderung der Anforderung fuehrt zu
          einer Aenderung/Neuausfuehrung des Tests." ;
176    rdfs:domain req:Requirement ;
177    rdfs:range rmnet:Test .
178
179
180 ###############################################################
181 #
182 #    Classes
183 #
184 ###############################################################
185
186 req:Stakeholder
187    rdfs:subClassOf rmnet:Person .
188
189
190 rmnet:AbstractUmfang
191    rdf:type owl:Class ;
192    rdfs:comment "Eine abstrakter Umfang kann ein Modul, ein
          Teil oder Software sein." .
193
194
```

```
rmnet:Modul
    rdf:type owl:Class ;
    rdfs:subClassOf rmnet:AbstractUmfang .


rmnet:Organisationsebene
    rdf:type owl:Class .


rmnet:Organisationseinheit
    rdf:type owl:Class .


rmnet:Person
    rdf:type owl:Class .


rmnet:Produkteigenschaft
    rdf:type owl:Class .


rmnet:Software
    rdf:type owl:Class ;
    rdfs:subClassOf rmnet:AbstractUmfang .


rmnet:Strukturelement
    rdf:type owl:Class .


rmnet:Teil
    rdf:type owl:Class ;
    rdfs:subClassOf rmnet:AbstractUmfang .


rmnet:Test
    rdf:type owl:Class .
```

Listing B.3: http://timguelke.de/produkt-artefakte.owl

```
@prefix rmnet: <http://timguelke.de/rmnet.owl#> .
@prefix art: <http://timguelke.de/produkt-artefakte.owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix core: <http://timguelke.de/core.owl#> .
```

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix req: <http://ns.softwiki.de/req#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

<http://timguelke.de/produkt-artefakte.owl>
    rdf:type owl:Ontology ;
    rdfs:label "Generische Entwicklungsprojekt Artefakt-
       Ontologie" ;
    rdfs:comment "Beschreibt eine generische
       Artefaktlandschaft fuer Entwicklungsprojekte" ;
    owl:imports
        <http://timguelke.de/core.owl> ,
        <http://timguelke.de/rmnet.owl> .


#################################################################
#
#    Object Properties
#
#################################################################

art:Modul_enthalten_in
    rdf:type owl:ObjectProperty ;
    rdfs:label "hat Modul"@de ;
    rdfs:comment "Ein modularer Baukasten besteht aus Modulen
       ." ;
    rdfs:subPropertyOf core:wirkt_auf ;
    rdfs:range art:ModularerBaukasten ;
    rdfs:domain rmnet:Modul .


art:Teil_von_Anforderungskatalog
    rdf:type owl:ObjectProperty ;
    rdfs:label "ist Teil von Anforderungtskatalog"@de ;
    rdfs:comment "Eine Anforderung ist immer Teil eines
       bestimmten Anforderungskatalogs. Eine Aenderung einer
       Anforderung zieht demnach eine Aenderung der
       uebergeordneten Entitaet nach sich, ohne jedoch andere
        Anforderungen zu beeinflussen." ;
    rdfs:domain req:Requirement ;
    rdfs:subPropertyOf core:wirkt_auf ;
```

```
41      rdfs:range art:Anforderungskatalog .
42
43
44  art:Teil_von_Eigenschaftskatalog
45      rdf:type owl:ObjectProperty ;
46      rdfs:label "ist Teil von Eigenschaftskatalog"@de ;
47      rdfs:comment "Eine Produkteigenschaft ist immer Teil
           eines bestimmten Eigenschaftskatalogs. Eine Aenderung
           einer Eigenschaft zieht demnach eine Aenderung der
           uebergeordneten Entitaet nach sich, ohne jedoch andere
            Produkteigenschaften zu beeinflussen." ;
                                rdfs:subPropertyOf
           core:wirkt_auf ;
48      rdfs:range art:Eigenschaftskatalog ;
49      rdfs:domain rmnet:Produkteigenschaft .
50
51
52  art:Umfang_enthalten_in
53      rdf:type owl:ObjectProperty ;
54      rdfs:label "hat Umfang"@de ;
55      rdfs:comment "Die Technikliste enthaelt abstrakte
           Umfaenge und deren Detaillierungen. Eine Aenderung
           einer Eigenschaft zieht demnach eine Aenderung der
           uebergeordneten Entitaet nach sich, ohne jedoch andere
            Produkteigenschaften zu beeinflussen." ;
56      rdfs:subPropertyOf core:wirkt_auf ;
57      rdfs:range art:Technikliste ;
58      rdfs:domain rmnet:AbstractUmfang .
59
60
61  art:bedient_Wert
62      rdf:type owl:ObjectProperty ;
63      rdfs:label "bedient Wert"@de ;
64      rdfs:comment "Produkteigenschaften bedienen gezielt
           bestimmte Werte und somit indirekt Milieus." ;
65      rdfs:range art:Wert ;
66      rdfs:domain rmnet:Produkteigenschaft .
67
68
69  art:enthaelt_Anforderung
70      rdf:type owl:ObjectProperty ;
71      rdfs:label "enthaelt Anforderung"@de ;
72      rdfs:comment "Der Anforderungskatalog gruppiert eine
```

```
       bestimmte Menge von Anforderungen." ;
73     rdfs:range req:Requirement ;
74     rdfs:subPropertyOf core:besteht_aus ;
75     rdfs:domain art:Anforderungskatalog .
76
77
78 art:enthaelt_Modul
79     rdf:type owl:ObjectProperty ;
80     rdfs:label "hat Modul"@de ;
81     rdfs:comment "Ein modularer Baukasten besteht aus Modulen
          ." ;
82     rdfs:subPropertyOf core:besteht_aus ;
83     rdfs:domain art:ModularerBaukasten ;
84     rdfs:range rmnet:Modul .
85
86
87 art:enthaelt_Produkteigenschaft
88     rdf:type owl:ObjectProperty ;
89     rdfs:label "enthaelt Produkteigenschaft"@de ;
90     rdfs:comment "Ein Eigenschaftskatalog gruppiert eine
          bestimmte Menge von Eigenschaften." ;
91     rdfs:subPropertyOf core:besteht_aus ;
92     rdfs:domain art:Eigenschaftskatalog ;
93     rdfs:range rmnet:Produkteigenschaft .
94
95
96 art:enthaelt_Umfang
97     rdf:type owl:ObjectProperty ;
98     rdfs:label "hat Umfang"@de ;
99     rdfs:comment "Die Technikliste enthaelt abstrakte
          Umfaenge und deren Detaillierungen." ;
100    rdfs:subPropertyOf core:besteht_aus ;
101    rdfs:domain art:Technikliste ;
102    rdfs:range rmnet:AbstractUmfang .
103
104
105 art:hat_Beschreibung
106    rdf:type owl:ObjectProperty ;
107    rdfs:label "hat Beschreibung"@de ;
108    rdfs:comment "Ein Produktkonzept enthaelt als einen
          strukturellen Baustein eine Beschreibung. Aendert sich
           das Produktkonzept, ist auch die Beschreibung
          betroffen." ;
```

```
109     rdfs:range art:Beschreibung ;
110     rdfs:domain art:Produktkonzept ;
111     rdfs:subPropertyOf rmnet:hat_Strukturelement .
112
113
114 art:hat_Fahrzeugklasse
115     rdf:type owl:ObjectProperty ;
116     rdfs:label "hat Fahrzeugklasse"@de ;
117     rdfs:comment "Ein Produktkonzept enthaelt als einen
            strukturellen Baustein Informationen ueber die
            Fahrzeugklasse. Aendert sich das Produktkonzept, ist
            eventuell auch die Fahrzeugklasse betroffen." ;
118     rdfs:range art:Fahrzeugklasse ;
119     rdfs:domain art:Produktkonzept ;
120     rdfs:subPropertyOf rmnet:hat_Strukturelement .
121
122
123 art:hat_KonzeptBestimmendeMasse
124     rdf:type owl:ObjectProperty ;
125     rdfs:label "hat Konzept-bestimmende Masse"@de ;
126     rdfs:comment "Ein Produktkonzept enthaelt als einen
            strukturellen Baustein Informationen ueber die das
            Konzept bestimmenden Masse. Aendert sich das
            Produktkonzept, sind eventuell auch diese betroffen."
            ;
127     rdfs:range art:KonzeptBestimmendeMasse ;
128     rdfs:domain art:Produktkonzept ;
129     rdfs:subPropertyOf rmnet:hat_Strukturelement .
130
131
132 art:hat_Produktkonzept
133     rdf:type owl:ObjectProperty ;
134     rdfs:label "hat Produktkonzept"@de ;
135     rdfs:comment "Ein Produkt hat ein Produktkonzept." ;
136     rdfs:domain core:Produkt ;
137     rdfs:range art:Produktkonzept .
138
139
140 art:hat_Wert
141     rdf:type owl:ObjectProperty ;
142     rdfs:label "hat Wert"@de ;
143     rdfs:comment "Einem Milieu sind ganz bestimmte Werte
            zugeordnet, die dieses Milieu definieren." ;
```

```
144     rdfs:domain art:Milieu ;
145     rdfs:range art:Wert .
146
147
148 art:wird_bedient_durch
149     rdf:type owl:ObjectProperty ;
150     rdfs:label "wird bedient durch Produkteigenschaft"@de ;
151     rdfs:comment "Ein Milieu wird durch bestimmte
            Produkteigenschaften gezielt bedient." ;
152     rdfs:subPropertyOf core:wirkt_auf ;
153     rdfs:domain art:Wert ;
154     rdfs:range rmnet:Produkteigenschaft .
155
156 art:hat_Produkt
157     rdf:type owl:ObjectProperty ;
158     rdfs:label "hat Produkt"@de ;
159     rdfs:domain art:Entwicklungsprojekt ;
160     rdfs:range core:Produkt .
161
162
163 ###################################################################
164 #
165 #    Data properties
166 #
167 ###################################################################
168
169
170 art:hat_Zieltermin
171     rdf:type owl:DatatypeProperty ;
172     rdfs:label "hat Zieltermin am"@de ;
173     rdfs:comment "Das Produkt besitzt einen Zieltermin an dem
            die Entwicklung abgeschlossen sein soll." ;
174     rdfs:domain art:Entwicklungsprojekt ;
175     rdfs:range xsd:nonNegativeInteger .
176
177
178 ###################################################################
179 #
180 #    Classes
181 #
182 ###################################################################
183
184
```

```
art:Anforderungskatalog
    rdf:type owl:Class ;
    rdfs:subClassOf core:Dokument ;
    rdfs:comment "Der Anforderungskatalog stellt die Sammlung
        aller Anforderungen dar." .


art:Beschreibung
    rdf:type owl:Class ;
    rdfs:subClassOf rmnet:Strukturelement .


art:Eigenschaftskatalog
    rdf:type owl:Class ;
    rdfs:subClassOf core:Dokument ;
    rdfs:comment "Der Eigenschaftskatalog stellt die Sammlung
        aller Eigenschaften eines Produkts dar." .


art:Eigenschaftsstruktur
    rdf:type owl:Class ;
    rdfs:subClassOf core:Struktur ;
    rdfs:comment "Die Eigenschaftsstruktur strukturiert den
       Eigenschaftskatalog" .


art:Entwicklungsprojekt
    rdf:type owl:Class .


art:Fahrzeugklasse
    rdf:type owl:Class ;
    rdfs:subClassOf rmnet:Strukturelement .


art:KonzeptBestimmendeMasse
    rdf:type owl:Class ;
    rdfs:subClassOf rmnet:Strukturelement .


art:Milieu
    rdf:type owl:Class ;
    rdfs:comment "Ein Milieu repreasentiert eine hinsichtlich
```

```
        gewisser Kriterien gleiche Menge von Personen" .


art:ModularerBaukasten
    rdf:type owl:Class ;
    rdfs:subClassOf core:Dokument ;
    rdfs:comment "Ein Baukasten ist eine Sammlung von Modulen
       , die alle bestimmten im Vorfeld definierten Kriterien
        genuegen." .


art:Produktkonzept
    rdf:type owl:Class ;
    rdfs:subClassOf core:Dokument .


art:Technikliste
    rdf:type owl:Class ;
    rdfs:subClassOf core:Dokument ;
    rdfs:comment "Die Technikliste stellt die Sammlung von
       Konzepten und Loesungen eines Produkts dar." .


art:Wert
    rdf:type owl:Class ;
    rdfs:comment "Ein Wert ist eine individuelle Vorstellung
       hinsichtlich gewisser Eigenschaften von Konzepten." .
```

Listing B.4: http://timguelke.de/fahrzeug-projekt.owl

```
@prefix rmnet: <http://timguelke.de/rmnet.owl#> .
@prefix art: <http://timguelke.de/produkt-artefakte.owl#> .
@prefix proj: <http://timguelke.de/fahrzeug-projekt.owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix core: <http://timguelke.de/core.owl#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix req: <http://ns.softwiki.de/req#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dcterms: <http://dublincore.org/documents/dcmi-terms
   /#> .

<http://timguelke.de/fahrzeug-projekt.owl>
    rdf:type owl:Ontology ;
    rdfs:label "Netz des Fahrzeugprojekts Golf" ;
```

```
14      rdfs:comment "Das Fahrzeugprojekt Golf. " ;
15      owl:imports <http://timguelke.de/produkt-artefakte.owl> .
16
17  ###################################################################
18  #
19  #    Individuals
20  #
21  ###################################################################
22
23  proj:AmbienteInnenraum
24      rdf:type
25          req:Requirement ,
26          owl:NamedIndividual ;
27      dcterms:title
28          "Ambiente des Innenraums" ;
29      dcterms:description
30          "Der Innenraum des Fahrzeugs soll über den
               Durchschnitt der Mittelklasse liegen." ;
31      rmnet:Anforderung_definiert_fuer proj:Golf .
32
33
34  proj:Anforderungserhebung
35      rdf:type
36          core:Prozessschritt ,
37          owl:NamedIndividual ;
38      core:hat_Ergebnis proj:AnforderungskatalogGolf .
39
40
41  proj:AnforderungskatalogGolf
42      rdf:type
43          art:Anforderungskatalog ,
44          owl:NamedIndividual ;
45      art:enthaelt_Anforderung
46          proj:Fahrverhalten ,
47          proj:Marktanteil ,
48          proj:Sicherheitssysteme .
49
50
51  proj:BuergerlicheMitte
52      rdf:type art:Milieu ,
53      owl:NamedIndividual ;
54      art:hat_Wert proj:Sicherheit .
55
```

```
56
57 proj:EigenschaftskatalogGolf
58     rdf:type
59         art:Eigenschaftskatalog ,
60         owl:NamedIndividual ;
61     art:enthaelt_Produkteigenschaft
62         proj:EinsatzNotbremsassistent ,
63         proj:EinsatzSpurhalteassistent ,
64         proj:VerbauLEDScheinwerfer ,
65         proj:VerbesserungBremsanlage .
66
67 proj:EinsatzNotbremsassistent
68     rdf:type
69         rmnet:Produkteigenschaft ,
70         owl:NamedIndividual ;
71     dcterms:title
72         "Einsatz Notbremsassistent im Fahrzeug" ;
73     dcterms:description
74         "Das Fahrzeug erhaelt als aktives Sicherheitssystem
              einen Notbremsassistenten." ;
75     art:Teil_von_Eigenschaftskatalog
          proj:EigenschaftskatalogGolf ;
76     art:bedient_Wert proj:Sicherheit ;
77     rmnet:erfuellt proj:Sicherheitssysteme .
78
79
80 proj:EinsatzSpurhalteassistent
81     rdf:type
82         rmnet:Produkteigenschaft ,
83         owl:NamedIndividual ;
84     dcterms:title
85         "Einsatz Spurhalteassistent im Fahrzeug" ;
86     dcterms:description
87         "Das Fahrzeug erhaelt als aktives Sicherheitssystem
              einen Spurhalteassistenten." ;
88     art:Teil_von_Eigenschaftskatalog
          proj:EigenschaftskatalogGolf ;
89     art:bedient_Wert proj:Sicherheit ;
90     rmnet:erfuellt proj:Sicherheitssysteme ;
91     rmnet:realisiert_durch proj:Spurhalteassistent .
92
93
94 proj:EntwicklungGolf
```

```
95      rdf:type
96          art:Entwicklungsprojekt ,
97          owl:NamedIndividual ;
98      art:hat_Produkt proj:Golf .
99
100
101 proj:ErstellungProduktkonzept
102     rdf:type
103         core:Prozessschritt ,
104         owl:NamedIndividual ;
105     core:hat_Ergebnis proj:KonzeptGolf .
106
107
108 proj:Exklusivitaet
109     rdf:type
110         art:Wert ,
111         owl:NamedIndividual ;
112     art:wird_bedient_durch proj:VerbauLEDScheinwerfer .
113
114
115 proj:Fahrverhalten
116     rdf:type
117         req:Requirement ,
118         owl:NamedIndividual ;
119     dcterms:title
120         "Durchschnittliches_Fahrverhalten" ;
121     dcterms:description
122         "Das Fahrzeug soll ein für seine Fahrzeugklasse
               durchschnittliches Fahrverhalten bieten." ;
123     rmnet:Anforderung_definiert_fuer proj:Golf ;
124     rmnet:wird_getestet_durch proj:TestFahrverhalten ;
125     rmnet:wird_erfuellt_durch proj:VerbesserungBremsanlage .
126
127
128 proj:Golf
129     rdf:type
130         core:Produkt ,
131         owl:NamedIndividual ;
132     rmnet:hat_Anforderung
133         proj:Fahrverhalten ,
134         proj:Marktanteil ,
135         proj:Sicherheitssysteme ;
136     art:hat_Produktkonzept proj:KonzeptGolf ;
```

```
137     rmnet:hat_Anforderung  ;
138     core:wird_erzeugt_durch proj:Produktentstehungsprozess .
139
140
141 proj:KonzeptGolf
142     rdf:type
143         art:Produktkonzept ,
144         owl:NamedIndividual .
145
146
147 proj:LEDScheinwerfer
148     rdf:type
149         rmnet:AbstractUmfang ,
150         owl:NamedIndividual ;
151     art:Umfang_enthalten_in proj:TechniklisteGolf ;
152     rmnet:realisiert proj:VerbauLEDScheinwerfer .
153
154
155 proj:Marktanteil
156     rdf:type
157         req:Requirement ,
158         owl:NamedIndividual ;
159     dcterms:title
160         "Gesteigerter Marktanteil" ;
161     dcterms:description
162         "Das Fahrzeug soll einen um 10% hoeheren Marktanteil
               im Vergleich zum Vorgaenger erreichen." ;
163     rmnet:Anforderung_definiert_fuer proj:Golf ;
164     rmnet:wird_erfuellt_durch proj:VerbauLEDScheinwerfer .
165
166
167 proj:ModernePerformer
168     rdf:type
169         art:Milieu ,
170         owl:NamedIndividual ;
171     art:hat_Wert proj:Spass .
172
173
174 proj:ModularerAssistentenBaukasten
175     rdf:type
176         art:ModularerBaukasten ,
177         owl:NamedIndividual ;
178     art:enthaelt_Modul proj:Notbremsassistent ;
```

```
179     art:enthaelt_Umfang proj:Spurhalteassistent .
180
181
182 proj:Notbremsassistent
183     rdf:type
184         rmnet:Modul ,
185         owl:NamedIndividual ;
186     art:Modul_enthalten_in proj:ModularerAssistentenBaukasten
            ;
187     art:Umfang_enthalten_in proj:TechniklisteGolf .
188
189
190 proj:Produktentstehungsprozess
191     rdf:type
192         core:Prozess ,
193         owl:NamedIndividual ;
194     core:erzeugt proj:Golf ;
195     core:hat_Prozessschritt
196         proj:ErstellungProduktkonzept ,
197         proj:Anforderungserhebung  ,
198         proj:TechnischeRealisierung ,
199         proj:Qualitaetssicherung .
200
201 proj:Qualitaetssicherung
202     rdf:type
203         core:Prozessschritt ,
204         owl:NamedIndividual .
205
206
207 proj:Sicherheit
208     rdf:type
209         art:Wert ,
210         owl:NamedIndividual ;
211     art:wird_bedient_durch proj:EinsatzNotbremsassistent ,
212     proj:EinsatzSpurhalteassistent .
213
214
215 proj:Sicherheitssysteme
216     rdf:type
217         req:Requirement ,
218         owl:NamedIndividual ;
219     dcterms:title
220         "Einsatz aktiver Sicherheitssysteme" ;
```

```
221     dcterms:description
222         "Das Fahrzeug soll durch den Einsatz aktiver
               Sicherheitssysteme einen Sicherheitsstandard
               bieten, der einem Mittelklassewagen angemessen ist
               ." ;
223     rmnet:wird_erfuellt_durch
224         proj:EinsatzNotbremsassistent ,
225         proj:EinsatzSpurhalteassistent ;
226     rmnet:Anforderung_definiert_fuer proj:Golf .
227
228
229 proj:Spass
230     rdf:type
231         art:Wert ,
232         owl:NamedIndividual ;
233     art:wird_bedient_durch proj:VerbesserungBremsanlage .
234
235
236 proj:Spurhalteassistent
237     rdf:type
238         rmnet:Modul ,
239         owl:NamedIndividual ;
240     rmnet:realisiert proj:EinsatzSpurhalteassistent ;
241     art:Modul_enthalten_in proj:ModularerAssistentenBaukasten
            ;
242     art:Umfang_enthalten_in proj:TechniklisteGolf .
243
244
245 proj:TechniklisteGolf
246     rdf:type
247         art:Technikliste ,
248         owl:NamedIndividual ;
249     art:enthaelt_Umfang
250         proj:LEDScheinwerfer ,
251         proj:Notbremsassistent ,
252         proj:Spurhalteassistent .
253
254
255 proj:TechnischeRealisierung
256     rdf:type
257         core:Prozessschritt ,
258         owl:NamedIndividual ;
259     core:hat_Ergebnis proj:TechniklisteGolf .
```

```
260
261
262 proj:TestFahrverhalten
263     rdf:type
264         rmnet:Test ,
265         owl:NamedIndividual .
266
267
268 proj:VerbauLEDScheinwerfer
269     rdf:type
270         rmnet:Produkteigenschaft ,
271         owl:NamedIndividual ;
272     dcterms:title
273         "Einsatz Voll-LED-Scheinwerfer" ;
274     dcterms:description
275         "Das Fahrzeug erhält Voll-LED-Scheinwerfer zur
               Erreichung erweiterter Zielgruppe." ;
276     art:Teil_von_Eigenschaftskatalog
           proj:EigenschaftskatalogGolf ;
277     art:bedient_Wert proj:Exklusivitaet ;
278     rmnet:realisiert_durch proj:LEDScheinwerfer ;
279     rmnet:erfuellt proj:Marktanteil .
280
281
282 proj:VerbesserungBremsanlage
283     rdf:type
284         rmnet:Produkteigenschaft ,
285         owl:NamedIndividual ;
286     dcterms:title
287         "Vorgaenger-Fahrwerk und Verbesserung der Bremsanlage
               " ;
288     dcterms:description
289         "Das Fahrzeug erhaelt Fahrwerk des Vorgängers mit
               einem um 5% reduzierten Bremsweg durch
               Neuentwicklung der Bremsanlage." ;
290     art:Teil_von_Eigenschaftskatalog
           proj:EigenschaftskatalogGolf ;
291     rmnet:erfuellt proj:Fahrverhalten ;
292     art:bedient_Wert proj:Spass .
```

Anhang C

Daten Werkzeug-Scouting

Produkt	**Firma**
Accept Requirements (Accept 360)	Accept Software
Acclaro DFSS	Axiomatic Design Solutions, Inc.
Aligned Elements	Aligned AG
Avenqo PEP	Avenqo
Blueprint Requirements Center 2010	Blueprint Software Systems, Inc.
CaliberRM	MicroFocus
Cameo Requirements+	No Magic Inc.
CASE Spec	Goda Software
Cognition Cockpit (Cockpit)	Cognition Corporation
consentor	consentor GmbH
Contour	Jama Software
CORE	Vitech Corporation
Cradle	3SL, Inc.
Dimensions RM (DimRM)	Serena Software
Enterprise Architect	Sparx Systems
Envision VIP	Future Tech Systems, Inc.
HP Requirements Management Modul	HP
IBM Rational DOORS	IBM
IBM Rational Requirements Composer	IBM
inteGREAT	eDev Technologies
IRQA	Visure Solutions
Kovair ALM Studio	Kovair Software, Inc.
MagicDraw and SysML Plugin	No Magic Inc.
MKS Integrity	MKS Inc.
PACE	Viewset Corporation
Polarion Requirements	Polarion Software
Project & Test Engineering System (PTESY)	Andromeda s.r.l.
Psoda	e-LM.com Limited
Rally	Rally Software Development
RaQuest	SparxSystems Japan Co., Ltd
ReqMan	RequirementOne Inc.
Reqtify	Geensoft
Requirements Manager (ReMa)	Accord Software and Systems Pvt. Ltd.
RRM - Requirement and Risk Management	Business Operation Systems GmbH
RTIME	QAVantage
Teamcenter Requirements (Tc RM)	Siemens PLM Software
TraceCloud	TraceCloud
workspace.com Requirements Management	workspace.com
ProR	Universität Düsseldorf

Tabelle C.1: Grundmenge Werkzeuge Requirements Management

Anhang D

Fragebogen zum Werkzeug-Scouting

- Funktionale Anforderungen
  - Anwender
    - Erstellung und Strukturierung der Anforderungen
      - Wie werden Anforderungen erstellt und strukturiert? Wird der Anwender bei der Formulierung und Einordnung z.B. durch automatisches Vervollständigen unterstützt? Wie sieht diese Unterstützung aus?
      - Wie werden Anforderungen spezifiziert? Können dabei selbst festgelegte Templates eingesetzt werden?
      - Wie werden Anforderungen gegliedert/strukturiert? Sind frei wählbare Strukturen möglich? Können Module vererbt werden?
      - Welche Möglichkeiten hat der Anwender zur Wiederverwendung von Anforderungen? Können einzelne Anforderungen oder ganze Module wiederverwendet werden?
      - Welche Funktionen zur Analyse von Anforderungen stellt die Anwendung zur Verfügung?
      - Welche Such-, Sortier- und Filterfunktionen können eingesetzt werden?
    - Management von Anforderungen
      - Welche Möglichkeiten zum Änderungsmanagement bietet das Produkt? Wie werden Änderungen dokumentiert?
      - Unterstützt die Anwendung Versions- und Variantenmanagement? Wie wird dies realisiert?
      - Welche Möglichkeiten zur Rückverfolgung von Anforderungen gibt es? (Traceability)
      - Können Meilensteine und Termine definiert und visualisiert werden?
    - Kollaboration und Abstimmung mit Lieferanten
      - Welche Möglichkeiten zum Datenaustausch mit Lieferanten bietet das Produkt?

- Wie unterstützt die Anwendung Teamwork? Können Funktionsgruppen mit einem gemeinsamen Arbeitsvorrat eingerichtet werden? Sind E-Mail Benachrichtigungen an Gruppen möglich?

* Darstellungen und Auswertungen
  - Wie ist die Oberfläche gestaltet? Fügen sie bitte aussagekräftige Screenshots bei, falls nötig.
  - Wie werden Auswertungen dargestellt?
  - Können Statistiken generiert werden? Welche sind möglich und wie werden sie visualisiert?
  - Welche Metriken stellt das Werkzeug zur Verfügung?

– Administration
* Nutzer- und Rechteverwaltung
  - Welche Möglichkeiten zur Nutzerverwaltung bietet das Produkt?
  - Wie ist die Rechteverwaltung umgesetzt? Wie fein granuliert können Rechte vergeben werden?
* Datenhaltung
  - Wie werden die Daten vorgehalten? Wird dazu eine Datenbank(welche?) oder ein anderes System eingesetzt?
* Werkzeuganpassung/Schnittstellen
  - Welche Möglichkeiten zur Integration mit anderen Tools (wie z.B. Teamcenter, SAP) bietet ihre Software? Können Schnittstellen selbst definiert werden? Werden Kommunikationsstandards wie ReqIF unterstützt?
  - Welche Möglichkeiten zur Anpassung an die Gegebenheiten innerhalb des Konzerns sind in der Anwendung möglich?
  - Stellt das Produkt ein API zur Verfügung, mit der andere Anwendungen live auf die Daten/Anwendung zugreifen können?
  - Welche Möglichkeiten bietet das Produkt Import und Export von Daten? Welche Formate werden unterstützt?
* Sicherheit
  - Wie ist die Kommunikation zwischen Client und Server verschlüsselt?
  - Ist der Datenbestand verschlüsselt? Wie?
  - Welche Systeme zur Benutzerauthentifikation werden unterstützt?
  - Wie wird der unerwünschte Zugriff auf Ressourcen verhindert?
* Workflow-Management
  - Welche Möglichkeiten zur Unterstützung oder Automatisierung von Geschäftsprozessen stellt das Produkt zur Verfügung?

            - Wird Testfallerstellung und Testfallmanagement unterstützt? Wie?
            - Welche Möglichkeiten zur Prozessunterstützung bietet das Tool?
            - Welche Möglichkeiten zur Prozesssteuerung unterstützt die Anwendung?
- Nichtfunktionale Anforderungen
    * Anwender
        - Wie performant ist die Anwendung? Mit welchen maximalen Wartezeiten ist zu rechnen?
        - Existieren verschiedene Lokalisierungen der Anwendung? Welche?
        - Kann die Anwendung in Teilbereichen auch Offline eingesetzt werden? In welchen?
        - Inwiefern ist die Anwendung ergonomisch gestaltet?
- Administration
    * Scalability
        - Wie gut lässt sich das Produkt skalieren? Wo liegen die Grenzen? Wie groß ist das größte System, dass erfolgreich eingesetzt wurde?
    * Support
        - Welchen Service umfasst der Support?
    * Kosten
        - Wie hoch sind die Entwicklungskosten?
        - Wie hoch sind die Lizenzkosten?
        - Wie hoch sind die Wartungskosten?
        - Welche Kosten fallen für Support an?
        - Fallen weitere, oben nicht genannte Kosten an?

Anhang E

Lebenslauf

Persönliche Daten :

geboren am 12.09.1985 in Hildesheim

Schule:

1991 - 1995	Grundschule Ochtersum in Hildesheim
1995 - 1997	Orientierungsstufe Ost in Hildesheim
1997 - 1999	Gymnasium Lahntalschule in Marburg - Biedenkopf
1999 - 2003	Scharnhorstgymnasium in Hildesheim

Studium:

Oktober 2003 – April 2009: Wirtschaftsinformatik-Studium an der TU Braunschweig
Abschluss des Vordiploms im November 2005 mit der Note 1,0
Stipendiat der MAN Nutzfahrzeuge AG Studienförderung
Abschluss des Diploms im April 2009 mit der Note 1,0

Vertiefungsrichtungen im Hauptstudium:
Finanzwirtschaft
Produktion und Logistik
Software Engineering

Berufserfahrung:

15.05.2006 – 13.04.2009	**Wissenschaftliche Hilfskraft am Institut für Software Systems Engineering an der TU Braunschweig.**

Aufgaben:

15.05.2006 – 31.07.2006:	Vorbereitung der Vorlesung [illegible] der Software-[illegible] nebst zugehöriger Übung
01.08.2006 – 31.12.2007	Mitglied der Projektleitung beim Instituts-übergreifenden Projekt „[illegible]“ mit dem die TU Braunschweig an der „DARPA Urban Challenge 2007“ des U.S. Militärs teilnimmt. Ziel ist die Entwicklung eines autonomen Fahrzeugs.
01.01.2008 – 13.04.2009	Organisatorische Betreuung des „Softwareentwicklungspraktikums 2008“ des Instituts mit insgesamt 11 Teilnehmern in 3 Gruppen. Diverse weitere Aufgaben.

14.04.2009 - 31.12.2011	**Wissenschaftlicher Angestellter am Lehrstuhl Software Engineering an der RWTH Aachen mit dem Ziel der Promotion**

Aufgaben:

14.04.2009 – 31.12.2011	Eigenverantwortliche Bearbeitung des Projekts „Vernetztes Anforderungsmanagement für modulare Produktstrukturen“ in Kooperation mit der Volkswagen AG in Wolfsburg.
01.01.2010 – 31.12.2010	Projektleiter „VIRE“ mit drei studentischen Hilfskräften und einem weiteren wissenschaftlichen Angestellten. Ziel ist die

	Entwicklung eines Web-basierten Frameworks zur Entwicklung von Online-Editoren sowie eines konkreten Prozess-Editors für die Volkswagen AG.
15.04.2010 – 31.12.2011	Leiter Arbeitsgruppe „Automotive“ mit drei zugeordneten Mitarbeitern und einem externen Doktoranden.

Seit 01.01.2012	**Systemanalytiker bei der Volkswagen AG im Bereich „IT-Projekte Controlling, Rechnungswesen“**
Aufgaben:	
01.01.2012 – 30.06.2013	Weiterentwicklung und Internationalisierung eines Mainframe-Systems zur Fahrzeugkalkulation.
Seit 01.07.2013	Lifecycle-Verantwortung für ein System zur weltweiten digitalen Rechnungsverarbeitung.

Veröffentlichungen:

T. Form, B. Rumpe, C. Berger, K. Cornelsen, M. Doering, J. Effertz, **T. Gülke**, K. Homeier, A. Movshyn, T. Nothdurft, M. Sachse, J. Wille
Caroline - Ein autonom fahrendes Fahrzeug im Stadtverkehr.
In: Proceedings des 8. Braunschweiger Symposiums "Automatisierungssysteme, Assistenzsysteme und eingebettete Systeme für Transportmittel", Februar 2007.

F. Rauskolb, K. Berger, C. Lipksi, M. Magnor, K. Cornelsen, J. Effertz, T. Form, F. Graefe, S. Ohl, W. Schumacher, J-M. Wille, P. Hecker, T. Nothdurft, M. Doering, K. Homeier, J. Morgenroth, L. Wolf, C. Basarke, C. Berger, **T. Gülke**, F. Klose, B. Rumpe.
Caroline: An Autonomoulsy Driving Vehicle for Urban Environments.
In: Journal of Field Robotics, Wiley Periodicals, Volume 25 Issue 9, pp. 674-724, September 2008.

C. Basarke, C. Berger, K. Berger, K. Cornelsen, M. Doering, J. Effertz, T. Form, **T. Gülke**, F. Graefe, P. Hecker, K. Homeier, F. Klose, C. Lipski, M. Magnor, J. Morgenroth, T. Nothdurft, S. Ohl, F. Rauskolb, B. Rumpe, W. Schumacher, J. Wille, L. Wolf.
Team CarOLO - Technical Paper.
Informatik-Bericht 2008-07
Technische Universität Braunschweig, Carl-Friedrich-Gauss-Fakultät, Oktober 2008.

C. Berger, **T. Gülke**, B. Rumpe
ProcDSL + ProcEd - a Web-based Editing Solution for Domain Specific Process-Engineering
In: Proceedings of the 9th OOPSLA Workshop on Domain-Specific Modeling (DSM '09), Orlando, Florida, USA, November 2009.

F. Rauskolb, K. Berger, C. Lipksi, M. Magnor, K. Cornelsen, J. Effertz, T. Form, F. Graefe, S. Ohl, W. Schumacher, J-M. Wille, P. Hecker, T. Nothdurft, M. Doering, K. Homeier, J. Morgenroth, L. Wolf, C. Basarke, C. Berger, **T. Gülke**, F. Klose, B. Rumpe.
Caroline: An Autonomoulsy Driving Vehicle for Urban Environments.
In: M. Buehler, K. Iagnemma, S. Singh (Eds.). The DARPA Urban Challenge - Autonomous Vehicles in City Traffic. Springer Tracts in Advanced Robotics, Volume 56, pp. 441-508, 2010.

T. Gülke, M. Jansen, B. Rumpe, J. Axmann: High-Level Requirements Management and Complexity Costs in Automotive Development Projects: A problem statement. In: Proceedings of the 18th international conference on Requirements Engineering: foundation for software quality (REFSQ 2012), Springer, 2012 (Lecture Notes in Computer Science (LNCS)), S. 94–100

Praktika:

08.01.2001 - 26.01.2001	Energieversorgung Hildesheim GmbH & Co. KG
24.06.2002 - 26.07.2002	EADS Deutschland GmbH, Ottobrunn
01.07.2003 - 26.09.2003	Siemens medical solutions, Forchheim
23.02.2004 - 30.03.2004	MAN Türkiye A.Ş. Evolution 2004, Ankara
09.03.2005 - 08.04.2005	NEOMAN Bus GmbH, Salzgitter
19.09.2005 - 28.02.2006	NEOMAN Bus GmbH, Salzgitter
13.02.2006 - 17.03.2006	Deutsche Bank Hannover, Spezialberatung Investment

Zertifikate:

EDV-Anwenderzertifikat (BNW) – Modul 1 und 2
Certified Tester Foundation Level der ISTQB

außerschulische Aktivitäten:

Teilnahme am 20. Bundeswettbewerb Informatik 2001/2002
Regional- und Landessieger bei „Jugend forscht 2002“ in Mathematik/Informatik
Sonderpreis der WYRE – Stiftung verliehen beim „Bundeswettbewerb Jugend forscht 2002“
Regional- und Vizelandessieger bei „Jugend forscht 2003“ in Mathematik/Informatik

Hobbies

Golf, Fotographie, Reisen

Sprachen:

Englisch in Wort und Schrift, Grundkenntnisse der französischen Sprache