Oguzhan Balandi

Ontologie-basierte Modellierung und Synthese von Bordnetzdaten für die Zuverlässigkeitsanalyse

kassel university press

Oguzhan Balandi

Ontologie-basierte Modellierung und Synthese von Bordnetzdaten für die Zuverlässigkeitsanalyse

kassel university press

Die vorliegende Arbeit wurde vom Fachbereich Elektrotechnik/Informatik der Universität Kassel als Dissertation zur Erlangung des akademischen Grades eines Doktors der Ingenieurwissenschaften (Dr.-Ing.) angenommen.

Gutachter: Prof. Dr. rer. nat. Ludwig Brabetz, Universität Kassel
Prof. Dr. rer. nat. Albert Zündorf, Universität Kassel

Prüfer: apl. Prof. Dr.-Ing. Mohamed Ayeb, Universität Kassel
Prof. Dr.-Ing. Stephan Frei, TU Dortmund

Tag der mündlichen Prüfung: 9. Dezember 2021

https://orcid.org/0000-0002-1631-4317 (Oguzhan Balandi)

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.dnb.de abrufbar.

Zugl.: Kassel, Univ., Diss. 2021
ISBN 978-3-7376-1020-9
DOI: https://doi.org/doi:10.17170/kobra-202202045713

Druck und Verarbeitung: Print Management Logistik Service, Kassel
Printed in Germany

Kurzfassung

Die steigende Elektrifizierung von Fahrzeugen stellt höhere Ansprüche an das zugrunde liegende Bordnetz. Gleichzeitig steigt die Komplexität des Bordnetzes parallel mit und es wird zunehmend zu einem sicherheitskritischen Produkt. Daher gewinnt die Zuverlässigkeitsanalyse von Bordnetzen zunehmend an Bedeutung. Um die Zuverlässigkeitsanalysen effektiv zu realisieren, ist die Zusammenführung von Daten aus verschiedenen Bordnetzanwendungen erforderlich, wobei die Diskontinuität der Systeme dies erschwert. Die Diskontinuität der Systeme ist eine allgemeine Herausforderung in vielen Bereichen und auch in der Domäne Entwicklung der Bordnetze. Ein integriertes und erweiterbares Produktmodell, das als „*Single-Source of Truth*“ dient, ist nach wie vor ein Ziel der Bordnetzentwicklung. Ein solches Produktmodell ist notwendig, um ein Bordnetz ausführlich digital auszuwerten und in naher Zukunft nur noch digital zu entwickeln. Weitere Vorhaben wie die automatisierte Produktion, machen dieses Ziel noch dringlicher. Dabei ist die Integration der Systeme mit hoher Komplexität und mit Hindernissen verbunden. Diese Arbeit untersucht die Kernprobleme auf der Datenebene, die robuste und nachhaltige Integration verhindern. Als Lösungsansatz werden in dieser Arbeit Ontologien bzw. Knowledge-Graph-Technologien und Methoden des Software-Engineerings für die strukturierte und semantische Datenintegration und -verarbeitung auf Basis des standardisierten Datenmodells der Bordnetzdaten vorgestellt. Aufbauend wird gezeigt, wie in diesem Ansatz die normgerechte Zuverlässigkeitsanalyse auf der Grundlage der funktionalen Sicherheit effektiv realisiert werden kann. Der konzeptionelle Ansatz wird als Prototyp umgesetzt, validiert und die gewonnenen Erfahrungen werden dargestellt. Anschließen wird gezeigt, dass dieser Ansatz zukunftsorientiert ist und zu den weiteren Anforderungen der Domäne wie Digital Twin, Digital Thread, Model-Based Systems Engineering beiträgt. Zum Schluss wird beschrieben, wie dieser Ansatz zu einem laufenden Entwicklungsprozess integriert werden kann.

Abstract

The expansion of vehicle electrification places higher demands on the underlying wiring harness. In parallel, the complexity of the wiring harness is also increasing, which makes it even more of a safety-critical product. Therefore, the reliability analysis of wire harnesses is becoming more important. In order to implement reliability analysis effectively, the integration of data from different wiring harness applications is required. However, the discontinuity of the systems makes this a bit more difficult. System discontinuity is a major challenge in the domain of wiring harness development like many other fields. An integrated and extensible product model that ensures as a "single source of truth" continues to be a goal of wiring harness development. Such a product model is necessary to evaluate a wiring harness digitally in detail and also to develop it exclusively digitally in the near future. Further developments, such as automated production, make this goal even more urgent. Thereby, the integration of the systems involves high complexity and various obstacles. This study explores the key data-level issues that prevent robust and sustainable integration. As a solution approach, this study presents ontologies or knowledge graph technologies and software engineering methods for structured and semantic data integration and processing based on the standardized wire harness data model. Based on this, it is shown how standard-compliant reliability analysis can be effectively realized on the base of functional safety. The conceptual approach will be implemented as a prototype, then it will be validated and the experiences will be presented. Subsequently, it is shown that this approach is future oriented and contributes to the further requirements of the domain such as digital twin, digital thread, and model-based systems engineering. Finally, it describes how this approach can be integrated into an ongoing development process.

Vorwort

Die vorliegende Arbeit entstand während meiner wissenschaftlichen Tätigkeit als Doktorand am Fachgebiet Fahrzeugsysteme und Grundlagen der Elektrotechnik an der Universität Kassel im Forschungsprojekt „Bordnetz Zuverlässigkeit“. Wissenschaftlich betreut wurde diese Arbeit von dem Leiter des Fachgebiets Prof. Dr. rer. nat. Ludwig Brabetz, dem ich herzlich für seine methodische und wissenschaftliche Begleitung der Arbeit und wertvollen Anregungen danke. Mein Dank gilt auch an apl-Prof. Dr.-Ing. Mohamed Ayeb, akademischer Oberrat im Fachgebiet, für die fachliche Betreuung und die förderlichen Ratschläge, die zum Erfolg dieser Arbeit beigetragen haben. Weiterhin gilt mein Dank Herrn Prof. rer. nat. Zündorf, Leiter des Fachgebiets Softwaretechnik an der Universität Kassel, für die Übernahme des Zweitgutachtens. Ferner möchte ich mich bei Herrn Prof. Dr.-Ing. Stephan Frei, Leiter des Fachgebiet Bordsysteme an der Technischen Universität Dortmund, für die Übernahme der Mitgliedschaft an der Prüfungskommission bedanken. Des Weiteren möchte ich mich bei allen Kollegen im Fachgebiet, insbesondre bei Johannes Korablin, Tobias Kerner und Oliver Baumgarten und den Ansprechpartnern Dr.-Ing. Ulrich Siebel, Dr.-Ing. Ralf Gemmerich und Kadir Aytemür bei dem Projektpartner Volkswagen AG für die gute Zusammenarbeit und die angenehme Arbeitsatmosphäre bedanken.

Besonderer Dank gilt an meinen Vorgesetzten Regine Stein und Dr. Christian Bracht an meiner späteren Arbeitsstelle an der Philipps-Universität Marburg für die gewährte Zeit zur Fertigung meiner Dissertation. Insbesondere möchte ich meinem Kollegen Werner Köhler für seine wertvollen Bemühungen danken, die zur Lesbarkeit dieser Arbeit beigetragen haben.

Ganz besonderer Dank gilt an meine Freunde für die großartige Unterstützung, die mir während des Studiums und der Promotion stets mit Rat und Tat zur Seite standen, insbesondere an Murat Şentaburlar, Erkam Atalay, Yusuf Altun und viele liebe Freunde aus Marburg.

Schließlich danke ich von tiefstem Herzen an meiner wundervollen Frau Neşe für ihre Begleitung mit Rückhalt, Geduld, Zuversicht und an unseren lieben Engelchen Elif und Feyza, die uns das Leben rundherum wunderschön und bunt machen. Ohne ihre Unterstützung wäre diese Arbeit nicht zustande gekommen. Gleichzeitig danke ich von ganzem Herzen an unseren Eltern, Geschwistern und liebevollen Nichten und Neffen, die uns moralisch unterstützt haben.

Inhaltsverzeichnis

1 Einleitung

In den letzten Jahren ist eine Vielzahl von Fahrerassistenzsystemen auf dem Markt erschienen, um die Sicherheit und den Fahrkomfort im Fahrzeug zu gewährleisten. Die Auswahl ist groß und reicht vom einfachen Geschwindigkeitsregler bis zum teilautomatisierten Fahren. Die zunehmenden Fahrerassistenzsysteme und die neuen Herausforderungen, wie hochautomatisiertes Fahren und die weitere Elektrifizierung der Komponenten, stellen dabei hohe Ansprüche an die Bewältigung der Komplexität in der Fahrzeugentwicklung. In erster Linie ist das physische Bordnetz von dieser Last betroffen, da das physische Bordnetz ein elementarer Bestandteil der Funktionsrealisierung ist. Die Komplexität im physischen Bordnetz steigt daher stetig mit. Das physische Bordnetz umfasst alle Hardwarekomponenten und Leitungen, die den Energie- und Signalfluss zwischen den elektrischen und elektronischen Bauteilen realisieren. Die vielfältige und große Anzahl von Komponenten, die Vernetzung und die Varianten komplizieren die Entwicklung, die Produktion und die Qualitätssicherung. Die zunehmende Elektrifizierung erfordert gleichzeitig höhere Sicherheitsanforderungen. Dabei spielen digitale Methoden eine wichtige Rolle, um die neuen Herausforderungen zu bewältigen und die Anforderungen abgesichert zu erfüllen. Parallel zur Entwicklung der Fahrzeuge müssen daher auch neue digitale Methoden für den Produktentwicklungsprozess mit entwickelt werden. Das vom Verband der Automobilindustrie standardisierte digitale Datenaustauschformat Vehicle Electric Container (VEC), das die digitale Beschreibung und somit auch die rechnergestützte Auswertung des physischen Bordnetzes ermöglicht, ist ein großer Fortschritt bei der digitalen Bordnetzentwicklung. Neben den Daten sind die Digitalisierung und Formalisierung des Wissens von Experten ein wichtiger Aspekt, um ein zuverlässiges Produkt in der digitalen Entwicklung zu erreichen. So beeinflussen beispielsweise die Materialentscheidungen der Komponenten bei der Entwicklung des Bordnetzes gleichzeitig die Kosten, das Gewicht und die Zuverlässigkeit des Bordnetzes. Daher ist es absolut wichtig, solches Wissen zu digitalisieren und auf die Daten anzuwenden, um ein zuverlässiges Bordnetz in der digitalen Entwicklung realisieren zu können. In dieser Arbeit wurde eine neue Methodik auf Basis von VEC entwickelt, die über die klassische Datenspeicherung hinausgeht und neue digitale Methoden bietet sowie die Komplexität der Bordnetzentwicklung reduziert und die digitale Entwicklung zuverlässiger Bordnetze gewährleistet. Im Folgenden werden das Problem und das Ziel erläutert und anschließend die Struktur der Arbeit vorgestellt.

1.1 Problemstellung

Diese Arbeit ist im Rahmen des Forschungsprojekts „Bordnetz Zuverlässigkeit“ am Fachgebiet Fahrzeugsysteme und Grundlagen der Elektrotechnik der Universität Kassel entstanden (Gemmerich, et al., 2016). Ziel dieses Forschungsprojektes war, die quantitative Zuverlässigkeit der Bordnetze von verschiedenen Fahrzeugmodellen auf der Grundlage der

digitalen Bordnetzdaten auszuwerten. Dabei wurde festgestellt, dass die Auswertung mit verschiedenen Herausforderungen verbunden ist. Um die Zuverlässigkeit des Bordnetzes zu bewerten, ist in erster Linie notwendig, die relevanten Bordnetz-Produktdaten zusammen mit den Daten aus den gesamten Produktlebenszyklen zu kombinieren. Dazu gehören beispielweise die Umweltbelastungszonen des Bordnetzes, die Simulationsergebnisse aus der Entwicklung, Funktionen, Ausfallraten aus den Katalogen, Ausfallinformationen aus Produktion und Wartung. Alle diese Daten sollten in einem strukturierten Modell kombiniert werden, um zuverlässige Ergebnisse zu erzielen. Dies ist nicht nur für die Auswertung, sondern auch für die Validierung der Ergebnisse wichtig. Darüber hinaus sollten die Daten und die Verfahren strukturiert erfasst werden, um die Überprüfbarkeit und Nachvollziehbarkeit der Ergebnisse zu gewährleisten. Zu diesem Zweck sollten Vielfalt und Komplexität der Daten für Menschen zugänglich und verständlich sein. Eine weitere wichtige Aufgabe besteht in der Bewältigung der Komplexität. Die Daten sollten für automatisierte Prozesse in einem validen Zustand vorliegen, da die Analysen sonst zu falschen Ergebnissen führen. Im Folgenden sind die Herausforderungen für eine abgesicherte Zuverlässigkeitsanalyse unter die vier Punkte (1) Integration der heterogenen Daten, (2) Integriertes Produktdatenmodell, (3) Automatisierung der Analysen und (4) Transparente Bewältigung der Komplexität zusammengefasst.

Integration der heterogenen Daten

Das Bordnetz als eines der grundlegenden und kompliziertesten Teile des Fahrzeugs besteht selbst aus vielen Modulen und wird arbeitsteilig von verschiedenen Abteilungen entwickelt. Der interne und externe Austausch von Informationen erfolgt dokumentenbasiert. Neben den Standardaustauschformaten werden auch diverse weitere Formate verwendet. Die Zusammenführung und das Management von gestreuten heterogenen Daten ist daher eine große und wiederkehrende Herausforderung in der Domäne. Die Daten stehen in der Regel in isolierten Silos und auch die Ergebnisse von Bordnetz-Entwicklungswerkzeugen stehen nebeneinander. Dies erschwert sowohl das Auffinden und Integrieren als auch den Zugriff auf die benötigten Daten. Eine weitere Herausforderung bei der Integration ist die Sicherstellung der Datenqualität und insbesondere der Vollständigkeit und Korrektheit der Daten.

Integriertes Produktdatenmodell

Die Beschreibung der Zusammenhänge der Daten ist ein wichtiger Aspekt bei der Integration der Daten. Die zu einem Produkt gehörenden relevanten Daten wie Simulationsergebnisse, Ausfallverhalten, Funktionen etc. sollten in einem Produktdatenmodell standardisiert beschrieben werden, damit die Daten nach diesem Standard zur Verfügung gestellt werden können. Das Produktdatenmodell ermöglicht einerseits die Datenintegration und andererseits die Interoperabilität der Entwicklungswerkzeuge. Entwicklungswerkzeuge können gemeinsam das standardisierte Modell anstelle eines eigenen nativen Datenmodells verwenden, so dass sie enger gekoppelt werden können. Die in den Prozessen aus der Verwendung divergenter nativer Datenmodelle resultierenden Lücken werden beseitigt. Ein

solches Produktdatenmodell sollte gleichzeitig skalierbar sein, so dass für den gesamten Lebenszyklus relevante Daten abgebildet werden können.

Automatisierung der Analysen
Die Automatisierung von Analysen ist ein wichtiges Kriterium, um manuelle, zeitaufwändige und fehleranfällige Prozesse zu vermeiden. Daher ist es notwendig, dass die Daten in einer flexiblen und algorithmierbaren Struktur vorliegen, um Analysen wie Datenvollständigkeit, Datenkorrektheit, Zuverlässigkeitsanalyse etc. zu automatisieren. Darüber hinaus werden für die maschinelle Verarbeitung entsprechende Schnittstellen benötigt.

Transparente Bewältigung der Komplexität
Der Einsatz neuer Technologien und Verfahren sollte nicht nur für Maschinen, sondern auch für den Menschen zugänglich sein. Die Daten und Methoden sollten transparent gestaltet und die Ergebnisse von Auswertungen für den Domänenexperten nachvollziehbar sein. Neben der zunehmenden Komplexität des Bordnetzes ist die Gestaltung der einfachen Handhabung eine besondere Herausforderung.

Im Laufe des Forschungsprojekts stellte sich heraus, dass die Zuverlässigkeitsanalyse mit den oben genannten Herausforderungen nicht in den Bordnetzentwicklungsprozess integriert werden kann, da die Bordnetzentwicklung selbst bereits mit den gleichen Herausforderungen zu kämpfen hat. In der Forschung existieren auch nur sehr wenige Arbeiten, die sich mit der Verbesserung der Entwicklungsprozesse von Bordnetzen beschäftigen. Im Forschungsprojekt wurde ein prototypisches Bordnetz-Analysetool für die quantitative Zuverlässigkeitsanalyse entwickelt. Ausgehend von in dieser Lösung gewonnenen Erkenntnissen und der Analyse der Bordnetzdomäne befasst sich die vorliegende Arbeit mit der Entwicklung eines allgemeinen Lösungsansatzes. Eine weitere bestehende Aufgabe ist die Analyse der Anforderungen aus der Sicht der funktionalen Sicherheit. Denn ein Ansatz ohne Berücksichtigung der funktionalen Sicherheit wäre unvollständig und nicht visionär, da dieses Thema bei den neuen Herausforderungen der Elektrifizierung und des autonomen Fahrens eine immer größere Priorität gewinnt. Die zunehmende Elektrifizierung bedeutet, dass Fahrzeuge immer mehr zu sicherheitskritischen Produkten werden. Die Zuverlässigkeitsanalysen am Bordnetz werden daher immer wichtiger. Deswegen gewinnt auch die ISO-Norm ISO 26262 (*Road vehicles – Functional safety*) für sicherheitsrelevante elektrische/elektronische Systeme in Kraftfahrzeugen in der Automobilindustrie immer mehr Bedeutung. ISO 26262 ist keine verbindliche Norm, sondern eine Empfehlung, bietet aber im Schadensfall einen relevanten Bezugspunkt, um nachzuweisen, dass die Anforderungen hinsichtlich des aktuellen Standes der Technik erfüllt sind. In der Bordnetzentwicklung ist die funktionale Sicherheit noch nicht vollständig umgesetzt. Zusammengefasst ist die Hauptforschungsfrage der Arbeit ***„Wie können die informationstechnischen Defizite in der Bordnetzentwicklung behoben werden, um die Bordnetzentwicklung zu verbessern und gleichzeitig normgerechte Zuverlässigkeitsanalysen auf Basis der funktionalen Sicherheit effektiv zu realisieren?“***.

1.2 Ziel und Vorgehensweise der Arbeit

Das Ziel dieser Arbeit ist es, den Einsatz von Semantic Web bzw. Knowledge-Graph-Technologien sowie Methoden des Software Engineering als Lösung für das vorgestellte Problem zu untersuchen. Das Semantic Web wurde entwickelt, um komplexe, heterogene und riesige Datenmengen im Internet zu bewältigen. Heutzutage werden diese Technologien zunehmend in der Industrie unter dem Namen Knowledge-Graph-Plattformen eingesetzt, weil Unternehmen mit den gleichen Problemen zu kämpfen haben. Die mit Hilfe der Ontologien semantisch vernetzten Daten, die sogenannten *Knowledge Graphs*, bilden eine Basis des Semantic Web. Knowledge Graphs stellen in der Industrie eine transparente und adaptive Plattform für kollaboratives Engineering und gleichzeitig eine algorithmierbare Ebene für die Automatisierung der Prozesse bereit. Es wird untersucht, wie die dokumentenbasierten und softwareintensiven Arbeitsabläufe der Bordnetzentwicklung in datenbasierte Abläufe umgewandelt werden können. Das variantenreiche Bordnetz wird dezentral von Teams entwickelt und schließlich zusammengeführt und getestet. Während der Entwicklung erstellen die Ingenieure digitale Produktdaten. Dieses Vorgehen findet ähnlich in der Softwareentwicklung statt. In der Softwareentwicklung wird der Quellcode auch dezentral entwickelt und getestet. Das Software Engineering bietet langjährige Erfahrung und zugleich entwickelte Methoden für die verteilte Entwicklung, das Testen und den Variantenreichtum von Software. Es wird untersucht, ob sich diese Methoden in die Entwicklung des Bordnetzes übertragen lassen.

1.3 Struktur der Arbeit

In dieser Arbeit werden zunächst die Grundlagen der Semantic Web bzw. Knowledge-Graph-Technologien vermittelt, die die Grundbausteine der Arbeit bilden und so zu einem besseren Verständnis der Arbeit beitragen. Im zweiten Kapitel wird der Stand der Technik in der Bordnetzentwicklung betrachtet. Dabei werden die Defizite im Entwicklungsprozess und in der Datenverarbeitung analysiert und dargestellt. Darauf aufbauend werden die informationstechnischen Anforderungen für die Realisierung normgerechter Zuverlässigkeitsanalysen für die Bordnetzentwicklung untersucht und aufgegliedert. Im Kapitel „Knowledge-Graph-basierter Lösungsansatz“ wird zunächst ein Lösungsansatz für die Datenverarbeitung in der Bordnetzentwicklung vorgestellt, anschließend mit den existierenden dokumentenbasierten Systemen verglichen und die Vorteile des Lösungsansatzes aufgezeigt. Aufbauend darauf wird die Knowledge-Graph-basierte Bordnetzentwicklung mit der Erweiterung durch Software-Engineering-Methoden vorgestellt. Es wird gezeigt, wie die Defizite in der Bordnetzentwicklung beseitigt und der Nutzen gegenüber den bestehenden Lösungsansätzen gesteigert werden kann. Der Ansatz wird im nächsten Abschnitt erweitert und es wird gezeigt, wie normgerechte Zuverlässigkeitsanalyse in der Bordnetzentwicklung umgesetzt werden kann. Im nachfolgenden Kapitel werden die prototypische Implementierung und die daraus

gewonnenen Erkenntnisse vorgestellt. Der Knowledge-Graph-basierte Ansatz wird dabei evaluiert. Im nächsten Kapitel wird untersucht, wie sich der Lösungsansatz technisch in die bestehenden Systeme integrieren lässt. Anschließend wird die Arbeit zusammengefasst und weitere Potentiale des Lösungsansatzes als Ausblick in die Zukunft dargestellt.

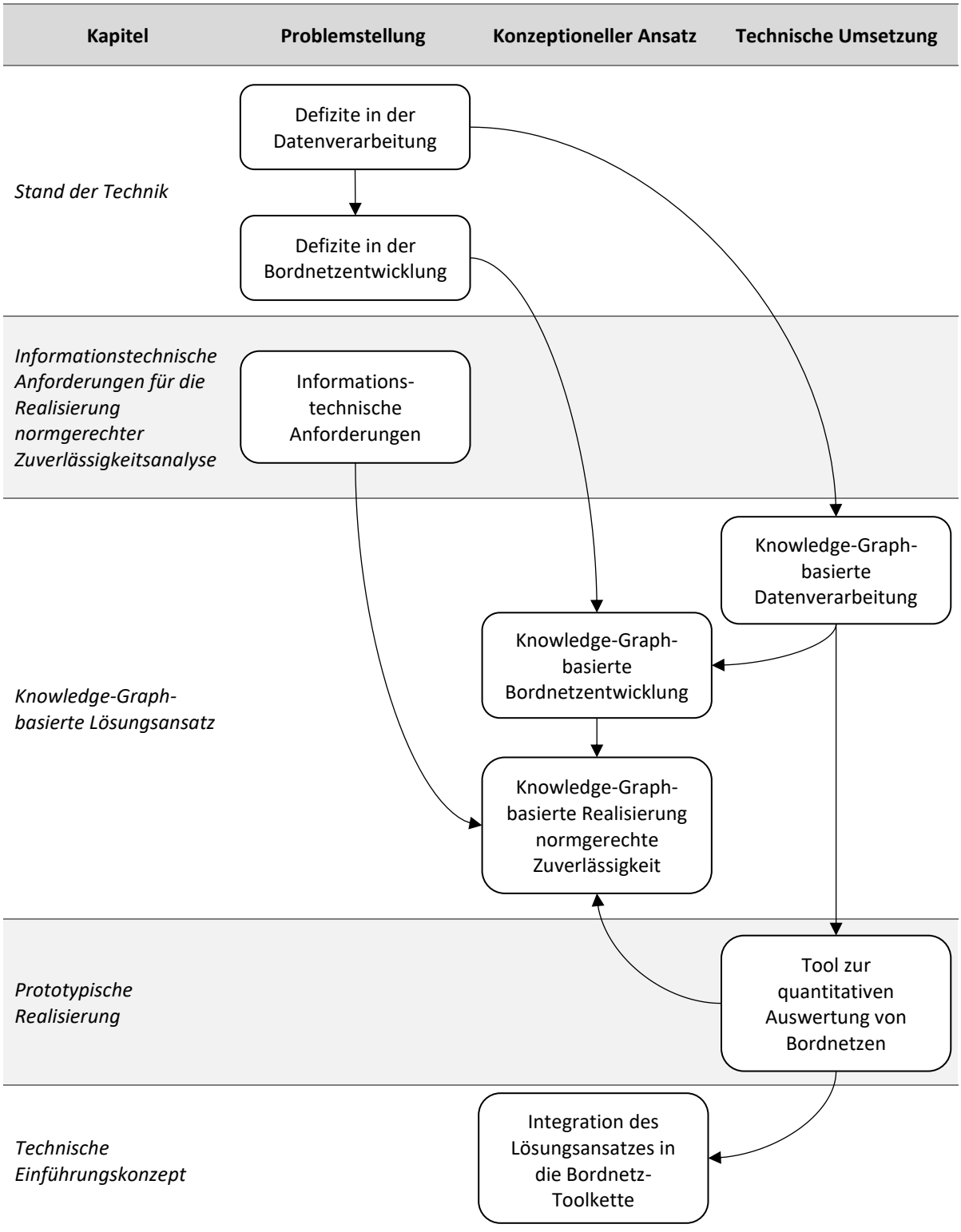


Abbildung 1 Struktur der Arbeit

2. Grundlagen

Das erste Kapitel gibt einen Überblick über die Semantic Web bzw. Knowledge-Graph-Technologien und zeigt Beispiele dafür, wie die Technologien eingesetzt werden können.

3. Stand der Technik

In diesem Kapitel werden die verwendeten Technologien, die Standards für den digitalen Datenaustausch und die Workflows in der Bordnetzentwicklung erläutert sowie die Defizite beschrieben.

4. Informationstechnische Anforderungen für die Realisierung normgerechter Zuverlässigkeitsanalyse

In diesem Kapitel werden Zuverlässigkeitsanalysen und die Norm zur funktionalen Sicherheit näher erläutert und die informationstechnischen Anforderungen zur Realisierung aufgegliedert.

5. Knowledge-Graph-basierte Bordnetzentwicklung

Dieses Kapitel bildet den Kern der Arbeit und stellt den Knowledge-Graph-basierten Lösungsansatz vor. Zunächst wird gezeigt, wie das Austauschformat *Vechile Electric Container* als Ontologie dargestellt wird, so dass anschließend der Lösungsansatz auf Basis dieser Ontologie entwickelt werden kann. Der Lösungsansatz wird präzisiert, und die sich daraus ergebenden Vorteile werden aufgezeigt und evaluiert.

6. Prototypische Realisierung

In diesem Kapitel werden technische Eigenschaften der im Forschungsprojekt entwickelten Anwendung vorgestellt. Die Knowledge-Graph-basierte Lösungsansatz wird anhand dieser Anwendung evaluiert. Dabei werden die durch den Einsatz der Knowledge-Graph-Technologien gewonnenen Erkenntnisse dargestellt.

7. Technische Einführungskonzept

Dieses Kapitel beschäftigt sich mit dem technischen Einführungskonzept der Knowledge-Graph-basierten Bordnetzentwicklung in Unternehmen. Es wird beschrieben, wie der Knowledge-Graph-basierte Ansatz durch die Erkenntnisse aus der prototypischen Realisierung in bestehende Systeme integriert werden kann.

8. Zusammenfassung und Ausblick

Das letzte Kapitel fasst die wesentlichen Inhalte und Ergebnisse dieser Arbeit zusammen und gibt einen Ausblick auf weitere Potentiale.

2 Grundlagen

In diesem Abschnitt werden die grundlegenden Begriffe und Themen erläutert, insbesondere der zentrale Begriff „Ontologie“, der in dieser Arbeit häufig verwendet wird. Die Verwendung von Ontologiesprachen im Kontext der Entwicklung des Semantic Web und weitere in diesem Zusammenhang wichtige Technologien werden näher erklärt. Daher bietet dieser Abschnitt die Bausteine für das weitere Verständnis der Arbeit.

Von Daten zum Wissen

Der Begriff „Ontologie“ wird in der Informatik als Instrument für Wissensrepräsentation definiert, wobei der Begriff „Wissensrepräsentation“ nicht selbsterklärend und nicht leicht verständlich ist. Daher ist es erforderlich die Grundzüge klarzustellen. „Datenverarbeitung“ und „Informationsverarbeitung“ sind durchaus bekannte Terme. Obwohl diese oft synonym verwendet werden, sind Daten und Informationen nicht gleichbedeutend. Die Daten sind die Kernelemente der Informatik. Sie bestehen aus einem elementaren Zeichenvorrat, der nach bestimmter Syntax zusammengestellt wird. Die Daten gewinnen ihre Bedeutung erst in einem Kontext. Daher ist eine Beschreibung der Daten auf einer höheren Ebene notwendig. In der Informatik werden dafür z. B. objektorientierte Datenmodelle eingesetzt, mittels derer Daten einer Domäne durch Klassen und Relationen beschrieben werden können, wodurch sie an Bedeutung gewinnen. Dabei kann durch Instanziierung, wie zum Beispiel `Sokrates` gehört zu der Klasse `Mensch`, Information gebildet werden. Wissen kann als die Fähigkeit, aus bestehenden Informationen neue Informationen abzuleiten, definiert werden. Hierbei geht es nicht um den Menschen als Zielgruppe, sondern um die Maschine, die in die Lage versetzt werden soll, Informationen interpretieren zu können und damit zu „verstehen“, um daraus

Wissen	Ableitung neuer Informationen aus vorhandenen Informationen	Sokrates ist sterblich.
Information	Kontext	Sokrates ist ein Mensch. Mensch ist sterblich.
Daten	Syntax	Sokrates
Zeichen	Zeichenvorrat	a, t, k, s, r, S, o, e

Tabelle 1 Zusammenhang zwischen Wissen, Information, Daten und Zeichen nach (Herrmann, 2012)

weitere Informationen abzuleiten bzw. automatisch Schlussfolgerungen zu ziehen. Die Datenmodelle reichen nicht aus, um Wissen zu betreiben, aber dennoch gibt es viele Ähnlichkeiten mit der ontologischen Wissensrepräsentation, da beide gleiche Sprachelemente

wie Klassen, Relationen etc. verwenden. Die Datenmodelle beschreiben eine Domäne und haben Vorbildcharakter und Nachbildcharakter (Hesse, 2006). Mit Hilfe der Beschreibung von Datenmodellen werden Datenbanken und Anwendungssysteme aufgebaut. Gleichzeitig werden Datenmodelle als Mittel der Kommunikation zwischen den Projektbeteiligten verwendet. Für die Datenmodellierung gibt es verschiedene Arten von Modellierungssprachen, wie z. B. relationale, hierarchische oder objektorientierte Datenmodellierung, die verschiedene Vorteile bieten. In der Wissensrepräsentation wird eine Domäne ähnlich wie bei der Datenmodellierung mit Hilfe von Sprachelementen beschrieben. Wissensrepräsentationssprachen ermöglichen es zusätzlich, eine Domäne noch expliziter durch formale Logik zu beschreiben, so dass eine Maschine das erstellte Modell auf formaler Basis interpretieren und weitere Informationen durch Algorithmen und Regeln ableiten kann. Das bedeutet, dass die Maschine die Daten nicht nur lesen, sondern auch verstehen kann. Obwohl Datenmodelle mit Ontologien der Wissensrepräsentation nicht direkt vergleichbar sind, weil sie für unterschiedliche Anforderungen entwickelt wurden, zeigt Abb. 2 die Beziehung zueinander, um den Unterschied besser einordnen zu können. Ontologien sind im Vergleich zu Datenmodellen ausdrucksstärker und formaler.

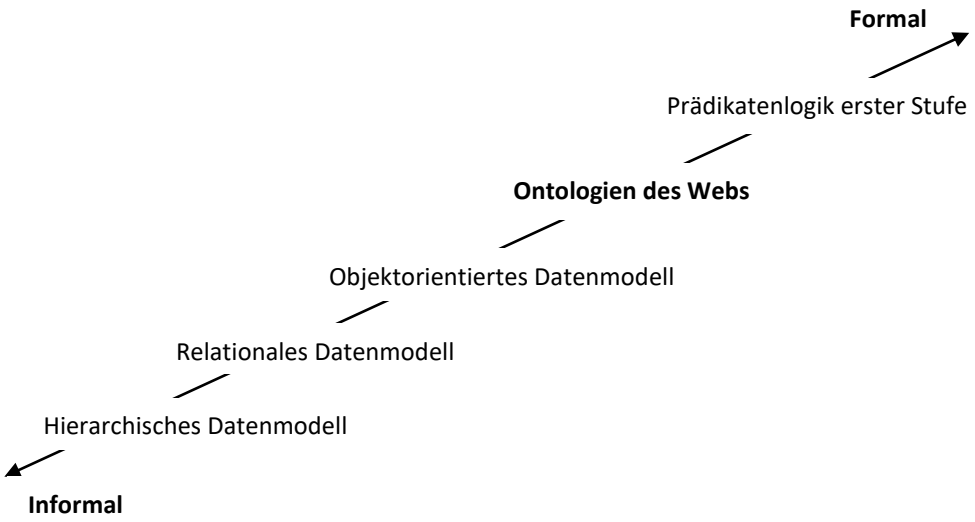


Abbildung 2 Überblick der Modelle von Informal zu Formal nach (Giunchiglia, et al., 2009)

Das World Wide Web Consortium (W3C) bietet zwei standardisierte Wissensrepräsentationssprachen zur Erstellung von Ontologien an, um Informationen im World Wide Web (im Folgenden abgekürzt: Web) formal zu beschreiben und somit die Informationen einerseits für Maschinen zugänglich zu machen und andererseits die Qualität der Informationen im Web auf einer formalen Ebene zu gewährleisten.

Ontologien

Der Begriff „Ontologie“ hat seine Wurzel in der Philosophie und repräsentiert die „Lehre vom Sein“ (Schuhbauer, et al., 2014). Die Ontologie, im Sinne einer philosophischen Fachdisziplin, befasst sich daher mit der Aufgabe, die Grundlagen des Seins zu erforschen.

Der Begriff wurde später von Informatikern übernommen. Ontologien werden in der Informatik für die formale Wissensrepräsentation einer Anwendungsdomäne verwendet. Laut meist verwendeter Definition in wissenschaftlichen Arbeiten ist eine Ontologie *„eine formale, explizite Spezifikation einer gemeinsamen Konzeptualisierung“* (Studer, et al., 1998). Ein solches Konzept, das gemeinsam, formal und explizit von interessierten Gruppen in einer Anwendungsdomäne spezifiziert und entwickelt wird, ermöglicht nicht nur eine bessere Kommunikation auf der zwischenmenschlichen Ebene, sondern auch die Integration und Interoperabilität auf der Maschinenebene. Das ist auch der größte Vorteil von Ontologien. Daher werden Ontologien besonders im Web eingesetzt, um die vielfältigen Daten mit sehr hohem Wachstum für Menschen und Maschinen leichter zugänglich zu machen. Die Vorteile von Ontologien wurden auch von der Industrie sehr schnell erkannt und werden zunehmend eingesetzt (Ege, 2015).

2.1 Semantic Web

Das Ziel der Entwicklung des Semantic Web ist die semantische Anreicherung des Web. Das Web verwendet Dokumente, die in der *Hypertext Markup Language* (HTML) dargestellt werden, um den Inhalt von Webseiten zu definieren. Die Inhalte der Dokumente werden durch die HTML-Auszeichnungssprache strukturiert und in maschinenlesbarer Form ins Web gestellt. Komplexe Inhalte können dabei auf mehrere Dokumente verteilt und die Dokumente untereinander mit Hyperlinks verknüpft werden. Hyperlinks ermöglichen somit eine Navigation von Dokument zu Dokument. Dieses Grundkonzept ist im Jahr 1990 von Tim Berners-Lee und Robert Cailliau am europäischen Forschungszentrum CERN entwickelt worden (Berners-Lee, et al., 1990). Die Dokumente im Web sind in erster Linie für die menschlichen Nutzer gedacht. Die Menschen können die Bedeutungen und die Beziehungen der Informationen vom Kontext her erkennen und intellektuell „verarbeiten“. Dagegen ist für Maschinen dieser Vorgang sehr komplex. Die Maschinen können z. B. in einem Text nicht direkt erkennen, ob mit dem Wort „Golf“ die Pkw-Modellreihe, die Sportart oder eine Meeresbucht gemeint ist. In diesem Fall wird durch Algorithmen die korrekte Bedeutung aus dem kontextuellen Umfeld als Schätzwert errechnet. Nach dieser Art funktionieren auch Suchmaschinen. Suchmaschinen lesen die Webdokumente ein und werten deren Inhalte und Hyperlinks aus, um eine Schlussfolgerung zu ziehen. Ein Nachteil liegt darin, dass die Informationen in Web stetig wachsen, was die Auswertung erschwert. Daher ist die Idee, Webinhalte mit den Ontologien des Semantic Web semantisch anzureichern, damit die Maschinen die Webinhalte besser interpretieren können und auf diese Weise semantische Suchen realisieren können. Das Netz des Web wird durch den *Uniform Resource Identifier* und die Hyperlinks realisiert. Ein *Uniform Resource Identifier* (URI) ist eine Zeichenkette, die eine abstrakte oder physische Ressource identifiziert (Berners-Lee, et al., 1998). Ein URI kann verschiedene Typen von Ressourcen, wie z. B. Webseiten, Dateien, E-Mail-Adressen usw. bezeichnen. Im Web wird beim Aufruf eines URI ein HTML-Dokument zurückgegeben, und dieses Dokument enthält weitere Hyperlinks, die zu weiteren URIs führen (Abb. 3).

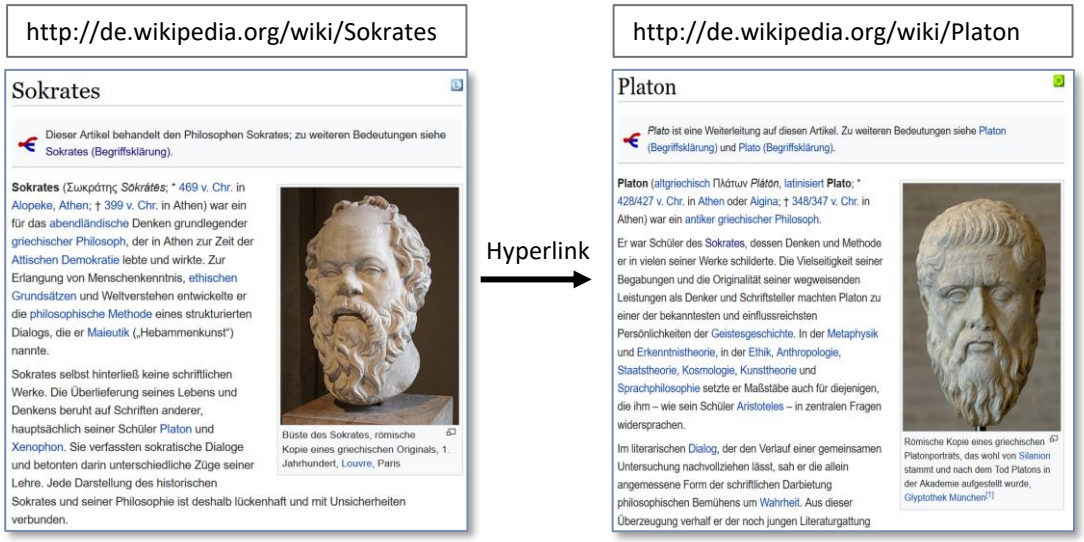


Abbildung 3 Ausschnitt aus Wikipedia – Sokrates und Platon

Eine Grundidee des Semantic Web ist, anstatt der HTML-Dokumente die beinhalteten Daten direkt zueinander zu verknüpfen. Dieses Konzept nennt sich *Linked Data* und bildet die Grundstruktur für das Semantic Web. In Linked Data referenzieren URIs auf Datentypen wie Zeichenkette, Zahl, Datum usw. und andere URIs. Ein bekanntes Beispiel ist das DBpedia-Projekt (DBpedia Association, 2014) in dem die Wikipedia-Inhalte als Linked Data dargestellt sind. In der Abbildung 4 identifiziert der URI dbr:Socrates[1] den Philosophen Sokrates. Die ausgehenden Links führen zu weiteren Informationen, und zwar entweder zu weiteren URIs wie bei rdf:type[2] und dbo:influenced[3] oder zu einfachen textlichen Informationseinheiten wie bei foaf:givenName[4] .

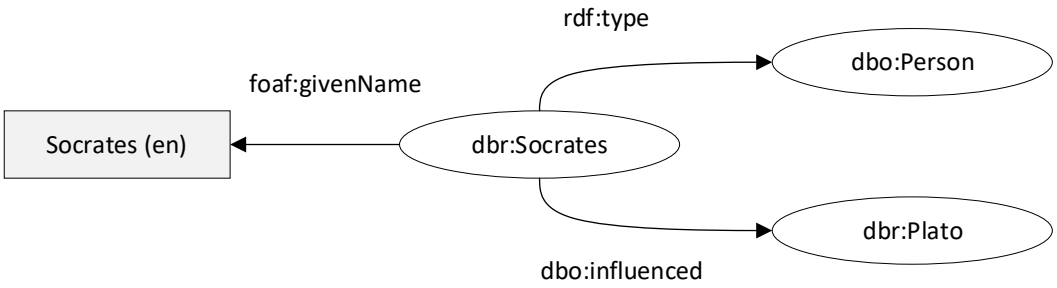


Abbildung 4 Ausschnitt aus DBpedia – Sokrates und Platon

In Linked Data werden Links mit Vokabularen beschrieben, die in einer Ontologie vordefiniert sind. Die Namensräume (*Namespace*) dbo, dbr, rdf und foaf zeigen, aus welcher Domäne und Ontologie die Elemente stammen. Auf diese Weise entsteht ein semantisches Netz, das es Maschinen ermöglicht, die Inhalte nicht nur linear zu verarbeiten, sondern auch

[1] http://dbpedia.org/resource/Socrates
[2] http://www.w3.org/1999/02/22-rdf-syntax-ns#
[3] http://dbpedia.org/ontology/influenced
[4] http://xmlns.com/foaf/spec/#term_givenName

zu interpretieren. In den folgenden Unterkapiteln werden die Funktionsweise und die Grenzen von Sprachen des Semantic Web detailliert erläutert.

2.2 Sprachen des Semantic Web

Das World Wide Web Consortium bietet standardisierte Technologien, um die Entwicklung des Semantic Web voranzutreiben. In diesem Unterabschnitt werden die Grundtechnologien, also die Sprachen für Datenvernetzung, Erstellung von Ontologien, Datenabfragen und Datenvalidierung vorgestellt. Dabei wird die Funktionsweise der maschinellen Inferenz erklärt. Anschließend wird der Begriff *Knowledge Graph* erläutert.

2.2.1 Resource Description Framework

Das Resource Description Framework (RDF) bildet das Fundament der Semantic-Web-Technologien. RDF wurde von der RDF Working Group im Jahr 2014 als Empfehlung für das Semantic Web veröffentlicht (RDF Working Group, 2014). RDF beruht auf einer einfachen Subjekt-Prädikat-Objekt-Struktur, genannt RDF-Tripel, um Ressourcen zu beschreiben bzw. Aussagen über Ressourcen zu treffen.

- **Subjekt** ist eine Ressource, über die eine Aussage formuliert wird.
- **Prädikat** ist eine Eigenschaft.
- **Objekt** ist der zugeordnete Eigenschaftswert.

Die Prädikate werden in RDF als *Property* bezeichnet. Ein RDF-Tripel ist entweder ein Ressource-Property-Ressource-Tripel oder ein Ressource-Property-Literal-Tripel. Literale sind einfache (primitive) Datentypen wie Zeichenketten, Ziffern oder Wahrheitswerte, die den Wert einer Eigenschaft repräsentieren. Die Menge der RDF-Tripel bildet zusammen einen gerichteten RDF-Graphen. RDF-Graphen können entweder serialisiert in Dokumenten ausgetauscht oder in RDF-Datenbanken (genannt auch Triplestores) (W3C, 2013) gespeichert werden. RDF-Tripel lassen sich formal in verschiedener Syntax wie XML, Turtle etc. ausdrücken.

```
@prefix dbo: <http://dbpedia.org/ontology/>.
@prefix dbr: <http://dbpedia.org/resource/>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix foaf:<http://xmlns.com/foaf/spec/>.



dbr:Socrates  a  dbo:Person;
                 foaf:givenName  “Sokrates”@(en);
                 dbo:influenced  dbr:Plato.
```

Quelltext 1 Serialisierung von RDF-Tripeln in Turtle-Syntax (RDF Working Group, 2014)

RDF-Graphen können sehr komplex wachsen. Daher bietet RDF eine Struktur, genannt *Named Graph*, die es mittels einer einfachen Vorgehensweise ermöglicht Untergraphen zu

bilden. Dabei werden Ressourcen, die zu einem gemeinsamen Namespace gehören, gemeinsam adressiert und können somit separat behandelt werden.

2.2.2 Resource Description Framework Schema

Das RDF-Schema (RDFS) ermöglicht durch wenige Sprachelemente einfache Ontologien zu erstellen, daher wird es auch als leichtgewichtige Ontologiesprache bezeichnet. RDFS wurde von der RDF Working Group im Jahr 2014 als Empfehlung für das Semantic Web veröffentlicht (RDF Working Group, 2014). RDFS basiert auf der Klassen- und Relationenstruktur, die aus der objektorientierten Modellierung bekannt ist, und bietet eine große Möglichkeit, Domänen detailliert zu modellieren. In RDFS werden Klassen mit rdfs:Class bezeichnet und Unterklassen können mit der Relation rdfs:subClassOf erstellt werden. Relationen zwischen Klassen werden durch rdf:Property realisiert. Es können in RDFS von bestehenden Relationen auch weitere Unterkategorien mit der Relation rdfs:subPropertyOf gebildet werden. Die Properties können mit rdfs:domain einer Urbild-Klasse und mit rdfs:range einer Abbild-Klasse zugeordnet werden. In Abbildung 5 ist eine Ontologie mit zwei Klassen und einer Property visuell dargestellt. Die Ontologie beschreibt eine Domäne auf der Metaebene. Dabei werden die Instanzen durch Verwendung der Property rdf:type aus den RDFS-Klassen erzeugt.

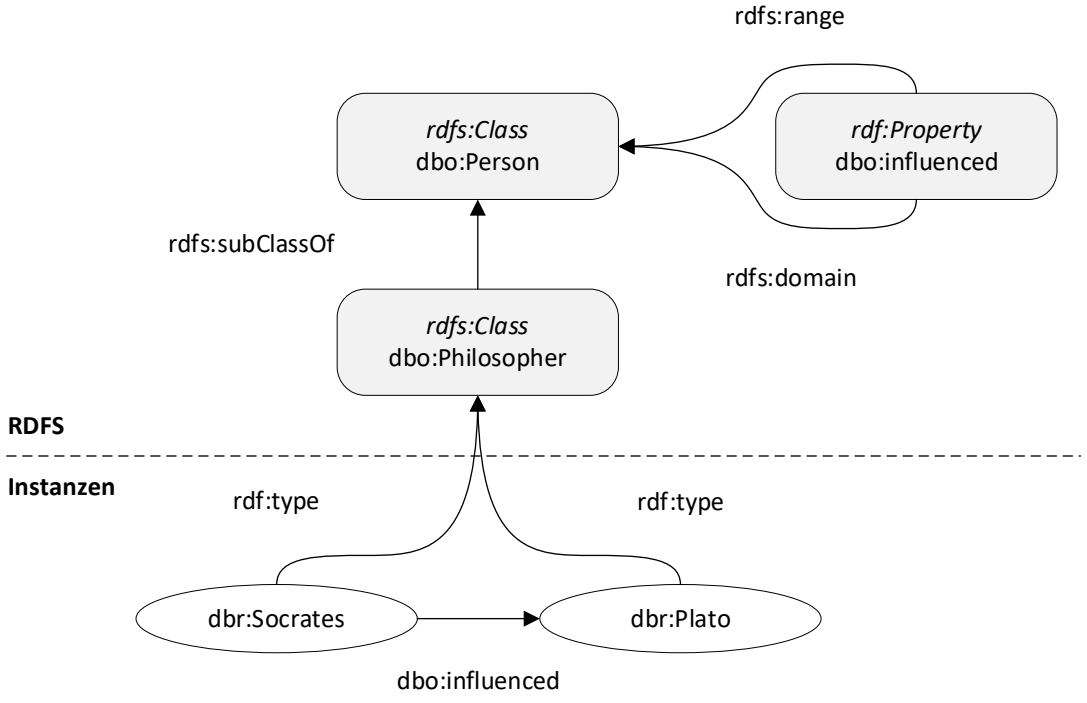


Abbildung 5 Beispiel von RDFS mit Instanzen

Auf diese Art und Weise wird das Semantic Web realisiert. RDF-Graphen müssen nicht unbedingt im Web umgesetzt und publiziert werden, sondern können auch in geschlossen Datenbanken realisiert werden. RDF-Graphen sind beliebig erweiterbar, auch ohne ein RDF-Schema. Ein RDF-Schema bietet jedoch die Vorteile, einerseits einer Überprüfung der

Datenkonsistenz, und andererseits können durch maschinelle Inferenz aus bestehenden Informationen weitere Informationen abgeleitet werden. Im Beispiel ist dbr:Socrates vom Typ dbo:Philosopher, und dbo:Philosopher ist Unterklasse von dbo:Person. Nach Durchführung der maschinellen Inferenz ist dbr:Socrates zugleich vom Typ dbo:Person. In diesem Abschnitt wurden nur die Grundelemente von RDF und RDFS behandelt. Beide Standards bieten darüber hinaus weitere nützliche Sprachelemente.

2.2.3 Web Ontology Language

Die Web Ontologie Language (OWL) ist eine vom W3C entwickelte Ontologiesprache für das Web, und die (zweite) Version OWL 2 wurde im Jahr 2012 standardisiert (OWL Working Group, 2012). OWL ist ausdrucksstärker als RDFS, basiert aber auf der gleichen Grundidee. OWL bietet viele weitere Sprachelemente zur detaillierten Beschreibung einer Anwendungsdomäne, von denen einige hier gezeigt werden. OWL besteht aus Klassen und Properties wie bei RDFS. Die Klassen in OWL werden durch owl:class definiert. In OWL existieren zwei Typen von Properties, zum einen owl:ObjectProperty für die Beziehung zwischen zwei Klassen, zum anderen owl:DataProperty für die Beziehung zwischen einer Klasse und einem einfachen (primitiven) Datentyp. OWL bietet die Möglichkeit die Klassen durch logische Konstrukte in Beziehung zu setzen und zu den Properties weitere Eigenschaften hinzuzufügen. Klassenkonstruktoren und Property-Eigenschaften ermöglichen eine noch strengere Konsistenzprüfung und erweiterte Ableitung neuer Informationen. Tabelle 2 zeigt einen Überblick mit Beispielen.

	Restriktionen	**Beispiel**
Klassen-konstruktoren	**Logisches UND** owl:intersectionOf	Ein Philosoph ist ein Denker und ein Mensch.
	Logisches ODER owl:unionOf	Ein Philosoph ist ein Dialektiker und/oder Theoretiker.
	Ausschließendes ODER owl:disjointWith	Ein Philosoph ist ein Empirist oder Rationalist.
	Logische Negation owl:complementOf	Ein Philosoph ist kein Sophist.
	Gleichheit owl:equivalentClass	Ein Philosoph ist ein Weiser.
Property-Eigenschaften	**Gleichheit** owl:equivalentProperty	**Ontologie:** „Elternteil von“ und „hat Kind“ sind gleiche Properties. **Daten:** Sokrates ist Elternteil von Menexenos. Sokrates hat Kind Menexenos. **Impliziert:** Die Beziehungen sind identisch.
	Disjunkt owl:propertyDisjointWith	**Ontologie:** „Elternteil von“ und „Ehepartner von“ sind disjunkte Properties. **Daten:** Sokrates ist Ehepartner von Xanthippe **Impliziert:** „Sokrates hat Kind Xanthippe.“ führt zu Inkonsistenz.

Reflexiv owl:ReflexiveProperty	**Ontologie:** Die Property „kennt“ ist reflexiv. **Daten:** Sokrates kennt Sokrates. **Impliziert:** Keine Inkonsistenz
Irreflexive owl:IrreflexiveProperty	**Ontologie:** „Elternteil von“ ist eine irreflexive Property. **Daten:** Sokrates ist Elternteil von Sokrates. **Impliziert:** Führt zu Inkonsistenz
Inverse owl:inverseOf	**Ontologie:** Die Properties „beeinflusst“ und „beeinflusst von“ sind invers. **Daten:** Sokrates beeinflusst Platon. **Impliziert:** Platon ist beeinflusst von Sokrates.
Transitiv owl:transitiveProperty	**Ontologie:** Die Property „beeinflusst“ ist transitiv. **Daten:** Sokrates beeinflusst Platon. Platon beeinflusst Aristoteles. **Impliziert:** Sokrates beeinflusst Aristoteles.
Symmetrisch owl:SymmetricProperty	**Ontologie:** „EhepartnerIn von“ ist eine symmetrische Property. **Daten:** Sokrates ist EhepartnerIn von Xanthippe. **Impliziert:** Xanthippe ist EhepartnerIn von Sokrates
Asymmetrisch owl:AsymmetricProperty	**Ontologie:** „Elternteil von“ ist eine asymmetrische Property. **Daten:** Sokrates ist Elternteil von Menexenos. **Impliziert:** „Menexenos ist Elternteil von Sokrates“ führt zu Inkonsistenz.
Funktional owl:FunctionalProperty	**Ontologie:** Die Property „geboren in“ ist funktional. **Daten:** Sokrates ist geboren in Athen. Sokrates ist geboren in Αθήνα. **Impliziert:** Athen und Αθήνα sind semantisch identisch.
Inverse-Funktional owl:InverseFuntionalProperty	**Ontologie:** „Vater von“ ist eine invers-funktionale Property. **Daten:** Sokrates ist Vater von Menexenos. Meister der Philosophie ist Vater von Menexenos. **Impliziert:** Sokrates und Meister der Philosophie sind semantisch identisch.

Tabelle 2 Übersicht zu Klassenkonstruktoren und Property-Eigenschaften in OWL

2.2.4 Maschinelle Inferenz

Maschinelle Inferenz bedeutet automatische Ableitung von neuen Aussagen aus einer bestehenden Ontologie und aus ihren Instanzen. Die Maschine bzw. Software, die diesen Vorgang durchführt, wird Inferenzmaschine genannt. Eine Inferenzmaschine überprüft die Konsistenz der Ontologie und leitet neue Aussagen ab. In beiden vorgestellten Sprachen, RDFS und OWL, ist maschinelle Inferenz möglich. Inferenzmaschinen werden im Kontext des Semantic Web auch *Reasoner* genannt, und es existieren sowohl kommerzielle als auch Open-Source-Inferenzmaschinen (W3C, 2011). Das RDFS basiert auf formaler Semantik,

weshalb durch Ableitungsregeln einfache syntaktische Schlussfolgerungen durchgeführt werden können (Hitzler, et al., 2008). Die Ableitungsregeln beschreiben durch Tripel sowohl Fall als auch Schlussfolgerung. Wenn z. B. die Tripel dbr:Socrates rdf:type dbo:Philosopher sowie dbo:Philosopher rdfs:subClassOf dbo:Person vorkommen, folgt nach untenstehender Ableitungsregel dbr:Socrates rdf:type dbo:Person. Die Ableitungsregeln werden in einem Algorithmus systematisch durchgeführt. In dieser Weise werden neue Aussagen produziert, die vorher nicht explizit vorhanden waren (Hitzler, et al., 2008):

$$\frac{y \;\; rdfs{:}subClassOf \;\; x\,.\;\; u \;\; rdf{:}type \;\; y\,.}{u \;\; rdf{:}type \;\; x.}$$

Gegenüber RDFS basiert OWL auf Beschreibungslogik als Teilsprache der Prädikatenlogik erster Stufe. Daher erfolgt die Inferenz in OWL durch Beweisverfahren mittels des Tableau-Algorithmus, der am häufigsten verwendeten Algorithmus in den Inferenzmaschinen (Hitzler, et al., 2008). Die Komplexität der Algorithmen hängt von der Ontologie und den eingegebenen Daten ab. OWL 2 existiert in zwei verschiedene Varianten: Full und DL (OWL Working Group, 2012). Sie unterscheiden sich durch ihre Komplexität. OWL 2 Full beinhaltet alle Sprachelemente von RDFS und OWL 2. OWL 2 Full ist nicht entscheidbar. Eine Ontologiesprache ist dann entscheidbar, wenn ein Algorithmus existiert, der entscheiden kann, ob eine Aussage von der Ontologie ableitbar ist oder nicht. Ein solcher Algorithmus wie der Tableau-Algorithmus existiert für OWL 2 Full nicht. Dagegen ist OWL 2 DL als eingeschränkte Teilsprache von OWL 2 Full entscheidbar. Der erweiterte Tableau-Algorithmus terminiert im Worst Case für die Sprache OWL 2 DL nach N2ExpTime, d. h. nichtdeterministisch in exponentieller Zeit. In der Praxis wird er trotzdem benutzt und für kleine Ontologien ist er ausreichend. Darüber hinaus gibt es drei OWL 2 Profile: EL, QL und RL (OWL Working Group, 2012), die für unterschiedliche Anwendungsfälle gedacht sind. Hierbei ist OWL 2 QL für die Anwendung im Kontext relationaler Datenbankmanagementsysteme (RDBMS) gedacht, worauf in Kapitel 5 rekurriert wird. Alle drei Profile sind Untersprachen (sub-languages) von OWL 2, und die maschinelle Inferenz läuft bei allen in Polynomialzeit (OWL Working Group, 2012)

Die Inferenzmaschinen unterstützen auch teilweise die regelbasierte Inferenz. Eine Regel besteht aus IF- und THEN-Klauseln. In der IF-Klausel wird die Voraussetzung definiert, und die THEN-Klausel beinhaltet die durchzuführende Aktion. Die IF-THEN-Regeln ermöglichen benutzerdefinierte Schlussfolgerungen auszuführen und neue Informationen in die Datenbank wie folgt hinzuzufügen:

$$hatKind(?x, ?y) \wedge weiblich(?y) \rightarrow hatTochter(?x, ?y)$$

Diese Inferenz kann z. B. in OWL nicht realisiert werden. Die regelbasierte Inferenz deckt solche Lücken von Ontologien ab. Das W3C bietet (nicht offizielle Empfehlung) hierzu die Sprache *Semantic Web Rule Language* (SWRL) (SWRL Working Group, 2004) für eine Regelbeschreibung. Der Nachteil von SWRL ist, dass sie nicht entscheidbar ist. Aber

trotzdem bieten viele Inferenzmaschinen die SWRL als Feature und in der Praxis ist sie für viele Probleme ausreichend. SWRL erweitert die OWL-Ontologie mit Regeln. Eine andere und weitverbreitete Regelsprache ist Jena Rules (*Generic rule reasoner*) von Apache Jena API (Apache Jena, 2012). Jena Rules bietet regelbasierte Inferenz auf RDF-Daten, so dass die IF-Klausel durch Tripel definiert werden kann und die Aktionen der THEN-Klausel dann ausgeführt werden, wenn die Tripel der IF-Klausel in den Daten auftreten.

2.2.5 Datenabfrage

Als Möglichkeit RDF-Daten abzufragen, bietet das W3C die standardisierte Abfragesprache *SPARQL Protocol And RDF Query Language* (SPARQL) (SPARQL Working Group, 2013). Eine SPARQL-Abfrage basiert auf Graph-Mustern. In einer Abfrage wird durch Variablen und Tripel ein Graph-Muster definiert.

```
@prefix dbo:  <http://dbpedia.org/ontology/>.
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.

SELECT ?philosopher ?influencedPhilosopher

WHERE
{
  ?philosopher rdf:type dbo:Philosopher.
  ?influencedPhilosopher rdf:type dbo:Philosopher.
  ?philosopher dbo:influenced ?influencedPhilosopher.
}
```

Quelltext 2 SPARQL-Abfrage

Eine SPARQL-Abfrage beginnt mit der Präfixdefinition für die Namenräume. Durch die Verwendung von Präfixdefinitionen werden die SPARQL-Abfragen übersichtlicher. Die `SELECT`-Klausel definiert die Variablen, und entsprechend der Reihenfolge der Variablen wird die Ausgabe formatiert. In der `WHERE`-Klausel wird, basierend auf den Variablen, das Graph-Muster beschrieben. Im obigen Beispiel wird abgefragt „Welcher Philosoph welchen Philosophen beeinflusst hat“. Das Abfrageergebnis ist eine Tabelle, in der die Variablen die Spalten bilden. Wenn das Muster nicht gefunden wird, ist die Ausgabe eine leere Tabelle. Neben der Konsolenausgabe bieten viele Tools die Möglichkeit, die Ergebnistabelle in verschiedene Ausgabeformate wie XML, Excel, etc. zu exportieren.

?philosopher	?influencedPhilosopher
dbr:Socrates	dbr:Plato
dbr:Socrates	dbr:Antisthenes
dbr:Antisthenes	dbr:Diogenes_of_Sinope

Tabelle 3 Ausgabe von SPARQL-Abfrage

SPARQL bietet viele weitere Funktionen, wie arithmetische und logische Operatoren, Filter, Sortierung etc., mit denen die Abfrage genauer spezifiziert werden kann. Neben der `SELECT`-

Anweisung existieren weitere Abfrageformen wie `CONSTRUCT`, `ASK` und `DESCRIBE`. Die `CONSTRUCT`-Abfrageform ermöglicht, neue Tripel in das Dataset einzufügen. Die `ASK`-Abfrageform dient zum Testen, ob ein Graph-Muster im RDF-Graph existiert, und erzeugt als Ausgabe einen booleschen Wert, *wahr* oder *falsch*. Die `DESCRIBE`-Abfrageform liefert RDF-Daten, die die Struktur des RDF-Graphs beschreiben. Diese Abfrageform wird benutzt, um herausfinden, wie die Struktur des RDF-Graphs aufgebaut ist.

2.2.6 Datenvalidierung

Die *Shapes Constraint Language* (SHACL) ist eine Sprache für die Validierung von RDF-Graphen gegen eine Reihe von Bedingungen (*Constraints*). SHACL wurde im Jahr 2017 vom W3C standardisiert (RDF Data Shapes Working Group, 2017). Eine in SHACL definierte Liste von Bedingungen wird Shapes-Graph genannt. Ein Shapes-Graph besteht aus einzelnen Shapes und in einem Shape können Constraints für ein bestimmtes Objekt (*targetNode*) oder für Objekte einer Klasse (*targetClass*) oder für Objekte, die eine bestimmte Property benutzen (*targetProperty*), definiert werden. Im folgendem Beispiel wird in dem Shape `ex:LessThanExample` als `targetClass` für Instanzen der Klasse `dbo:Person` folgende

```
#Shape-Graph
ex:  http://example.com/ns#
sh:  http://www.w3.org/ns/shacl#

ex:LessThanExample
  a sh:NodeShape ;
  sh:targetClass dbo:Person ;

  sh:property [
    sh:path dbo:birthDate;
    sh:lessThan dbo:deathDate;
  ] .

#RDF-Graph:
dbr:Socrates    a dbo:Person ;
      dbo:birthDate "0469-01-01"^^xsd:date.
      dbo:deathDate "0399-01-01"^^xsd:date.
```

Quelltext 3 SHACL-Beispiel

Einschränkung definiert: die Eigenschaft `dbo:birthDate` darf nicht kleiner als `dbo:deathDate` sein. Gegenüber diesem Shapes-Graph ist der RDF-Graph im Beispiel invalide. Bei der Validierung werden als Input sowohl der Shapes-Graph als auch die RDF-Daten übergeben. Nach der Validierung wird ein Report erstellt, der die invaliden Stellen repräsentiert. SHACL bietet viele Sprachelemente um z. B. stringbasierte, logische, typenbasierte und weitere Einschränkungen zu definieren. SPARQL-Abfragen können auch in SHACL eingebettet werden, um komplexe Bedingungen zu definieren, und es ist auch möglich, SHACL mit eigenem Code zu erweitern. (RDF Data Shapes Working Group, 2017).

2.3 Vom Semantic Web zum Knowledge Graph

Der Vorteil von Semantic-Web-Technologien ist in den letzten Jahren von der Industrie sehr schnell erkannt worden, so dass viele renommierte Softwarehäuser, auf diesen Technologien basierende Plattformen für Unternehmen entwickelt haben. Heutzutage werden diese Technologien, die ursprünglich für das Web entwickelt worden sind, zunehmend in der Industrie eingesetzt. Daher erschien der Begriff „Semantic-Web-Technologien" in den letzten Jahren auch unter anderen Namen, z. B. „Semantische Technologien", „Enterprise Semantic Web", „Corporate Semantic Web", „Enterprise Linked Data" und aktuell wird häufig die Bezeichnung „Knowledge Graph" verwendet. In (Ehrlinger, et al., 2016) wird beschrieben, dass „Knowledge Graph" ein *buzzword* ist, das von Google geprägt und von anderen Unternehmen und Wissenschaftler übernommen wurde, um verschiedene Anwendungen zur Wissensrepräsentation zu beschreiben. Es wird in (Gutierrez, et al., 2021) argumentiert, dass der Knowledge Graph seinen Ursprung nicht nur im Bereich Semantic Web hat, sondern dass er das Ergebnis vieler Fachdisziplinen der Informatik ist, die mit den ersten digitalen Computern und Programmiersprachen begonnen haben, um das Wissen zu materialisieren und mit Daten zu konzeptualisieren. Der Begriff „Knowledge Graph" (KG) ist hier vom Begriff „Labeled Property Graph" (LPG) abzusetzen. LPGs werden in Graph-Datenbanken benutzt und wurden im Kontext von NoSQL (*Not only SQL)* entwickelt, um Problematiken von relationalen Datenbaken zu lösen. Um zum Beispiel in relationalen Datenbaken komplexe Beziehungen herauszufinden, müssen zuerst die Tabellen mit vielen JOIN-Operatoren verknüpft werden. Dagegen können mittels LPGs die Beziehungen direkt zueinander verknüpft und ohne JOIN-Operatoren abgefragt werden. LPGs basieren auf der mathematischen Graphentheorie, und die Daten werden als Knoten und Kanten gespeichert. Es existieren Frameworks, wie z. B. Gremlin (Apache TinkerPop, 2018), die es ermöglichen, auf Graphen effizient zu traversieren und so effektive Graph-Algorithmen zu implementieren. Nachteil dabei ist, dass LPGs keine Metasprache, wie Ontologien bei KGs, bieten. Gegenüber KGs haben LPGs den Vorteil, dass Informationen auch unter Kanten gespeichert werden können, was bei RDF nicht möglich war. In der Zwischenzeit sind viele hybride Anwendungen entstanden, die beide Technologien gleichzeitig anbieten. Um die Lücke zu LPGs zu schließen, gibt es für RDF Ansätze zu Erweiterungen RDF* und SPARQL* (Hartig, 2017), die es ermöglichen, auch zu Properties Eigenschaften hinzuzufügen und diese abzufragen.

Knowledge-Graph-Plattform

Eine Knowledge-Graph-Plattform besteht aus einer RDF-Datenbank und den implementierten Sprachen des Semantic Web als Werkzeugen:

- OWL und RDFS dienen für die Erstellung einer Ontologie.
- Die Daten werden als RDF-Triple nach erstellter Ontologie in die RDF-Datenbank gespeichert. Es existieren auch Mapping-Sprachen (Dimou , et al., 2020) (RDB2RDF

Working Group, 2012) und Werkzeuge, um verschiedene Datenquellen mit der RDF-Datenbank zu verbinden.

- Inferenzmaschine und Regeln ermöglichen, Daten automatisch und semantisch zu verarbeiten.
- Durch die Abfragesprache SPARQL können komplexe Beziehungen abgefragt werden.
- Die Qualität der Daten wird durch SHACL sichergestellt.
- Darüber hinaus ermöglichen zusätzliche APIs die Entwicklung von qualitativen Anwendungen auf dem Knowledge-Graphen mit wenig Code. Die Graphenstruktur der Daten und der objektorientierte Zugriff auf die Daten ermöglichen eine hohe Flexibilität beim Entwurf komplexer Algorithmen.

Knowledge-Graph-Plattformen bieten neben einem Gesamtpaket an Semantic-Web-Technologien weitere Tools zur Datenanalyse, Datenvisualisierung (Abb. 6), visuellen

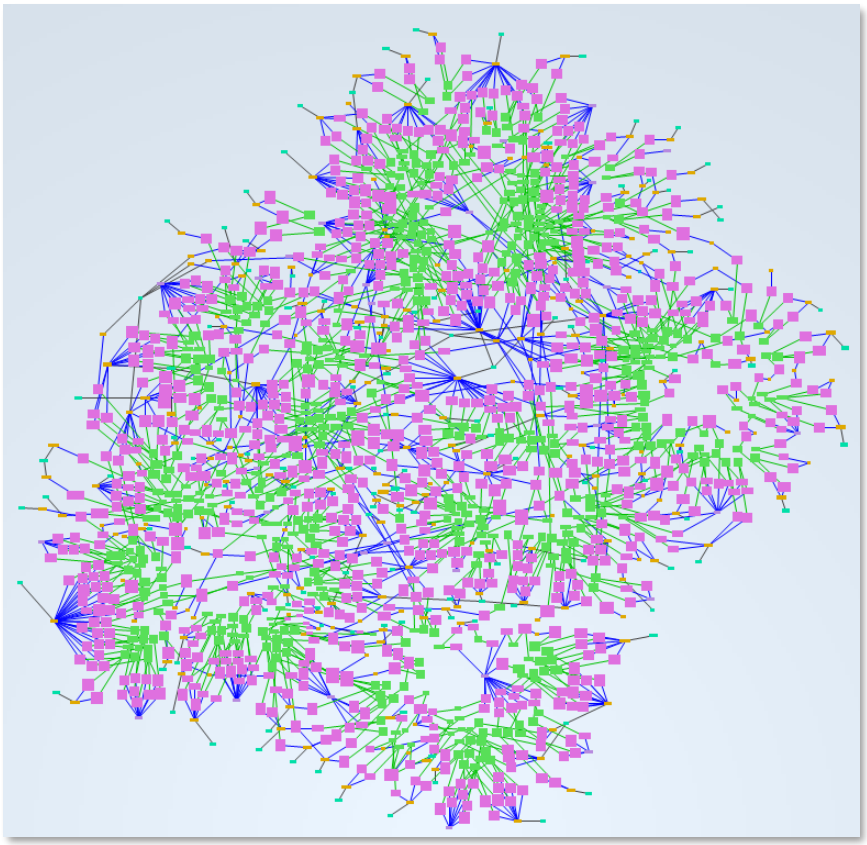


Abbildung 6 Visualiserung von Mittelklasse-Bordnetzdaten als Graph mit Gruff (Franz Inc., 2021)

Abfragesprachen etc. Die Plattformen sind heutzutage so weit entwickelt, dass sie mit sehr großen Datenmengen im Milliardenbereich zurechtkommen (W3C, 2020). Die Knowledge-Graph-Plattformen bieten hohe Flexibilität gegenüber klassischen Datenbanken, wie relationalen oder XML-basierten Datenbanken. Sowohl die Ontologie als auch die Daten sind skalierbar und gleichzeitig lässt sich die Qualität der Ontologie und der Daten durch die

bereitgestellten Werkzeuge leicht verwalten. Daher sind sie sehr gut für Anwendungen mit hoher Dynamik geeignet. Im Gegensatz dazu sind klassische Datenbanksysteme sehr starr, und die Realisierung komplexer Algorithmen auf diesen ist sehr aufwendig.

2.4 Zusammenfassung

In den vorhergehenden Abschnitten sind die grundlegenden Sprachen des Semantic Web, RDF, RDFS, OWL, SPARQL und SHACL, vorgestellt worden. Die wichtigsten Sprachelemente wurden erklärt und eine Übersicht wurde gegeben. Bevor nachfolgend in Kapitel 6 dargestellt wird, wie diese Technologien im Bereich der Bordnetzentwicklung eingesetzt werden können und welche Vorteile sich daraus ergeben, werden zunächst in den folgenden Kapiteln 3 und 4 die Defizite in der Bordnetzentwicklung und die informationstechnischen Anforderungen für die Realisierung normgerechter Zuverlässigkeitsanalysen dargestellt.

3 Stand der Technik

In diesem Abschnitt werden zuerst das physische Bordnetz, dessen Entwicklungsprozess und dabei die Herausforderungen bei der Entwicklung erläutert. Anschließend werden die Standards für die digitalen Datenaustauschformate in der Bordnetzentwicklung erklärt. Zum Schluss werden die Defizite in dem Entwicklungsprozess und in der Datenverarbeitung im Fokus der Digitalisierung der Prozesse dargestellt.

3.1 Das physische Bordnetz

Kraftfahrzeuge waren ursprünglich mit sehr geringen elektrischen Anteilen ausgestattet. Die elektrische Energie diente nur dafür, den Verbrennungsmotor zu starten, und war für die Front-, Heck- und Innenbeleuchtung notwendig. Mit der Zeit kamen stetig neue Energieverbraucher hinzu, von elektrischen Fensterhebern bis hin zu vollelektrischen Fahrzeugen. Seit Anfang der 80er Jahre, ausgehend von elektronisch gesteuerten Motorfunktionen, wurden immer mehr Steuergeräte in das Fahrzeug eingebaut. Beispielsweise enthält ein *Golf 5* je nach Ausstattung bis zu 40 Steuergeräte sowie 140 Aktoren und Sensoren (Gräbel, 2013). Nach dem starken Anstieg der Steuergeräte und Energieverbraucher war es notwendig geworden, sowohl die Energieverteilung als auch die Signalverteilung anzupassen. Deswegen wurden Bussysteme eingeführt, die die Kommunikation der Steuergeräte ermöglichen. Heutzutage besteht somit ein modernes Bordnetz aus der Gesamtheit aller elektrischen und elektronischen Komponenten - zusammengefasst aus Steuergeräten, Bussystemen, Sensoren, Aktuatoren, Energiespeichern, Generatoren und Verkabelung. Darunter liegt ein physisches Bordnetz, auch Kabelbaum genannt, welches die Infrastruktur für die Energie- und Signalverteilung zur Verfügung stellt. Zu den Komponenten eines physischen Bordnetzes gehören u. a. Leitungen, Stromverteiler, Sicherungen, Relais, Stecker, Kontakte, Kabelverbinder, Befestigungsteile, Schutzelemente und Formteile. Ein Mittelklassefahrzeug-Bordnetz beinhaltet ca. 750 verschiedene Leitungen mit einer Länge von insgesamt rund 1500 Metern und tausenden Kontaktstellen. Die Vielfalt und die hohe Anzahl von Komponenten erhöhen die Komplexität und gleichzeitig die Kosten eines Bordnetzes erheblich. Ein weiterer Faktor der Komplexität sind die Varianten, die eine kundenspezifische Fahrzeugkonfiguration ermöglichen und ebenfalls berücksichtigt werden müssen. Neben der Komplexität und den enormen Auswirkungen auf die Kosten und die Qualität eines Fahrzeugs ist die Entwicklung eines physischen Bordnetzes eine große Herausforderung zur Erreichung der Qualitätsziele. Die Entwickler müssen parallel viele Anforderungen bei der Entwicklung berücksichtigen. Nach (Reif, 2014) sind die Hauptanforderungen *„die Dichtheit, die EMV[5]-Kompatibilität, die Temperaturen der*

[5] Elektromagnetischen Verträglichkeit (EMV): "*die Fähigkeit eines Betriebsmittels, in seiner elektromagnetischen Umgebung zufriedenstellend zu arbeiten, ohne dabei selbst elektromagnetische Störungen zu verursachen, die für andere Betriebsmittel in derselben Umgebung unannehmbar wären;*" (EU-Verordnung, 2014)

Bauzonen, der Beschädigungsschutz der Leitungen, die Leitungsauslegung und die Belüftung des Kabelbaums“. In diesem Zusammenhang beschreibt (Reif, 2014) die folgenden wichtigsten Aufgaben von Kabelbaumentwicklern:

- *Dimensionierung der Leitungsquerschnitte,*
- *Werkstoffauswahl,*
- *Auswahl geeigneter Steckverbinder,*
- *Verlegen der Leitungen unter Berücksichtigung von Umgebungstemperatur, Motorbewegungen, Beschleunigungen und EMV-Einfluss,*
- *Betrachtung des Umfelds, in dem der Kabelbaum verlegt wird (Topologie, Montageschritte bei der Fahrzeugherstellung und Vorrichtungen am Montageband).*

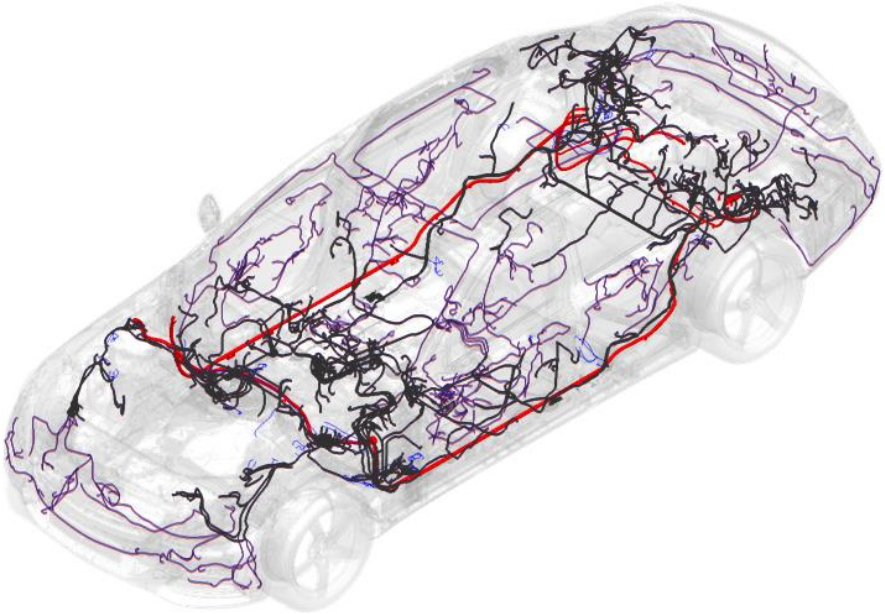


Abbildung 7 Beispielhafte Verteilung verschiedener Leitungen (Neckenich, 2017)

Durch die steigenden Anforderungen wie Elektrifizierung und autonomes Fahren wird das Fahrzeug ein sicherheitskritisches Produkt. Die Entwicklung eines kosten-, gewichts- und platzoptimalen und gleichzeitig zuverlässigen Bordnetzes wird daher auch weiterhin mit immer komplexeren Herausforderungen verbunden bleiben.

3.2 Entwicklungsprozess des physischen Bordnetzes

In der Bordnetzdomäne existieren sehr wenige Arbeiten und Lehrmaterial, welche sich mit der Bordnetzentwicklung im Detail befassen. Daher bildet die Arbeit von Neckenich (Neckenich, 2017) die Hauptquelle für diesen Abschnitt. Darin (Neckenich, 2017) werden die Bordnetzentwicklungsprozesse von Daimler AG und Volkswagen AG ausführlich analysiert und daraus resultierend der folgende allgemeine Entwicklungsprozess dargestellt. Im Entwicklungsprozess des physischen Bordnetzes wird je nach Anforderungen und nach den abgestimmten Funktionalitäten des Fahrzeugs zuerst das System-Design entworfen.

System-Design: In dieser Phase wird die Architektur des Systems entwickelt. Das abstrakte Design zeigt die Vernetzung der Komponenten.

Logik-Design: Basierend auf dem System-Design werden die fahrzeugspezifischen Schaltpläne entworfen. Die elektrische Verschaltung aller Bauteile mit der Detaillierung hinsichtlich Kontaktnummern, Potentialen und Strombelastung der Verbindungen sowie unter Einbeziehung der Topologie des Leitungssatzes aus der Leitungsverlegung werden realisiert.

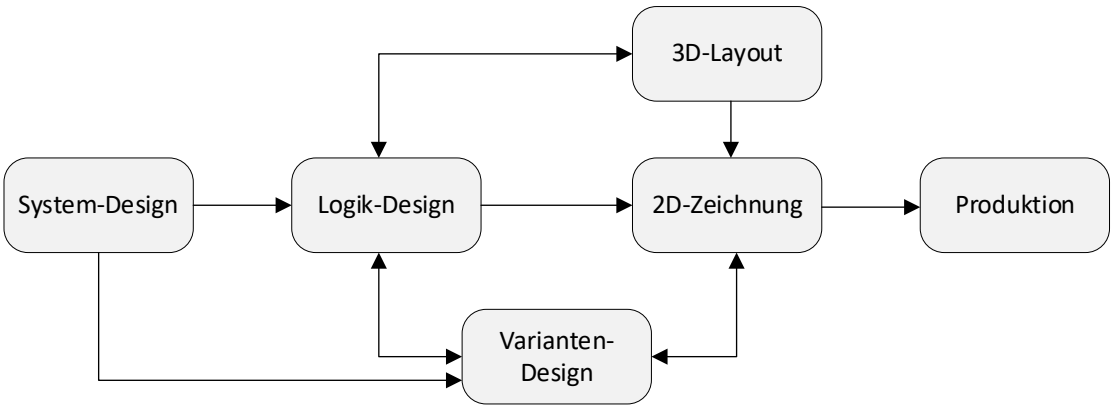


Abbildung 8 Allgemeiner Kabelbaum-Entwicklungsprozess nach (Neckenich, 2017)

3D-Layout: In dieser Phase wird parallel zum Logik-Design das 3D-Modell entworfen. Die Positionen der Komponenten sowie die Verbindungspfade werden festgelegt. Die Überschneidungen und Freiräume für die Leitungen werden sichergestellt.

Varianten-Design: Im Varianten-Design wird das Bordnetz in Module zerlegt. Je nach Fahrzeugkonfigurationen bilden die entsprechenden Modulkombinationen die Varianten. Diese Aufgabe sollte nicht unterschätzt werden, denn selbst bei zehn frei wählbaren Ausstattungsmerkmalen ergeben sich bereits 2^{10} Varianten. Nicht nur die Handhabung einer hohen Variantenvielfalt, sondern auch die optimale montagegerechte Festlegung der Module ist eine komplexe Aufgabe. Daher ist die Bewältigung der Variantenvielfalt in der Entwicklung und auch in der Produktion eine große Herausforderung.

2D-Zeichnung: Im Entwicklungsprozess ist die 2D-Zeichnung die Senke. Alle Informationen der Design-Phasen werden in der 2D-Zeichnung gesammelt. Somit ist die 2D-Zeichnung der Hauptinformationsträger für die Entwicklung des physischen Bordnetzes. In dieser Phase werden die Kontaktgehäuse entworfen und der Leitungssatzaufbau der einzelnen Verbindungen zwischen den Komponenten detailliert festgelegt, wie Befestigungsteile, Schutzelemente, Umwicklung etc. Die zweidimensionale Abwicklung des Leitungssatzes zur Erstellung einer 2D-Zeichnung erleichtert die Auslegung eines Legebrettes, da die physischen Bordnetze in der Produktion auf Legebrettern erstellt werden.

Nach der Erstellung der 2D-Zeichnung und Sicherstellung der Anforderungen wird das physische Bordnetz zur Produktion freigegeben. In der Entwicklung kommen viele

Softwaretools zum Einsatz. Neben der Entwicklung ist die Orchestrierung dieser Softwarelandschaft eine weitere Herausforderung. Um die Kommunikation in dieser Toolkette zu realisieren, sind standardisierte Austauschformate entwickelt worden. Diese werden im nächsten Abschnitt erläutert.

3.3 Digitale Austauschformate für den Entwicklungsprozess

In der physischen Bordnetzentwicklung werden die Daten durch ein standardisiertes und XML-basiertes maschinenlesbares Format für die Toolkette zur Verfügung gestellt. Das Datenmodell hierzu wird von der Projektgruppe *Vehicle Electric Systems Workflow Forum* (VES-WF) unter dem Dach des prostep ivip Vereins entwickelt. Der prostep ivip Verein beschreibt seine Aktivitäten als *"Entwicklung von zukunftsweisenden Lösungsansätzen und Standards für das Produktdatenmanagement und die virtuelle Produktentstehung"* (prostep ivip, 1993). Der Verein bringt Hersteller, Zulieferer, IT-Anbieter und Forscher zusammen, um für die Mitglieder neue IT-Lösungen für die durchgängige Prozess-, System- und Datenintegration in allen Phasen der Produktentstehung zu realisieren. Die Arbeiten der Projektgruppe VES-WF zielen darauf, die methodischen und technischen Grundlagen für eine modellbasierte Bordnetzentwicklung zu schaffen. Die Kabelbaumliste (KBL) und insbesondere der Vehicle Electric Container (VEC) bilden dabei die Grundlage als digitales Produktmodell für den gesamten Produktlebenszyklus des Bordnetzes (VES-WF, 2019). Die Arbeiten wurden im Jahr 2000 mit dem Motto „*Wir wollen Kabelbaumdaten vollständig, zu 100% elektronisch auswertbar, standardisiert und offen austauschen*“ angefangen und werden bis heute kontinuierlich weiterentwickelt (Becker, et al., 2013). Nach fünfjähriger Zusammenarbeit von Automobilherstellern und Kabelbaum-Konfektionären wurde der erste Standard KBL im Jahr 2005 als Empfehlung VDA 4964 (Project Group Car Electric, 2005) vom Verband der Automobilindustrie (VDA) veröffentlicht (Becker, et al., 2013). KBL wird seit vielen Jahren von deutschen OEMs in der Entwicklung eingesetzt und besonders beim Datenaustausch zwischen OEM und Zulieferer verwendet. Die langjährigen Erfahrungen aus der Praxis des Formats KBL haben zu einer Weiterentwicklung des Standards in das neue Format VEC geführt. Im Gegensatz zu KBL soll VEC nicht nur zum Datenaustausch für den Kabelbaum dienen, sondern ein vollständig digitales Produktmodell des gesamten Bordnetzes realisieren. Zuletzt ist die Version 1.2 von VEC im Jahr 2020 als Empfehlung VDA 4968 veröffentlicht worden (prostep ivip, 2020). VEC wurde in den letzten Jahren kontinuierlich weiterentwickelt und hat sich so weit etabliert, dass es heute die meisten Anforderungen erfüllt und im Einsatz ist (VES-WF, 2019). Diese Arbeit konzentriert sich speziell auf den neuen Standard VEC, der Vorgängerstandard KBL wird in dieser Arbeit nicht näher betrachtet. Der Lösungsansatz in Kapitel 5 wird auf Basis von VEC entwickelt. In der prototypischen Realisierung wurde jedoch KBL verwendet, daher wird es in Kapitel 6 kurz erläutert.

Vehicle Electric Container

VEC als digitales Produktmodell dient einerseits als digitale Produktdokumentation und wird anderseits in verschiedenen Systemen für verschiedene Aufgaben und Analysen wie EMV-Analyse, Energie-Simulation, Kostenoptimierung, Gewichtskalkulation etc. als Grundlage verwendet. Daher müssen die benötigten Informationen vollständig enthalten sein, um die Analysen durführen zu können. VEC enthält als Basis die folgenden Grundinformationen (Becker, et al., 2013):

- *Stückliste und Komponentendaten,*
- *Verbindungsinformationen,*
- *Geometrie- und Topologie-Informationen,*
- *Elektrologische Verschaltung der Komponenten.*

Diese aufgeführten Basisinformationen wurden bereits bei KBL modelliert. Der Nachteil bei KBL war, dass diese Informationen in separaten Modellen modelliert wurden. In VEC wurde diese diskrete Basis zu einer flexiblen und durchgängigen Struktur ausgebaut und darauf aufbauend weiterentwickelt.

Das Datenmodell zu VEC wurde mit der grafischen Modellierungssprache UML entwickelt. Das Datenmodell ist aufgrund des hohen Detaillierungsgrades der Beschreibung, der für die Prozesse erforderlich ist, recht komplex. Es besteht aus ca. 250 Klassen mit ca. 300 Beziehungen zwischen diesen Klassen sowie ca. 200 Datenfeldern. Die Struktur und die Benennung der Klassen und Beziehungen bilden zusammen quasi die Sprache der Domäne „Bordnetz“ und sind den langjährigen und intensiven Arbeiten der Projektgruppe zu verdanken. Im weiteren Vorgehen werden aus dem UML-Modell ein XML-Schema generiert, die Daten nach diesem Schema erfasst und als XML-Datei ausgetauscht.

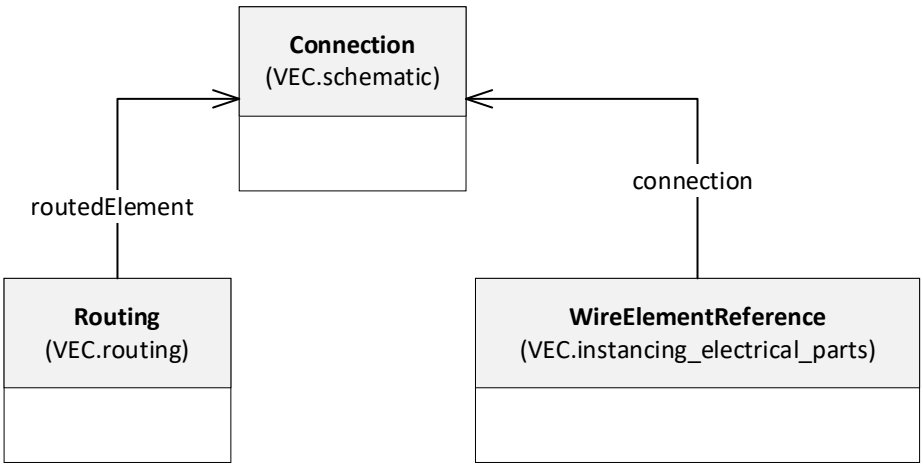


Abbildung 9 Ausschnitt aus Modulen des VEC (prostep ivip, 2020)

VEC ist modular aufgebaut. Jede Klasse gehört zu einem Modul. Die Module beschreiben eine bestimmte Unterdomäne. Zum Beispiel beschreibt die Klasse *Connection* aus dem

Schaltplan-Modul (*schematic*) die Beziehung zwischen den Bordnetzkomponenten. Diese Beschreibung kann durch die Klasse *Routing* erweitert werden, die die Verlegungswege in der Topologie beschreibt, oder durch die Klasse *WireElementReference*, die die *Connection* mit konkreten Kabelinformationen erweitert. Auf solche Weise kann ein Bordnetz in VEC in unterschiedlicher Detaillierung, von sehr abstrakt bis zu sehr detailliert, dargestellt werden. Diese grundlegende Eigenschaft von VEC ermöglicht einerseits flexiblen Datenaustausch und andererseits unterschiedliche Perspektiven auf das Bordnetz. So können beispielsweise verschiedene Domänenexperten in einem gemeinsamen Modell sowohl an geometrischen als auch an elektrologischen Eigenschaften des Bordnetzes arbeiten.

3.4 Defizite

Die Bewältigung der Komplexität im Bordnetzentwicklungsprozess und die digitale Datenverarbeitung im Prozess ist eine große Herausforderung. In den letzten Jahren ist dies zu einem wichtigen Thema der Domäne geworden, und es wurden einige Lösungsansätze entwickelt. Die steigenden Herausforderungen und die digitale Transformation werden die Domäne mit dieser Problematik noch eine Weile beschäftigen, bis ein reibungsloser Prozess erreicht wird. In diesem Abschnitt werden die Ursachen der Komplexität näher betrachtet und die daraus resultierenden Defizite klargestellt. Dieser Abschnitt ist in zwei Unterabschnitte aufgeteilt. Im ersten Teil werden die Defizite im Entwicklungsprozess, im zweiten Teil werden die Defizite in der Datenverarbeitung vorgestellt. Der vorgeschlagene Lösungsansatz zur Bewältigung der Defizite wird dann im Kapitel 5 konkretisiert und im Vergleich mit anderen Lösungsansätzen behandelt.

3.4.1 Defizite im Entwicklungsprozess

In diesem Abschnitt werden die durch die Vorgehensweise und aufgrund steigender Komplexität in der Entwicklung entstehenden Defizite näher betrachtet.

3.4.1.1 Dualität in der Entwicklung

Das physische Bordnetz entsteht durch einen dualen Prozess, in dem zwei Fachdisziplinen zusammenarbeiten, wobei die eine die Elektrologik, die andere die Topologie entwickelt. Die Experten arbeiten jeweils mit ihrer eigenen Sicht auf das Produkt und mit eigenen Softwaretools. Daraus entstehen zwangsläufig Datensilos in der Entwicklung, wie der US-amerikanischen Informatiker Melvin Edward Conway es formuliert und dies auch als das Gesetz von Conway bekannt ist: *„that organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations“* (Conway, 1968). Die Dualität in der Entwicklung führt zu Diskontinuität in Prozess und Daten. Diese Problematik ist wohl die bekannteste in der Bordnetzentwicklung. In einer ausführlichen Fallstudie (Kuhn, et al., 2019), an der 54 Experten von 26 Unternehmen aus OEM und der Bordnetzdomäne teilgenommen haben, ist die Diskontinuität in Prozess und

Daten als Hautproblem bezeichnet worden. Die Diskontinuität führt zu inkonsistenten Zuständen und verlangsamt den Gesamtprozess erheblich.

3.4.1.2 Linearität in der Entwicklung

Die dual laufenden Entwicklungsstränge führen den gesamten Bordnetzentwicklungsprozess zu einem linearen Prozess. Die Prozessschritte werden wie bei einem Wasserfallprozess nacheinander durchgeführt. Am Ende des Entwicklungsprozesses werden die Informationen in der 2D-Zeichnung (Abb. 8) gesammelt und weitere Aktionen werden auf der 2D-Zeichnung durchgeführt. (Neckenich, 2017) beschreibt, dass „*die Änderungen hauptsächlich in den Zeichnungen dokumentiert werden, was zu einer Diskrepanz der Daten zwischen 2D- und 3D-Design führt“*. Gleichzeitig betont er, dass „*in der Regel keine automatischen Aktualisierungen und Ableitungen zwischen den beiden Dokumentationssystemen stattfinden, so dass die manuelle Dokumentation auf beiden Seiten zu einem hohen Pflegeaufwand und einer entsprechenden Fehleranfälligkeit führt“*. Ein iterativer Prozess im Gesamtprozess ist in Bezug auf die Arbeitsstruktur und die Infrastruktur nicht möglich. Die Rückverfolgbarkeit wird in den Systemen nicht vollständig unterstützt. Die neu hinzukommenden Anforderungen werden getaktet in die nächsten Zeichnungsversionen eingepflegt.

3.4.1.3 Änderungsmanagement

Die Entwicklung ist ein aktiver Prozess und Änderungen sind ein Teil davon. Ein Bordnetz wird in einem OEM verteilt, sowohl von internen als auch externen Mitarbeitern, entwickelt. Es ist daher normal, dass während der Entwicklung viele Änderungen vorgenommen werden müssen. Dabei unterliegt laut der Studie von Kuhn und Nguyen (Kuhn, et al., 2019) die Entwicklung einer massiven Änderungsdynamik mit ca. 1000 Änderungen/Monat, und die Ingenieure verbringen ca. 80% ihrer Arbeitszeit mit der Implementierung der Änderungen. Das zeigt wiederum, wie die verteilte Struktur des Entwicklungsprozesses das Änderungsmanagement enorm herausfordert. Eine Minimierung des Änderungsaufwandes auf Null ist ein unrealistisches Ziel. Daher ist es im Änderungsmanagement notwendig, dass Änderungen rückverfolgbar sind und dass nach einer Änderung die möglichen Auswirkungen angezeigt werden, damit die Änderungen strukturiert erfasst und schneller bearbeitet werden können. In den meisten Fällen werden die Änderungen durch andere Abteilungen ausgelöst, z. B. Änderungen an der Karosserie erfordern Änderungen am Bordnetz, so dass ein integriertes Änderungsmanagement erforderlich ist. Um solche Anforderungen realisieren zu können sind neue digitale Methoden in der Entwicklung notwendig.

3.4.1.4 Funktionsorientierte Vorgehensweise

Die funktionsorientierte Vorgehensweise ist ein Konzept aus der Entwicklung der Datenmodellierung und beschreibt nach (Hellmuth, 1994) einen Ansatz, in dem zuerst die für eine Anwendung relevanten Funktionen definiert werden. Darauf aufbauend werden dann in einem nächsten Schritt die für die definierten Funktionen benötigten Datenmodelle entworfen. Diese Datenmodelle sind dabei jeweils aus dem Blickwinkel der

Anwendungsfunktionen definiert. Diese Vorgehensweise führt dazu, dass mehrere Schnittstellen implementiert werden müssen, um die Interoperabilität der Anwendungen zu gewährleisten. In meisten Fällen stehen am Ende trotz der Schnittstellen die Daten in unterschiedlichen Versionen und die Ergebnisse der Anwendungssysteme nebeneinander. Daher müssen am Ende eine aufwändige Integration und Aktualisierungsarbeit durchgeführt werden. In der Bordnetzentwicklung ist die funktionsorientierte Vorgehensweise zu beobachten (Abb. 10). Die Modelle KBL und VEC erledigen teilweise die Interoperabilitäts-aufgabe bzw. die Orchestrierung der Softwarelandschaft, aber eine 100%ige Interoperabilität ist mit dieser Vorgehensweise nicht möglich, da die Aufrechterhaltung der vielen Schnittstellen in einem stabilen Zustand mit hohem Aufwand und Kosten verbunden ist.

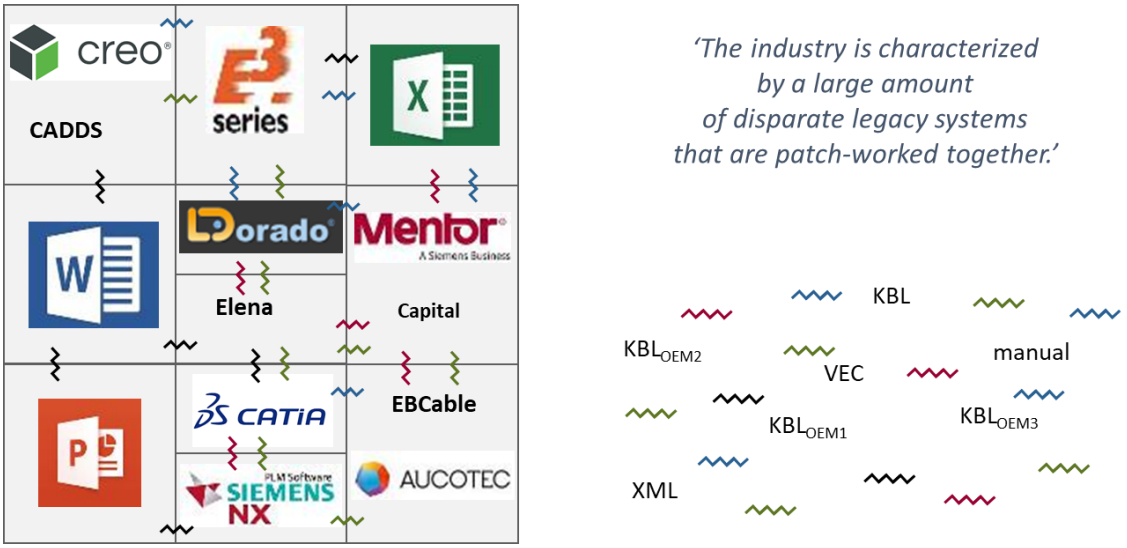


Abbildung 10 Digitale Technologien in der Bordnetzentwicklung (Kuhn, et al., 2019)

Im Gegensatz zur funktionsorientierten Vorgehensweise wird in der datenorientierten Vorgehensweise - nachdem zuerst die Daten anwendungsunabhängig analysiert wurden und je nach relevanten Funktionen - auf ein umfassendes Datenmodell abgezielt. Bei diesem

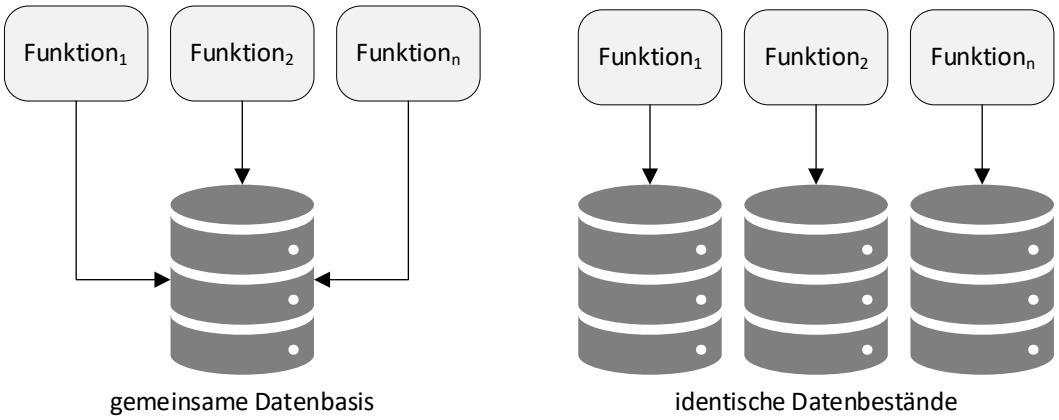


Abbildung 11 Datenorientierte - versus Funktionsorientierte-Vorgehensweise nach (Wikipedia, 2018)

Ansatz greifen die Funktionen bzw. Anwendungssysteme auf eine gemeinsame Datenbank zu und führen Aktionen darauf aus (Hellmuth, 1994). So werden Redundanzen vermieden und die Daten synchron gehalten. Auf einer gemeinsamen Plattform sind Änderungen jeweils direkt für die anderen Anwendungssysteme sichtbar und können somit in abgesicherter Weise verarbeitet werden. Auf dieser Basis kann der Entwicklungsprozess kollaborativ und iterativ gestaltet werden.

3.4.2 Defizite in der Datenverarbeitung

Die Komplexität des physischen Bordnetzes und die Komplexität des Entwicklungsprozesses spiegeln sich auch auf der Datenebene. Um das Bordnetz und die im Entwicklungsprozess benötigten Informationen zu beschreiben, enthält das Datenmodell von VEC ca. 300 Klassen und ca. 400 Klassenrelationen. Diese komplexe Struktur ist eine große Herausforderung bei der Handhabung und Verarbeitung der Daten. In diesem Abschnitt werden die Defizite in der Datenverarbeitung im Hinblick auf die Komplexitätsbewältigung näher betrachtet. Insbesondere wird der XML-basierte Ansatz mit Fokus auf Interoperabilität diskutiert, und es wird gezeigt, dass er für viele Probleme nicht ausreichend ist.

3.4.2.1 Notwendigkeit einer höheren Beschreibungssprache für semantische Interoperabilität

Das Datenmodell VEC wird mit der Modellierungssprache UML entwickelt (Abb. 12). UML ist eine objektorientierte Modellierungssprache, die ermöglicht, eine Domäne mit Klassen und Relationen visuell zu beschreiben. Dank ihrer Diagrammstruktur ist die Sprache sehr verständlich und leicht nachvollziehbar. Daher wird UML in vielen Projekten und auch bei der Entwicklung von VEC eingesetzt, um ein gemeinsames Verständnis und eine gemeinsame Sprache zwischen den Stakeholdern zu erreichen und abzusichern.

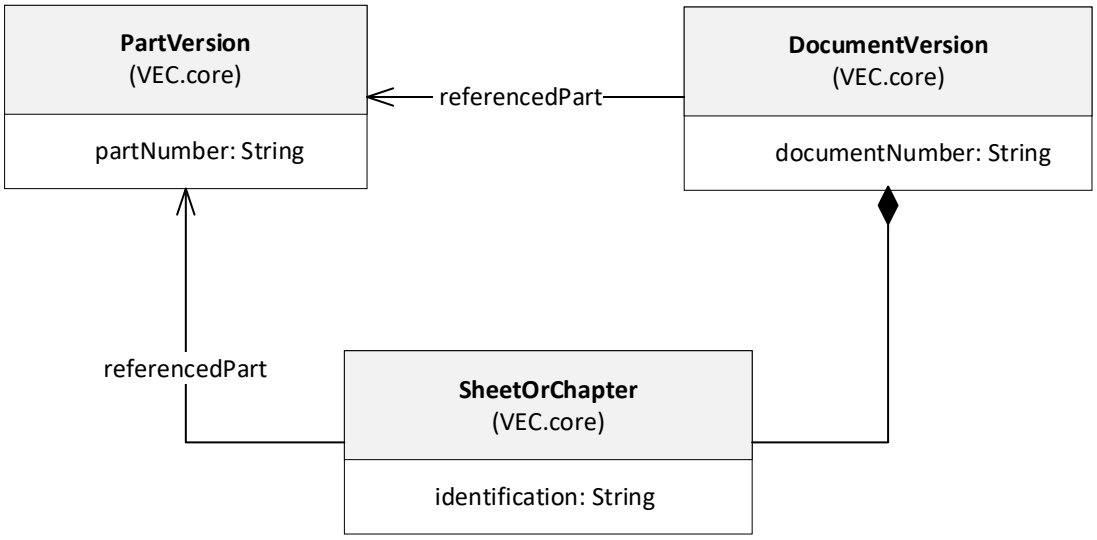


Abbildung 12 Abschnitt aus VEC/UML (prostep ivip, 2020)

In der praktischen Anwendung von VEC wird in einem zweiten Schritt aus dem UML-Datenmodell ein XML-Schema generiert. XML (Extensible Markup Language) ist eine

erweiterbare Auszeichnungssprache, und XML-Technologien bieten die Möglichkeit, Daten strukturiert auszutauschen. XML-Schema definiert die Struktur von XML-Dokumenten, und die XML-Dokumente enthalten die konkreten Daten. XML-Dokumente haben eine hierarchische Struktur. Bei der Generierung des XML-Schemas aus dem UML-Datenmodell von VEC werden die UML-Klassen in das XML-Schema-Element `<complexType>` und die UML-Klasseneigenschaften in das XML-Schema-Element `<element>` transformiert. So wird zum Beispiel die UML-Klasse `PartVersion` im XML-Schema als `<complexType name="PartVersion">` definiert. Die Instanzen der UML-Klasse erscheinen in einem XML-Dokument als hierarchisch strukturierte Daten in XML-Elementen.

```
<!--XML Schema-->
<complexType name="PartVersion">
   <sequence>
      <element name="partNumber" type="xs:string"/>
      ...
   </sequence>
</complexType>

<!--XML Dokument-->
<PartVersion>
   <partNumber>pn_1</partNumber>
   ...
</PartVersion>
```

Quelltext 4 VEC Daten in XML

Bei dieser Transformation von UML in XML ist zu beachten, dass die Semantik verloren geht. Die Semantik, dass `<complexType name="PartVersion">` eine Klasse beschreibt und `<PartVersion>` Elemente in der XML-Datei die Instanzen dieser Klasse bilden, ist bei der Verarbeitung des XML-Dokuments nicht direkt von einer Maschine interpretierbar. XML-Dokumente ermöglichen strukturierten Datenaustausch, und die XML-Strukturen müssen an den jeweiligen Schnittstellen definiert werden, damit die Maschinen die Inhalte interpretieren können. Im folgenden Quelltext ist Java-Code zu sehen, der durch eine Programmschnittstelle namens JAXB (*Java Architecture for XML Binding*) für die Maschine die Inhalte der XML-Dokumente definiert. Nach dieser Definition werden ausgehend vom XML-Schema Java-Klassen generiert. Dieser Prozess wird XML-Datenbindung genannt. Auf diese Art und Weise können Softwaresysteme Klassen und Objekte in XML-Dokumenten erkennen,

```
@XmlType
public class PartVersion implements Serializable
{
    @XmlElement
    public String partNumber;
}
```

Quelltext 5 VEC/XML-Datenbindung nach (Becker, 2018)

serialisieren und verarbeiten. Bei diesem Ansatz wird der Datenaustausch auf der Grundlage der hierarchischen Syntax realisiert. Zur Aufrechterhaltung der Interoperabilität der Systeme müssen daher die Schnittstellen der Entwicklungswerkzeuge bei jeder Änderung angepasst werden. Dies ist auch eine der Ursachen für die Diskrepanz von Daten und Software in der Bordnetzentwicklung. Dagegen können bei semantischer Interoperabilität Werkzeuge enger gekoppelt werden und Daten ohne semantischen Verlust ausgetauscht werden. Dafür muss die Semantik der Daten ebenfalls explizit und formal ausgetauscht werden. Um dies zu realisieren, ist eine höhere Beschreibungssprache, wie sie z. B. die vorgestellten Ontologien bieten, notwendig, um die Klassen und Beziehungen aus dem VEC-Datenmodell eins zu eins abbilden und damit auch die Bedeutung von Daten mit transformieren zu können.

3.4.2.2 Komplexes Datenmodell

Das VEC-Datenmodell mit seinen vielschichtigen Referenzierungsstrukturen (Abb. 13) und das durch das Datenmodell beschriebene Bordnetz haben eine komplexe Netzwerkstruktur.

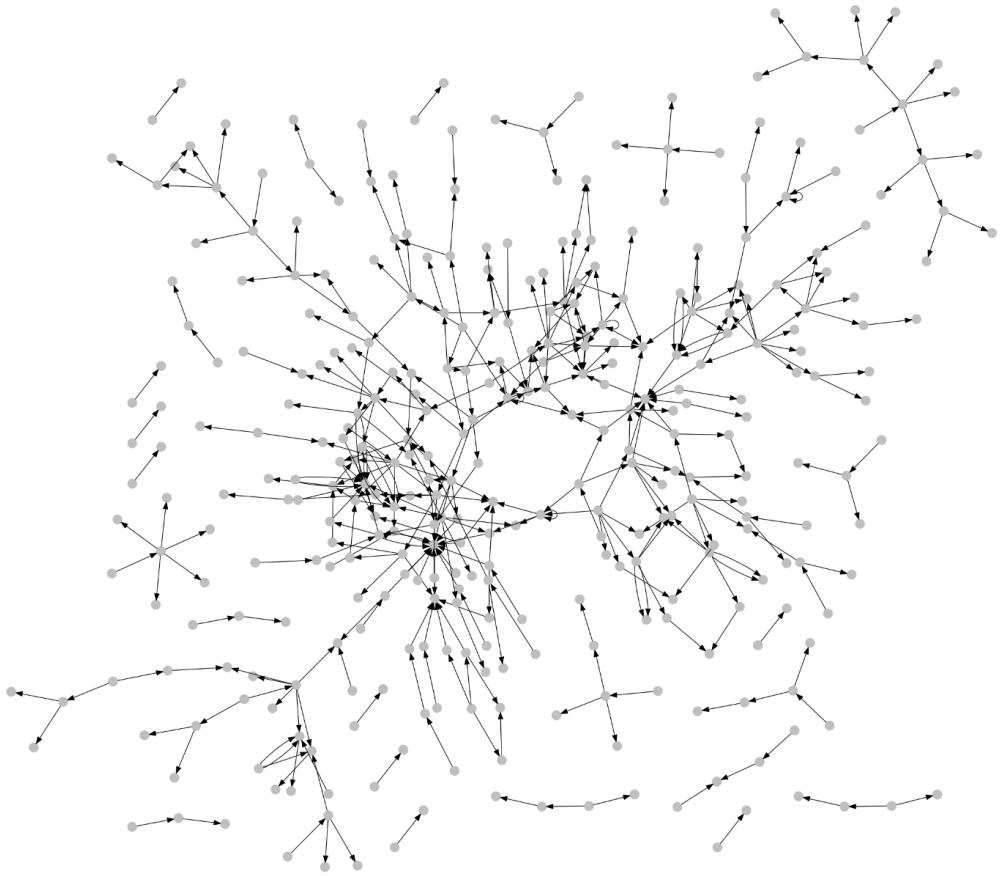


Abbildung 13 Assoziationen im VEC-Datenmodell

Die Objekte werden in den XML-Dokumenten jedoch in hierarchischer Struktur abgelegt, und eine netzwerkartige Darstellung auf der Datenebene ist nicht gewährleistet. Zwar gibt es Sprachen wie GraphML um Graphen mittels XML bereitzustellen (GraphML Working Group, 2019). Diese werden jedoch in erster Linie für Visualisierungswerkzeuge entwickelt, und obwohl ihre Graphenstruktur vorteilhaft ist, muss deren Semantik wiederum an den

Schnittstellen definiert werden. Beziehungen von Objekten werden in VEC/XML durch Zuordnung referenzierender IDs realisiert. Daher ist es die Aufgabe von Bordnetz-entwicklungswerkzeugen diese Referenzierungsstruktur zu generieren (Abb. 14).

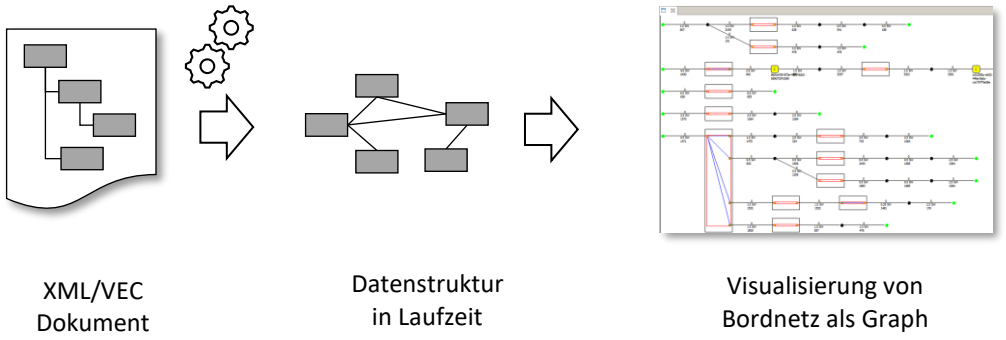


Abbildung 14 Generierung der Graphenstruktur aus VEC/XML in Laufzeit

Die Entwicklung eines solchen Bordnetzentwicklungswerkzeugs, z. B. für Visualisierung, Analyse etc., ist mit hohem Implementierungsaufwand verbunden. Da jedes Werkzeug sein natives Format hat, müssen in den meisten Fällen zusätzlich spezielle Anpassungen am Datenmodell vorgenommen werden. Wegen der hohen Komplexität des Datenmodells sind Implementierungs- und Wartungsaufwand dementsprechend hoch. Daher muss die notwendige Beschreibungssprache neben der Semantik auch die Graphenstruktur von VEC unterstützen.

3.4.2.3 Datenabfrage

In der Bordnetzentwicklung müssen ständig bestimmte Daten wie Leitungsstränge, Merkmale von Komponenten etc. bearbeitet oder in Analysen verwendet werden, weshalb die Entwicklungswerkzeuge die angeforderten Daten flexibel bereitstellen können müssen. Hierzu sind Schnittstellen und eine Abfragesprache erforderlich, die in der Lage sein muss, Abfragen wie „*Querschnitte aller ausgehenden Leitungen einer Komponente ausgeben*“

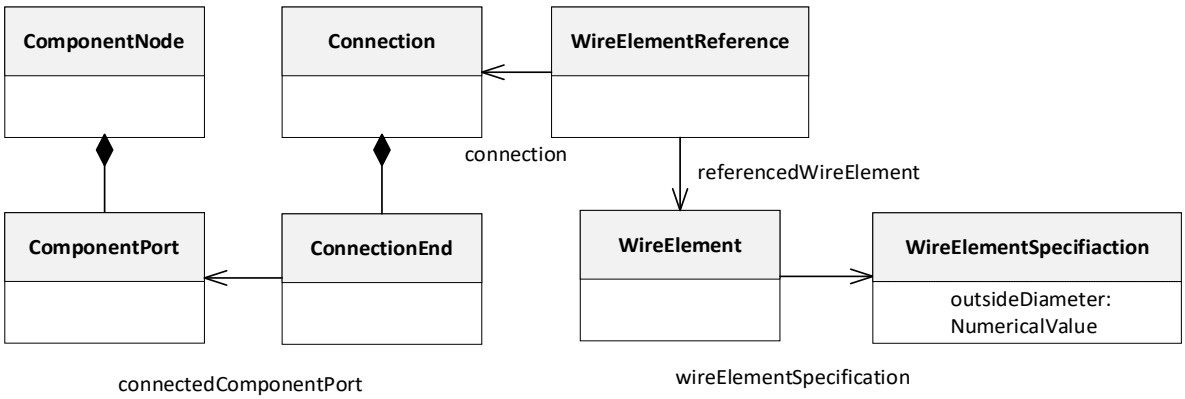


Abbildung 15 Der Pfad zwischen *ComponentNode* und *WireElementSpecificetion* (prostep ivip, 2020)

realisieren zu können. In einem XML-Dokument können Daten durch XML-Abfragesprachen wie XQuery oder XPath abgefragt werden. Dabei liegt die Schwierigkeit darin, komplexe Muster in einer Abfrage abzubilden. Um die oben genannte Abfrage zu realisieren, muss der Pfad von ComponentNode aus über fünf Klassen bis hin zur Klasse WireElementSpezification definiert werden (Abb. 15). Auch wenn solche Abfragen mit XML-Technologien möglich sind, werden selbst einfache Abfragen aufgrund der komplexen Zusammenhänge in VEC schnell unübersichtlich. Eine weitere Lösung für dieses Problem in der Bordnetzentwicklung bietet das Bordnetz-Design-Werkzeug LDorado, in dem durch die Einführung von Makros derartige Abfragen realisiert werden können. Makros sind Programmierschnittstellen bzw. Unterprogramme, die dem Benutzer ermöglichen, eine Software an zugelassenen Stellen durch eigenen Code zu erweitern. Mit diesen zusätzlichen Werkzeugen können komplexe Abfragen programmiert werden (Quelltext 6).

```
//Travesiert alle Connector
foreach( Connector curConnector in
                   Document.GetEntitiesOfType(typeOf(Connector))){

 //Überprüft ob der Connector ein Splice ist
 if(string.Compare(curConnector.GetUserAttribute("Usage","",false),"splice")){

  //Hinzufügen Splice zur Ergebnismenge
  ResultSet.Add(curConnector);
 }
}
```

Quelltext 6 Makros bei LDorado Vestigo (COMSA, 2018)

3.4.2.4 Datenvalidierung und Variantenmanagement

Die diskrete Struktur sowie die massive Änderungshäufigkeit sind innerhalb des Entwicklungsprozesses die Hauptursachen, dass in den VEC/XML-Dokumenten Inkonsistenzen auftauchen, wie z. B. fehlende Leitungen. Inkonsistente Dokumente führen zu falschen Ergebnissen bei den Analysen und Auswertungen des Bordnetzes. Daher sind eine Konsistenzprüfung bzw. eine Datenqualitätsprüfung notwendig. Zwar werden die VEC-Dokumente nach der Generierung durch das VEC-Schema validiert. Diese schemabasierte Validierung ist jedoch eine rein syntaktische Überprüfung. Daher ist es möglich, dass ein VEC/XML-Dokument mit fehlenden Daten z. B. fehlenden Leitungen, trotzdem valide sein kann. Eine semantische Überprüfung, z. B. ob alle Komponenten zueinander korrekt und vollständig verbunden sind, ist mittels XML-Schema-Validierung ohne zusätzliche Algorithmen nicht möglich. Infolgedessen müssen derartige Überprüfungen durch spezielle Funktionalitäten innerhalb der Entwicklungswerkzeuge durchgeführt werden. Solche standardisierten Anwendungen sind in der Domäne nicht vorhanden. Daher ist auch hier ein strukturiertes Vorgehen notwendig, um möglichst fehlerfreie und effiziente Interoperabilität in der Entwicklung zu erreichen.

Die Datenvalidierung betrifft notwendig auch die Bordnetzkonfigurationen. Hier muss z. B. geprüft werden, ob die gewählte Konfiguration eine zulässige Kombination ist. Ein Bordnetz wird verteilt von Teams in Modulen entwickelt, z. B. Motorraum, Tür-links/rechts, Heck etc. Als Resultat werden viele einzelne Dokumente zu den Modulen und deren Versionen produziert. Eine zulässige Kombination von Modulen bildet eine Konfiguration bzw. eine Variante. In VEC existieren Datenfelder, die als Zeichenketten definiert sind, für die Beschreibung der Varianten, eine formale Beschreibung und Überprüfung, wie *„wenn Leitungsstrang A ausgewählt ist, dann soll auch Leitungsstrang B ausgewählt werden“*, ist mit XML-Schema-Validierung aber nicht möglich, da die Sprache nicht hinreichend ausdruckstark ist.

3.5 Zusammenfassung

In diesem Kapitel wurde der Bordnetzentwicklungsprozess erläutert und erklärt, wie die Standards KBL und VEC zum Einsatz kommen. Dabei wurde gezeigt, dass die duale und lineare Prozessstruktur, die funktionsorientierte Vorgehensweise und die hohe Änderungsdynamik der Domäne zu inkonsistenten Daten und einer Diskrepanz der Anwendungen führt. Damit wird durch fehlende Interoperabilität in der Softwarelandschaft kollaboratives und iteratives Arbeiten in der Entwicklung verhindert. Neben den Defiziten im Entwicklungsprozess wurden auch Defizite in der Datenverarbeitung vorgestellt. Die XML-basierten Standards ermöglichen zwar den Datenaustausch zwischen Bordnetzentwicklungswerkzeugen, aber nur auf einer syntaktischen Ebene und dies reicht nicht aus, solide Interoperabilität zu realisieren. Zugleich können viele Basisaufgaben, wie Datenabfragen, Sicherung der Datenqualität, Variantenmanagement etc., auf Datenebene wegen der Komplexität des Datenmodells mittels XML-Technologien nicht gewährleistet werden. Bevor in Kapitel 5 ein Lösungsansatz für die dargestellten Defizite vorgestellt wird, werden im folgenden Kapitel 4 zunächst die informationstechnischen Anforderungen aus Sicht der Zuverlässigkeitsanalyse betrachtet.

4 Informationstechnische Anforderungen für die Realisierung normgerechter Zuverlässigkeitsanalyse

In dem vorherigen Abschnitt *Stand der Technik* wurde erklärt, wie ein physisches Bordnetz in der Entwicklung mit den digitalen Technologien bearbeitet wird. Es wurde dabei dargelegt, dass die Defizite im Entwicklungsprozess und in der Datenverarbeitung die effektive Umsetzung eines digitalen Bordnetzes beeinträchtigen. Neben der Herausforderung der Interoperabilität und Datenkonsistenz in der Entwicklung ist die digitale Absicherung der Zuverlässigkeit des Bordnetzes eine weitere Herausforderung. Aufgrund der Komplexität heutiger Bordnetze und der verkürzten Time-to-Market wird versucht, so wenig Prototypen wie möglich zu entwickeln und in naher Zukunft nur noch digital zu entwickeln. Die digitale Absicherung der Zuverlässigkeit ist daher ein wichtiges Thema, und die neuen Herausforderungen wie die Elektrifizierung des Fahrzeuges und autonomes Fahren werden das Thema noch einige Zeit aktuell halten, da das Fahrzeug zu einem zunehmend sicherheitskritischen Produkt wird. Die Zuverlässigkeitssicherung eines Fahrzeugs wird nach Normen realisiert. In der Automobilindustrie ist die entsprechende Norm zur funktionalen Sicherheit ISO 26262 „*Road Vehicles – Functional Safety*“ (ISO 26262, 2018). Die ISO-Norm beschreibt die Methoden und Aktivitäten, die während des Entwicklungsprozesses von sicherheitsrelevanten Systemen in Fahrzeugen durchgeführt werden müssen. Die Norm ist aktuell noch nicht vollständig in die Bordnetzentwicklung integriert. In diesem Abschnitt wird daher betrachtet, welche informationstechnischen Anforderungen eine effektive Integration der Norm ISO 26262 für die Bordnetzentwicklung mit sich bringt, um die Zuverlässigkeitsanalysen normgerecht realisieren zu können. Ohne Berücksichtigung dieser Herausforderungen wäre eine Lösung zur Beseitigung der bestehenden Defizite bei der Bordnetzentwicklung unvollständig. Zu Beginn dieses Kapitels werden die Grundlagen zu Zuverlässigkeit und funktionaler Sicherheit erläutert und anschließend die informationstechnischen Anforderungen dargestellt.

4.1 Zuverlässigkeitsanalyse

Nach DIN 40041 ist Zuverlässigkeit *„die Beschaffenheit bezüglich der Eignung, während oder nach vorgegebenen Zeitspannen bei vorgegebenen Arbeitsbedingungen die Zuverlässigkeitsanforderungen zu erfüllen“* (DIN 40041, 1990). Die Bewertung der Zuverlässigkeit hängt daher von der genauen Definition von Zuverlässigkeitsanforderungen an die Komponenten eines Produkts ab, die die Funktionalität des Produkts für eine bestimmte Betriebszeit und unter bestimmten Bedingungen realisieren. Für das Bordnetz bedeutet dies, dass es unter der typischen Lebensdauerannahme von 300.000 km / 15 Jahre / 8.000 Betriebsstunden eines Fahrzeugs (Hauck, et al., 2019) und unter festgestellten Betriebsbedingungen funktionsfähig bleiben soll. Die Erfüllung und der Nachweis dieser Anforderungen sind nicht trivial. Dafür müssen alle möglichen Fehlerursachen betrachtet und

eine Prognose über die Lebensdauer des Bordnetzes erstellt werden. Die Zuverlässigkeitstechnik stellt quantitative und qualitative Methoden bereit, um solche Prognosen zu treffen und durch systematische Untersuchungen Produktfehler zu beseitigen. Die folgende zwei Abschnitte geben einen Überblick über die quantitativen und qualitativen Methoden.

4.1.1 Quantitative Methoden

Die quantitativen Methoden in der Zuverlässigkeitstechnik liefern quantitative Werte, die die Zuverlässigkeit der Betrachtungseinheit beschreiben. In diesem Abschnitt wird die Ausfallratenanalyse als die wichtigste quantitative Methode betrachtet. In der Ausfallraten-analyse wird die Zuverlässigkeit der Betrachtungseinheit bzw. des Produkts mit der Ausfallrate λ(t) zum Zeitpunkt t beschrieben. Die Ausfallrate eines Produktes ist abhängig von der Ausfallrate seiner einzelnen Bauteile, wie z. B. Kontaktstelle, Stecker etc., für das Bordnetz. Die Ausfallrate für ein Bauteil kann in vier verschiedenen Weisen bestimmt werden: (1) Statistische Auswertung der Felddaten, falls diese strukturiert erfasst sind; (2) Experimentelle Untersuchung, indem eine große Menge von Bauteilen unter beschleunigten Arbeitsbedingungen getestet und ihr Ausfallverhalten beobachtet wird; (3) Bestimmung durch physikalische Modelle (Physics of failure); (4) Bestimmung durch Ausfallraten-kataloge, falls Daten für das Bauteil existieren. Wobei die Inhalte der Kataloge durch (1) - (3) generiert werden. Nach Häufigkeit der Ausfälle kann das Ausfallverhalten durch eine Wahrscheinlichkeitsdichtefunktion beschrieben werden (Abb. 16). „*Die Ausfallrate λ zu einem Zeitpunkt t_x lässt sich interpretieren als ein Maß für das Risiko eines Teiles auszufallen, unter der Voraussetzung, dass es bereits bis zu diesem Zeitpunkt t_x überlebt hat, und ergibt sich als Quotient aus Dichtefunktion $f(t_x)$ und Überlebenswahrscheinlichkeit $R(t_x)$.*“ (Betsche, et al., 2004).

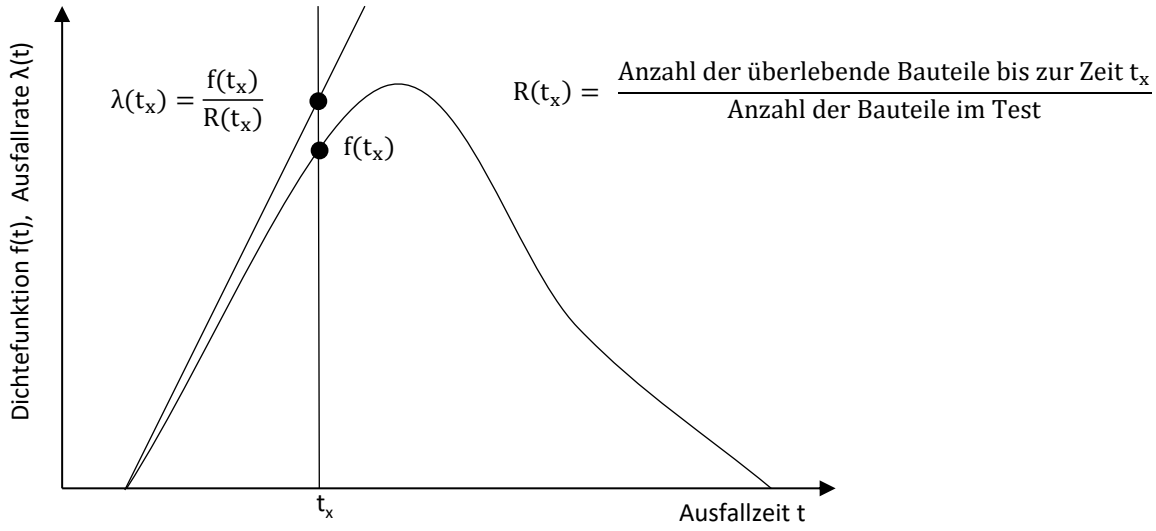


Abbildung 16 Ermittlung der Ausfallrate aus Dichtefunktion und Überlebenswahrscheinlichkeit (Betsche, et al., 2004)

Der Ausfall eines Systems ist nicht nur von den Ermüdungsausfällen abhängig, sondern es können auch Früh- und Zufallsausfälle auftreten. Daher wird in der Zuverlässigkeitstechnik das Ausfallverhalten eines Systems in drei Phasen Früh-, Zufalls- und Verschleißausfälle aufgeteilt. Die drei Phasen lassen sich in einer sogenannten „Badewannenkurve“ deutlich unterscheiden (Abb. 17). Die Frühausfälle treten am Anfang auf und nehmen mit zunehmender Zeit schnell ab. Frühausfälle werden auch „Kinderkrankheiten“ genannt und ihre Ursachen sind meistens Konstruktions- oder Fertigungsfehler. In dem sogenannten Burn-In-Verfahren werden die Systeme anfangs unter extremen Arbeitsbedingungen getestet, weswegen die Ursachen von Frühausfällen meistens früh erkannt und beseitigt werden können. In der zweiten Phase erreichen die Systeme eine konstante Ausfallrate. Die Ausfälle treten in dieser Phase zufällig auf. Nach einem Zufallsausfall sind die Ursachen zwar technisch erklärbar, aber nicht vorhersehbar. In der letzten Phase der erwarteten Lebensdauer der Systeme treten die Verschleißausfälle auf und nehmen kontinuierlich zu. Im normalen Alterungsprozess gelten die Komponenten nach einer gewissen Zeit als nicht mehr funktionsfähig. Die Verschleißausfälle können auch während des Normalbetriebs auftreten, in solchen Fällen können die Verschleißkomponenten in regelmäßigen Abständen ausgetauscht oder repariert werden und so das System wieder in seinen normalen Betriebszustand versetzt werden.

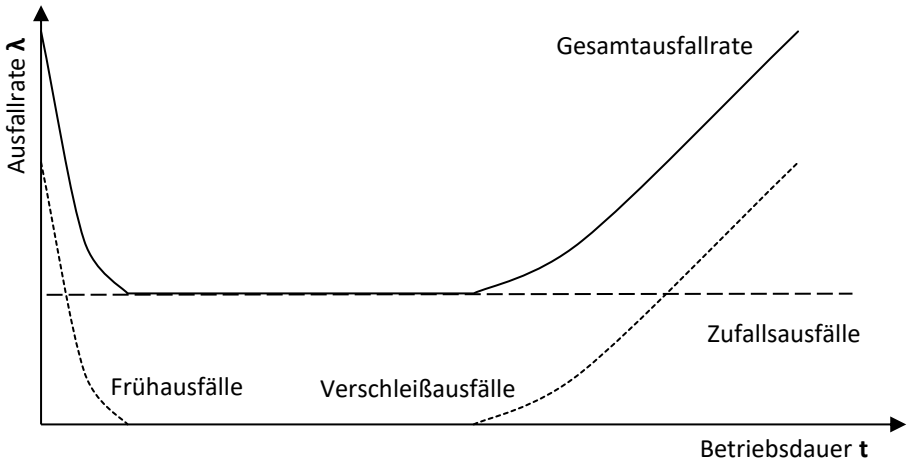


Abbildung 17 Zeitlicher Verlauf der Summe alle Ausfallraten nach (Eberlin, et al., 2014)

Die Ausfallrate eines Systems wird stufenweise ermittelt. Zuerst werden die funktionalen Zusammenhänge der Subsysteme festgestellt. Die Subsysteme werden bis zu den einzelnen Bauteilen weiter zerlegt, wobei der Ausfall eines Bauteils zu einem Ausfall eines Subsystems führt und letzterer wiederum zum Systemausfall. Anschließend wird die Ausfallrate der einzelnen Bauteile bestimmt und im einfachsten Fall zusammenaddiert (Parts-Count-Methode), falls keine Redundanzen vorliegen. Der Einbau von Redundanz- oder Diagnosesystemen kann Systemausfälle erheblich reduzieren. Aufgrund der hohen Kostensteigerungen werden diese Lösungen meist für sicherheitskritische Funktionen eingesetzt.

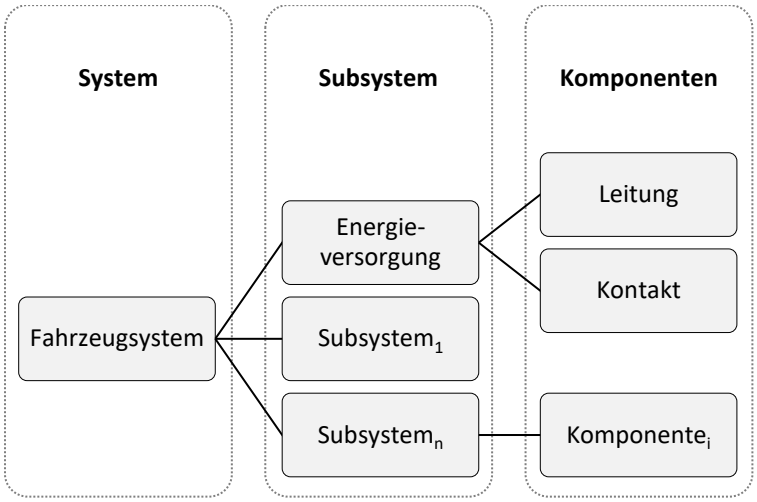


Abbildung 18 Kategorisierung eines Systems für die Zuverlässigkeitsanalyse

Die Bestimmung der Ausfallrate eines Systems wird meistens auf schnellste Weise durch Ausfallratenkataloge realisiert. In den standardisierten Ausfallratenkatalogen sind für einzelne Bauteile Ausfallraten aufgeführt, die sich auf Normalbetrieb beziehen. In den Katalogen wird häufig die Einheit FIT (*Failures In Time*) verwendet. Der Wert 1 FIT steht dabei für 1 Fehler in 10^9 Betriebsstunden. Zunächst erscheint diese Einheit recht unpraktisch, denn Systeme mit einer Lebensdauer von 114.000 Jahren sind eher unrealistisch. Wenn man aber mehrere tausend Systeme betrachtet, werden die Zahlen bereits praktisch verwendbar. In den meisten Ausfallratenkatalogen werden die Ausfallraten in der folgenden Form angegeben:

$$\lambda = \lambda_b \pi_i$$

Die Ausfallrate λ ergibt sich aus der Multiplikation von λ_b mit π_i Faktoren. λ_b beschreibt die Basisausfallrate für die jeweilige Komponente in einem bestimmten Normalbetriebszustand. In einem komplexen System haben die verwendeten Komponenten nicht die gleichen Ausfallraten. Je nach Einsatzort der Komponenten ändern sich Umgebungs- und Funktionsbedingungen, wie Motorraum, Innenraum etc. Daher wird der Faktor λ_b je nach Art der Belastung, wie hohe Temperatur, elektrische Beanspruchung, mechanische Belastung, Feuchtigkeit etc., mit den Belastungsfaktoren multipliziert. Die Belastungsfaktoren werden von den Domänenexperten auf der Grundlage von Felddatenanalysen, experimentellen Untersuchungen oder physikalischen Modellen (*Physics of failure*) bestimmt und ihre Anzahl beträgt im Normalfall drei bis fünf Faktoren. Die Ursachen, wie fehlerhafte Auslegung, schlechte Fertigung etc., werden in die Ausfallrate nicht eingerechnet, sondern werden qualitativ beseitigt. Zu den wichtigsten Ausfallratenkatalogen (sowie Anwendungsbereichen) gehören nach (Weinrich, 2018):

- *MIL-HDBK-217 (Militärelektronik)*

- *NRPD-2016 (Zuverlässigkeitsdaten für nicht elektronische Bauteile aus kommerziellen und militärischen Quellen)*
- *British Telecom HRD-5 (Telekommunikation)*
- *Chinesischer Standard GJB/Z 299B (Militärelektronik)*
- *CNET-2000 (Militärelektronik, am Boden)*
- *DIN EN 61709 (genereller Geräteeinsatz der Bauelemente)*
- *FIDES (Luftfahrt- und Militärelektronik)*
- *IEC 62380 / RDF-2000 (Telekommunikation)*
- *PRISM - Reliability Analysis (Zivil- und Militärelektronik)*
- *SAE 870050 (Automobiltechnik)*
- *Siemens Norm 29500 (Siemens-Produkte)*
- *Telcordia SR-332 / Bell TR 332 (hauptsächlich Telekommunikation)*

Die Ausfallratenkataloge bieten den Vorteil, dass sie einfach zu benutzen sind und dabei ein System quantitativ zügig ausgewertet werden kann, wenn die Daten zur Verfügung stehen. Der Nachteil liegt darin, dass die Kataloge in unterschiedlichen Domänen veröffentlicht sind, so dass es nicht einfach ist, die passende Komponente zu finden. In (Meyna, et al., 2010) wird beschrieben, dass nach Untersuchungen der Georg Washington University die Angaben für gleiche Komponenten im Verhältnis Bell: SAE : MIL = 1 : 2 : 4 variieren und daher relativ verlässliche Daten nur aus den *Reliability Reports* der Zulieferer erhalten werden können.

Eine weitere Quelle für die Bestimmung der Ausfallraten sind die Felddaten, die durch Wartungsdaten gewonnen werden. Basierend auf diesen Daten können Ausfallraten berechnet werden. Nach (Weinrich, 2018) sollen die Ausfallbeschreibungen und Auswirkungen professionell und aussagekräftig dokumentiert sein, um effektiv benutzt werden zu können. Außerdem können die Felddaten in einem kombinierten Ansatz für die Validierung der Ausfallraten, die durch Ausfallratenkataloge berechnet wurden, verwendet werden. Dieser Ansatz wurde in dem in (Gemmerich, et al., 2016) beschriebenen Projekt verfolgt. In dem Projekt wurde basierend auf den Ausfallratenkatalogen zuerst die Ausfallrate für das physische Bordnetz berechnet, indem Komponenten Leitungen, Kontakte, Kabelschuhe, Splices, Sicherungen, Relais und Stecker berücksichtigt wurden. Danach wurde die Ausfallrate aus den Felddaten über die Häufigkeit der erfassten Ausfälle bestimmt. Anschließend wurden die Ergebnisse verglichen und eine Korrektur der Ausfallraten durchgeführt. Wobei zu beachten ist, dass die Felddaten anders strukturiert und mit zu berücksichtigenden Einschränkungen behaftet sind (Gemmerich, et al., 2016):

- *Die Anzahl der registrierten Ausfälle wird mit steigendem Alter der Fahrzeuge immer unvollständiger, da die Besuche in Vertragswerkstätten mit steigendem Fahrzeugalter abnehmen.*

- *Es ergibt sich eine Unschärfe bei der Anzahl von fehlerabhängigen Ausfällen bei der Erhebung der Daten. Eine korrekte Zuordnung der Ausfälle zu den vorgegebenen Ausfallkategorien kann nicht sichergestellt werden, da nicht ausgeschlossen werden kann, dass der Ausfall eines Fahrzeuges in der Werkstatt nicht der korrekten Ausfallkategorie zugeordnet wird.*
- *Die Ausfallkategorien im Feld entsprechen nicht den beschriebenen Ausfällen bzw. Fehlerfällen in der Zuverlässigkeitsberechnung. Zum einen ist keine eindeutige Zuordnung der Fehler zu den unterschiedlichen Bauräumen und damit zu den unterschiedlichen Belastungen durchzuführen. Zum anderen sind die beschriebenen Fehler, die zum Ausfall geführt haben, in der Zuverlässigkeitsberechnung detaillierter als in den Felddaten.*

In dem Forschungsprojekt war zu beobachten, dass die aus den Ausfallratenkatalogen bestimmten Ausfallraten zu wesentlich konservativeren Ergebnissen im Vergleich mit den Felddaten geführt haben. Dies kann als Vorteil angesehen werden, solange die berechnete Ausfallrate die in den Normen beschriebenen Grenzen nicht überschreitet, in diesem Fall hat man einen bestimmten Puffer zwischen der Ausfallrate aus Felddaten und der berechneten Ausfallrate.

4.1.2 Qualitative Methoden

Im Gegensatz zu quantitativen Methoden analysieren qualitative Methoden nicht die Ausfallraten, sondern die Ausfallarten, indem systematisch die Ursachen der Ausfälle analysiert werden. Dieser Abschnitt gibt einen Überblick über die bekannten Methoden FMEA (*Failure Mode and Effects Analysis*) und FTA (*Fault Tree Analysis*) der qualitativen Analyse.

FMEA

Ziel der FMEA Methode ist die frühzeitige Erkennung und Beseitigung von Schwachstellen eines Produkts. Daher ist vorgesehen, dass die Methode schon in der Entwicklungsphase des Produkts eingesetzt wird, um in späteren Phasen eine kostenintensivere Korrektur einzusparen. Die Methode wird auch in der Produktionsphase eingesetzt, um mögliche Fehler in der Produktion zu beseitigen. Die Methode besteht aus drei Phasen. In der ersten Phase werden die möglichen Ausfallarten, Ausfallfolgen und die Ausfallursachen für ein System oder für ein Bauteil analysiert. Die möglichen Ausfallarten werden von einem interdisziplinären Expertenteam erstellt und gepflegt. Jede nicht erfasste und analysierte Ausfallart stellt ein Risiko dar und wird spätestens nach dem Auftauchen erfasst. Die Ausfallarten für ein Gesamtsystem werden bis zur Bauteilebene systematisch analysiert. Die Ausfallarten, Ursachen und Folgen werden, wie in Tabelle 4 dargestellt, in den sogenannten FMEA-Formblättern für Systeme und Bauteile erfasst.

System/Bauteil	Art	Ursache	Folge
Leitung	Eigenerwärmung	Unterdimensionierung	Schädigung

Tabelle 4 Beispiel Ausfallart, Ursache und Folge für eine Leitung

In der zweiten Phase der FMEA-Methode werden die Risiken der Fälle nach den Kriterien Auftretenswahrscheinlichkeit, Schweregrad der Folgen und nach Entdeckbarkeit bewertet und priorisiert. In der letzten Phase der FMEA werden die Fälle nach ihrer Priorität bearbeitet und Optimierungsmaßnahmen werden ergriffen.

FTA

Die FTA-Methode sucht ähnlich wie die FMEA-Methode die Auswirkungen und Ursachen von Ausfallarten. Der Unterschied liegt darin, dass die FTA einen systematischeren Ablauf hat und die Ergebnisse in einer logischen und visuellen Fehlerbaumstruktur abgebildet werden. In FTA werden zunächst die unerwünschten Systemausfälle identifiziert und die funktionale Abhängigkeitskette bis auf Komponentenebene untersucht. Nach der Untersuchung entsteht einen Fehlerbaum (Abb. 19), der durch logische Beziehungen die Ausfallabhängigkeiten zeigt. Eine qualitative FTA-Analyse ist auf quantitative Analyse hin erweiterbar, indem die Ausfallwahrscheinlichkeiten mitberücksichtigt werden.

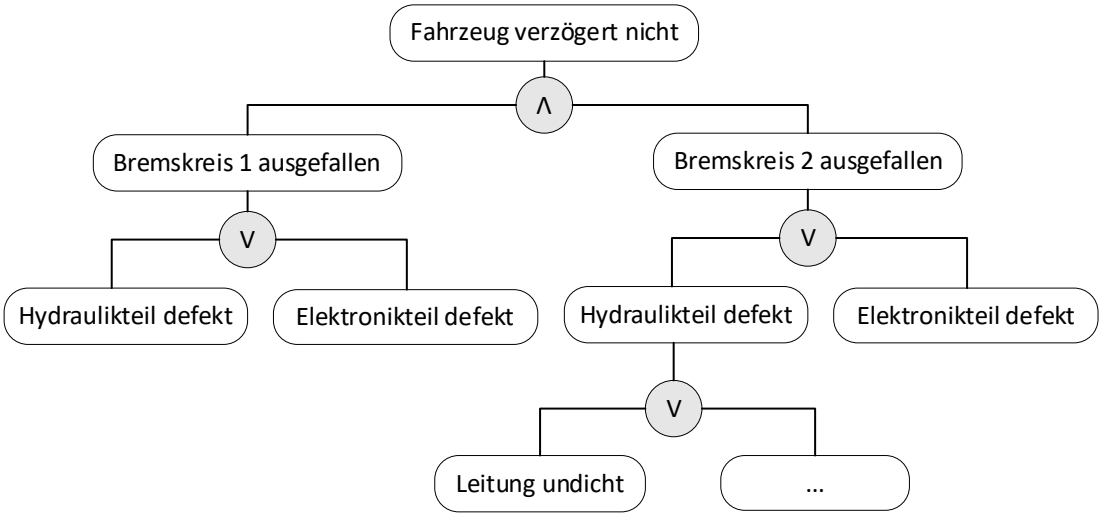


Abbildung 19 Beispiel zur Fehlerbaumanalyse nach (Hillenbrand, 2012) (Wappis, et al., 2010)

4.2 Funktionale Sicherheit

Ziel der funktionalen Sicherheit ist es, die möglichen Fehlfunktionen sicherheitskritischer Systeme nach Gefährdungsgrad und Auftretenswahrscheinlichkeit zu analysieren und abzuwenden. Die Bedeutung der funktionalen Sicherheit in der Automobilindustrie nimmt stetig zu, da die Fahrzeuge immer mehr zu sicherheitskritischen Produkten werden. Infolgedessen wurde, basierend auf der allgemeinen Sicherheitsnorm IEC 61508 für elektrische, elektronische und programmierbare elektronische Systeme, im Jahr 2011 die ISO-

Norm ISO 26262 *Road vehicles - Functional safety* speziell für die Automobilindustrie entwickelt. Die ISO 26262 besteht aus zwölf Teilen und beschreibt Methoden und Aktivitäten, die während der Entwicklung und Produktion von Fahrzeugen durchgeführt werden müssen. Die Umsetzung der Norm ist nicht obligatorisch, sondern eine Empfehlung. Die Einhaltung der Norm ist für die Automobilhersteller aus Sicht der Produkthaftung wichtig, da im Schadensfall nachgewiesen werden muss, dass bei der Produktentstehung die Methoden dem „Stand der Technik und Wissenschaft“ entsprechend umgesetzt wurden.

<table>
<tr><td colspan="4">1. Vocabulary</td></tr>
<tr><td colspan="4">2. Management of functional safety</td></tr>
<tr><td>3. Concept phase</td><td colspan="2">4. Product development at the system level</td><td rowspan="2">7. Production, operation, service and decommissioning</td></tr>
<tr><td>12. Adaption of ISO 26262 for motorcycles</td><td>5. Product development at the hardware level</td><td>6. Product development at the software level</td></tr>
<tr><td colspan="4">8. Supporting Process</td></tr>
<tr><td colspan="4">9. Automotive safety integrity level (ASIL)-oriented and safety-oriented analyses</td></tr>
<tr><td colspan="4">10. Guideline on ISO 26262</td></tr>
<tr><td colspan="4">11. Guidelines on application of ISO 26262 to semiconductors</td></tr>
</table>

Tabelle 5 Übersicht der ISO 26262-Norm nach (ISO 26262, 2018)

Im ersten Teil sind die Vokabulare der Norm definiert. Teil Zwei der Norm beschreibt, wie organisatorisch vorgegangen soll, um die Norm zu realisieren. In der Konzeptphase (dritter Teil) werden Sicherheitsziele definiert, die in Entwicklung und Produktion erreichet werden sollen. Die Sicherheitsziele werden in vier Schritten von der allgemeinen bis auf die Komponentenebene hin spezifiziert (Birch, et al., 2013):

- *Safety Goals (Ebene 1) - Das Fahrzeug in seiner Umgebung*
- *Functional Safety Requirements (Ebene 2) – Die Fahrzeugfunktionen und Systeme*
- *Technical Safety Requirements (Ebene 3) - Die E/E-Systeme*
- *Hardware and Software Requirements (Ebene 4) – Komponenten und Bauteile*

Die Norm schlägt in den Teilen vier bis sieben Methoden und in Teil acht unterstützende Prozesse vor, wie diese Ziele erreicht werden können. Der Teil neun beschreibt ASIL-basierte Sicherheitsanalysen und Teil zehn umfasst weitere informative Erläuterungen. Die neuere Version der Norm wurde um die Teile elf und zwölf, um Richtlinien für Motorräder und Halbleiter, erweitert. In der Vorgehensweise werden die Funktionen bewertet und je nachdem mit der Funktion in Beziehung stehende Systeme und Komponenten analysiert. In der Norm wird beschrieben, was Funktionen, Systeme und Komponenten sind und wie sie gegliedert werden sollen. Nach (Ross, 2014) werden die Elemente wie folgt beschrieben:

- *Ein **Fahrzeugsystem** besteht aus einem oder mehreren Systemen.*
- *Ein **System** besteht mindestens aus drei Komponenten: einem Sensor, einer Verarbeitungseinheit und einem Aktuator. Die **Komponenten** sind entweder*

Hardware-Bauelemente oder Software Units. Ein System kann mehrere Subsysteme haben.

- *Eine **Funktion** wird von einem oder mehreren Systemen realisiert. Wobei ein System mehrere Funktionen realisieren kann.*

Im nächsten Schritt wird die Gefährdungs- und Risikoanalyse durgeführt. *„Ziel der Gefährdungs- und Risikoanalyse ist die Identifizierung und Kategorisierung der potentiellen Gefahren, die durch Fehlfunktionen des Fahrzeugsystems hervorgerufen werden können“* (Ross, 2014). Hierbei werden die möglichen Fehlfunktionen und daraus resultierende mögliche Gefährdungen analysiert und nach dem *Automotive Safety Integrity Level* (ASIL) klassifiziert. Die Klassifikation ist nicht trivial, da viele mögliche Fälle bewertet werden müssen. Bei der Analyse werden die Fehlfunktionen nach den ASIL-Kriterien *Severity* (Schweregrad), *Exposure* (Gefährdungsgrad) und Controllability (Beherrschbarkeit)

Funktion	Fehlfunktion	Betriebszustand	Fahrsituation	Gefahrenszenario	Gefährdung
Beschleunigung	Höhere Beschleunigung als erwartet	aus dem Stand	Fahrt auf Landstraße	Fahrt in den Gegenverkehr	Unfall mit Gegenverkehr
	Niedrigere Beschleunigung als erwartet	aus dem Stand	Fahrt auf Landstraße	Keine Gefährdung	-

Tabelle 6 Fahrsituations- und Betriebszustandsmatrix nach (Ross, 2014)

ausgewertet. Nach diesen Kriterien kann die Fehlfunktion „Höhere Beschleunigung als erwartet“ (Tabelle 6) dem Level QM (Tabelle 7) zugeordnet werden, da der mögliche Schaden lebensgefährlich sein kann (S3), die Fahrsituation selten auftritt (E1) und die Situation vom Fahrer durch Bremsen einfach beherrschbar ist (C1). Die Ergebnisse der Gefährdungs- und Risikoanalyse sind die Sicherheitsziele. Im nächsten Schritt soll ein Konzept entwickelt und umgesetzt werden, damit die Fahrzeugsysteme, die die Funktionen umsetzen, die Ziele durch die Zuverlässigkeitsanalysen erreichen und damit von den Gefahren befreit werden. Die ASIL

Severity	Exposure	Controllability		
		C1	C2	C3
S1	E1	QM	QM	QM
	E2	QM	QM	QM
	E3	QM	QM	**ASIL A**
	E4	QM	**ASIL A**	**ASIL B**
S2	E1	QM	QM	QM
	E2	QM	QM	**ASIL A**
	E3	QM	**ASIL A**	**ASIL B**
	E4	**ASIL A**	**ASIL B**	**ASIL C**
S3	E1	QM	QM	**ASIL A**
	E2	QM	**ASIL A**	**ASIL B**
	E3	**ASIL A**	**ASIL B**	**ASIL C**
	E4	**ASIL B**	**ASIL C**	**ASIL D**

Tabelle 7 Automotive Safety Integrity Level nach (ISO 26262, 2018)

fordert dafür klare Anforderungen. Für das Level QM (Qualitätsmanagement) sollen einfache qualitative Analysen und für die Level A bis D sowohl qualitative als auch quantitative Zuverlässigkeitsanalysen in Entwicklung und Produktion durchgeführt werden. Es soll quantitativ nachgewiesen werden, dass die Ausfallraten bei A < 1000 FIT, bei B < 100 FIT, bei C < 10 FIT und bei D <1 FIT liegen. Um dies nachzuweisen, werden in ISO 26262 auch einige Ausfallratenkataloge zur Verwendung vorgeschlagen. In der Norm wird keine konkrete Umsetzung vorgeschrieben. Daher müssen die Experten die Anforderungen in ihrer Domäne interpretieren und Konzepte entwickeln, um diese zu realisieren. Ein weiterer wichtiger Punkt, der hier zu erwähnen ist, dass in Teil 8 *Supporting Processes* neben vielen Anforderungen auch Maßnahmen zum Konfigurations- und Änderungsmanagement gefordert sind. „*Die Aufgabe des Konfigurationsmanagements ist es, alle Arbeitsergebnisse samt ihrer Historie und Zusammenhänge zu archivieren und den relevanten Beteiligten strukturiert zur Verfügung zu stellen. Das Ziel des Änderungsmanagement ist, den Überblick über alle laufenden und geplanten Änderungen zu wahren und den formalen Rahmen für die Entscheidungen über diese Änderungen zu definieren.“* (Gebhardt, et al., 2013). Das Konfigurations- und Änderungsmanagement ermöglicht zum einen ein strukturiertes Vorgehen bei der Umsetzung des Standards, zum anderen liegen die Ergebnisse in nachvollziehbarer Weise zum Nachweis der Zuverlässigkeit vor.

4.3 Informationstechnische Anforderungen

Das physische Bordnetz bildet zusammen mit den Energie- und Signalleitungen die Grundlage für die Realisierung der Fahrzeugfunktionen und spielt daher eine wichtige Rolle bei der Zuverlässigkeitsbewertung des Gesamtfahrzeugs. Eine Funktionssicht in der Bordnetzentwicklung existiert derzeit nicht (Gemmerich, et al., 2016). Mit den Herausforderungen der Elektrifizierung und des autonomen Fahrens werden einerseits mehr Bordnetzkomponenten und anderseits mehr Redundanzen und Diagnosesysteme benötigt, um die Ausfallwahrscheinlichkeit der Funktionen gering zu halten und somit die definierten Sicherheitsziele zu erreichen. Die Sicherstellung der Zuverlässigkeit und die Erstellung eines kostenoptimalen Bordnetzes werden infolgedessen komplexer. Daher ist die Integration der Norm zur funktionalen Sicherheit ISO 26262 in die Bordnetzentwicklung unerlässlich, um eine frühzeitige Qualitätssicherung zu gewährleisten. In diesem Abschnitt werden zwei wichtige informationstechnische Anforderungen „Rückverfolgbarkeit“ und „Semantische Datenqualität“ behandelt, die auf Datenebene für eine effektive Umsetzung funktionaler Sicherheit in der Bordnetzentwicklung notwendig sind. Das Ergebnis der Bordnetz-entwicklung ist ein digitales Produkt, das auf dem VEC-Datenmodell basiert. Es stellen sich daher die Fragen, „Wie können sicherheitsrelevante Daten mit dem Produktmodell harmonisiert werden?“ und „Wie können die Zuverlässigkeitsanalysen auf digitaler Ebene durchgeführt werden?“.

4.3.1 Rückverfolgbarkeit

Der Standard zur funktionalen Sicherheit ISO 26262 erfordert, dass die Ergebnisse und Aktivitäten transparent dargestellt und rückverfolgbar sind. Das ermöglicht einerseits eine strukturierte Vorgehensweise und anderseits erleichtert es den Nachweis der Zuverlässigkeit des Produktes. Um dies in der Bordnetzentwicklung umsetzen zu können, ist in erster Linie eine semantische Integration der Domänen erforderlich, in der die Beziehungen zwischen den Daten, Informationen und Dokumenten der funktionalen Sicherheit und den Bordnetz-Komponenten dargestellt werden (Abb. 20). Die benötigten Daten für die funktionalen Sicherheit sind sehr heterogen und umfassen vieles, wie Strukturelemente, Funktionen und Systeme, Anforderungen, Daten aus Katalogen für Ausfälle und Ausfallarten, Source Code von Simulationen, Dokumentationen etc., und stehen in semantischen Beziehungen zueinander. Daher ist eine digitale Plattform erforderlich, die gleichzeitig die Realisierung dieser semantischen Beziehungen sowie das Speichern und Abrufen heterogener Daten ermöglicht. Auf diese Weise können die Domänenexperten die Beziehungen nachvollziehen, bei kritischen Fällen den Ursachen auf den Grund gehen und komplexe Zusammenhänge erkennen. Gleichzeitig sind die Beziehungen bei Änderungen hilfreich, um die Auswirkungskette zu verfolgen.

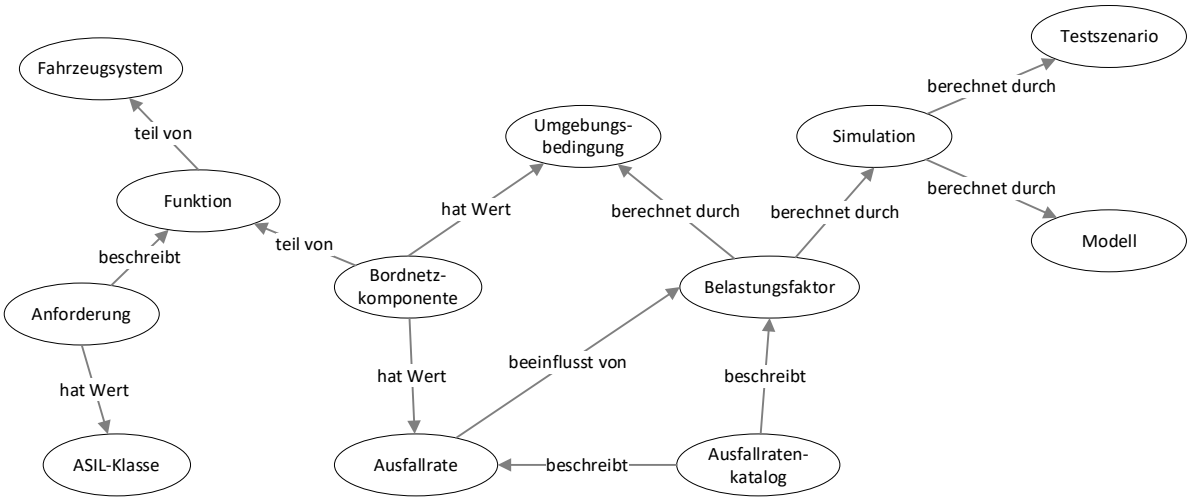


Abbildung 20 Beispiel zur Veranschaulichung der Zusammenhänge der Entitäten für die normgerechte Zuverlässigkeitsanalyse

4.3.2 Semantische Validierung

Auf der Grundlage der vorangegangenen Abschnitte lässt sich zusammenfassend sagen, dass das Ergebnis der Zuverlässigkeitsanalyse ein Produkt sein sollte, das durch die Ausfallursachen und damit durch Ausfälle für eine definierte Zeit und unter bestimmten Betriebsbedingungen normgerecht abgesichert ist. Um dieses Ziel im digitalen Bordnetz zu erreichen, müssen neben den Entwicklungs- und Konstruktionsrichtlinien, wie *„Im*

Motorraum müssen Leitungen mindestens einen Leiterquerschnitt von 0,5mm^2 besitzen“ oder *„Bei der Dimensionierung von Leitungen und Sicherungen müssen Leiterquerschnitt, Sicherungswert und Strom zueinander passen“* (Gemmerich, 2008), sowohl Fehlerursachen, wie *„Eine Leitung kann nur eine bestimmte Anzahl von Biegungen haben“*, die zu bestimmen Fehlern führen (Abb. 21), als auch quantitative Analysen, wie *„Erhöhe die Ausfallrate der Leitungen im Motorraum je nach Belastungsfaktor“*, auf dem digitalen Bordnetz realisierbar und überprüfbar sein. Die Ausfallursachen können, basierend auf Bauteilmerkmalen, Verlegung, Umgebungsbedingungen und Simulationsergebnissen, mit Regeln und Algorithmen erkannt und beseitigt werden. Eine solche Absicherung, bei der die realen Regeln auf die Daten reflektiert und überprüft werden, bezieht sich auf die semantische Validierung des Produkts. Wie in der folgenden Abbildung zu erkennen ist, sind die untersuchten Ausfallursachen vielfältig. Daher ist eine systematische Vorgehensweise und semantische Gliederung der Fälle notwendig, um die Komplexität zu bewältigen.

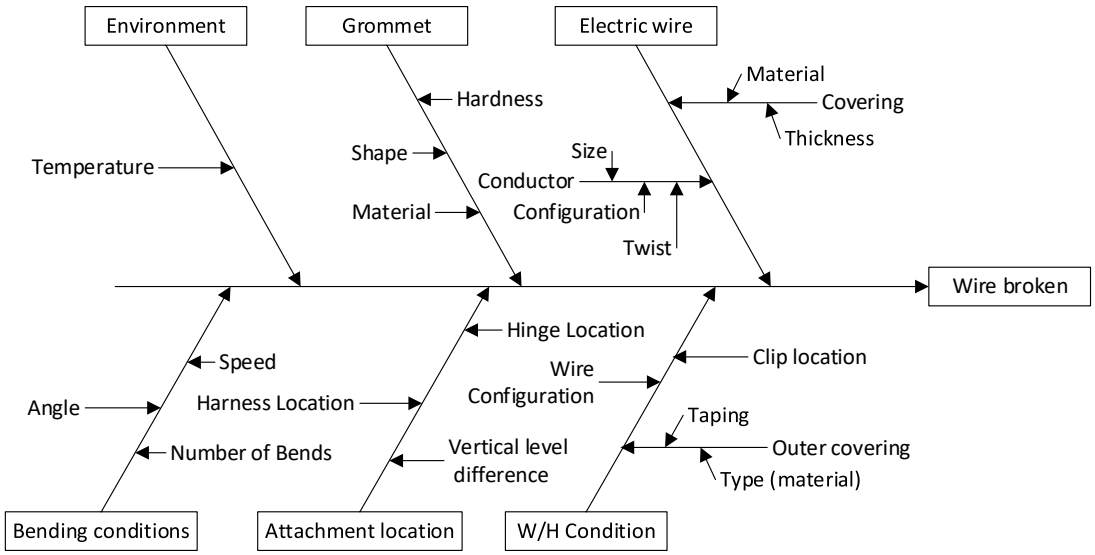


Abbildung 21 Ursache-Wirkungs-Diagramm nach (Inoue, et al., 2000)

Eine semantisch valide Datenbasis ist eine Voraussetzung, um zuverlässige Ergebnisse zu erhalten. Die Umsetzung ist dabei eine Herausforderung. In (Schwimmbeck, et al., 2020) wird der Datenprozess besonders betont (Abb. 22). Es wird beschrieben, dass eine synchrone und aktuelle Datenbereitstellung über standardisierte Schnittstellen notwendig ist, um ausreichende und genaue Simulationen durchführen zu können und gleichzeitig die große Datenmenge, d. h. über hundert Komponenten und ca. 50 Parameter pro Komponente, während der Simulation zu bewältigen. Darüber hinaus ist die Nachvollziehbarkeit der durchgeführten Aktivitäten ein wichtiges Kriterium, damit sie in den Anwendungssystemen nicht als Blackboxes fest kodiert sind, sondern transparent bleiben, um den Domänenexperten

die Möglichkeiten der Weiterpflege und Wiederverwendbarkeit zu bieten. Denn die Anforderungen können sich je nach Produktversionen und -konfiguration ändern.

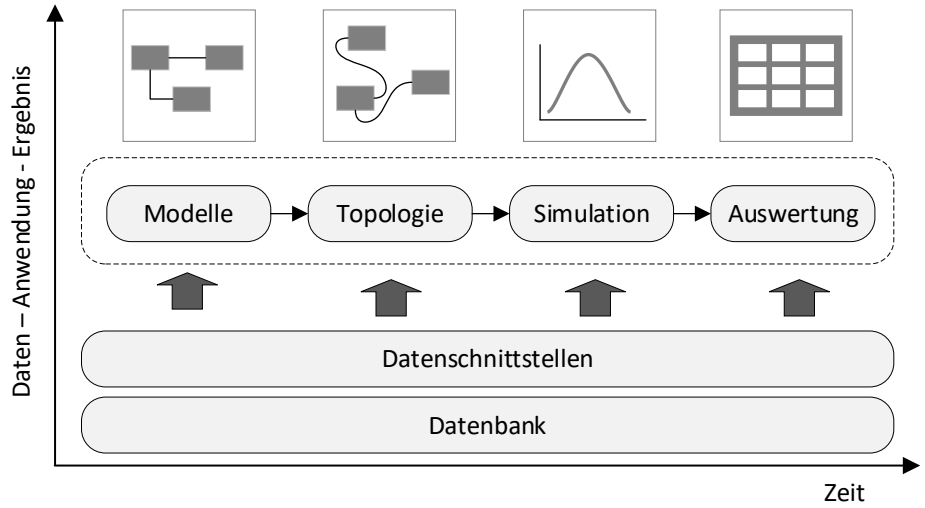


Abbildung 22 Datenmanagement für die virtuelle Energiebordnetz-Entwicklung nach (Schwimmbeck, et al., 2020)

4.4 Zusammenfassung

In diesem Kapitel wurde eine Einführung zu den Themen Zuverlässigkeitsanalysen und funktionale Sicherheit nach ISO-Norm 26262 gegeben. Anschließend wurden die informationstechnischen Anforderungen für die Realisierung einer normgerechten Zuverlässigkeitsanalyse auf der Basis funktionaler Sicherheit betrachtet. Die Schlussfolgerung ist, dass eine Plattform erforderlich ist, auf der die heterogenen Bordnetzdaten, Zuverlässigkeitsanalysedaten und Prozessdaten der funktionalen Sicherheit semantisch miteinander verknüpft werden können, um die geforderte Rückverfolgbarkeit zu realisieren. Eine weitere Anforderung ist, dass für die digitale Absicherung der Zuverlässigkeit und semantischen Validierung weitere transparente digitale Instrumente notwendig sind, um die Analysen effektiv umsetzen zu können. Im Kapitel 5 wird basierend auf Knowledge-Graph-Technologien ein Lösungsansatz zu den dargestellten Anforderungen vorgestellt.

5 Knowledge-Graph-basierter Lösungsansatz

Dieses Kapitel ist der Kern dieser Arbeit und beschreibt einen Lösungsansatz für die in den vorangegangenen Kapiteln beschriebenen Defizite und Anforderungen auf Grundlage der Knowledge-Graph-Technologien. Zunächst wird die ontologiebasierte Darstellung von VEC vorgestellt, die die Grundlage für den Lösungsansatz bildet. Anschließend werden die Vorteile dieses Ansatzes für die Datenverarbeitung, den Entwicklungsprozess und die Realisierung normgerechter Zuverlässigkeitsanalysen erläutert.

5.1 Knowledge-Graph-basierte Datenverarbeitung

Dieser Abschnitt beschreibt, wie die in Abschnitt 3.4.2 beschriebenen Datenverarbeitungsdefizite in XML-basierter VEC mit Hilfe von Knowledge Graphs beseitigt werden können.

5.1.1 Ontologie-basierte Modellierung der Bordnetzdaten

In Kapitel 2 „Grundlagen“ ist bereits behandelt worden, was eine Ontologie ist, daher ist die Einführung hier ausgespart, und dieser Abschnitt konzentriert sich auf die Erstellung der *Vehicle Electric Ontology* (VEO) auf der Basis von VEC und dessen Eigenschaften.

VEC ist ein mit UML erstelltes Datenmodell, und die XML-basierte Implementierung, die für die Anwendungen zur Verfügung gestellt wird, ermöglicht dokumentenbasierte Workflows. In dieser Arbeit wird vorgeschlagen, VEC als Ontologie darzustellen und die Daten als RDF-Graphen zu speichern, um datenbasierte Workflows in der Bordnetzentwicklung zu realisieren. Bereits im Jahr 2008 hat Gerlicher die ontologiebasierte Vorgehensweise unter Verwendung domänenspezifischer Sprachen vorgeschlagen, ausgehend von den folgenden Problematiken in KBL (Gerlicher, 2008): (1) *Komplexe Verweise zwischen den Elementen sind ein Implementierungshindernis.* (2) *Weitere Tools sind zur Validierung erforderlich. Die Definition von XML-Schemas ist nicht präzise genug.* (3) *Unterschiedliche Interpretationen in den Implementierungen*. Dieser Vorschlag wurde jedoch damals nicht weiter spezifiziert und umgesetzt. Zu jener Zeit waren auch die Knowledge-Graph-Technologien noch nicht ausgereift genug für industrielle Anwendungen. Ein weiterer Vorschlag für die Ontologie-basierte Modellierung befindet sich in (Hillenbrand, 2012), in dem die Ontologien für Elektrik/Elektronik Architekturmodelle vorgeschlagen werden, um Domänenwissen im Zusammenhang mit der funktionalen Sicherheitsnorm ISO 26262 zu formalisieren.

Semantische Interoperabilität

Ontologien ermöglichen eine höhere semantische Interoperabilität bei geringerem Implementierungsaufwand im Vergleich zu XML-basierten syntaxorientierten Ansätzen. Auf einer Knowledge-Graph-Plattform können die Softwaresysteme direkt über APIs und objektorientiert auf der Grundlage einer Ontologie auf die Daten zugreifen, ohne das Modell

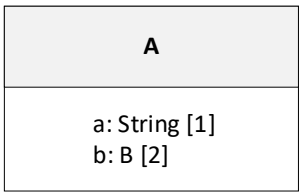
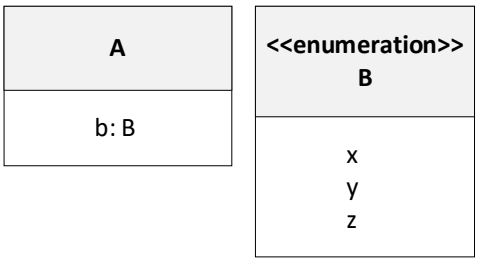
zu interpretieren und in ihr eigenes natives Format zu konvertieren. Programmierschnittstellen zur Verarbeitung von Ontologien und RDF-Daten stehen für alle gängigen Programmiersprachen zur Verfügung (W3C, 2009). Auf diese Weise können Daten zwischen den Systemen ohne Informationsverlust ausgetauscht werden, und die aufwendigen Anpassungen der Schnittstellen entfallen.

```
//Java Code: Gibt alle Instanzen der Klasse Connection zurück
OntClass connection = model.getOntClass("Connection");
connection.listInstances();
```

Quelltext 7 Zugriff auf die Daten durch Jena API (Apache Jena, 2012)

Die Transformation von UML nach OWL

Bei der Transformation des VEC/UML-Modells in OWL wird das Vokabular sowie die Klassen- und Beziehungsstruktur des Datenmodells nicht verändert. In (Zedlitz, et al., 2011) und (El Hajjamy, et al., 2016) wird ein detailliertes Mapping vorgestellt, wie die Elemente von UML auf OWL abgebildet werden können. Basierend auf diesen Arbeiten wird das VEC-Modell, wie in der folgenden Tabelle dargestellt, transformiert (die Transformation wird zur besseren Lesbarkeit mit abstrakter Syntax dargestellt):

UML	OWL
	Klasse, Klasseneigenschaft und Assoziation
	**Declaration**(**Class**(:A)) **Declaration**(DataProperty(:a)) **DataPropertyDomain**(:a :A) **DataPropertyRange**(:a xsd:String) **Declaration** (ObjectProperty(:b)) **ObjectPropertyDomain**(:b :A) **ObjectPropertyRange**(:b :B) **Restriction** (:A **ObjectExactCardinality**(:a 1)) **Restriction** (:A **ObjectMinCardinality**(:b 0)) **Restriction** (:A **ObjectMaxCardinality**(:b 2))
	**Aufzählungen** **Declaration**(DataProperty(:b)) **DataPropertyDomain**(:b :A) **DataPropertyRange**(:b **DataOneOf**("x" "y" "z"))

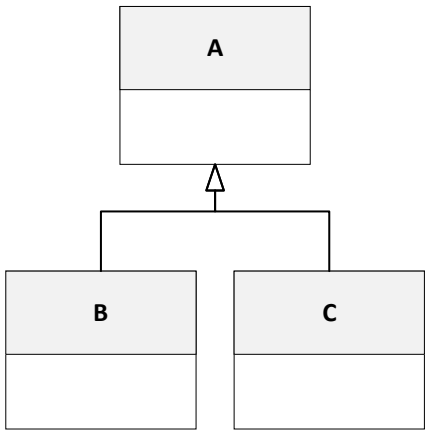
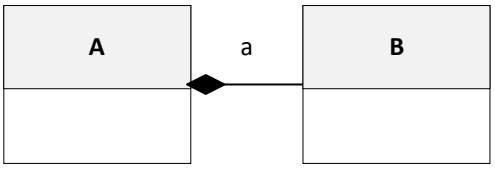
	**Vererbung** **Declaration**(**Class**(:A)) **Declaration**(**Class**(:B)) **Declaration**(**Class**(:C)) **SubClassOf**(:B :A) **SubClassOf**(:C :A) **DisjointClasses**(:B :C)
	**Komposition** **Declaration**(**Class**(:A)) **Declaration**(**Class**(:B)) **Declaration** (**ObjectProperty**(:a)) **ObjectPropertyDomain**(:a :A) **ObjectPropertyRange**(:a :B)

Tabelle 8 Mapping von UML nach OWL nach (El Hajjamy, et al., 2016)

In (El Hajjamy, et al., 2016) wird vorgeschlagen, die *Komposition*-Beziehung mit einer konkreten Beziehung ObjectProperty: isPartOf zu modellieren. Da in VEC die Kompositionen konkrete Namen haben, wird dieser Designentscheidung gefolgt. Die Kardinalitäten werden mit den OWL-Sprachelementen ObjectMaxCardinality, ObjectMinCardinality, ObjectExactCardinality, DataMaxCardinality, DataMinCardinality, DataExactCardinality ausgedrückt. Mit den verwendeten Sprachelementen von OWL ist VEO ein Teil von OWL QL (OWL Working Group, 2012). OWL QL ist ein Profil von OWL DL (siehe Abschnitt 2.2.4), das speziell für die Anwendung mit relationalen Datenbanken und UML entworfen ist. Die VEO 1.2.0 (Balandi, 2018) basiert auf dem publizierten XML-Schema von VEC Version 1.2.0.

Qualität der Ontologie

Die technische Bewertung der Qualität der Ontologie ist eine wichtige Tätigkeit im *Ontology-Engineering*, um die Effizienz zu steigern und Fehler zu vermeiden. Eine nicht präzis genug definierte Ontologie führt z. B. zu falschen Ergebnissen bei der maschinellen Inferenz. Das Ziel der Ontologie-Bewertung hängt vom jeweiligen Aspekt ab. Je nach gewähltem Aspekt werden unterschiedliche Ansätze und Maßnahmen ergriffen. Laut (Sabou, et al., 2012) sind die folgenden Aspekte, die am häufigsten bewerteten:

1. *Domain coverage*
2. *Quality of the modelling*

3. *Suitability for an application/task - Is the ontology suitable for a specific application/task?*
4. *Adoption and use – Has the ontology been reused (imported) as part of other ontologies?*

VEC ist ein verbreiteter Standard und wird von Domänenexperten sowohl weiterentwickelt als auch an die Aufgaben und Anwendungen angepasst. VEO basiert auf der gleichen Struktur und den gleichen Vokabularen, so dass sie parallel zum VEC aktualisiert werden kann. Der Punkt (1) wird daher in dieser Arbeit nicht behandelt. In diesem Abschnitt wird die Qualität der Modellierung betrachtet (Punkt 2). In den folgenden Abschnitten wird erläutert, welche Aufgaben durch VEO besser bewältigt werden können (Punkt 3). Punkt (4) kann in einer späteren Phase bewertet werden, nachdem die Ontologien in der Domäne angewendet wurden. Abschnitt 5.3 zeigt jedoch, wie VEO mit der Kombination von Ontologien der OSLC (Open Services for Lifecycle Collaboration) die informationstechnischen Anforderungen der normgerechten Zuverlässigkeitsanalyse umsetzt.

Nach (Sabou, et al., 2016) kann die Qualität der Ontologie mit einer Vielzahl von Ansätzen bewertet werden, die sich auf logische Korrektheit oder syntaktische, strukturelle und semantische Qualität konzentrieren. Gleichzeitig wird beschrieben, dass die logische Korrektheit von den Inferenzmaschinen automatisch bewertet wird, während die Ansätze, die die syntaktischen, strukturellen und semantischen Qualitätsaspekte der Ontologie betrachten, auf den Erfahrungen der Experten basieren. Die festgestellten Fehlermuster können gesammelt und systematisch überprüft werden. In (Poveda-Villalón, et al., 2012) und (Poveda-Villalón, et al., 2015) wird eine Validierungsmethode namens OOPS! (*OntOlogy Pitfall Scanner!*) vorgeschlagen, die die Ontologie nach potenziellen Fehlern bewertet. In dem Ablauf werden Fehlermuster, sogenannte *Pitfalls* (i. e. *„Fallstricke“*), nach Qualitäts-dimensionen in einem Katalog gesammelt und automatisch auf eine Ontologie angewendet. Die Qualitätsdimensionen sind in (Poveda-Villalón, et al., 2012) wie folgt definiert:

- ***Human understanding:*** *Diese Dimension bezieht sich darauf, ob die Ontologie genug Informationen bietet, so dass sie aus menschlicher Sicht verstanden werden kann. Dieser Aspekt ist in hohem Maße mit der Dokumentation der Ontologie und der Klarheit des Codes verbunden.*
- ***Logical consistency:*** *Diese Dimension bezieht sich darauf, ob logische Inkonsistenzen bestehen, und es wird überprüft, ob es Teile der Ontologie gibt, die möglicherweise zu einer Inkonsistenz führen könnten.*
- ***Modelling issues****: Diese Dimension bezieht sich darauf, ob in der Ontologie von den Ontologie-Implementierungssprachen bereitgestellte Konstrukte richtig verwendet werden oder ob es Modellierungsentscheidungen gibt, die verbessert werden könnten.*
- ***Ontology language specification:*** *Diese Dimension bezieht sich auf die syntaktische Korrektheit der Ontologie.*

- ***Real world representation:*** *Diese Dimension bezieht sich darauf, wie genau die Ontologie die für die Modellierung vorgesehene Domäne repräsentiert. Diese Dimension sollte von Domänenexperten überprüft werden.*
- ***Semantic applications****: Die Dimension bezieht sich darauf, ob die Ontologie für die Software, die sie verwendet, geeignet ist. Diese Dimension ist sehr abhängig von der Anwendung, daher ist sie schwer mit Fehlermustern zu bewerten.*

Human understanding	**Modelling issues**
P1. Creating polysemous elements	P2. Creating synonyms as classes
P2. Creating synonyms as classes	P3. Creating the relationship “is” instead of using ''rdfs:subClassOf'', ''rdf:type'' or ''owl:sameAs''
P7. Merging different concepts in the same class	P4. Creating unconnected ontology elements
P8. Missing annotations	P5. Defining wrong inverse relationships
P11. Missing domain or range in properties	P6. Including cycles in the hierarchy
P12. Missing equivalent properties	P7. Merging different concepts in the same class
P13. Missing inverse relationships	P10. Missing disjointness
P19. Swapping intersection and union	P17. Specializing too much a hierarchy
P20. Misusing ontology annotations	P11. Missing domain or range in properties
P22. Using different naming criteria in the ontology	P12. Missing equivalent properties
Logical consistency	P13. Missing inverse relationships
P5. Defining wrong inverse relationships	P14. Misusing ''owl:allValuesFrom''
P6. Including cycles in the hierarchy	P15. Misusing “not some” and “some not”
P14. Misusing ''owl:allValuesFrom''	P18. Specifying too much the domain or the range
P15. Misusing “not some” and “some not”	P19. Swapping intersection and union
P18. Specifying too much the domain or the range	P21. Using a miscellaneous class
P19. Swapping intersection and union	P23. Using incorrectly ontology elements
P27. Defining wrong equivalent relationships	P24. Using recursive definition
P28. Defining wrong symmetric relationships	P25. Defining a relationship inverse to itself
P29. Defining wrong transitive relationships	P26. Defining inverse relationships for a symmetric one
Real world representation	P27. Defining wrong equivalent relationships
P9. Missing basic information	P28. Defining wrong symmetric relationships
P10. Missing disjointness	P29. Defining wrong transitive relationships

Tabelle 9 Katalog der Pitfalls gruppiert nach Ontologie-Qualitätsdimensionen (Poveda-Villalón, et al., 2012)

Der Katalog wird unter OOPS!-Projektseite (Poveda-Villalón, 2021) (Poveda-Villalón, et al., 2015) gepflegt und nach neuen Erkenntnissen erweitert. Nach der Durchführung der Überprüfung liefert die Anwendung die vorkommenden *Pitfalls* kategorisiert nach Prioritäten *Minor* (z. B. P8), *Important* (z. B. P10) und *Critical* (z. B. P18). Die *Minor-Pitfalls* zu verhindern, wird für bessere Nutzbarkeit empfohlen, die *Important-Pitfalls* sind zu beachten, um mögliche Fehler zu vermeiden, und die *Critical-Pitfalls* müssen unbedingt angepasst werden, ansonsten führen sie zu inkonsistenten Zuständen.

In VEC/UML besitzen viele Assoziationen den gleichen Namen, daher sind bei der Transformation viele ObjectProperty Beziehungen entstanden, die mehrere Domains und Ranges haben. Daher verweist OOPS! auf den *Critical-Pitfall* P18 *„Specifying too much the domain or the range”*. Diese Problematik kann so gelöst werden, dass die beteiligten Klassen zueinander disjoint definiert werden. Eine bessere Lösung ist es, das Vokabular zu verfeinern,

indem man verschiedene und genauere Namen verwendet. Ansonsten gibt es keinen weiteren kritischen Mangel, so dass die Ontologie VEO in Anwendungen eingesetzt werden kann.

Weiterentwicklung der VEO

Die VEO kann mit Open-Source-Editoren weiterentwickelt werden. Es gibt einige Editoren in der Praxis, darunter Protégé (BMIR, 2019) welches der meistbenutzte Editor ist, der zudem mit zusätzlichen Modulen (Plugins) erweitert werden kann. Protégé ist im Kontext der Biomedizinischen Informatik entstanden, wird aber heutzutage in allen Domänen, in denen Ontologien entwickelt werden, eingesetzt. Neben der Modellierung können Klassen und Properties mehrsprachig, detailliert mit verschiedenen Annotationen, dokumentiert werden. Detaillierte Dokumentation ist auch ein Kennzeichen einer guten Ontologie. Somit wird die Dokumentation direkt mit dem Modell integriert. Eine Ontologie dient als objektorientierte Beschreibung der Domäne und auch als Daten-Schema für RDF-Dokumente oder

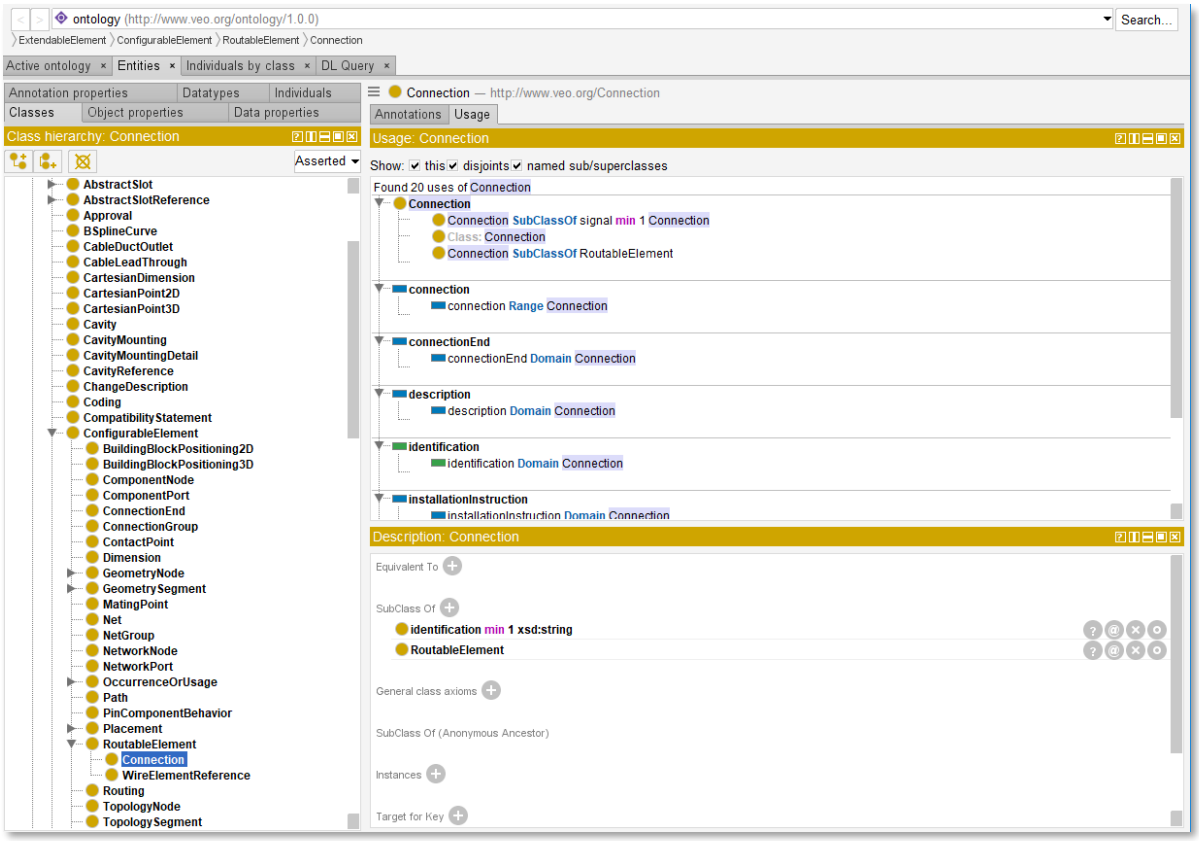


Abbildung 23 VEO in Protégé

RDF-Datenbanken. Damit werden sowohl die Aufgaben der UML-Modellierung als auch die Definition von XML-Schemas bei der Entwicklung von VEC erfüllt. Der Protégé-Editor ist sehr benutzerfreundlich und ermöglicht einen strukturierten Ansatz für die Entwicklung von Ontologien. Wie in Abbildung 23 zu sehen ist, werden die Klassen und Properties auf der linken Seite angezeigt, und in der Detailansicht auf der rechten Seite stehen sie für die Bearbeitung zur Verfügung. Der Editor kann mit hilfreichen Plugins, z. B. zur Visualisierung

der Ontologie mit UML-ähnlichen Diagrammen, interaktive Debugger (Schekotihin, et al., 2018) etc., erweitert werden.

5.1.2 Graphen-basierte Datenstruktur

In der ontologiebasierten Vorgehensweise stehen die Daten im RDF-Format zur Verfügung. RDF, wie in Kapitel 2 „Grundlagen“ erläutert, ermöglicht eine einfache Vernetzung der Daten mit der Subjekt-Prädikat-Objekt-Struktur. Daher ist RDF sehr gut dafür geeignet, die komplexe Referenzierungsstruktur von VEC auf der Datenebene zu realisieren. Indem die Daten zueinander vernetzt sind, entsteht ein riesiger Graph. Die folgende Abbildung zeigt wie schon die Instanzen von fünf Klassen, Component Node, Component Connector, Component Port, Connection und Connection End, die die elektrologische Konnektivität der Bordnetzkomponente zeigen, eine hohe Vernetzung darstellen. Die Graphenstruktur und die durch die Ontologie typisierten Knoten und Kanten ermöglichen es den Domänenexperten, statt mit abstrakten

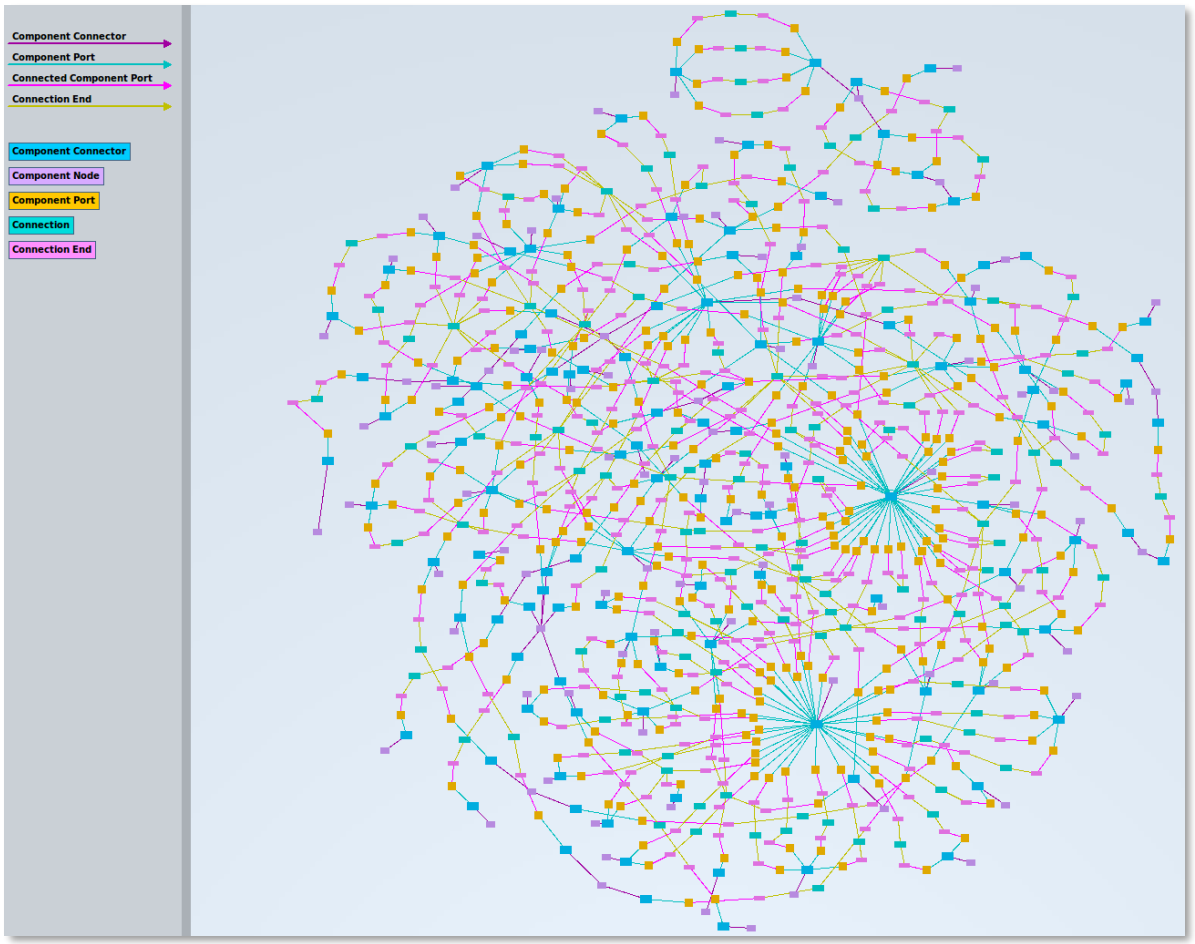


Abbildung 24 Bordnetzdaten als Graph

Visualisierungen direkt mit den Daten zu arbeiten, was es ihnen erleichtert, Zusammenhänge zu verifizieren und besser zu verarbeiten. Auf der anderen Seite bringen die Graphen nicht nur den Endanwendern Erleichterung, sondern vereinfachen auch die Entwicklung performanter Algorithmen. Die Graphen bilden eine wichtige Datenstruktur in der Informatik. Die effiziente Navigation auf den Daten ermöglicht es, performante Algorithmen für viele

Problematiken zu realisieren. Daher sind die Graphen sehr gut für die Konsistenzüberprüfungen und für die Analyse der Topologie, Signal- und Energiepfade des Bordnetzes geeignet, da das Bordnetz – wie der Name schon sagt – selbst auch ein Netz ist. Die Bordnetzwerkzeuge können, ohne den VEC in ein eigenes natives Format umzuwandeln, direkt auf die Daten zugreifen und Algorithmen darauf ablaufen lassen. Die Graphen und Graphenalgorithmen sind in der Bordnetzdomäne nicht völlig neu, z. B. schlägt (Zündorf, et al., 2010) vor, das Bordnetz mit Graph-Grammatiken zu modellieren, und in (Gemmerich, 2008), (Simon, 2008), (Zhu, et al., 2012) werden Graphenalgorithmen vorgeschlagen, um kostenoptimale Energieversorgungen zu realisieren und Absicherungskonzepte des Bordnetzes zu bewerten.

5.1.3 Abfragen der verknüpften Daten

In vielen Fällen erfordert die Struktur von VEC, dass mehrere Klassen durchlaufen werden müssen, um bestimmte Informationen abzufragen. Zum Beispiel um herauszufinden, welche Bordnetzkomponenten über elektrische Leitungen mit welchen Komponenten verbunden sind, müssen die sieben Klassen unter ConnectionSpecification (Abb. 25) durchlaufen werden.

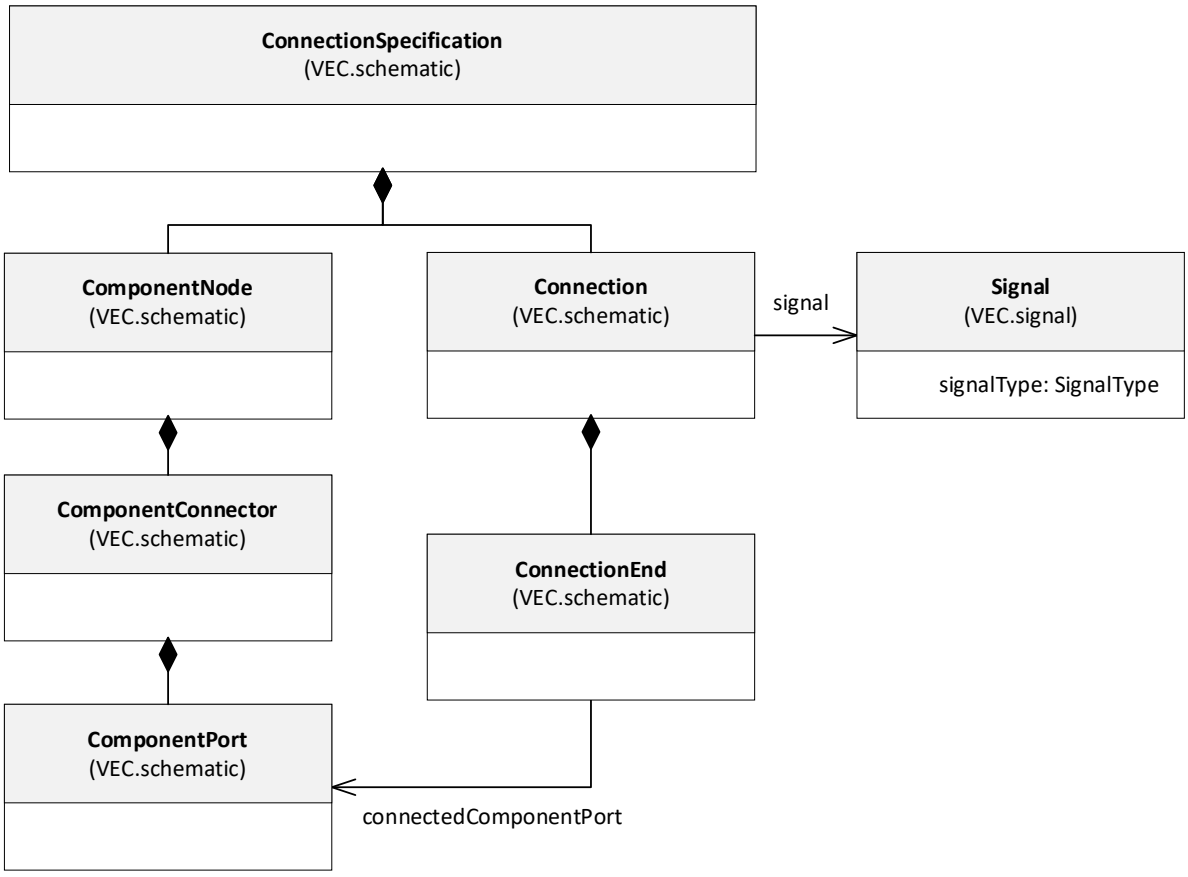


Abbildung 25 Spezifikation der elektrologischen Konnektivität in VEC (prostep ivip, 2020)

Die Knowledge-Graph-Plattformen stellen die SPARQL-Abfragesprache zur Verfügung, um solche verknüpften Daten abzufragen. Die Erstellung einer Abfrage in SPARQL ist im

Vergleich zu den bekannten Abfragesprachen XQuery oder SQL sehr einfach. Die Beziehungen in dem Modell müssen einfach auf eine Abfrage abgebildet werden. Die Beziehungen in SPARQL werden ähnlich wie in RDF mit einer Subjekt-Prädikat-Objekt-Struktur definiert. Wie im folgenden Quelltext zu sehen ist, wird das Modellmuster aus Abbildung 25 mit den Subjekt-Prädikat-Objekt-Tripeln auf eine SPARQL-Abfrage abgebildet.

```
SELECT ?componentNode ?connection

WHERE {
          #Electrological connection
          ?connectionSpecification   rdf:type   veo:ConnectionSpecification.
          ?connectionSpecification   veo:connection   ?connection.
          ?connection   veo:connectionEnd   ?connectionEnd.
          ?connectionEnd   veo:connectedComponentPort   ?componentPort.
          ?componentConnector   veo:componentPort   ?componentPort.
          ?compoenentNode   veo:componentPort   ?componentConnector.

          #Signal type
          ?connection veo:signal ?signal.
          ?signal veo:signalType “Enegry”.
}
```

Quelltext 8 SPARQL-Abfrage der Energie Leitungen zwischen Bordnetzkomponenten

5.1.4 Semantische Datenverarbeitung und –validierung

Die Daten können mit Hilfe von Knowledge-Graph-Technologien effizienter verarbeitet werden. In vielen Fällen sind die Rules Engine und die Query Engine für viele Probleme ausreichend. In anderen Fällen können APIs verwendet werden, um durch Algorithmen die Probleme zu lösen. In diesem Abschnitt wird anhand eines Beispiels gezeigt, wie die Technologien für die Bordnetzdaten zum Einsatz kommen können. Die Konsistenzhaltung der Bordnetzdaten ist eine wichtige Aufgabe. Durch ständige und verteilte Entwicklung kann es schnell zu inkonsistenten Zuständen kommen. Syntaktische Prüfungen reichen in vielen Fällen nicht aus, um Inkonsistenzen zu finden. Um z. B. sicherzustellen, dass alle Aktuatoren mit Sicherungen verbunden sind, müssen die Datenfelder Signal Type der Connection, Signal Direction am Component Port sowie der Typ des Component Node geprüft und die Leitungen traversiert werden. Erschwert wird die Aufgabe, wenn die Aktuatoren nicht wie in Abbildung 26 direkt an die Sicherung angeschlossen sind, sondern über Steuergeräte und Schalter versorgt werden. Die Aufgabe kann mit einer einfachen Regel und Abfrage anstelle von Algorithmen gelöst werden. Zunächst werden, basierend auf den vorhandenen Informationen, neue Informationen mit den IF-THEN-Regeln in den Graphen eingefügt, also Wissen eingebracht, daher heißen die Technologien auch Knowledge Graphs. Die IF-THEN-Regeln bestehen aus zwei Klauseln. Die IF-Klausel beschreibt das Muster mit Tripeln und die THEN-

Klausel beschreibt die auszuführende Aktivität, wenn das beschriebene Muster im Graph auftritt. Die Aktivitäten können z. B. Einfügen, Löschen oder die Berechnung einer bestimmten Information sein. Im Quelltext 9 wird eine IF-THEN-Regel gezeigt, die den ausgehenden Port mit dem eingehenden Port mit der Property "energy" verbindet, wenn der Signaltyp "energy" ist. Wenn die Regel ausgeführt wird, wird der elektrische Versorgungs-baum automatisch durch die eingefügten Energy-Properties als Graph realisiert. Die Regel kann für Masse- und Signal-Leitungen und für Innenverschaltungen erweitert werden, womit ein lückenloser Graph entsteht, der Energie-, Masse- und Signalfluss darstellt. Anhand eines solchen Graphen kann die Konsistenz durch einfache Abfragen überprüft werden.

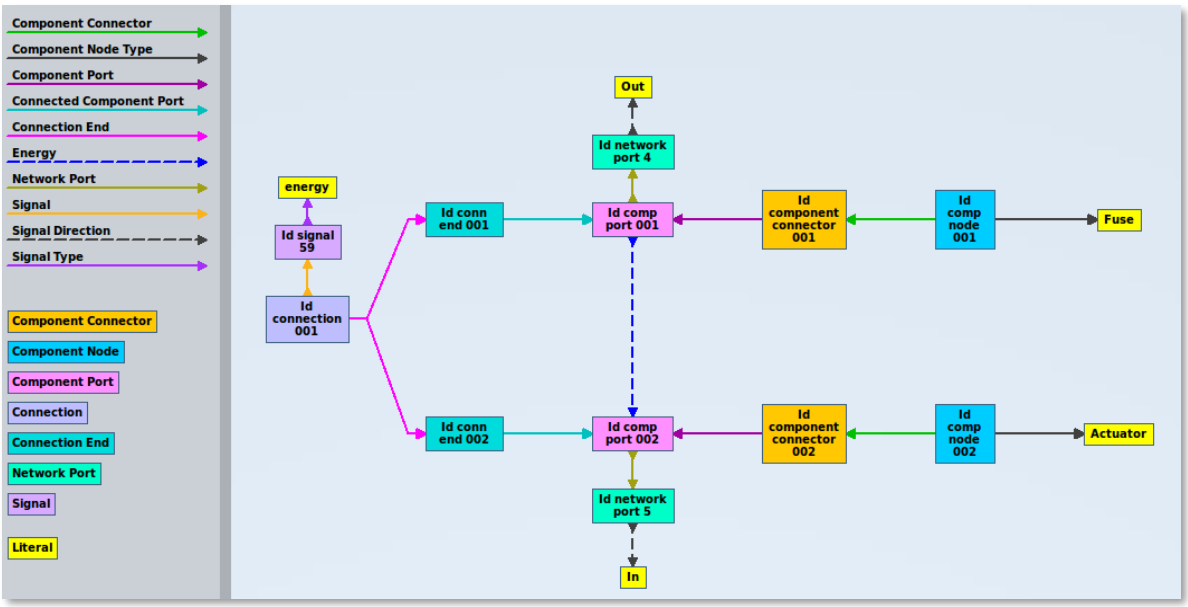


Abbildung 26 Vernetzung der Component Ports

```
IF {   ?connection rdf:type veo:Connection.

     #componentPortOut Out
       ?connection veo:connectionEnd ?connectionEndOut.
       ?connectionEndOut veo:connectedComponentPort ?componentPortOut.
       ?componentPortOut veo:networkPort ?networkPortOut.
       ?networkPortOut veo:signalDirection "Out".

     #componentPortOut In
       ?connection veo:connectionEnd ?connectionEndIn.
       ?connectionEndIn veo:connectedComponentPort ?componentPortIn.
       ?componentPortIn veo:networkPort ?networkPortIn.
       ?networkPortIn veo:signalDirection "In".

     #signaltype
       ?connection  veo:signal ?signal.
       ?signal veo:signalType "energy".}

THEN { ?componentPortOut veo:energy ?componentPortIn. }
```

Quelltext 9 IF-THEN-Regel für die Vernetzung der Component Ports

Im nächsten Schritt werden die generierten Beziehungen überprüft. Der folgende Quelltext beschreibt eine ASK-Abfrage, ob die Aktuatoren von den Sicherungen durch Vorkommen von mindestens einer oder mehrerer Energie-Properties erreicht werden können. Die SPARQL-Abfrage als einzelner Aufruf würde TRUE liefern, wenn dies für mindestens einen Aktuator gilt. Die Abfrage kann als `SPARQLConstraint` in eine SHACL-Regel eingebettet werden, so dass alle Aktuatoren überprüft werden. Im inkonsistenten Fall wird `sh:message` zurück-gegeben. Um eine nachvollziehbare Verarbeitung zu realisieren, spielen die Reihenfolge der ausgeführten Regeln und die Verständlichkeit der Fehlermeldungen eine wichtige Rolle.

```
ActuatorShape a sh:NodeShape ;
  sh:targetClass veo:Actuator;

  sh:sparql [
    a          sh:SPARQLConstraint ;

    sh:message "Der Aktuator ist NICHT an eine Sicherung gebunden!";

    sh:ask
      """ask
        {
          $this rdf:type veo:Actuator.
          $this veo:componentConnector ?actuatorConnector.
          ?actuatorConnector veo:componentPort ?actuatorPort.

          ?fuse rdf:type veo:Fuse.
          ?fuse veo:componentConnector ?fuseConnector.
          ?fuseConnector veo:componentPort ?fusePort.

          ?fusePort veo:energy+ ?actuatorPort.
      }""" ;
 ] .
```

Quelltext 10 SPARQL-Constraint in SHACL

Die SHACL-Regeln können mit eigenentwickelten Funktionen erweitert werden, so dass domänenspezifische komplexe Überprüfungen auf den Daten realisiert werden können. Auf diese Art und Weise können die Regeln für die semantische und syntaktische Datenprüfung strukturiert erfasst und wiederverwendet werden. Der Vorteil ist, dass das Wissen der Domänenexperten durch die Regeln effektiv formalisiert werden kann. Da die Definition der Regeln auf einer höheren intellektuellen Ebene erfolgt, sind sie für den Domänenexperten leichter zu verstehen und zu handhaben.

5.1.5 Zusammenfassung

In diesem Abschnitt wurde gezeigt, wie die Ontologie VEO auf Basis des VEC-Datenmodells realisiert werden kann. Anschließend wurde anhand von Beispielen erläutert, wie die Defizite in der Datenverarbeitung beseitigt und die Technologien eingesetzt werden können. Die

folgende Tabelle vergleicht die ontologiebasierten und XML-basierten Ansätze und gibt einen Überblick.

Kriterien	VEO	VEC
Metamodell	Formales Modell	Schema
Semantik	In der Ontologie	Verteilt in UML und XML
Datenstruktur	Graphenbasiert	Hierarchische Struktur
Algorithmierbarkeit	Graphen sind flexibel für Algorithmen	Muss meist umgewandelt werden auf natives Format der Software
Datenabfrage	Graphenbasiert	Komplexe Abfragen in externen Tools
Skalierbarkeit	Die Ontologien lassen sich einfach semantisch erweitern	Umständlich mit UML- und XML-Interaktion
Datenintegration	Semantische Integration mit verschiedenen Ontologien möglich	Syntaktische Integration durch externe Tools
Interoperabilität	Semantischer Austausch der Daten	Syntaxorientiert mit Interpretierungsaufwand
Konsistenzprüfungen	Semantisch durch zur Verfügung gestellte Werkzeuge wie SHACL	Syntaxbasiert oder durch externe Tools
Verständlichkeit	Semantische Netze ermöglichen einfachen Zugang	Die hierarchische Struktur ist ein Hindernis

Tabelle 10 Vergleich von VEO und VEC

5.2 Knowledge-Graph-basierte Bordnetzentwicklung

In diesem Kapitel wird der auf dem Knowledge Graph basierende Lösungsansatz vorgestellt, mit dem verteilte, kollaborative und iterative Arbeitsabläufe in der Bordnetzentwicklung realisiert und die Defizite beseitigt werden können. Um diese Ziele zu erreichen, spielt ein integriertes Produktmodell mit professionellem und sicherem Versions- und Variantenmanagement eine wichtige Rolle. Dabei können die Dualität und Linearität in der Bordnetzentwicklung behoben und die hohe Änderungsrate bei der Entwicklung besser bewältigt werden. Es werden die Ansätze aus dem Software Engineering betrachtet und das Vorgehen mit bestehenden Lösungsansätzen verglichen.

Knowledge-Graph-basierter Lösungsansatz

Der Knowledge-Graph-basierte Lösungsansatz wird durch VEO, der Knowledge-Graph-Plattform und mit folgenden vier weiteren Komponenten realisiert:

- **Feature Modell** für die Bewältigung der Produktvarianten (Abschnitt 5.2.2)
- **Versionsverwaltungssystem** für die Bewältigung der hohen Änderungsrate bei der Entwicklung (Abschnitt 5.2.3)
- **OSLC** (Open Services for Lifecycle Collaboration) für die Realisierung der Rückverfolgbarkeit durch Vernetzung von Informationen mit Standards (Abschnitt 5.3.1)
- **Unit-Test** für die Validierung der Zuverlässigkeitsanforderungen (Abschnitt 5.3.2)

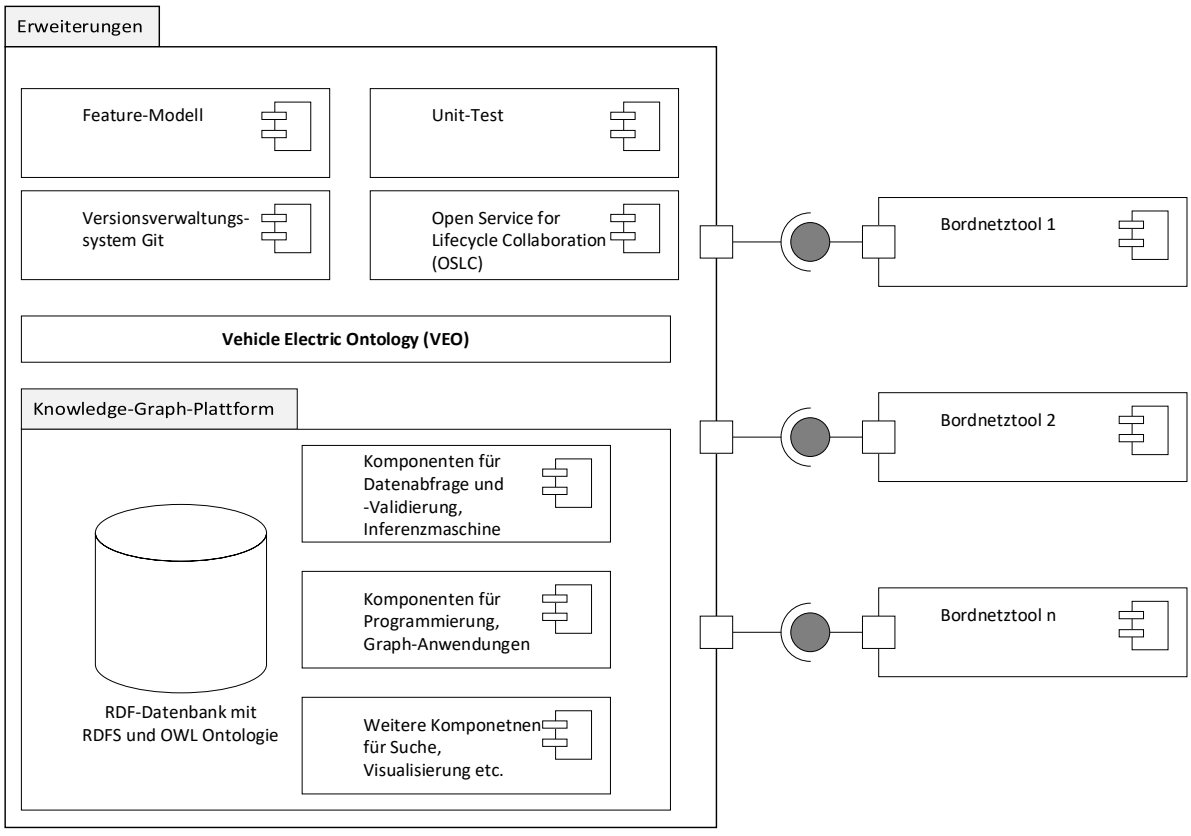


Abbildung 27 Knowledge-Graph-basierter Lösungsansatz

Die VEO und die Vorteile der Knowledge-Graph-Technologien zur Datenverarbeitung wurden bereits in den vorherigen Abschnitten erläutert. In diesem und den folgenden Abschnitten werden weitere Komponenten des Lösungsansatzes zu den Defiziten im Entwicklungsprozess und die informationstechnischen Anforderungen zur Realisierung normgerechter Zuverlässigkeitsanalyse vorgestellt.

5.2.1 Semantisches und integriertes Produktmodell

Die dokumentenbasierte und duale Vorgehensweise in der Bordnetzentwicklung führt schnell zu inkonsistenten Zuständen, und die Ergebnisse der Bordnetzentwicklungswerkzeuge stehen zusammenhanglos nebeneinander. Als Lösung für diese Problematik werden Produktmodelle vorgeschlagen, um damit das Prinzip *„Single Source of Truth“* zu realisieren. Nach (Picard, 2015) (Grabowski, et al., 1993) sind Produktmodelle wie folgt definiert: *„sie zielen ab auf die interdisziplinäre Definition, realitätsnahe Beschreibung und Darstellung von Produktinformationen sowie deren anwendungsbezogene Bereitstellung und Verarbeitung“*. In (Kyriazis, 2016) wird für den Bereich Bordnetzentwicklung ein Paradigmenwechsel von dokumenten- zum produktmodellbasierten Ansatz vorgeschlagen. Dabei werden die folgenden Ziele gesetzt: *(1) Durchgängigkeit der Daten, (2) Beherrschen der Produktvarianz, (3) Abbildung der Änderungsprozesse als integrale Bestandteile des Prozesses, (4) Traceability, (5) Absicherung der Produktqualität, (6) Verteilte Entwicklung mit mehreren Standorten, (7) Erhöhung des Einsatzes von Generatoren für algorithmierbare Aufgaben*. Um diese Ziele zu erreichen, wird ein auf VEC basierendes Master-Produktmodell (Logical Mock-Up) vorgeschlagen, bei dem die Bordnetztools direkt am Produktmodell arbeiten (Abb. 28). Dabei ist durch zentrale Datenhaltung darauf abgezielt, die Konsistenz der Daten zu sichern und kontrollierbares Änderungsmanagement zu gewährleisten.

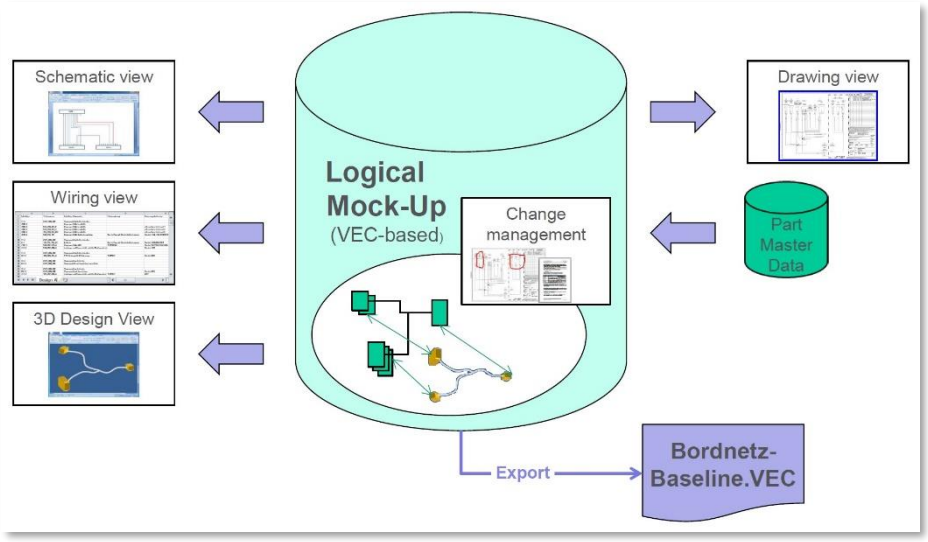


Abbildung 28 Master-Produktmodell (Kyriazis, 2016)

Ein Lösungsansatz, der das Master-Produktmodell implementiert, wird in (Neckenich, 2017) vorgeschlagen, in dem der 3D-Master-Lösungsansatz (Kitsios, et al., 2014) verwendet wird,

bei dem alle Produktinformationen unter dem CAD-Modell (*Computer Aided Design Model*) gespeichert werden. So wird der 3D-Master zum Hauptinformationsträger, und damit wird angestrebt, die Dualität in der Entwicklung aufzulösen. Der 3D-Masteransatz wird in CATIA V5 (Kitsios, et al., 2014) realisiert und ist ein softwarebasierter Lösungsansatz, bei dem die CAD-Visualisierung im Vordergrund steht und die Daten im Hintergrund als XML zur Verfügung stehen. Im Gegensatz dazu ist der Knowledge-Graph-basierte Ansatz ein datenbankgestützter Ansatz, bei dem nicht die Visualisierung, sondern die Daten im Vordergrund stehen. In (Kirsch, et al., 2019) ist beschrieben, dass der 3D-Master allein noch keinen digitalen Prozess liefert, da die Informationen aus den technischen Zeichnungen nicht ganz digitalisiert werden. Daher wird ein Lösungsansatz vorgeschlagen, bei dem die Dokumente aufgelöst und Inhalte zusammengeführt und navigierbar verknüpft werden. In (Kirsch, et al., 2019) bedeutet die Digitalisierung, *„die Umwandlung von Daten und Informationen in digitales Format mit dem Nutzen, dass diese informationstechnisch, d.h. vernetzt und automatisiert verarbeitet werden können. Vernetzung bedeutet das Verknüpfen von Daten mit Hilfe semantischer Beziehungen“.*

Auflösung der Dualität

Die Integration der Domänen Elektrologik und 3D-Entwicklung auf der Datenebene ist notwendig, um die Dualität in der Bordnetzentwicklung aufzulösen. In der 3D-Entwicklung werden die Geometriedaten in verschiedenen Datenformaten ausgetauscht. Die Inhalte der Datenformate sind ganz oder teilweise maschinenlesbar. Das bedeutet, dass die teilweise maschinenlesbare Datenformate Binär-Code-Teile haben, die nur von den CAD-Werkzeugen gelesen und interpretiert werden können. Aber solche Datenformate können mit Begleitformaten erweitert werden, die Struktur und Metadaten der Geometrieinformationen durch XML-Schemata beschreiben (Sindermann, 2014). Das am weitesten verbreitete und ISO-10303 standardisierte Datenformat ist STEP *(Standard for The Exchange of Product Model Data)* (Sindermann, 2014). STEP ist sehr umfangreich und bietet Datenmodelle (so genannte *Application Protocols*) für verschiedene Anwendungsbereiche und ermöglicht damit, Produktmodellinformationen für den gesamten Produktlebenszyklus zu beschreiben. In STEP sind die Datenmodelle mit der Datenmodellierungssprache EXPRESS (ISO 10303, 2004) definiert. Die durch EXPRESS definierten Geometriedaten sind in strukturiertem ASCII-Code erfasst und somit maschinenlesbar. Ein weiteres Format, das in der Bordnetzdomäne verwendet wird, ist JT (Jupiter Tesselation). JT-Dateien werden in Binär-Code ausgetauscht und im Vergleich zu STEP sind die Dateigrößen sehr klein, weshalb JT weit verbreitet ist (Sindermann, 2014). In (Barbau, et al., 2012) wird ein semantisches Produktmodell vorgeschlagen, in dem die Geometriedaten mit den Produktmodelldaten integriert werden. Dabei wird zuerst das in EXPRESS definierte Schema des STEP AP nach OWL 2 transformiert. Diese Ontologie wird OntoSTEP (Krima, et al., 2009) genannt. Danach werden die Instanzen bzw. die Geometriedaten, die gemäß STEP in einer Part-21-Datei definiert sind, zu Instanzen gemäß OntoSTEP konvertiert. Damit können sie dann mit den

Produktdaten aus CPM (*Core Product Model*) semantisch integriert werden, wobei das CPM auch mit Ontologien beschrieben ist. Die Anwendung ist als Protégé-Plugin implementiert.

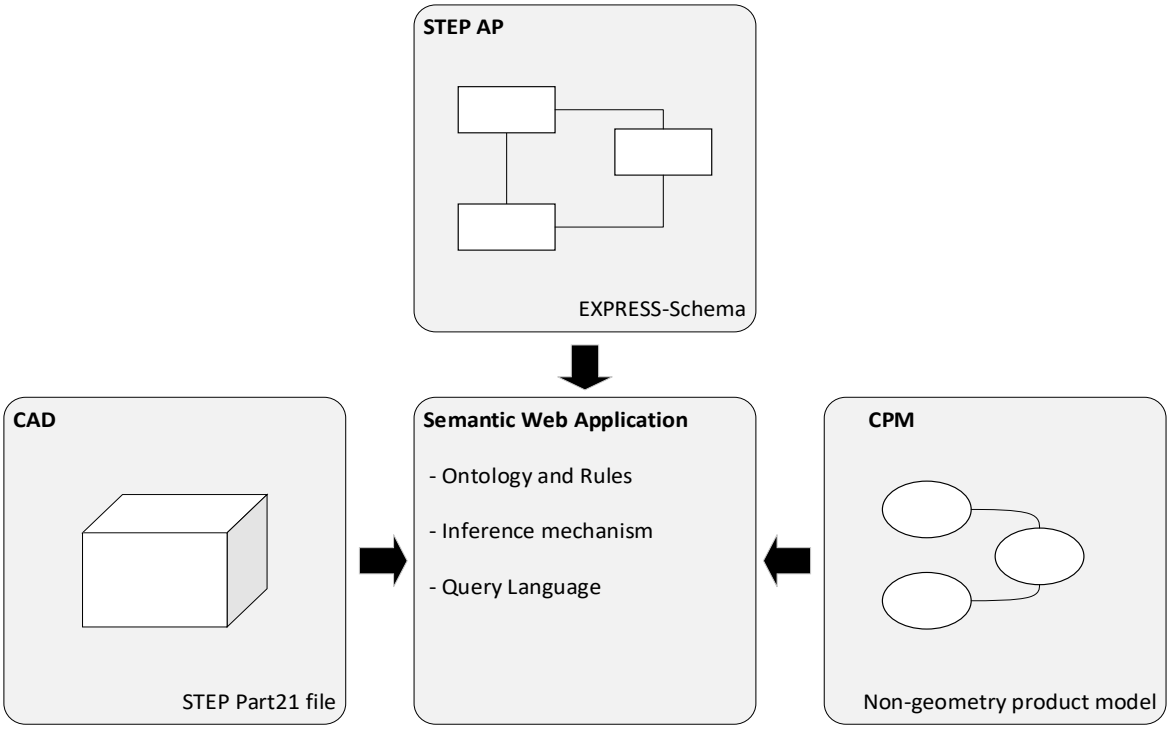


Abbildung 29 Aufbau eines semantischen Produktmodells. CAD – Computer Aided Design CPM – Core Product Model nach (Barbau, et al., 2012)

Eine weiterer ontologiebasierter Lösungsansatz wird in (Perzylo, et al., 2015) beschrieben, in dem die mathematische Darstellungsform *Boundary Representation* (B-Rep) als OWL-Ontologie OntoBREP implementiert wird. B-Rep ist ein in CAD-Werkzeugen benutzter Ansatz, um Volumenkörper auf mathematischen Grundlagen durch Punkte, Kurven, Flächen und Volumina darzustellen (Sindermann, 2014). Durch OntoBREB ist es möglich, die komplexen Inhalte von CAD-Daten semantisch zu analysieren (Perzylo, et al., 2015). Das JT-Datenformat ist kombinierbar mit B-Rep, somit könnte der Ontologie-Ansatz auch auf JT verwendet werden. Die vorgestellten Ansätze OntoSTEP bzw. OntoBREP können semantisch mit VEO integriert werden, oder zumindest die Begleitformate, so dass die CAD-Daten und Bordnetzdaten semantisch miteinander verknüpft werden. Auf dieser Basis können die Bereiche Elektrologik und 3D-Entwicklung auf der Datenebene integriert werden. Durch die Integration könnte die Entwicklung näher zusammengeführt werden, so dass beide Domänen auf ein gemeinsames Repository zugreifen, so dass Änderungen gegenseitig verfolgt werden können und die Dualität aufgehoben werden kann. Mit einer auf VEO basierenden Knowledge-Graph-Plattform ist es möglich, auf diese Weise ein semantisches und integriertes Produktmodell[6] zu realisieren.

[6] *„Integrierte Produktmodelle zielen auf ein umfassendes, konsistentes Produktmodell über alle Entwicklungsdomänen und über den gesamten Produktlebenszyklus ab“* (Picard, 2015) (Grabowski, et al., 1990) (Gausemeier, et al., 2006) .

Realisierung von Digital Twin und Digital Thread

In letzter Zeit sind die beiden Begriffe *Digital Twin* (Digitaler Zwilling) und *Digital Thread* (Digitaler Faden) in der virtuellen Produktentwicklung populär geworden und tauchen in verschiedenen Artikeln und Arbeiten sogar mit unterschiedlichen Beschreibungen auf. Folglich werden diese Begriffe hier betrachtet, um sie in Bezug auf das integrierte Produktmodell zu klären. Nach (Kuhn, 2017) ist die Definition von *Digital Twin* wie folgt: „*Digitale Zwillinge sind digitale Repräsentanzen von Dingen aus der realen Welt. (…) Ein zentraler Aspekt von digitalen Zwillingen ist daher die Fähigkeit, verschiedene Informationen in einem einheitlichen Format zu repräsentieren. Digitale Zwillinge sind jedoch mehr als reine Daten. Sie beinhalten Algorithmen, die ihr Gegenstück aus der realen Welt akkurat beschreiben*“. Nach dieser Definition ist Digital Twin nichts anderes als ein integriertes Produktmodell. Die 3D-Zeichnung als statisches Modell oder das Simulationsmodell als dynamisches Modell können als Digital Twin betrachtet werden. Sie stellen jedoch nur einen Teil des realen Produkts dar. Daher ist ein Digital Twin notwendig, unter dem alle Informationen integriert werden können. Der Digital Twin muss ein logisches Modell sein, damit das Produkt und seine Beziehungen zu allen anderen Dingen logisch definiert werden können, wie in Abbildung 30 dargestellt. Beim Digital Twin ist das Besondere, dass das reale Produkt mit seiner digitalen Abbildung gekoppelt bleibt. Das bedeutet beispielsweise, dass Sensoren, die in das Produkt eingebaut sind, aktiv Informationen an den Digital Twin senden oder dass die Wartungsergebnisse des Produkts an den Digital Twin weitergeleitet werden, so dass die Informationen in Simulationen und Produktentwicklungen zur Verbesserung der Produkt- und Fehlprognose verwendet werden können.

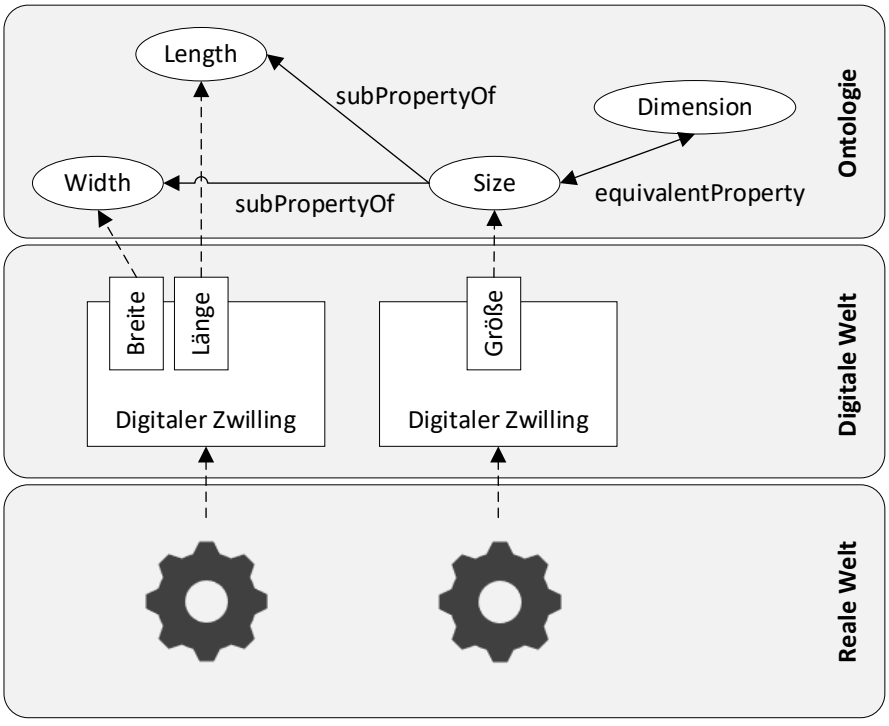


Abbildung 30 Ontologie zur Definition von Eigenschaften nach (Kuhn, 2017)

Darüber hinaus bedeutet Digital Thread die digitale Vernetzung von Produktinformationen über den gesamten Produktlebenszyklus hinweg. Um Digital Twin zu realisieren, ist daher Digital Thread notwendig, um verschiedene Informationen eines Produkts zu vernetzen. Ziel ist es, Prozesse durch die Vernetzung verschiedener Informationen zu verbessern, um Aufgaben schneller und sicherer zu erledigen. Die Verknüpfung von Anforderungen und Testfällen ist dafür ein gutes Beispiel. Damit können Qualitätsingenieure schnell und sicher überprüfen, ob die Anforderungen in den Testfällen abgedeckt sind. Da die heutigen Produkte sowohl komplex sind als auch interdisziplinär entwickelt werden, spielen Digital Twin und Digital Thread eine wichtige Rolle von der Produktentwicklung über die Produktion bis hin zur Produktwartung.

Die Knowledge-Graph-Technologien bieten die große Möglichkeit für die Realisierung von Digital Twin und Digital Thread. Ontologien sind sehr ausdrucksstark, daher können die realen Dinge realitätsnah abgebildet werden. Ein Digital Twin im Knowledge Graph ist kein statisches Dokument wie in Legacy-Systemen, sondern eine eindeutige Instanz einer Klasse aus einer Ontologie, und kann bei Bedarf flexibel erweitert werden. Dabei können verschiedene Digital Twins durch Ontologien semantisch integriert werden (Abb. 30). Die Beziehungen der Merkmale von Digital Twins, die Abhängigkeiten und sogar die unterschiedlichen Bezeichnungen können klar definiert werden. In Knowledge Graphs sind Digital Twins unter einem bestimmten URI adressierbar und somit global erreichbar, so wie die realen Dinge. Durch URIs und RDF können verschiedene Informationen semantisch miteinander verknüpft werden, so wird Digital Thread realisiert. Es existiert auch eine Initiative namens *Open Services for Lifecycle Collaboration (OSLC)*, die auf Basis dieser Technologien Spezifikationen entwickelt, um standardisierte Digital Threads zu realisieren (siehe Abschnitt 5.3.1).

In der Bordnetzentwicklung ist die Datendurchgängigkeit ein fehlendes Merkmal. Derzeit wird das digitale Ergebnis, also die XML-Dokumente der Bordnetzentwicklung, in ein anderes Format für die Produktion überführt und in Papierform zur Wartung weitergegeben. Daher findet kein kontinuierlicher Datenaustausch statt, so dass Meldungen aus Produktion und Wartung nicht einwandfrei in der Entwicklung platziert werden können. Mit VEO und Knowledge-Graph-Technologien ist es möglich, die Informationen und Modelle über den gesamten Produktlebenszyklus zu verknüpfen und so Digital Twin und Digital Thread zu realisieren.

5.2.2 Variantenmanagement mit Feature-Modell

Kraftfahrzeuge sind heutzutage variantenreiche Produkte. Die Kunden können auf millionenfach verschiedene Weise ein Fahrzeug konfigurieren. Die Entwicklung eines variantenreichen Produktes ist eine komplexe Aufgabe, da die Aufrechterhaltung der Funktionalität in allen möglichen Varianten eine große Herausforderung darstellt. Dasjenige Modell, das alle möglichen Varianten enthält, wird 150-Prozent-Modell, das Modell, das eine

spezifische Variante bzw. eine spezifische Konfiguration darstellt, wird 100-Prozent-Modell genannt. Dabei sind zwei Aufgaben wichtig, die erste ist die Beschreibung der Variantenvielfalt im 150-Prozent-Modell, und die zweite ist die automatische Überprüfung, ob ein bestimmtes 100-Prozent-Modell eine zulässige Konfiguration ist. Die Beschreibung der Variantenvielfalt erfolgt in heutigen Produktdatenmanagement-Systemen mit Code-Regeln, so auch in der Bordnetzentwicklung. Das VEC-Datenmodell hat eine eigene hierarchische Variantenbeschreibung mit UML in dem Modul VEC.variants und die Code-Regeln werden unter den Klassen VariantConfiguration und VariantCode erfasst. Eine spezielle Software kann diese Code-Regeln entschlüsseln und auf Gültigkeit überprüfen.

Feature-Modell

In (Seibertz, et al., 2013) werden Code-Regeln und andere Ansätze für das Variantenmanagement in verschiedenen Phasen der Produktentwicklung beschrieben und vorgeschlagen, anstelle der bestehenden Ansätze die Feature-Modellierung zu verwenden, da die anderen Ansätze bei einer großen Anzahl von Konfigurationen schnell unübersichtlich werden, zu einem immensen Dokumentationsaufwand bei Änderungen führen und die Absicherung des Modells erschweren. Im Gegensatz dazu bieten Feature-Modelle die Möglichkeit in einer kompakten Repräsentation alle Instanzen einer Produktlinie anhand von *Features* auszudrücken. Feature-Modelle stellen die Varianten überschaubar graphisch dar, und dabei können die Abhängigkeiten schnell erkannt und basierend auf dem Modell strukturierte Analysen durchgeführt werden. Feature-Modelle können daher in allen Phasen eines Projekts, wie bei Erstellung der Anforderungen, Variantenbeschreibung eines Systems, Validierung des Systems etc., eingesetzt werden und bieten ein modellbasiertes Variantenmanagement (Junk, et al., 2015). Feature-Modelle sind zuerst in einer Machbarkeitsstudie an der Carnegie-Mellon-Universität (Kang, et al., 1990) entstanden und ihr Einsatz ist in Softwareproduktlinien und Forschungen stark verbreitet. Ein konkretes Feature in einem Modell beschreibt eine Konfigurationsoption, z. B. eine Komponente, Module etc., die baumartig strukturiert dargestellt werden können. Dagegen ist ein abstraktes Feature ein Strukturierungselement, um Feature Modelle übersichtlich zu halten (Thüm, et al., 2011). Ein Feature-Modell hat vier Grundmodellelemente *Mandatory*, *Optional*, *Alternative*, *Or* und zusätzlich die Modellelemente *Implies* und *Excludes* um baumübergreifend Einschränkungen zu definieren. Neben den Grundelementen finden sich weitere Erweiterungen in verschiedenen Forschungsarbeiten.

- *Mandatory:* Dieses Feature ist in einer Konfiguration obligatorisch.
- *Optional:* Dieses Feature ist in einer Konfiguration optional.
- *Alternative*: Eins von den Features ist in einer Konfiguration obligatorisch.
- *Or:* Eins oder mehrere Features können in einer Konfiguration ausgewählt werden.
- *Implies:* Das Vorkommen eines Features in einer Konfiguration erfordert ein anderes Feature.

- *Excludes:* Das Vorkommen eines Features in einer Konfiguration schließt ein anderes Feature aus.

Die folgende Abbildung zeigt ein beispielhaftes Feature-Modell für ein Bordnetz. Das Modell ist erstellt mit dem FeatureIDE-Framework (FeatureIDE, 2007). Das Framework ermöglicht die Realisierung eines Konsistenzmodells mit aktivem Debugger. Im kleinen Modell besteht das Bordnetz aus zwei obligatorischen und einem optionalen Feature. Die konkreten Features bilden die Bordnetz-Module und die entsprechenden Bordnetz-Komponenten können unter diesen Features zugeordnet werden. Zusätzlich enthält das Modell zwei Regeln. Die erste Regel beschreibt, dass das Modul_m_2 das Modul_i_2 anfordert, und die zweite Regel beschreibt, dass das Modul_i_2 das Modul_e_2 ausschließt.

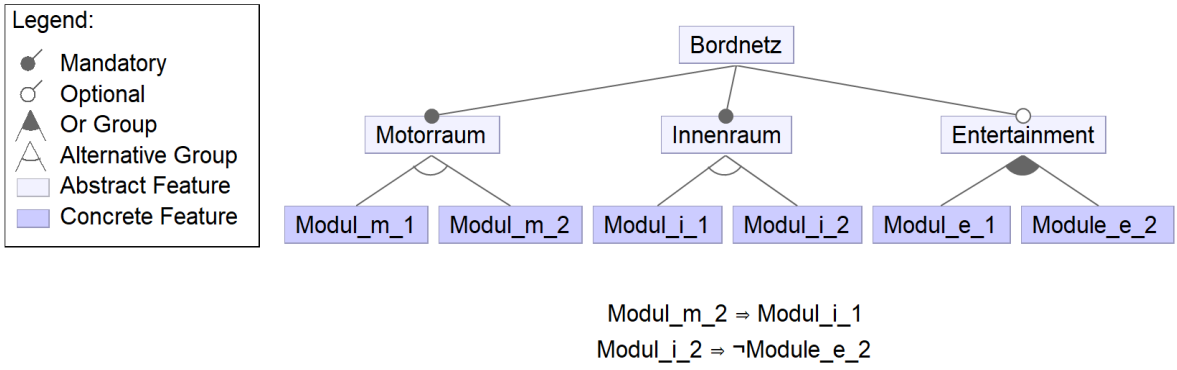


Abbildung 31 Beispielhaftes Feature-Modell einer Bordnetz-Produktlinie

Im nächsten Schritt kann in FeatureIDE auf Basis des Feature-Modells ein Konfigurator generiert werden, der es ermöglicht, zulässige Konfigurationen zu erstellen. Auf diese Weise kann ein vollständiges Bordnetz zusammengestellt werden. Der Konfigurator berechnet alle möglichen Konfigurationen (in diesem Fall 10). Während der Auswahl unterstützt er, welche Auswahlen und Anzahl von Konfigurationen im nächsten Schritt möglich sind. In dieser Art und Weise werden Feature-Modelle eingesetzt, um die Produktvarianz zu bewältigen.

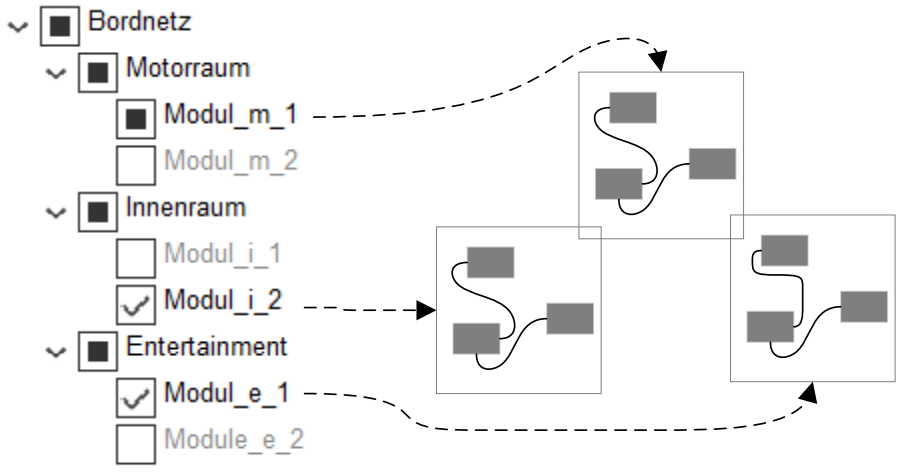


Abbildung 32 Konfiguration des Bordnetzes durch Verlinkung nach (Rock, 2009)

Feature-Modell und Ontologie

Ein Feature-Modell kann mit verschiedenen Sprachen und Technologien realisiert werden. Die Publikationen (Benavides, et al., 2010) und (Galindo, et al., 2018) geben einen umfassenden Überblick über den Bereich automatischer Analyse-Möglichkeiten mit Feature-Modellen. Die ontologiebasierte Darstellung von Feature-Modellen ist in dieser Arbeit zur Integration mit VEO von besonderem Interesse. In (Wang, et al., 2007) ist eine detailliertere Vorgehensweise für die OWL-DL-basierte Modellierung von Feature-Modellen beschrieben. Die dargestellte Vorgehensweise ermöglicht einerseits die Repräsentation von Feature-Modellen und andererseits die Überprüfung von Konfigurationen auf Gültigkeit. Dabei werden die einzelnen Features als Klassen definiert und zu jeder Klasse wird zugleich eine Regelklasse erstellt, wobei die Feature-Modellelemente *Mandatory*, *Optional*, *Or*, *Alternative*, *Requires*, *Excludes* in diesen Regelklassen implementiert werden. Eine Konfiguration ist dabei eine Unterklasse dieser Regelklassen. Mit der Inferenz-Maschine wird die Ontologie auf Inkonsistenzen geprüft. Wenn die Konfiguration nicht vollständig ist, wird die Inkonsistenz erkannt. In (Wang, et al., 2007) wird berichtet, dass ein großes Feature-Modell mit rund 1000 Features mit zehn Konfigurationen erstellt wurde, um die maschinelle Inferenz und das Debugging von Feature-Modellen/Konfigurationen zu testen. Dabei stellte sich heraus, dass der Ansatz sehr effektiv und präzise ist. Im Vergleich zu diesem Test haben Bordnetze viel weniger Features (< 100). Basierend auf dieser Vorgehensweise ist die Kombination von VEO und Feature-Modell eine geeignete Methode, um auf gleicher Modellebene die Produktdaten und die Produktvariabilität zu repräsentieren. Die Feature-Modelle bieten neben der kompakten Darstellung der Varianten und Konfigurationsbildung den großen Vorteil, dass die Beziehungen der Produktfeatures präzise dargestellt sind, so dass eine bessere Auswirkungsanalyse und strukturierte Tests durchgeführt werden können.

5.2.3 Änderungsmanagement mit Versionsverwaltungssystem

Das Bordnetz wird von verteilten Teams an verschiedenen Standorten und von verschiedenen Zulieferern entwickelt. Die Entwicklung wird modular aufgeteilt, aber trotzdem sind die Abhängigkeiten stark und die hohe Rate von Änderungen erschweren die Aufrechterhaltung der Konsistenz der Daten. Nach der Entwicklung werden die in Einzeldokumenten gelieferten Module zusammengeführt, und das gesamte Bordnetz wird durch Validierung der Dokumente getestet. Der Prozess beinhaltet viele manuelle Schritte. Bei Änderungen muss der aufwändige Prozess erneut ausgeführt werden. Laut (Kuhn, et al., 2019) verbringen Ingenieure 80% ihrer Zeit mit der Verarbeitung von Änderungen. Daher ist die Automatisierung dieses Prozesses enorm wichtig. Im Software Engineering wird bei der Quellcode-Entwicklung dieses Problem durch Versionsverwaltungssysteme gelöst. Die Versionsverwaltungssysteme ermöglichen es, die Versionen von Quellcode bzw. Dateien zu verwalten, indem die Änderungen nach Benutzerkennungen erfasst und die Versionen archiviert werden. Somit können die Versionen jederzeit wiederhergestellt werden. In diesem

Abschnitt wird betrachtet, wie dieser Lösungsansatz aus dem Software Engineering in die Bordnetzentwicklung übertragen und durch das Versionsverwaltungssystem Git realisiert werden kann.

Versionsverwaltungssystem Git

Git ist ein Versionsverwaltungssystem und nach (Chacon, et al., 2014) das derzeit meistverwendete Versionsverwaltungssystem mit den Haupteigenschaften schnelle Geschwindigkeit, einfaches Design, starke Unterstützung für nichtlineare Entwicklung (Tausende von parallelen Zweigen möglich) und vollständig verteiltes System. In einem verteilten Versionsverwaltungssystem stehen die Dateien nicht nur in einem zentralen Repository, sondern jede Instanz, die an das Projekt angebunden ist, hat ihr eigenes Repository, wobei durch Abgleich die Repositorien synchron gehalten werden. Dadurch kann jeder Projektpartner an dem Projekt lokal arbeiten. Ein Hauptbestandteil von Git sind die Entwicklungszweige, die sogenannten *Branches*. Der Hauptentwicklungszweig wird *Master* genannt. Parallel zum Master können weitere Zweige für verschiedene Versionen, Varianten, Funktionalitäten etc. erstellt werden. So findet eine strukturierte und kontrollierte Entwicklung statt, und die Zweige können später zusammengeführt werden. Die Aktionen in Git werden durch Befehle ausgeführt. Diese können entweder über die grafische Oberfläche oder über Terminal als Code aufgerufen werden. Einige von den wichtigsten Befehlen sind hier als Beispiel aufgelistet:

- *commit*: Änderungen werden im Respository erfasst.
- *push*: Aktualisieren von Remote-Referenzen.
- *pull*: Abrufen von Änderungen aus einem Remote-Repository und integrieren der Daten in ein anderes Repository oder eine lokale Zweigstelle.
- *merge*: Verbindet zwei oder mehrere *Branches* miteinander.
- *revert*: Macht die *Commits* rückgängig.
- *blame***:** Zeigt an, welche Revision und welcher Autor die einzelnen Zeilen einer Datei zuletzt geändert hat.

Git und Knowledge Graph

In den Arbeiten (Arndt, et al., 2019), (Quit Store Project, 2019), (Koneksys, 2019) werden Git-basierten Versionsverwaltungssystemen für RDF-Graphen vorgestellt. Von besonderem Interesse ist der in (Arndt, et al., 2019) beschriebene protypisch implementierte *Quit Store*, bei dem Änderungen in einem separaten Repository gespeichert werden und separat abgefragt werden können. Die Anwendung ist ursprünglich für das Forschungsdatenmanagement in der Geschichtswissenschaft implementiert worden, so dass die Änderungen oder Provenienzen genau die gleiche Relevanz haben wie die Daten. Im *Quit Store* (Abb. 33) werden die RDF-Graphen im N-Quads-Format in der Komponente *Quad Store* in kanonischer Form serialisiert und gespeichert. N-Quads ist ein zeilenbasiertes, transparentes Textformat zur Kodierung

eines RDF-Datensatzes (RDF Working Group, 2014). Der *Quad Store* wird mit dem *Git Repository* synchron gehalten. Um den Zugriff auf den versionierten RDF-Datensatz zu ermöglichen, werden mehrere virtuelle Endpunkte bereitgestellt. Der Default-Endpunkt ermöglicht den Zugriff auf die aktuelle Version des Datensatzes. Für jeden Zweig und für jede Version wird ein zusätzlicher Endpunkt angegeben. Für eine bessere Zugänglichkeit ist immer der aktuelle Datensatz im *Quad Store* gespeichert, wohingegen die älteren Versionen als *Git-Blobs* (*binary large objects*) gespeichert und gegebenenfalls so aufgerufen werden können. Die Komponente *Virtual Graph* stellt dabei die vollständige Historie des Datensatzes zur Verfügung. Die *Quit-API* bietet Abfrage- und Aktualisierungsschnittstellen nach den Semantic-Web-Standards SPARQL und RDF sowie Schnittstellen, um das Git-Repository zu steuern. Die Änderungen werden separat als *Provenance Graph* gespeichert und können ebenso mit SPARQL abgefragt werden.

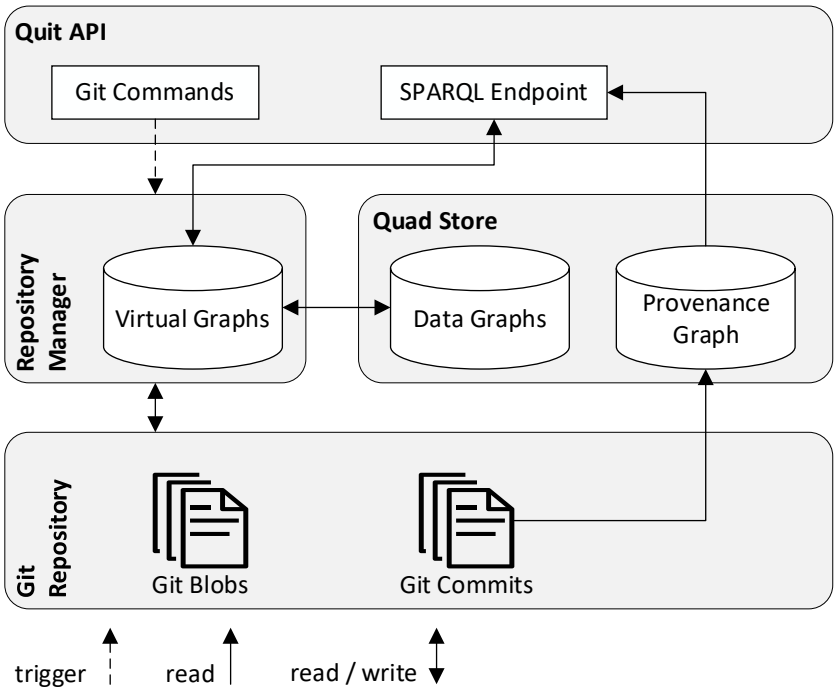


Abbildung 33 Bestandteile von Quit Store nach (Arndt, et al., 2019)

Der Lösungsansatz von *Quit Store* ist sehr gut übertragbar auf die Bordnetzentwicklung. In diesem Knowledge-Graph-basierten Lösungsansatz kann das gesamte Bordnetz, das in RDF-Datensätzen zur Verfügung steht, nach Modulen, also einzelnen Untergraphen (Named Graphs), zerlegt werden und in separaten Branches mit verschieden Teams entwickelt werden. Die Varianten von Modulen können ebenso in weiteren Branches entwickelt werden. Die Entwicklungen können später abgeglichen und am Ende im Master Branch zusammengeführt werden. Eine solche Vorgehensweise bietet für die Bordnetzentwicklung nicht nur ein kontrolliertes Änderungsmanagement, sondern ebenso eine strukturierte, verteilte und iterative Vorgehensweise, und darüber hinaus kann zugleich die Entwicklung von Varianten besser bewältigt werden.

5.2.4 Datenorientierte und modellbasierte Vorgehensweise

In den vorangegangenen Abschnitten wurde dargestellt, wie die Knowledge-Graph-Technologien die Möglichkeiten bieten, die Defizite in der Bordnetzentwicklung zu beheben. Für die Realisierung und Aufrechterhaltung dieses Lösungsansatzes ist die Umstellung vom funktionsorientierten Ansatz zum datenorientierten Ansatz vorgesehen. Beim datenorientierten Ansatz werden die Daten unabhängig von den Entwicklungswerkzeugen analysiert, und je nach den jeweiligen Anforderungen wird ein Datenmodell und ein gemeinsames Repository angestrebt. Beim funktionsorientierten Ansatz hingegen stehen die Entwicklungswerkzeuge im Mittelpunkt und die Daten werden in den jeweiligen Werkzeugen verarbeitet, so dass die Ergebnisse am Ende oft nebeneinanderstehen. Beim Knowledge-Graph-basierten Ansatz greifen die Werkzeuge auf einen gemeinsamen Knowledge Graph zu, und auf diese Weise kann eine bessere Interoperabilität zwischen den Werkzeugen erreicht werden. Bei Bedarf wird die Ontologie, die den Graphen beschreibt, so erweitert, dass die Werkzeuge weiterhin einwandfrei arbeiten können. Mit dieser Vorgehensweise wird auch das *Model-based System Engineering (MBSE)* realisiert. Der International Council on Systems Engineering (INCOSE) (INCOSE, 1990) definiert das System Engineering wie folgt: *„Systems Engineering is a transdisciplinary and integrative approach to enable the successful realization, use, and retirement of engineered systems, using systems principles and concepts, and scientific, technological, and management methods"*. Wie in der Definition sehr klar mit den Begriffen transdisciplinary und integrative ausgedrückt ist, ist die interdisziplinäre Zusammenarbeit eine Grundvoraussetzung für das Systems Engineering, um die erfolgreiche Realisierung eines technischen Produkts zu ermöglichen. Im Jahr 2007 wurde von INCOSE MBSE folgende Vision für 2020 formuliert: *"Model-based[7] systems engineering […] MBSE is part of a long-term trend toward model-centric approaches adopted by other engineering disciplines, including mechanical, electrical and software. In particular, MBSE is expected to replace the document-centric approach that has been practiced by systems engineers in the past and to influence the future practice of systems engineering by being fully integrated into the definition of systems engineering processes"* (MBSE Initiative, 2007). Der Ausgangspunkt für den modellbasierten Ansatz des System Engineering war, dass diskrete Dokumente in traditionellen dokumentenbasierten Systemen ein wesentliches Hindernis für die Erreichung von MBSE darstellten (Hart, 2015). Darüber hinaus beschäftigt sich das Ontology Action Team (OAT, 2013), eine Initiative von INCOSE, mit der praktischen Nutzung von Ontologien zur Verbesserung des MBSE, und es beschreibt die Vorteile des ontologiebasierten Ansatzes wie folgt „*In particular, Ontology is an enabler of good modeling in that it focuses on establishing well defined domain concepts in terms of the terminology, definitions, and relationships as needed to model real world applications. In*

[7] Model-Based Engineering: "*An approach to engineering that uses models as an integral part of the technical baseline that includes the requirements, analysis, design, implementation, and verification of a capability, system, and/or product throughout the acquisition life cycle"* (Hart, 2015).

addition, the use of formal semantics is essential for modeling languages to properly represent the concepts, and to enable additional analysis to be performed about the systems. The combination of a controlled vocabulary and underlying formalism provides the opportunity to create more consistent models and improve semantic interoperability". Auch die NASA (National Aeronautics and Space Administration) empfiehlt (NASA, 2013) und berichtet über die erfolgreiche Umsetzung von MBSE mit Ontologien und Knowledge Graphs (Stardog, 2019). Daher verbessert der Einsatz von Knowledge-Graph-Technologien nicht nur den Entwicklungsprozess, sondern modernisiert auch die Systeme in Richtung MBSE.

5.2.5 Zusammenfassung

In diesem Abschnitt wurde gezeigt, wie die Defizite bei der Entwicklung von Bordnetzen mit Hilfe des Knowledge-Graph-basierten Ansatzes beseitigt werden können. Zunächst wurde gezeigt, dass die Bereiche Elektrologik und 3D-Entwicklung auf der Datenebene semantisch für kollaboratives Arbeiten integriert werden können. Es wurde gezeigt, dass auf Basis von VEO ein semantisches und integriertes Produktmodell realisierbar ist. Darauf aufbauend wurde erläutert, wie Methoden des Software Engineering, das Feature-Modell und das Versionsverwaltungssystem Git zur Handhabung der Bordnetzvarianten und -versionen in der Entwicklung sowie zur Realisierung eines kontrollierten Änderungsmanagements und einer verteilten Entwicklung eingesetzt werden können. Dieser Ansatz bietet neben der Erfüllung der Anforderungen die Realisierung von modellbasiertem System Engineering (MBSE) und die Umsetzung der neuen Anforderungen *Digital Twin* und *Digital Thread*. Demgegenüber bieten die existierenden Legacy-Systeme keine effektiven konkreten Lösungen. Der vorgestellte Ansatz ist modellbasiert und datenorientiert, während die existierenden Legacy-Systeme dokumentenbasiert sind. In dokumentenbasierten Systemen lässt sich der modellbasierte Ansatz nicht eindeutig umsetzen, da eine Diskrepanz zwischen Dokument und Modell besteht und jene daher oft nicht synchron gehalten werden können. Die folgende Tabelle 11 fasst die Vorteile von Knowledge-Graph-basierten Lösungen gegenüber Legacy-Systemen zusammen. Der nächste Abschnitt zeigt dann, wie die informationstechnischen Anforderungen zur Realisierung normgerechter Zuverlässigkeitsanalyse auf der Grundlage dieses Ansatzes umgesetzt werden können.

Kriterien	Knowledge Graph	Legacy Systems
Vorgehensweise	Modelbasierte und datenorientierte Architektur	Dokumentenbasierte und softwareorientierte Architektur
Datenhaltung	Im RDF-Triplestore und verteilt im Netz durch URIs	P2P-Datenaustausch oder All-in-One-Softwarelösungen
Variantenmanagement	Modelbasiert mit Feature-Modell	Code-basiert in Software
Versionsmanagement	Versionsverwaltungssysteme	Keine konkrete Lösung
Technische Implementierung	Konkrete Werkzeuge für Anforderungen, daher transparent	Blackbox-Implementierungen
Realisierung von MBSE	Mit Ontologie und Instanzen eindeutig und verfolgbar	Diskrepanz zwischen Modell und Dokumenten
Digital Twin	Semantisch eindeutig und identifizierbar im Netz durch URI	Dokumente in den Softwaresystemen
Digital Thread und Rückverfolgbarkeit	Mit OSLC	Keine konkrete Lösung

Tabelle 11 Vergleich von Knowledge-Graph-basiertem Lösungsansatz und Legacy-Systeme

5.3 Knowledge-Graph-basierte Realisierung normgerechter Zuverlässigkeitsanalyse

In den vorherigen Abschnitten wurde ausführlich erklärt, wie die Knowledge-Graph-Technologien für die Behebung der Defizite in der Datenverarbeitung und in der Bordnetzentwicklung eingesetzt werden können. Darauf aufbauend wird in diesem Abschnitt die Umsetzung der dargestellten informationstechnischen Anforderungen „Rückverfolgbarkeit“ und „Semantische Validierung“ zur Realisierung normgerechter Zuverlässigkeitsanalyse behandelt.

5.3.1 Rückverfolgbarkeit mittels Semantic Web

Die Realisierung der Rückverfolgbarkeit von Arbeitsergebnissen ist eine obligatorische Aufgabe in der funktionalen Sicherheit gemäß der Norm ISO 26262. Die Aufgabe ist unabdingbar, und das zugrundeliegende System muss in der Lage sein, die Abhängigkeiten aufzuzeigen und damit nachzuweisen, dass und wie die Sicherheitsanforderungen (*Safety Requirements*) umgesetzt werden, z. B. in welchen konkreten Simulationen, Testfällen etc. Die Architektur des Semantic Web ist sehr gut geeignet, solche Beziehungen semantisch darzustellen. Es existiert auch eine Initiative namens *Open Services for Lifecycle Collaboration* (OSLC), die die Semantic-Web-Architektur als Grundlage nutzt und Spezifikationen veröffentlicht, um Daten, Domänen und Anwendungen in einem Unternehmen zu vernetzten und somit Interoperabilität der Systeme zu erreichen (Abb. 34). Die Arbeiten von OSLC wurden im Jahr 2009 bei IBM als internes Projekt begonnen und haben sich so weiter entwickelt, dass die Initiative seit 2013 Mitglied von OASIS (*Organization for the Advancement of Structured Information Standards)* ist und das offene Projekt heute von weiteren namhaften Unternehmen verwendet und unterstützt wird (OASIS, 2013). Die zugrundeliegende Idee ist, dass die Daten und Informationen der Werkzeuge mittels URIs offen zugänglich gemacht werden, so dass auch andere Werkzeuge darauf zugreifen können.

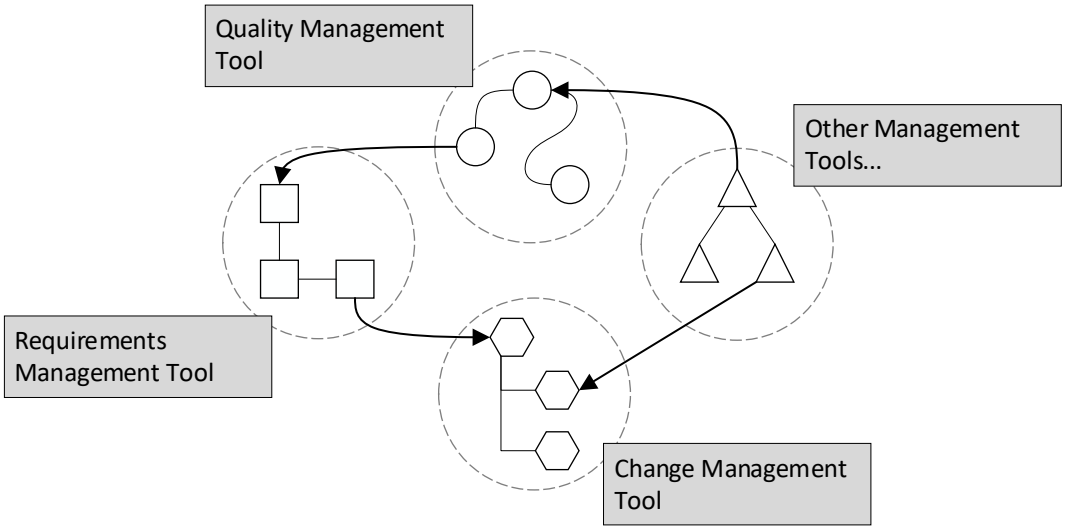


Abbildung 34 Linked Lifecycle Data nach (Rigolet, 2011)

Die Grundlage bildet das RDF, und die Beziehungen und Beschreibungen der Ressourcen basieren auf den Vorgaben bzw. Spezifikationen der OSLC-Initiative. Auf diese Weise kann z. B. eine Anforderung semantisch mit einem Testfall verknüpft werden, obwohl beide von unterschiedlichen Werkzeugen erstellt werden. Nach den OSLC-Spezifikationen muss die Anforderung in einer mit oslc_rm:Requirement[8] typisierten Ressource und der Testfall in einer mit oslc_qm:TestCase[9] typisierten Ressource stehen. Danach können diese Ressourcen mit der Beziehung oslc_rm:validatedBy, ausgehend von der Anforderung zum Testfall, verknüpft werden. Somit wird ein semantisches Netz zwischen den heterogenen Arbeitsprodukten der Werkzeuge erstellt. Einem URI können verschiedene Arbeitsprodukte wie Quelltexte von Simulationen, Tabellen, XML-Dateien etc. zugeordnet werden. Der Zugriff auf die Daten erfolgt über HTTP (Hypertext Transfer Protocol), eine SPARQL-Schnittstelle oder über eine REST-API (Representational State Transfer Application Programming Interface), die den Austausch von Informationen auf verschiedenen Systemen ermöglicht.

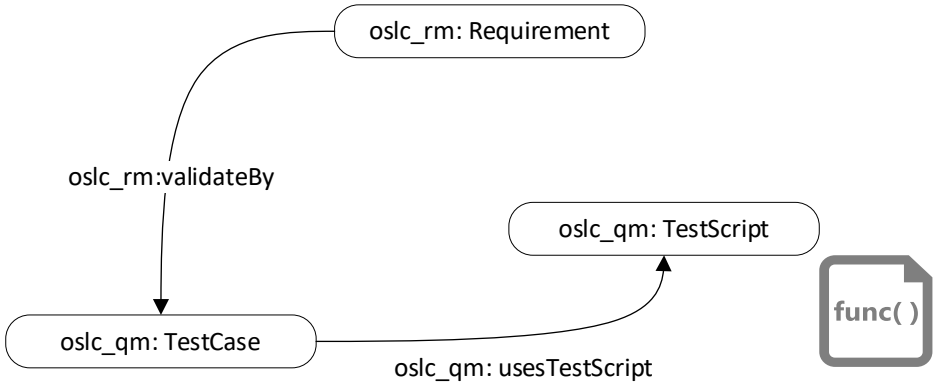


Abbildung 35 OSLC Beispiel

In (Bräuchle, et al., 2015) wird berichtet, dass in dem CRYSTAL-Projekt *(Critical Systems Engineering Acceleration)* (CRYSTAL-Projekt, 2016) eine Integrationsplattform für die Automotive-Domäne mit OSLC entwickelt wurde, die die Verwaltung von Anforderungen, Tests, Simulationen und Co-Simulationen umfasst. In der Arbeit wird basierend auf dem Szenario „automatische Validierung der Anforderungen“ von ersten Erfahrungen, Erkenntnissen und vorläufigen Resultaten mit der Plattform berichtet. Als Nachteil wird dort genannt, dass keines der verwendeten Tools (Tab. 12) web-konform war, so dass der Implementierungsaufwand hoch war. Dennoch wird berichtet, dass *„die Automatisierung der Tool-Schnittstellen eine höhere Transparenz, beschleunigte Arbeitsabläufe und verbesserte Ergebnisqualität“* ermöglicht. Im Allgemeinen handelt es sich bei vielen Entwicklungswerkzeugen im Engineering um Desktop-Anwendungen, und dies gilt auch für die Bordnetzentwicklung, so dass webbasierte Lösungen eine Anwendungsmodernisierung erfordern. Die OSLC-Initiative bietet verschiede Frameworks wie OSLC4Net, Eclipse Lyo

[8] oslc_rm: http://open-services.net/ns/rm#
[9] oslc_qm: http://open-services.net/ns/qm#

und OSLC4JS etc. (OSLC, 2019) um verschiedene Anwendungen, die auf verschiedenen Plattformen entwickelt werden, OSLC-konform zu erweitern.

Tool	Einsatzzweck im Demonstrator	Verwendete Schnittstellen
PTC Integrity Lifecycle Manager	Verwaltung der Anforderungen und Testfälle im Entwicklungsprozess	• ReqIF (Requirements Interchange Format) zum Import des Kundenlastenhefts • OSLC-RM (Requirements Management) Provider zur Anbindung von MBSE Tools • OSLC-Asset Client zur Kopplung von Systemlastenheft und Sicherheitskonzept • OSLC-QM (Quality Management) Provider zur Anbindung von Testausführungs-Tools
IBM DOORS	Quelle des Kundenlastenhefts	• ReqIF zur dateibasierten Übertragung des Kundenlastenhefts
PTC Integrity Modeler	Entwurf der Systemarchitektur unter Verwendung von SysML	• OSLC-RM Consumer zur Repräsentation textbasierter Anforderungen
Sparx Enterprise Architect	Erstellen des Funktionalen Sicherheitskonzepts nach ISO 26262	• OSLC-RM Consumer zur Repräsentation der Sicherheitsziele und Update von Sicherheitsfunktionen in der Systemspezifikation • OSLC-Assset Provider zur Kopplung des Modells an das Lifecycle Management Werkzeug
ICOS	Virtuelle Integrationsplattform zur Durchführung von dynamischen Co-Simulationen mit heterogenen Simulationstools	• FMI (Functional Mockup Interface) zur dynamischen Kopplung verschiedener domänenspezifischer Simulationsmodelle
AVL VeVaT/Magic	Testplanung und Analyse der Simulationsergebnisse	• OSLC-RM Client zur Darstellung der getesteten Anforderungen • OSLC-QM Client zur Aktualisierung von Teststatus und -Ergebnis

Tabelle 12 Im CRYSTAL-Projekt verwendete Tools, Funktionalitäten und Schnittstellen (Bräuchle, et al., 2015)

Die Bordnetzdaten, die unter VEO als RDF-Daten zur Verfügung stehen, sind OSLC-kompatibel und können mit anderen Informationen, wie z. B. Sicherheitsanforderungen, EMV-Simulation, CAD-Modell etc., semantisch verknüpft werden, so dass über die semantischen Verknüpfungen die Rückverfolgbarkeit realisiert wird und über die Navigation auf den Links die notwendigen Informationen gefunden werden können.

5.3.2 Unit-Test-basierte Validierung der Sicherheitsanforderungen

Um die definierten Sicherheitsanforderungen zu erreichen, müssen entsprechende quantitative und qualitative Zuverlässigkeitsanalysen durchgeführt werden (Abschnitt 4.1). Besonders bei den Anforderungen mit höheren ASIL-Stufen ist man gefordert, eine Ausfallrate zu erreichen, die unter dem zugelassenen FIT-Wert steht. Die Ausfallrate hängt von verschiedenen Parametern ab, wie Komponenteneigenschaften, Betriebstemperatur, elektrische Wärmeleistung etc. Um einen aussagekräftigen Wert zu ermitteln, muss

sichergestellt werden, dass das ausgewertete Bordnetz vollständig und konsistent ist und gleichzeitig die verwendeten Daten, wie Konfiguration, Simulationsmodell, Testszenarien etc., vollständig zur Rückverfolgung dokumentiert sind, damit die Ursachen der unerwünschten Ergebnisse verfolgt werden können. Die Knowledge-Graph-Technologien sind an dieser Stelle sehr nützlich, nicht nur für die semantische Verknüpfung von Daten, sondern darüber hinaus kann mit Regeln die Datenvollständigkeit und Konsistenz gewährleistet werden. Um den Nutzen zu veranschaulichen, wird folgender Fall top-down betrachtet:

(1) Angenommen, die Ausfallrate eines elektrischen Kontakts verdoppelt sich bei 110°C Betriebstemperatur. Dies könnte mit einer Regel ausgewertet werden:

```
(operatingTemperatur + heatingOnContact) > 110°C -> FailureRateOfContact x 2
```

(2) Die Betriebstemperatur ist abhängig von der Zone, wo der Kontakt sich befindet, und kann ebenso mit Regeln bearbeitet werden und die Bordnetzdaten so erweitert werden:

```
contact isIn engineCompartment -> operatingTemperaturOfContact = 100°C
```

(3) Dagegen werden die Eigenerwärmungswerte von Kontakten von der Simulation abgeleitet und können nach der Simulation, semantisch an die Daten verknüpft werden:

```
contact hasHeatingBasedOnSimulinkSimulation 10°C
```

(4) Um zu einem qualitativen Wert in Punkt (3) zu gelangen, ist neben der Simulation auch die Qualität der Bordnetzdaten wichtig, da die Simulationsmodelle daraus generiert werden. Daher müssen die Bordnetzdaten vollständig und fehlerfrei sein. Auch zu diesem Zweck – nämlich um Fehlerfreiheit und Vollständigkeit zu erreichen - können Regeln aufgestellt werden, in denen die Konstruktion, Richtlinien oder das Wissen von Ingenieuren formalisiert werden (Angele, et al., 2008) und semantische Datenqualitätsüberprüfungen ausgeführt werden. Zur Durchführung der Konsistenzprüfung (*), ob die Nennstrom- und Betriebsstromwerte übereinstimmen, muss die Datenvollständigkeit (**) sichergestellt werden, indem geprüft wird, ob die Komponente die erforderlichen Attribute aufweist und ob die Aktuatoren an die Sicherungen angeschlossen sind.

```
(*) (fuse ratedCurrent ?x) and (actuator operatingCurrent ?y) and
    (actuator connectTo fuse) and (?x >= ?y)
(**) (fuse has ratedCurrent) and (actuator has operatingCurrent) and
     (actuator connectTo fuse)
```

Basierend auf den Erkenntnissen aus dem Forschungsprojekt wird, wie im vorliegenden Fall gezeigt, in dieser Arbeit ein strukturiertes Vorgehen in fünf Schritten zur Validierung der quantitativen Zuverlässigkeit vorgeschlagen.

1. Bestimmung der Konfiguration: Im ersten Schritt ist die Erstellung einer Bordnetz-Konfiguration in Abhängigkeit von den Sicherheitsanforderungen erforderlich. In diesem Schritt ist das Feature-Modell (Abschnitt 5.2.2) ein nützliches Werkzeug, da die Features und ihre gegenseitigen Auswirkungen getestet und die Gültigkeit der Konfigurationen überprüft werden können.

2. Überprüfung der Vollständigkeit: Im zweiten Schritt kann durch die SHACL-Regeln (Abschnitt 5.1.4) die Datenvollständigkeit der erstellten Konfiguration überprüft werden.

3. Überprüfung der Konsistenz: Darauf aufbauend können in einem dritten Schritt die Konsistenzüberprüfungen durchgeführt werden. Die Erfassung der Regeln in SHACL ermöglicht die Wiederverwendung der Regeln in verschiedenen Projekten und Fahrzeugmodellen. Wenn die Regeln nicht ausreichen, können entweder die SHACL-Regeln mit Funktionen erweitert oder die RDF-APIs verwendet werden. Dabei können die Regeln nach Mustern sortiert werden. In (Kontokostas, et al., 2014) sind *Data Quality Test Patterns* definiert, um die Datenqualität strukturiert und testbasiert zu überprüfen.

4. Ausführung der Simulation: Im vierten Schritt werden Simulationsmodell und Testszenarien erstellt und ausgeführt und die Ergebnisse mit den Daten verknüpft.

5. Überprüfung der Ergebnisse: Im letzten Schritt können wieder SHACL-Regeln verwendet werden, um die Komponenten nach Zuverlässigkeitskriterien auszuwerten und gleichzeitig zu validieren, ob die Sicherheitsanforderungen erfüllt sind, z. B. mit einer solchen Regel: `(sectionOfEngineWiringHarness > 1 FIT) -> (RequirementForWiringHarness is Invalid)`.

Alle notwendigen Regeln und Algorithmen können in Unit-Tests gekapselt werden. Dies ermöglicht zum einen eine strukturierte Erfassung und zum anderen können sie kontrolliert auf Daten durchgeführt werden. In der Softwareentwicklung wird das Unit-Testing eingesetzt, um Funktionseinheiten (Units) von Computerprogrammen auf korrekte Funktionalität zu überprüfen (Witte, 2019). Der modulare Aufbau (Abb. 36) eignet sich sehr gut für die

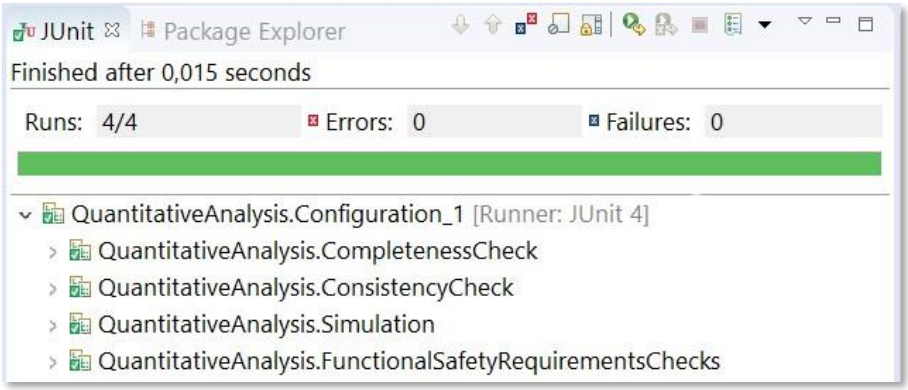


Abbildung 36 Beispiel für die Skript-Ausführung eines Unit-Tests in Java mit JUnit (JUnit, 2021)

Realisierung der vorgestellten Validierungsschritte quantitativer Zuverlässigkeitsanalysen. Die Unit-Test-Struktur gibt einen Überblick, welche Teile funktionieren und welche nicht, so dass Inkonsistenzen nachvollzogen werden können. Andererseits können die durchgeführten Aktivitäten und das Wissen strukturiert erfasst werden, so dass sie dauerhaft gepflegt und in anderen Projekten wiederverwendet werden können.

In (Witte, 2019) wird beschrieben, dass nach ISTQB (*International Software Testing Qualifications Board*) ein standardisierter Test aus den fünf Hauptaktivitäten *(1) Planung und Steuerung, (2) Analyse und Design, (3) Realisierung und Durchführung, (4) Auswertung und Bericht, (5) Abschluss* besteht. Die OSLC-Qualitätsmanagement-Spezifikation beschreibt, wie die Aktivitäten der ISTQB basierend auf RDF erfasst werden können. In dieser Weise kann die Testung rückverfolgbar abgelegt werden. Die Aktivitäten werden durch die folgenden fünf Klassen definiert (OSLC, 2011):

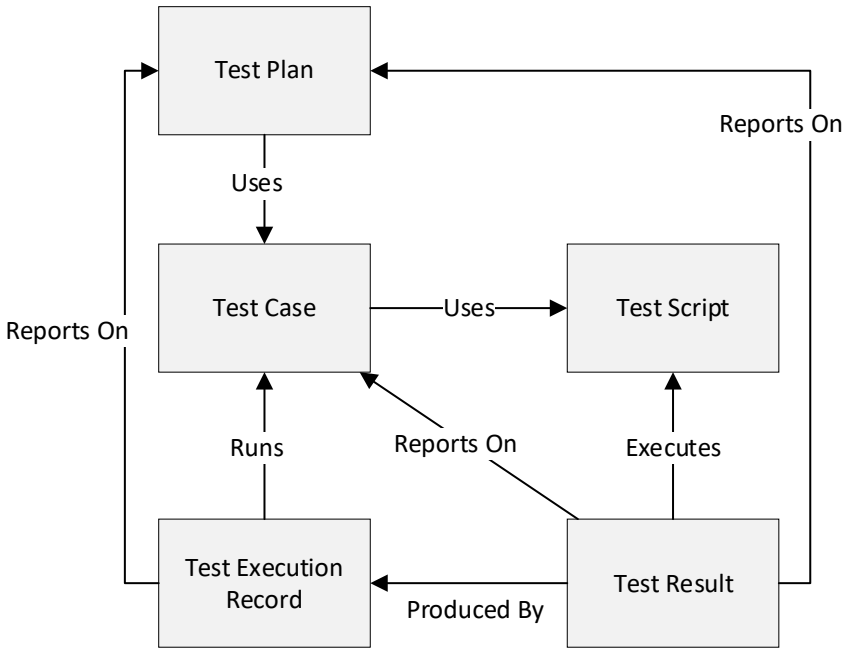


Abbildung 37 Qualitätsmanagement-Spezifikation (OSLC, 2011)

- *TestPlan*: Definiert den Gesamtprozess und die Strategie für den Test eines Systems.
- *TestCase*: Definiert die Kriterien, die bestimmen, ob ein System unter bestimmten Umständen das richtige Verhalten zeigt.
- *TestScript*: Definiert ein Programm oder eine Liste von Schritten, die zur Durchführung eines Tests verwendet werden.
- *TestExecutionRecord*: Planung zur Durchführung eines Tests.
- *TestResult*: Beschreibt das Ergebnis des Versuchs, einen Test auszuführen.

Knowledge-Graph-Technologien ermöglichen zusammen mit OSLC und Unit-Tests eine semantische, strukturierte und nachvollziehbare Validierung. Das Unit-Test-Verfahren eignet sich nicht nur für die quantitative Analyse des Bordnetzes, sondern kann auch dazu verwendet

werden, andere Analysen wie EMV-Simulation, Optimierungsalgorithmen etc. in den Unit-Tests zu kapseln und so einen vollständigen Überblick über alle Analysen am Bordnetz zu erhalten. Das Verfahren bietet die Möglichkeit, die Qualität des Bordnetzes bereits in der Entwicklungsphase strukturiert zu erfassen und zu sichern. Neben den quantitativen Zuverlässigkeitsanalysen können auch die qualitativen Zuverlässigkeitsanalysen in das Verfahren eingebettet werden. In den qualitativen Analysen wie FMEA und FTA werden die Fehlermuster an den Produkten auf der Grundlage des Wissens der Ingenieure untersucht und beseitigt. Knowledge-Graph-Technologien ermöglichen es, dieses Wissen zu formalisieren und damit die Analysen zu automatisieren. In den Arbeiten (Rehman, et al., 2016) (Venceslau, et al., 2014) wird eine ontologie- und regelbasierte Vorgehensweise für FMEA und FTA vorgeschlagen und in (Steenwinckel, et al., 2018) ist diese Idee umgesetzt. Ein weiterer Vorteil dieses Vorgehens ist, dass die Prozesse bei Änderungen automatisch wiederholt werden können und Fehlerorte schnell erkannt und die erforderlichen Maßnahmen schnell angepasst werden können.

5.3.3 Zusammenfassung

In diesem Abschnitt wurde gezeigt, wie auf der Grundlage des Knowledge-Graph-basierten Ansatzes die informationstechnischen Anforderungen für die Realisierung der normgerechten Zuverlässigkeitsanalyse umgesetzt werden können. Dabei ist zu beobachten, dass der Ansatz viele Vorteile mit sich bringt. Zunächst wurde gezeigt, dass die Semantic-Web-Architektur und die Spezifikationen von OSLC die große Möglichkeit bieten, die Sicherheits-anforderungen und deren Umsetzung zueinander semantisch zu verknüpfen, so dass sie rückverfolgbar sind. Darauf aufbauend wurde erklärt, wie durch die Unit-Tests die Zuverlässigkeitsanalyse strukturiert durchgeführt und validiert werden kann. Die Knowledge-Graph-Technologien und das Feature-Modell helfen formal sicherzustellen, dass das analysierte Bordnetz eine gültige Konfiguration ist und dass die Daten vollständig, konsistent und die Ergebnisse rückverfolgbar sind. Dieser Ansatz kann sowohl für quantitative und qualitative Zuverlässigkeitsanalysen als auch für andere Analysen verwendet werden. Daher ist dieser Ansatz keine Insellösung, sondern ein modular erweiterbarer Ansatz.

6 Prototypische Realisierung

In diesem Kapitel wird die Software, die in dem vorgestellten Forschungsprojekt „Bordnetz Zuverlässigkeit“ (Abschnitt 1.1) entwickelt wurde, vorgestellt. Die zuverlässigkeitsbezogenen technischen Ergebnisse des Projekts wurden in Abschnitt 4.1.1 erwähnt. In diesem Kapitel werden daher die technischen Eigenschaften der Software betrachtet. Der Arbeitsablauf der Software, die Architektur und die eingesetzten Technologien werden erklärt. Dabei wird gezeigt, wie die Knowledge-Graph-basierte Datenverarbeitung realisiert wird. Am Ende des Kapitels werden die während der Entwicklung gewonnenen Erkenntnisse (*Lessons Learned*) erläutert.

6.1 Funktionsweise der Software

Das *Wiring Harness Analysis Tool* (WHAT) wurde im Fachgebiet „Fahrzeugsysteme und Grundlagen der Elektrotechnik“ der Universität Kassel interdisziplinär entwickelt. WHAT ist eine Desktop-Anwendung, die die eingegebenen Daten des Bordnetzes auswertet und einen Zuverlässigkeitsbericht erstellt. WHAT läuft in einer Java-Umgebung und die Simulationen werden in einer Matlab-Umgebung ausgeführt. Der Datenaustausch zwischen den beiden Umgebungen läuft automatisch über eine XML-Schnittstelle. Der Arbeitsablauf erfolgt in neun Arbeitsschritten.

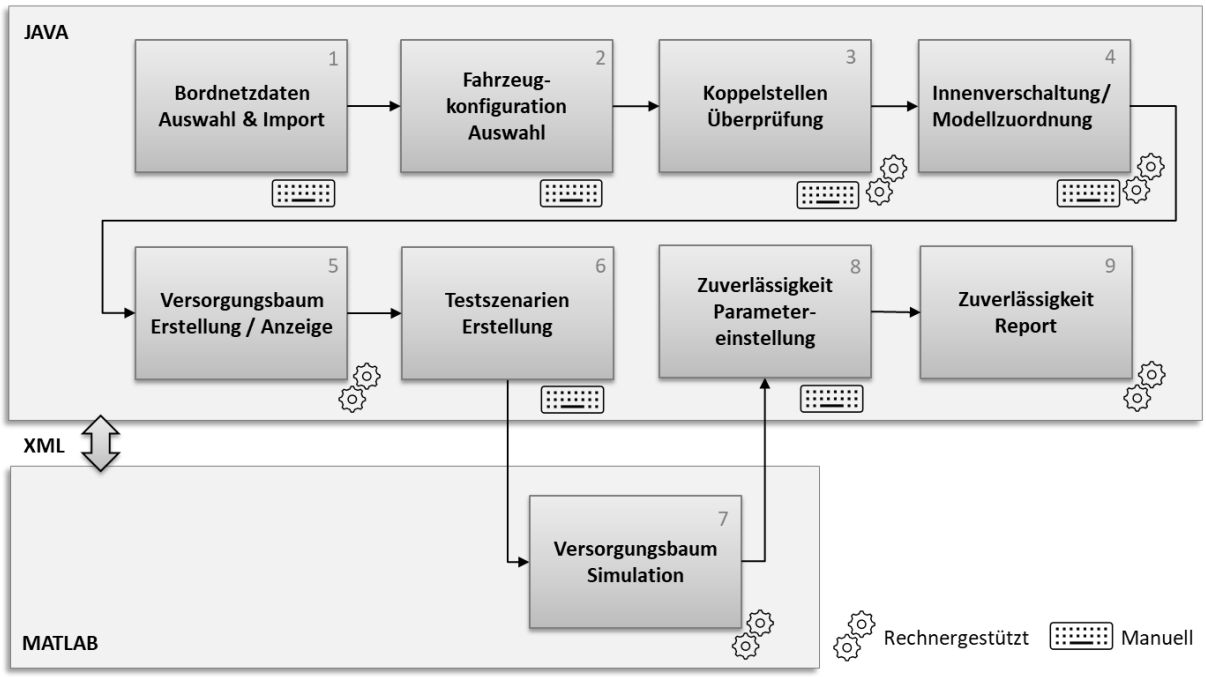


Abbildung 38 Arbeitsablauf von WHAT

1. Auswahl und Import von Bordnetzdaten: Im ersten Schritt werden die Bordnetzdaten von verschiedenen Quellen ausgewählt und importiert. Dabei dienen als Hauptdatenquelle die KBL-Daten. Da die KBL-Daten aber das physikalische Bordnetz nicht vollständig

beschreiben, um eine elektrische Simulation zu realisieren, werden zusätzliche Daten aus verschiedenen Quellen geholt und integriert.

2. Auswahl der Fahrzeugkonfiguration: Die Bordnetzdaten bestehen aus Modulen. Eine zulässige Kombination dieser Module ist eine Bordnetzkonfiguration. In diesem Schritt können Domänenexperten eine Konfiguration durch eine Zusammenstellung dieser Module erstellen.

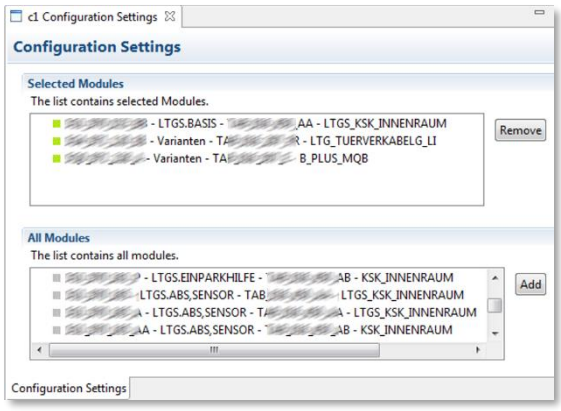


Abbildung 39 Konfigurationserstellung

3. Koppelstellen Überprüfung: Nach dem Erstellen der Konfiguration werden zunächst die Koppelstellen (Steckerverbindungen) konfigurationsbezogen überprüft und automatisch nach den Kriterien Steckerbezeichnung (1), räumlicher Abstand der Stecker (2), Eigenschaften der verbundenen Leitungen (3) miteinander gekoppelt. Nach Durchlaufen der Reihenfolge kann der Domänenexperte die Ergebnisse überprüfen und manuell erweitern. Die gefundenen Koppelstellen werden grün und die nicht gefundenen rot markiert.

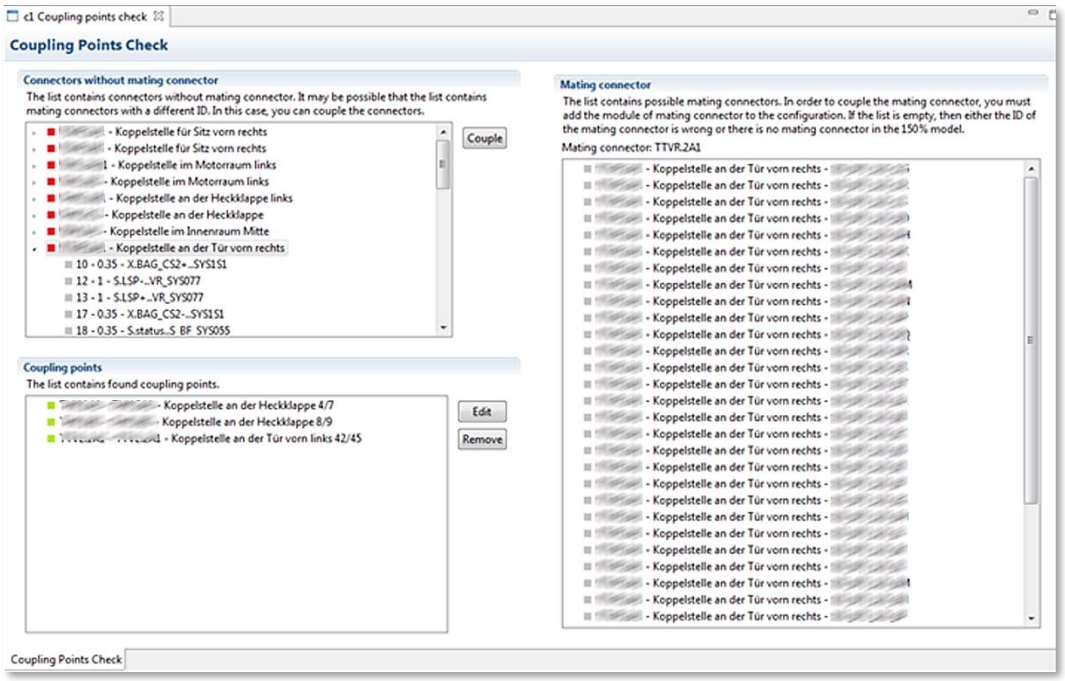


Abbildung 40 Koppelstellen-Überprüfung

4. Innenverschaltung und Modellzuordnung: Die Innenverschaltung der elektrischen und elektronischen Komponenten läuft wie bei der Koppelstellenüberprüfung automatisch und kann nachgepflegt werden. In diesem Schritt können zusätzlich die Simulationsmodelle den elektrischen Verbrauchern zugeordnet sowie Potentialverteiler der Sicherung und Stromwerte angepasst werden. Gleichzeitig bietet die Benutzeroberfläche eine Übersicht, die elektrische und elektronische Komponenten der gewählten Konfiguration beinhaltet.

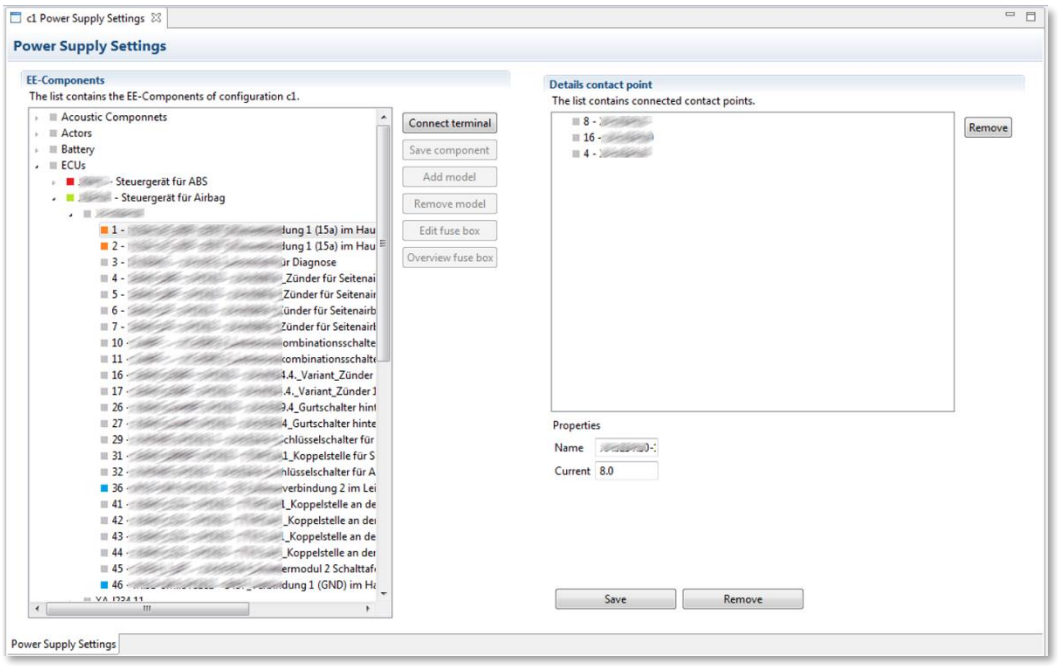


Abbildung 41 Einstellung des Versorgungsbaums

5. Erstellung von Versorgungs- und Massebaum: In diesem Schritt werden automatisch Versorgungs- und Massebaum aus vorhanden Daten generiert und visuell dargestellt. Die übersichtliche Darstellung ermöglicht es Domänenexperten, die Ergebnisse besser auf Vollständigkeit und Korrektheit zu analysieren.

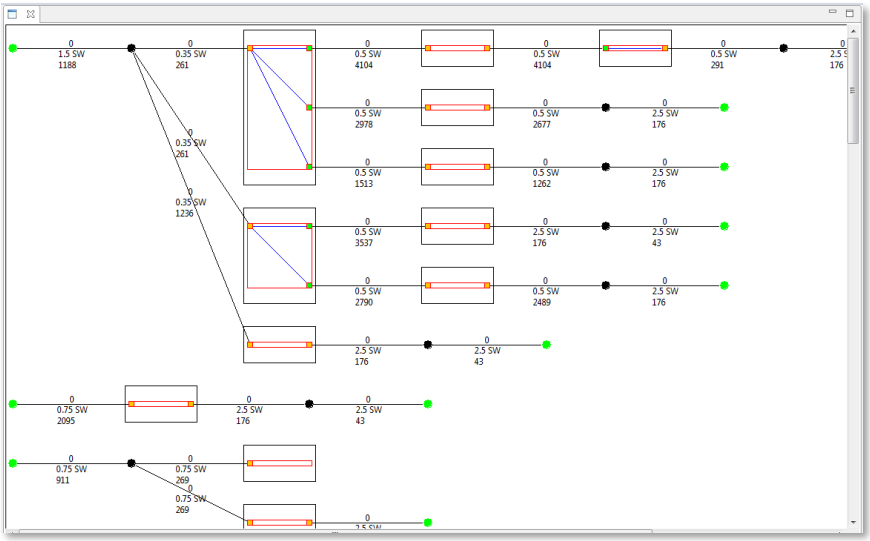


Abbildung 42 Visuelle Darstellung von Versorgungs- und Massebaum

Bei der Erstellung werden die Versorgungsleitungen von Sicherungen bis zu Massepunkten verfolgt und so der Versorgungs- und Massebaum erzeugt. Sicherungen und Masse-punkte werden in grüner Farbe dargestellt (Abb. 42). Der Prozess generiert simulierbare Schaltungen aus konstruktionsbasierten Daten und stellt deren Konsistenz sicher.

6. Erstellung des Testszenarios: Der Testszenario-Editor ermöglicht die Erstellung von Testszenarien durch die *Gantt-Chart*-Darstellung, basierend auf der gewählten Konfiguration. In der Gantt-Chart-Ansicht findet die eigentliche Bearbeitung des Gantt-Diagramms statt. Auf der linken Seite befindet sich die Liste der verfügbaren Aktionen, und ihre Namen basieren auf der Bezeichnung des Verbrauchers. Die Aktionen beziehen sich auf die Aktivierung der elektrischen Verbraucher. Auf der rechten Seite befindet sich der Editorbereich. Direkt über diesem wird ein Zeitstrahl angezeigt. Ein Gantt-Diagramm ist hierarchisch aufgebaut. Das Diagramm beinhaltet Zeilen, welche wiederum Aktionen beinhalten. Zeilen fassen Aktionen vom gleichen Typ zusammen. Sie sind selbst ebenfalls von einem bestimmten Typ, der stets mit dem der beinhalteten Aktionen übereinstimmt. Er wird am linken Rand des Editorbereichs auf Höhe der Zeile angezeigt. Es ist nicht möglich mehr als eine Zeile pro Typ in einem Gantt-Diagramm zu platzieren. Aktionen sind die kleinsten Elemente eines Gantt-Diagramms. Jede Aktion ist von einem bestimmten Typ, der sie klassifiziert. Sie können nur in einer Zeile vom gleichen Typ platziert werden. Außerdem besitzen Aktionen einen Start- und Endzeitpunkt sowie eine Dauer. Eine Aktion muss erst beendet sein, bevor eine andere vom gleichen Typ beginnt, da es sonst zu einer Überlappung kommt. Ein Sonderfall von Aktionen sind Gruppenaktionen. Gruppenaktionen fassen mehrere einzelne Aktionen zusammen und können daher als Ausschnitt eines Gantt-Diagramms verstanden werden. Die Dauer einer Gruppenaktion entspricht dem Endzeitpunkt der letzten darin enthaltenen einzelnen Aktion. Die Gantt-Diagramm-Darstellung ermöglicht somit eine übersichtliche Darstellung der Aktionen der Verbraucher in einem Testszenario.

Abbildung 43 Testszenario-Editor

7. Simulation: Nach der Erstellung von Versorgungsbaum und Testszenario werden aus beiden Daten ein Matlab-Simulink-Modell generiert, in der Matlab-Umgebung ausgeführt und die Ergebnisse in die zugehörigen Daten integriert.

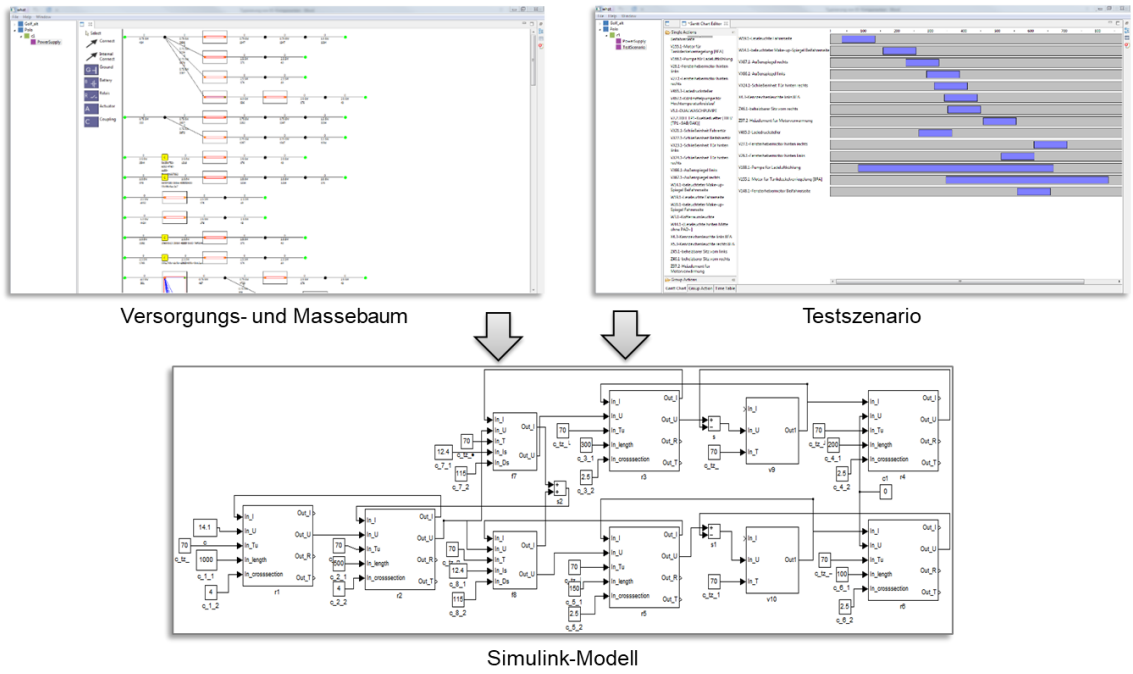


Abbildung 44 Erstellung einer Simulation in WHAT

8. Einstellung von Parametern: Die Zuverlässigkeitsparameter und Regeln bestimmen die quantitative Auswertung des Bordnetzes. Im vorletzten Schritt können die Standardparameter zur Analyse der Auswirkungen zusätzlich per Bedienfeld eingestellt werden.

9. Zuverlässigkeitsbericht: Im letzten Schritt wird die quantitative Auswertung durchgeführt und es werden Zuverlässigkeitsberichte erstellt, die die Ausfallrate der physikalischen Bordnetzkomponenten übersichtlich darstellen.

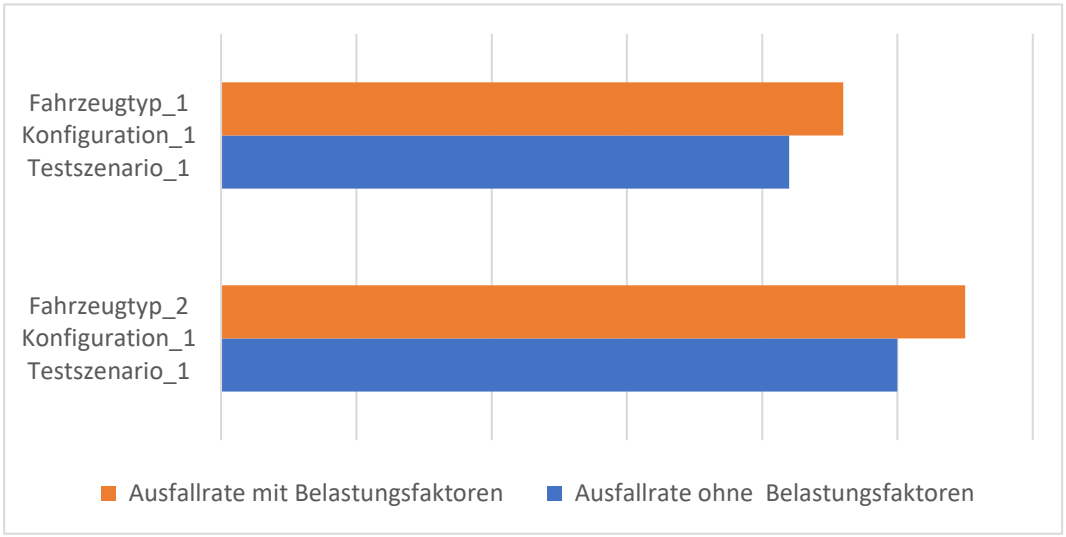


Abbildung 45 Vergleich der Bordnetze nach Fahrzeugtyp, Konfiguration und Testszenario

Die Anwendung ermöglicht die Konfiguration von Bordnetzen verschiedener Fahrzeugtypen. Zu den Fahrzeugtypen können verschiedene Konfigurationsvarianten zugeordnet werden. Die Konfigurationen können mit verschiedenen Testszenarien simuliert werden. Dadurch ist es möglich, verschiedene Bordnetzvarianten auszuwerten und zu vergleichen (Abb. 45). Dabei kann auch der Einfluss der Belastungsfaktoren beobachtet werden. Neben der Gesamtausfallrate können die ausgewertete Bordnetzkomponenten im Detail betrachtet werden. Die Belastungsfaktoren erhöhen die Ausfallrate und zugleich führen sie zu verschiedenen Fehlern, wie z. B. „hohe Temperatur führt zu Kurzschluss“. In der Detailansicht werden die Beiträge der verschiedenen Fehlerarten zur Gesamtausfallrate eines Komponententyps dargestellt. Die Fehlerverteilung erlaubt es, die Ergebnisse mit den Felddaten zu vergleichen und Schlussfolgerungen zu ziehen. Die Felddaten werden nach den gleichen Fehlerarten erfasst. Daher können die Daten auf dieser Basis abgeglichen werden. Die folgende Abbildung 46 zeigt eine Beispielauswertung von sieben Bordnetzkomponenten (Sicherung, Relais, Splice, Kabelschuh, Kabel, Kontakte und Stecker) mit Gesamtausfallraten verteilt nach fünf Fehlerarten (Erhöhung des Übergangswiderstandes, Trennung, Kurzschluss, Signalpfad, Aktiv/Inaktiv).

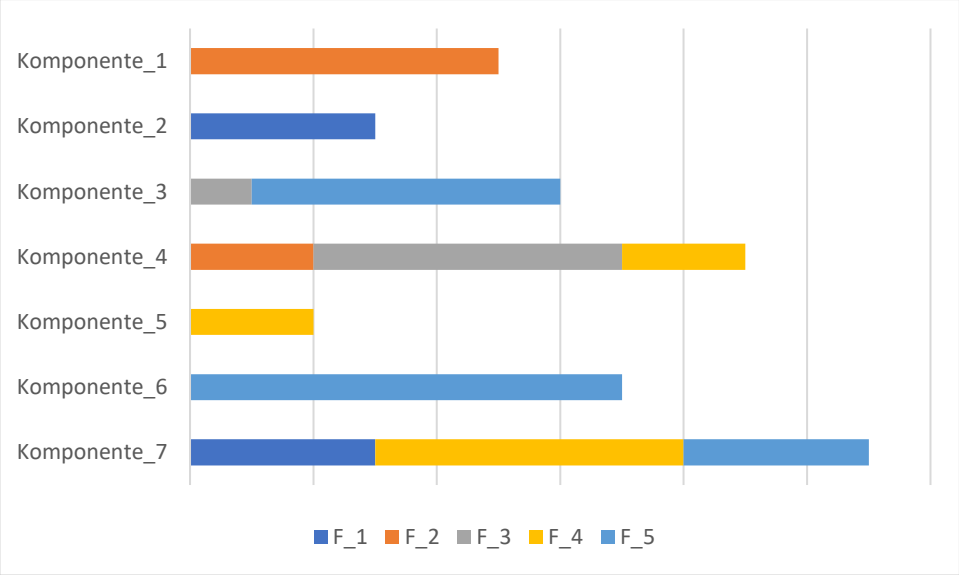


Abbildung 46 Beispielhafte Darstellung von ausgewerteten Komponenten mit Fehlerverteilung

6.2 Architektur und eingesetzte Technologien

WHAT ist mit der Open-Source-Entwicklungsumgebung Eclipse (Eclipse IDE, 2013) implementiert und ist eine Desktopanwendung, die auf der *Eclipse Rich Client Platform (RCP)* (Eclipse RCP, 2013) basiert. Die Interaktion zwischen Desktopanwendung und Matlab wird durch die Java API *matlabcontrol* (Matlabcontrol, 2013) realisiert. Die Architektur von WHAT folgt dem Architekturmuster *Model-View-Controller* (MVC, Modell-Präsentation-Steuerung). Das MVC-Muster besteht aus drei Komponenten und der Quelltext wird nach diesen Komponenten klassifiziert. Die Präsentationskomponente ist für die Präsentation der

Daten und den Aufruf der Aktionen zuständig. Die Benutzeroberfläche und die editierbare grafische Oberfläche sind mit *RCP* und dem *Eclispe Graphical Modelling Framework (GMF)* (Eclipse GEF, 2013) implementiert. Die Steuerungskomponente ist für die Interaktion zwischen Daten und Präsentation verantwortlich. Sie enthält daher die auszuführenden Aktionen und Algorithmen. Die Modellkomponente läuft im Backend und ist zuständig für die Bereitstellung und Haltung von Daten. Die Modell- und Steuerungskomponenten sind mit Knowledge-Graph-Technologien implementiert. Das Modell ist eine auf KBL basierende RDFS-Ontologie und die Daten werden im RDF-Format bereitgestellt. Zur Verarbeitung und Abfrage der Daten wird die *Apache Jena API* (Apache Jena, 2012) verwendet. Vorteile der Verwendung des MVC-Architekturmusters sind, dass die Erweiterungsmöglichkeiten der Software sehr flexibel und die einzelnen Komponenten wiederverwendbar sind. Während des Forschungsprojektes wurde die Software permanent weiterentwickelt. Daher war der Einsatz des MVC-Musters von großem Nutzen.

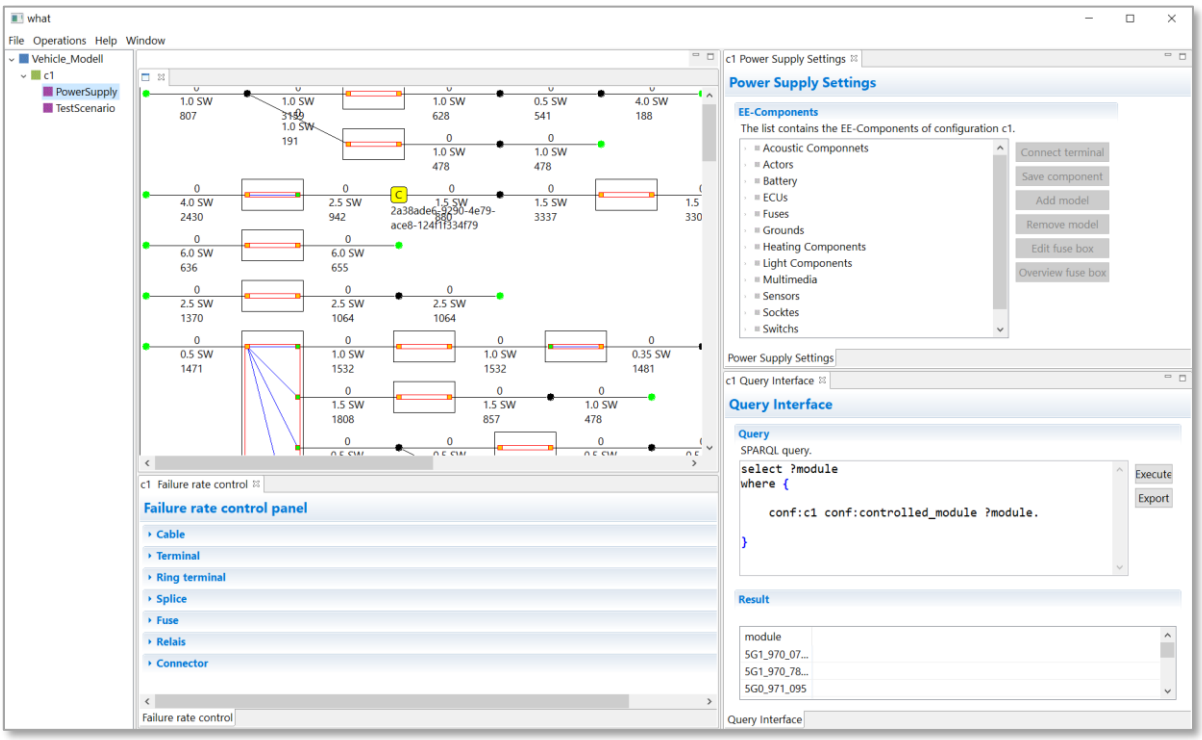


Abbildung 47 Benutzeroberfläche mit diversen Ansichten

Die Benutzeroberfläche von WHAT orientiert sich an der Entwicklungsoberfläche von Eclipse und ist einfach zu bedienen. Auf der linken Seite können Fahrzeugmodelle, Konfiguration, Testszenario, etc. ausgewählt werden. Über verschiedene Ansichten, die auf Steuerungskomponenten zugreifen, können verschiedene Operationen an den Bordnetzdaten durchgeführt werden.

6.3 Implementierung

In diesem Abschnitt wird erläutert, wie der Knowledge-Graph-basierte Lösungsansatz zur Datenverarbeitung im Prototyp umgesetzt wurde. Zunächst wird der Aufbau des Knowledge-

Graphs erläutert, anschließend wird der Anwendungsfall „Quantitative Zuverlässigkeits-analyse“ gezeigt und zum Schluss wird dargestellt, wie die Komplexität der Anwendung durch den Einsatz des Knowledge Graph reduziert wurde.

6.3.1 Aufbau des Knowledge Graph

Im Projekt wurde eine Ontologie auf Basis der heterogenen Bordnetzdaten entwickelt. Hauptbestandteil dieser Ontologie ist die Kabelbaumliste (KBL). Die KBL-Dateien allein reichen nicht aus, um ein vollständiges simulierbares Bordnetz zu generieren, daher wurden weitere Daten aus unterschiedlichen Quellen und Formaten in die Anwendung integriert. Nicht alle Daten lagen in einer strukturierten Form vor, in der sie automatisch verknüpft werden könnten. Für diese Fälle wurden Benutzerschnittstellen implementiert, um die fehlenden Informationen zu ergänzen.

Kabelbaumliste als Ontologie

Die KBL-Ontologie wird basierend auf dem KBL-Schema 2.3 entwickelt und ist eine RDFS-Ontologie (Balandi, 2018). Die Ontologie besteht aus 82 Klassen, 100 Object Properties und 85 Data Properties. Im Vergleich zur VEC-Ontologie ist die KBL-Ontologie klein, hat aber ebenfalls eine komplexe Referenzierungsstruktur. Die KBL-Ontologie hat zwei wichtige Grundklassen *„Part“* und *„Occurence“*. Die Part-Klasse hat 12 Unterklassen und dient für die Beschreibung der Bauteile. Die Bauteile sind Bordnetzkomponenten wie Stecker, Kontakte, Leitungen etc. Dagegen beschreiben die Occurence-Klassen die Bauteilverwendung. Die Bauteilverwendungen enthalten verwendungsbezogene Informationen, wie z. B. die Platzierung oder Zusammensetzung von Komponenten, die Länge oder den drei-dimensionalen Verlauf von Leitungen etc. Die Referenzierungsstruktur der KBL-Ontologie wird auf RDF-Graphen abgebildet, so dass die verlinkten Informationen flexibel mit der Abfragesprache SPARQL abgefragt und verarbeitet werden können.

Datenintegration

Das Ziel des Knowledge Graph ist es, die für den Zweck benötigten Daten zu identifizieren und verschiedene Datenquellen unter einer Ontologie zu vereinheitlichen. Die grundlegende KBL-Ontologie wurde daher entsprechend erweitert, um eine vollständige quantitative Auswertung eines Bordnetzes zu ermöglichen. Wie in der folgenden Abbildung dargestellt, sind verschiedene Datenquellen über XML-Datenschnittstellen mit der Anwendung verbunden.

Kabelbaumliste: Die KBL-Dateien enthalten die Leitungsinformationen. Dazu gehören neben den Kabeln auch Steckverbinder, Kontaktstellen, Sicherungen, Relais, Schutz- und Befestigungselemente mit dreidimensionalen Positionsinformationen.

Komponentendatenbank: Die Komponentendatenbank enthält die elektrischen und elektronischen Komponenten, in denen sich die Stromwerte und die Informationen zu den

Innenverschaltungen befinden. Bei der Datenintegration werden durch die Bezeichnungen der Komponenten die dazugehörigen Stecker bestimmt und zueinander zugeordnet. Auf diese Weise wird das Bordnetz, wie folgt dargestellt ist, zusammengestellt.

Andere Datenquellen: Weitere Datenquellen sind Fahrzeugkonfigurationen und Funktionen, die bestimmen, was ausgewertet wird. Darüber hinaus kommen Informationen wie Bauräume, Ausfalldaten, Simulationsmodelle für die quantitative Auswertung hinzu.

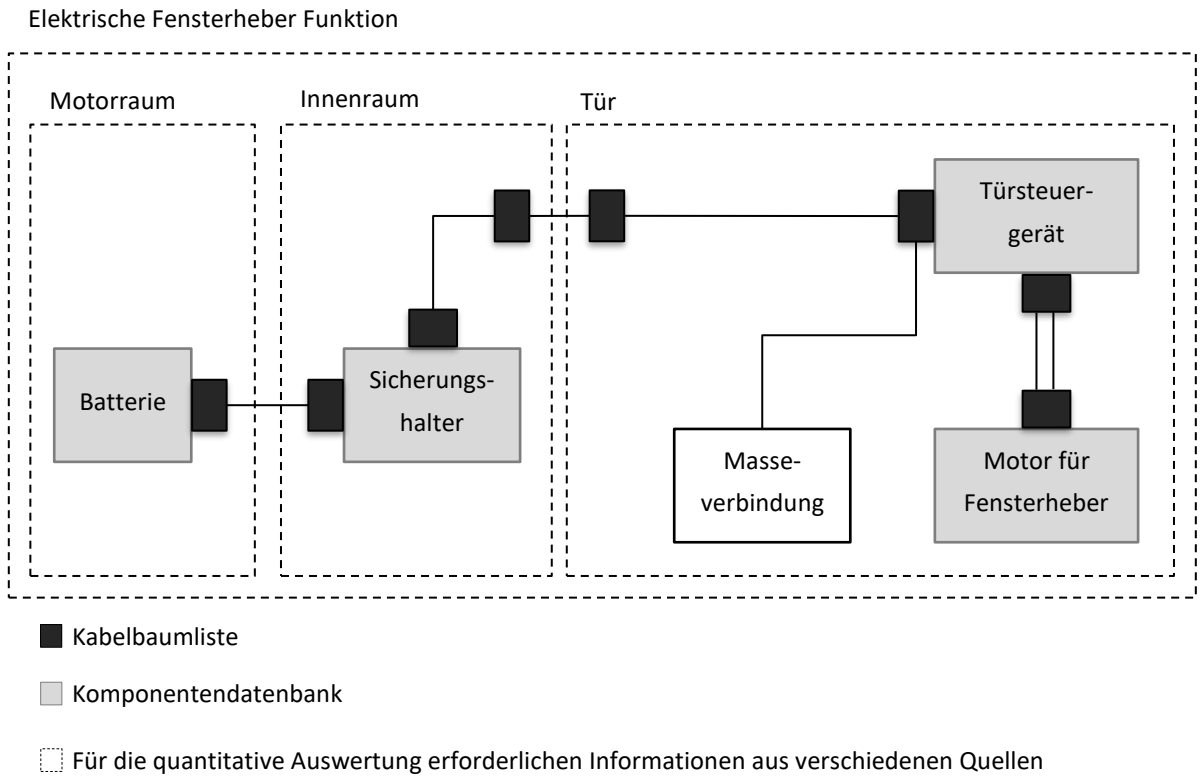


Abbildung 48 Zusammenstellung des Bordnetzes

Nachdem die Daten integriert sind, werden Konsistenzüberprüfungen durchgeführt und der Knowledge-Graph erweitert und verfeinert. Zuerst werden die Koppelstellen überprüft. Zwei Stecker werden an Koppelstellen miteinander verbunden. Koppelstellen liegen zwischen den Bauräumen z. B. zwischen Innenraum und Tür (Abb. 48). Die zusammengehörigen Stecker werden durch Eigenschaften wie Bezeichnung, 3D-Position und angeschlossene Leitungen bestimmt. Dabei kommen SPARQL-Abfragen und Programmierschnittstellen zum Einsatz. Wenn die notwendigen Informationen vorhanden sind, reichen meistens die SPARQL Abfragen aus, um den Knowledge-Graph zu verfeinern. In KBL-Dateien sind indirekt noch viel mehr Informationen enthalten. Diese impliziten Informationen können explizit dargestellt werden zwecks einer formalen Verarbeitung der Daten. Zum Beispiel sind die Koppelstellen Stecker (`Connector_Occurrence`), die mit Anfangsbuchstaben „T“ im Identifikator bezeichnet sind. Diese implizite Information kann durch eine SPARQL-Construct-Abfrage explizit dargestellt werden, indem anhand der Bezeichnung die `Connector_occurence` typisiert wird. Der folgende Quelltext zeigt, wie eine solche Abfrage implementiert wird. Die

`Connector_occurence` enthält zusätzlich den Typ `base:CouplingPoint`. Durch die neue Klasse wird auch die Ontologie erweitert.

```
construct{
    ?connectorOcc rdf:type base:CouplingPoint.
}

where{

    ?connectorOcc rdf:type kbl:ConnectorOccurrence.
    ?connectorOcc kbl:id ?id.

    FILTER(strStarts(?id,"T")) .
}
```

Quelltext 11 Typisierung von Steckern

Im zweiten Schritt können die Koppelstellen zueinander gekoppelt werden. Der folgende Quelltext zeigt, wie zwei Stecker anhand ihres räumlichen Abstands mit der Property `base:connected_connector` zueinander referenziert werden können. Zwischen zwei Steckern wird die räumliche Distanz $d(X,Y) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$ berechnet und falls sie unter 20 mm liegt, werden sie gekoppelt.

```
construct {
    ?couplingPoint_source base:connected_connector ?couplingPoint_target.
}
where {

    #source
    ?couplingPoint_source rdf:type base:CouplingPoint.
    ?couplingPoint_source base:coordinate_x ?x_1.
    ?couplingPoint_source base:coordinate_y ?y_1.
    ?couplingPoint_source base:coordinate_z ?z_1.

    #target
    ?couplingPoint_target rdf:type base:CouplingPoint.
    ?couplingPoint_target base:coordinate_x ?x_2.
    ?couplingPoint_target base:coordinate_y ?y_2.
    ?couplingPoint_target base:coordinate_z ?z_2.

    FILTER NOT EXISTS {
        ?connectorOcc_source base:connected_connector ?any_1.
        ?connectorOcc_target base:connected_connector ?any_2.
    }

    FILTER(?connectorOcc_source != ?connectorOcc_target)
    FILTER(abs(
            sqrt(
             pow((?x_1-?x_2),2) + pow((?y_1-?y_2),2) + pow((?z_1-?z_2),2))
              )< 20)
}
```

Quelltext 12 Kopplung von zwei Steckern basierend auf der räumlichen Distanz

Anschließend wird das Bordnetz auf Vollständigkeit geprüft. Von den Sicherungen aus werden die Leitungen im Netz verfolgt und es wird geprüft, ob die Verbraucher und Masseverbindungen erreicht werden können. Es müssen unterschiedliche Fälle berücksichtigt werden. Auf diese Weise wird eine einheitliche und qualitätsgesicherte Datenumgebung durch den Knowledge-Graph erreicht. Am Ende werden die Daten zur Simulation und Auswertung freigegeben.

Ausgewertete Daten und Komponenten

In dem Prototyp wurden drei Fahrzeuge aus den Kompakt-, Klein- und Kleinstwagen-Klassen ausgewertet. Die Leitungsinformationen von den Fahrzeugen befinden sich in den KBL-Dateien und pro Fahrzeug umfassen die Daten 5 bis 30 KBL-Dateien je nach Export. Zusätzlich für jedes Bauteil werden Datensätze aus der Komponentendatenbank importiert und für die quantitative Analyse zusätzliche Daten integriert. Nach der Integration steigt die Anzahl der Datensätze enorm an. Bei der Kompaktklasse liegt die Anzahl der RDF-Datensätze bereits bei mehr als einer halben Million. Bei höheren Fahrzeugklassen geht die Zahl exponentiell in die Millionen. Die Datensätze werden für VEC sogar noch höher sein, da VEC einen höheren Grad an Detailbeschreibung bietet.

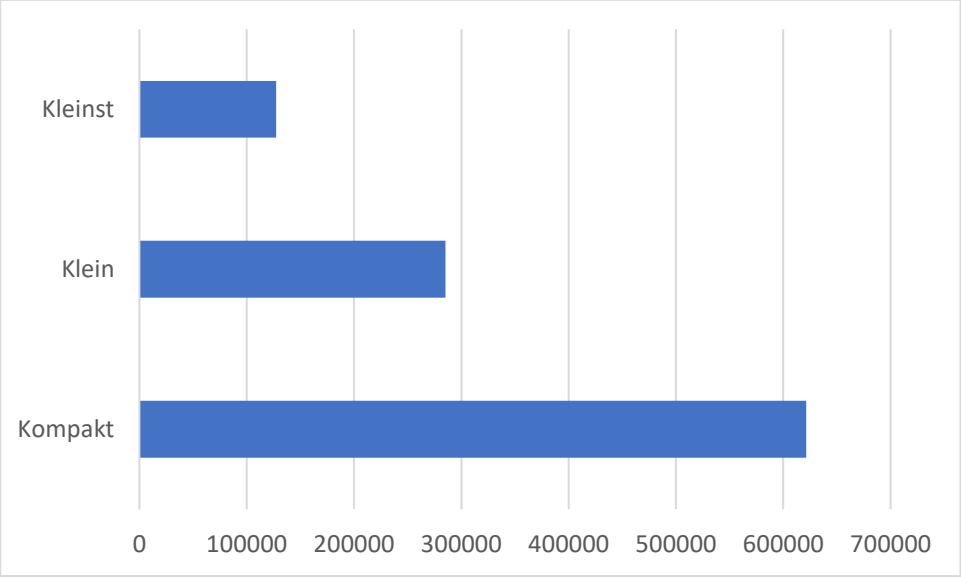


Abbildung 49 Fahrzeugdaten mit Varianten sortiert nach Klassen und Anzahl der RDF-Triples

Die Auswertung der Bordnetzkomponenten erfordert eine große Menge von Daten. In dem Prototyp wurden sieben Bordnetzkomponenten quantitativ ausgewertet. Die ausgewerteten Komponenten besitzen auch Untertypen. Die Komponenten werden auf Basis ihrer Eigenschaften, Umgebungsinformationen und Simulationsergebnisse ausgewertet. Um auf ein aussagekräftiges Ergebnis zu kommen, sollen alle notwendigen Informationen vorhanden sein. Aufgrund der vielen Datenquellen nahm die Vervollständigung der Daten bei der Entwicklung des Prototyps viel Zeit in Anspruch. Die folgende Abbildung zeigt die Anzahl der Komponenten von vollausgestatteten Fahrzeugen, die im Prototyp ausgewertet wurden.

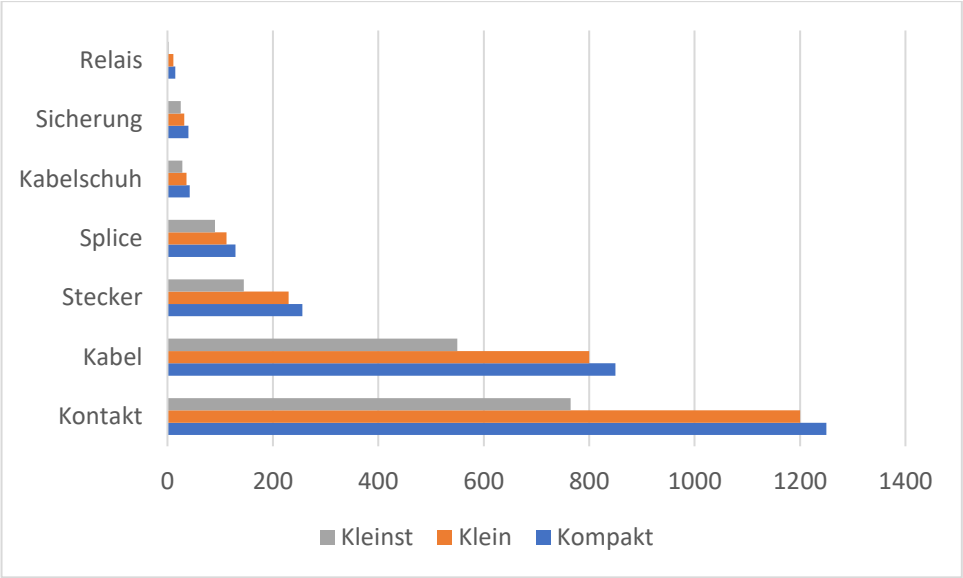


Abbildung 50 Komponentenanzahl der ausgewerteten Fahrzeugklassen

6.3.2 Anwendungsfall Quantitative Zuverlässigkeitsanalyse

Die quantitative Zuverlässigkeitsanalyse im Prototyp wurde anhand von Ausfalldatenbüchern realisiert. Die Bauteile haben Standardausfallraten und je nach Verwendung werden sie durch Belastungsfaktoren multipliziert (siehe Abs. 4.1.1.). Die Belastungsfaktoren werden durch Fehlermuster aus Ausfalldatenbüchern bestimmt. Die Abfragesprache SPARQL ermöglicht es, solche Muster nachvollziehbar und semantisch zu definieren und auszuführen. Im Prototyp wurden 43 Fehlermuster auf Basis von Umgebungs- und Simulationsinformationen sowie Komponenteneigenschaften implementiert. Die Bauteilverwendungen (Occurrence-Elemente) werden nach diesen Mustern ausgewertet. Im folgenden Quelltext wird gezeigt, wie der Temperaturlastfaktor an Kontakten hinzugefügt wird. Der Temperaturlastfaktor wird auf Basis der Umgebungstemperatur und der Eigenerwärmung (ermittelt durch Simulation) generiert, wenn deren Summe 120 Grad überschreitet.

```
construct{
    ?terminalOcc base:pi_temperature "1.5"^^xsd:float .
}

where{

    ?terminalOcc rdf:type kbl:TerminalOccurrence.
    ?terminalOcc base:simulation_temperature ?simulation_temperature.
    ?terminalOcc base:installation_space  ?installation_space.
    ?installation_space base:operating_temperature ?operating_temperature

    FILTER((?simulation_temperature + ?operating_temperature) > 120)
}
```

Quelltext 13 Zuordnung der Belastungsfaktoren

Im nachfolgenden Schritt kann die Standardausfallrate der Bauteile mit den Belastungsfaktoren multipliziert werden, um die belastungsgewichtete Ausfallrate zu bestimmen, falls dem Bauteil ein Belastungsfaktor zugeordnet ist.

```
select ?terminalOcc ((?lambda_basis * IF(bound(?pi_temperature),
                                        ?pi_temperature, 1)) as ?failureRate)

where{

    ?terminalOcc rdf:type kbl:ContactPointOccurrence.
    optional{?terminalOcc base:pi_temperature ?pi_temperatur.}
    ?terminalOcc kbl:part ?terminal.
    ?terminal base:lambda_basis ?lambda_basis.
}
```

Quelltext 14 Berechnung der Ausfallrate

Die quantitative Auswertung ist ein dynamischerer Prozess. Die SPARQL-Abfragesprache und die weiteren Knowledge-Graph-Technologien ermöglichen diese Dynamik umzusetzen, indem die Informationen flexibel verändert, erweitert und abgefragt werden können. Die Analysen können unter verschiedenen Perspektiven, wie bauteil-, konfiguration-, modul-, funktionsbezogen, durchgeführt werden. Zusätzlich sind spezielle Abfragen, wie die Bestimmung des Leitungsstrangs mit der größten Ausfallrate etc., ebenfalls möglich. Durch solche Abfragen wird das Bordnetz umfangreich und detailliert analysiert.

6.3.3 Reduktion der Komplexität

Der Einsatz von Knowledge-Graph-Technologien ermöglichte eine enorme Reduktion der Komplexität im Prototyp. Gleichzeitig wurde eine Menge Code eingespart. Sowohl die im vorherigen Abschnitt vorgestellten Analysen als auch die Kommunikation zwischen der Benutzeroberfläche und dem Datenrepository laufen weitgehend über die SPARQL-Schnittstelle. Die komplexe Referenzierungsstruktur von KBL und VEC ist ein großes Hindernis beim Abfragen von zusammenhängenden Informationen. Werden die KBL/XML-Dateien in eine objektorientierte Umgebung transformiert und von dort aus verarbeitet, müssen viele Iterationen im Code implementiert werden, um die zugehörigen Informationen zu ermitteln und zu filtern. Der Quelltext 15 zeigt einen solchen Fall in der objektorientierten Programmiersprache Java. In diesem Bespiel wird die Summe der Querschnitte aller angeschlossen Leitungen einer Komponente „Component_X“ berechnet. Es ist zu beachten, dass solche Abfragen nur modulspezifisch abgefragt werden können, da die gleiche Komponente in verschiedenen Modulvarianten unterschiedliche Leitungen haben kann. Deshalb werden nur die *Connections,* die vom gleichen Modul stammen ausgewertet. Wie im Quelltext zu sehen ist, wird der Java-Code durch mehrfache und verschachtelte Iterationen schnell unübersichtlich und komplex. Wenn für jede dieser Anforderungen eine Funktion implementiert werden sollte, dann müsste eine Programmierbibliothek implementiert werden.

```
// Traversierung von Klassen:
// ComponentOccurence → connectorOccurence →
// Connection → [ WireOccurence → GeneralWire/outsideDiameter |
// SpecialWireOccurence → CoreOccurence → GeneralWire/outsideDiameter]

double sumOfOutgoingWires = 0;

Module module = harness.getModule("Module_X");

Component component = module.getComponentOccurence("Component_X");

List<Connector> connectorOccurences = component.getConnectorOccurence();

for(ConnectorOccurence connectorOccurence : connectorOccurences){

   List<Contactpoint> contactpoints = connectorOccurence.getContactPoints();

}

List<Connection> connections = module.getAllConnection();

for(Connection connection : connections){

   for(contactpoint contactpoint: contactpoints){

      if(connection.getExtemities.getContactPoints().contains(contactpoint)){

         if(harness.getWireOccurrence(connection.getWireOccurence()) != null){

            WireOccurence wireOcc =
               harness.getWireOccurrence(connection.getWireOccurence());

            GeneralWire generalWire =
               harness.getGeneralWire(wireOccurence.getPart());

            sumOfOutgoingWires += generalWire.getOutsideDiameter();

         } else {

            SpecialWireOccurence specialWireOccurence =
               harness.SpecialWireOccurence(connection.getWireOccurence());

            for(CoreOccurence coreOccurence:
               specialWireOccurence.getAllCoreOccurence() ){

               GeneralWire generalWire =
                  harness.getGeneralWire(coreOccurence.getPart());

               sumOfOutgoingWires += generalWire.getOutsideDiameter();
            }
         }
      }
   }
}
```

Quelltext 15 Berechnung der Summe der Leitungsquerschnitte einer Komponente in Java

Im Gegensatz dazu ist die gleiche Berechnung in SPARQL aufgrund der Musterbeschreibung sehr einfach, übersichtlich und nachvollziehbar. In SPARQL können solche komplexen Abfragen wesentlich flexibler gestaltet werden. Die Informationen können sowohl in Richtung der referenzierenden Struktur als auch in umgekehrter Richtung, d. h. vorwärts und rückwärts, gesucht werden. Durch das Einfügen von Shortcuts in den Knowledge-Graphs kann sogar die Abfrage verkürzt werden. Im anderen Fall muss im Code jede Traversierung einzeln implementiert werden.

```
select (sum(?outsideDiameter) as ?sumOfOutgoingWires)
where {
        ?module kbl:id "Module_X".
        ?module kbl:controlled_component ?component.
        ?module kbl:controlled_component ?connection.
        ?component kbl:id "Component_X".
        ?component kbl:connector_Occurrence ?ConnectorOccurence.
        ?ConnectorOccurence kbl:contactpoint ?Contactpoint.
        ?connection kbl:extremity ?extremity.
        ?extremity kbl:contactpoint ?Contactpoint.
        ?Connection kbl:wire ?wireOccurence.
        ?wireOccurence kbl:part ?part.
        ?part kbl:ousideDiameter ?outsideDiameter.

        optional{
            ?wireOccurence rdf:type kbl:SpecialWireOccurence.
            ?wireOccurence kbl:part ?part.
            ?part kbl:core ?core.
            ?part kbl:ousideDiameter ?outsideDiameter
          }
       }
```

Quelltext 16 Berechnung der Summe der Leitungsquerschnitte einer Komponente in SPARQL

6.4 Gewonnene Erkenntnisse

Dieser Abschnitt beschreibt die durch das Forschungsprojekt und die Entwicklung des WHAT-Prototyps gewonnenen Erkenntnisse. Insbesondere werden die Vorteile des Einsatzes der Knowledge-Graph-Technologien behandelt.

Daten in Graphenstruktur

Die Bordnetzdaten sind stark zueinander verknüpft, und es gibt eine Menge indirekt voneinander abhängiger Informationen. Daher ist die Graphenstruktur sehr gut geeignet und sogar "must to have" für die Realisierung von Anforderungen wie Versorgungsbaum-erstellung, Abfrage komplexer Zusammenhänge, Datenqualitätssicherung etc. Bei der Entwicklung kamen die Vorteile der RDF-Graphenstruktur stark zum Tragen, da Versorgungs- und Massebaum durch Traversieren auf dem RDF-Graph erstellt und die Abfragesprache SPARQL zur Abfrage der Datenqualitätsprüfungen und komplexer Beziehungen verwendet werden konnte.

Erweiterbares Datenmodell

Im Laufe des Projektes wurden alle notwendigen Informationen, die für die Bewertung des Bordnetzes notwendig sind, wie KBL-Daten, Belastungszonen, Ausfallraten, Simulation, etc., als Erweiterungen basierend auf dem existierenden RDF-Graphen im Repository sukzessive gesammelt. Die Graphenstruktur ermöglichte einfache Datenintegration und die RDFS-Ontologie, die RDF-Daten beschreibt, konnte entsprechend erweitert werden.

Datenorientiert anstatt Software-orientiert

In WHAT konnte durch Einsatz der Knowledge-Graph-Technologien eine datenorientierte Lösung realisiert werden. Durch die SPARQL-Schnittstelle (Abb. 47) konnten die Domänenexperten mitverfolgen, wie Daten im Backend gehalten und durch Algorithmen verarbeitet werden. Dies ermöglicht zum einen ein transparentes und nachvollziehbares Vorgehen, zum anderen können die Algorithmen und Daten von der Software entkoppelt und wiederverwendet werden, da die Geschäftslogik nicht mit einem nativen Datenmodell in der Software fest verbunden und einprogrammiert ist.

Dynamischer Prozess

Sowohl die Bordnetzentwicklung als auch die Zuverlässigkeitsanalyse sind hochdynamisch, angetrieben von ständigen Veränderungen, Anpassungen und einem bereichsübergreifenden Informationsfluss. Die starren Legacy-Systeme reichen nicht aus, um diese Dynamik zu bewältigen und daher geraten die Daten schnell in inkonsistente Zustände. Die Knowledge-Graph-Technologien bieten die Möglichkeit, diese Dynamik durch Werkzeuge besser zu unterstützen, abzubilden und zu verarbeiten, indem die Daten noch spezifischer beschrieben und strenger überprüft werden und damit gleichzeitig komplexe Fälle besser abgedeckt werden.

Basierend auf der gründlichen Analyse der Bordnetzdomäne und den Erkenntnissen, die mit der Prototypensoftware WHAT gewonnen wurden, wird im nächsten Kapitel ein technisches Einführungskonzept des Lösungsansatzes für die Bordnetzentwicklung vorgestellt.

7 Technisches Einführungskonzept

Dieser Abschnitt beschäftigt sich mit dem technischen Einführungskonzept der Knowledge-Graph-basierten Bordnetzentwicklung in Unternehmen. Es wird beschrieben, wie der Knowledge-Graph-basierte Ansatz in bestehende Systeme integriert werden kann. Darüber hinaus wird auf die Implementierung und Verwendbarkeit des Ansatzes eingegangen. In (Kuhn, et al., 2019) wird beschrieben, dass bei der Entwicklung von Bordnetzen die Legacy-Systeme teilweise bis zu 20 Jahre alt sind und die stark manuell gesteuerten Prozesse an ihre Grenzen stoßen, so dass neue Lösungen notwendig sind. Nicht nur Systeme in der Bordnetzentwicklung, sondern auch PLM-Systeme (Product Lifecycle Management Systems) sind an ihre Grenze gestoßen. „*Die PLM-Lösungen bilden die funktionale und administrative Basis (Backbone) in Unternehmen. Sie sind Ende der 90er-Jahre aus einer Erweiterung von Produktdatenmanagement-Systemen entstanden.“* (Eigner, et al., 2009). In dem vom Verein *prostep ivip* erstellten Thesenpapier (Rosenplänter , et al., 2016) wurden unter *"Future PLM"* eine Reihe von Thesen von Domänenexperten zur Zukunft von PLM-Systemen aufgestellt, um eine Diskussion über Anforderungen und Lösungsansätze zu initiieren. Ein besonders wichtiger in dem Thesenpapier diskutierter Punkt, der auch diese Arbeit abstützt, ist, dass ein Systemwechsel kein Ausweg ist, sondern die existierenden Systeme an die neuen Prozesse angepasst werden sollen. Das Thesenpapier betont auch, dass die semantischen Netzwerke eine Voraussetzung für die Integration der Systeme und MBSE sind. Eine weitere innovative Entwicklung in diesem Bereich ist das Projekt *Code of PLM Openness*, das darauf abzielt, Offenheit zu schaffen und damit die einfache Integration der IT in vernetzte IT-Umgebungen zu realisieren sowie die PLM-Systeme voranzutreiben (prostep ivip, 2019).

7.1 Integration des Knowledge Graph in die Bordnetz-Toolkette

Die Bordnetzentwicklung ist ein komplexer und kontinuierlicher Prozess, der über viele Jahre hinweg gepflegt wird. Ein schneller Technologiewechsel ist daher kein geeigneter Ansatz. Infolgedessen kann bei der Umsetzung des Verfahrens zunächst eine Knowledge-Graph-Plattform (KGP) als Analyseplattform in die Toolkette des Bordnetzes integriert werden. Ein großer Vorteil von RDF-Graphen an dieser Stelle ist, dass sie auch dokumentenbasiert ausgetauscht werden können. Die RDF-Graphen können als XML serialisiert werden (genannt RDF/XML), und die RDF/XML-Dokumente können wiederum in eine KGP importiert werden. Derzeit wird die dokumentenbasierte Bordnetzentwicklung von KBL auf VEC umgesetzt. Daher können die VEC/XML-Dokumente auf Basis von VEO in RDF/XML umgewandelt und in eine KGP importiert werden. Zum Beispiel werden in der VOBES^{PLUS}-Toolkette von Volkswagen die Elektrologik und Mechanik-Daten in der ELENA-Plattform zusammengeführt und KBL-Dokumente generiert. Damit ist es möglich, die KGP mit der ELENA-Plattform zu verknüpfen und die Dokumente ohne zusätzlichen Aufwand zu

synchronisieren. Ein VEC2VEO-Konverter ist auf der VEO-Projektseite (Balandi, 2018) verfügbar und kann entsprechend angepasst werden.

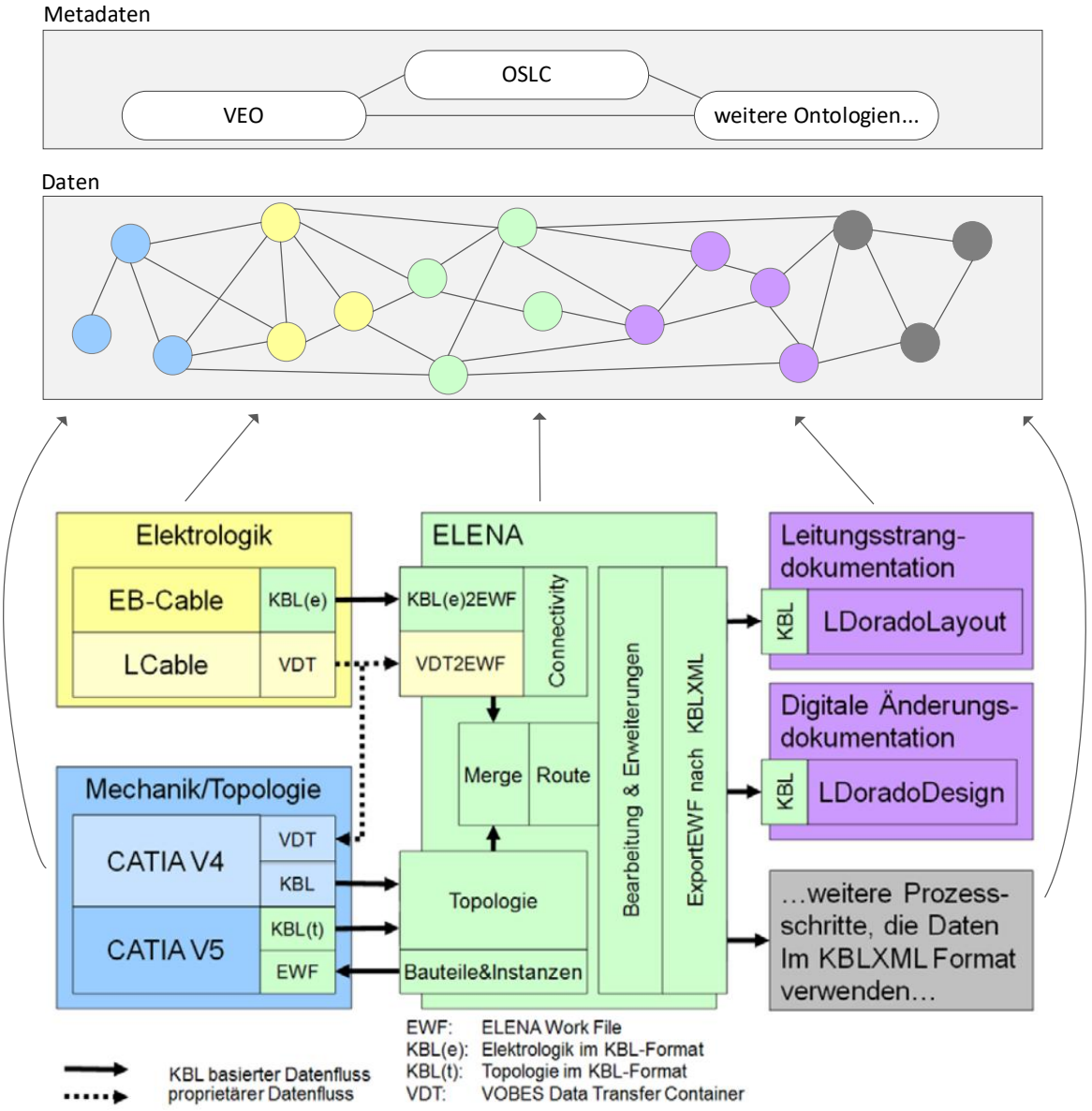


Abbildung 51 VOBES^{PLUS} Toolkette (Kyriazis, 2019) und Knowledge Graph

In der ersten Phase können auf KGP semantische Datenqualitätsanalysen, Optimierungs-algorithmen etc. durchgeführt werden und zusätzlich als Linked Data Layer verwendet werden, um systemübergreifend die Daten zu vernetzten. Darauf aufbauend können dann in einer weiteren Phase sowohl die in dieser Arbeit vorgestellte Architektur Schritt für Schritt auf der KGP realisiert als auch die verwendeten Tools in die Bordnetzentwicklung eingebunden werden. An dieser Stelle müssen die Bordnetz-Tools mit Schnittstellen erweitert werden, damit sie auf RDF-Graphen zugreifen und diese bearbeiten können.

7.2 Implementierung und Technologien

Die Implementierung fordert in der ersten Phase den Einsatz einer KGP und in späteren Phasen die Modifizierung der Bordnetzentwicklungstools. In den letzten Jahren sind viele Softwarehäuser in den Bereich der Knowledge-Graph eingestiegen und haben effiziente KGPs auf den Markt gebracht. Für die Auswahl von KGPs stehen daher viele Optionen offen. In der Domäne bieten sowohl renommierte Softwarehäuser als auch andere, die sich in diesem Bereich spezialisiert und bekannt gemacht haben, Lösungen an. Die existierenden KGPs können sehr großen Datenmengen im Milliardenbereich verarbeiten (W3C, 2018) (W3C, 2020). In der Bordnetzentwicklung liegen die Daten im Millionen-Bereich, daher sind die Anwendungen völlig ausreichend. Die KGPs ermöglichen darüber hinaus, dass auf der Plattform mehrere Triplestores erstellt und zueinander verlinkt werden können, so dass mit Federated SPARQL-Abfragen (SPARQL Working Group, 2013) mehrere Triplestores gleichzeitig angesprochen werden können. Daher ist es nicht notwendig, alle Daten in einem Triplestore zu speichern, sondern sie können in verschiedenen Triplestores parallel gehalten werden. Bei der Auswahl ist eine hybride Plattform wichtig, da gleichzeitig die Vorteile von Knowledge Graphs und Labeled Property Graphs benutzt werden können, wobei viele Anbieter in ihrem Produktspektrum hybride Anwendungen im Portfolio haben. Die Anpassung der Bordnetzentwicklungstools auf KGPs erfolgt durch Modifizierung der Schnittstellen der Tools. In diesem Fall ist die Kommunikation nicht dokumentenorientiert, sondern die Werkzeuge können über die bereitgestellten RDF-APIs auf RDF-Graphen zugreifen, was zu einer semantischen Interoperabilität führt. Es existieren Programmier-schnittstellen für alle gängigen Programmiersprachen Java, Javascript, Python, etc. und auch für die Softwareplattform Microsoft .NET, so dass verschiedene Bordnetztools einfacher modernisiert werden können. Bei der Implementierung der Schnittstellen ist darauf zu achten, dass die Datenkonsistenz erhalten bleibt, wenn Änderungen durch die Bordnetztools vorgenommen werden. Ein weiterer wichtiger Punkt bei der Integration der Tools ist, dass die Ergebnisse der Tools in die KGP integriert werden, damit die Ergebnisse der einzelnen Tools nicht ohne Bezug nebeneinanderstehen.

7.3 Benutzerfreundlichkeit

Die Einführung neuer Technologien in Unternehmen ist mit hoher Einarbeitungszeit verbunden. Die Vorteile semantischer Technologien liegen darin, dass die vernetzten Daten sowie die Abfrage- und Regelsprachen a priori verständlich sind. Um darüber hinaus die Einarbeitungszeit zusätzlich zu verkürzen und die Produktivität zu steigern, können visuelle Tools für Datenvisualisierung und -abfrage integriert werden. In (Vargas, et al., 2019) wird ein solches System namens *RDF Explorer* vorgestellt, dass es Nicht-Experten ermöglicht, gleichzeitig auf einem Knowledge-Graphen zu navigieren und diesen abzufragen, indem sie den Datensatz schrittweise explorieren. Im *RDF Explorer* können Objekte und Klassen durchsucht werden, und die ausgewählten Daten können visuell miteinander verknüpft

werden, so dass eine SPARQL-Abfrage visuell erstellt werden kann. Das System unterstützt den Benutzer dabei, wie die Daten miteinander verknüpft werden sollen.

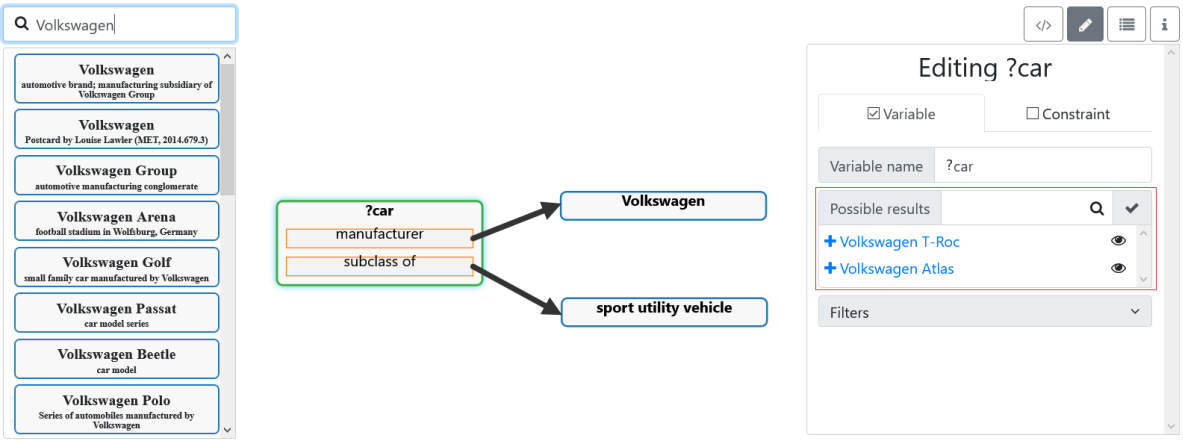


Abbildung 52 Beispiel für eine visuelle Abfrage im RDF-Explorer: SUV-Fahrzeuge des Herstellers VW

In einem weiteren Beitrag (Soylu, et al., 2018) wird ein visuelles Abfragesystem namens *OptiqueVQS* für die Industrie vorgestellt, mit dem basierend auf der Ontologie visuell Muster definiert, automatisch in Abfragen umgewandelt und ausgeführt werden können. Die Usability-Studien (Soylu, et al., 2016) weisen darauf hin, dass die Benutzer mit *OptiqueVQS* komplexe Abfragen formulieren konnten, ohne dass sie technische Vorkenntnisse benötigten.

8 Zusammenfassung und Ausblick

Dieses Kapitel fasst diese Arbeit zusammen und zeigt mit einem Ausblick, dass der vorgeschlagene Ansatz neue Türen öffnet und keine Insellösung darstellt.

8.1 Zusammenfassung

In dieser Arbeit wurde die Forschungsfrage ***„Wie können die informationstechnischen Defizite in der Bordnetzentwicklung behoben werden, um die Bordnetzentwicklung zu verbessern und gleichzeitig normgerechte Zuverlässigkeitsanalysen auf Basis der funktionalen Sicherheit effektiv zu realisieren?“*** ausführlich bearbeitet und ein konkreter Lösungsansatz entwickelt. Zuerst wurde die Bordnetzentwicklung, aufbauend auf bestehende wissenschaftliche Arbeiten, analysiert und festgestellt, dass die Diskontinuität der Prozesse und Daten das Hauptproblem darstellen. Daraufhin wurde die Analyse auf der Datenebene fortgesetzt und zu der Erkenntnis gelangt, dass ohne eine semantische Integration der Daten die Integration der Prozesse nicht realisiert werden kann. Für die semantische Integration der Daten wurden Knowledge-Graph-Technologien eingesetzt und eine Ontologie auf Basis des standardisierten Datenmodells der Bordnetzdaten entwickelt. Der Ansatz wurde mit den existierenden dokumentenbasierten Systemen verglichen und die Vorteile wurden aufgezeigt. Darauf aufbauend wurde dargestellt, wie die Defizite der Bordnetzentwicklung, die Diskontinuität von Prozessen, das Variantenmanagement und das Änderungsmanagement, auf der Basis von Knowledge-Graph-Technologien und mit den Methoden des Software Engineerings behoben werden können. Dabei wurde der Lösungsansatz mit den Legacy-Systemen verglichen und dargelegt, dass der Lösungsansatz keine Insellösung darstellt, sondern mit der modellbasierten und datenorientierten Architektur zum modellbasierten Systems Engineering beiträgt und die Realisierung der neuen Anforderungen wie Digital Twin und Digital Thread ermöglicht. Anschließend zeigte die Arbeit auf, wie die normgerechten Zuverlässigkeitsanalysen auf Basis der funktionalen Sicherheit effektiv realisiert werden können, basierend auf den Erfahrungen aus dem Projekt „Bordnetz Zuverlässigkeit“, das am Fachgebiet für Fahrzeugsysteme und Grundlagen der Elektrotechnik der Universität Kassel durchgeführt wurde. Anhand einer prototypischen Implementierung wurde gezeigt, wie der Lösungsansatz umgesetzt wird und welche Erfahrungen gesammelt werden konnten. Der Ansatz ermöglicht es, einerseits die Sicherheitsanforderungen und deren Umsetzung semantisch zu verknüpfen, so dass die Rückverfolgbarkeit, d. h. die Hauptanforderung der funktionalen Sicherheit, realisiert wird und andererseits die Zuverlässigkeitsanalyse strukturiert und formal durchgeführt werden kann. Mit den bereitgestellten Werkzeugen, der flexiblen Datenstruktur und dem skalierbaren Modell der Knowledge Graphs kann der Ansatz um weitere Prozesse und Analysen erweitert werden. Am Ende der Arbeit wurde gezeigt, wie die Technologien schrittweise im Entwicklungsprozess eingesetzt werden können.

8.2 Ausblick

In dieser Arbeit wurde vorgestellt, wie die Knowledge-Graph-Technologien in der Bordnetzentwicklung und darauf aufbauend bei der Realisierung normgerechter Zuverlässigkeitsanalyse Nutzen stiften. Die Vorteile, wie realitätsnahe Modellierung, semantische Vernetzung, maschinelle Interpretierbarkeit, sind wichtige und nützliche Eigenschaften im Digitalisierungszeitalter. Die Technologien haben sich daher in den letzten Jahren sehr stark in vielen Bereichen verbreitet, so z. B. bei intelligenten Lösungen im Finanzsektor, komplexer Modellierung in den Biowissenschaften, der Digitalisierung des kulturellen Erbes und auch im Bereich des Engineerings. Im Engineering können die Technologien vielseitig eingesetzte werden. Die OSLC bietet die große Möglichkeit verschiedene Engineering-Daten semantisch zu vernetzen und trägt das Produkt Lifecycle Management auf eine neue Ebene. Auch für neue Herausforderungen in der Industrie 4.0 werden die Technologien zur Integration und Verwaltung von Engineering-Daten eingesetzt, vom Anforderungsmanagement über Simulation bis hin zum Projektmanagement (Sabou, et al., 2017). Darüber hinaus werden die Technologien weiterhin eine wichtige Rolle bei der semantischen Integration der Bereiche Mechanik, Elektrotechnik und Softwaretechnik spielen, um modellbasiertes Systems Engineering für die interdisziplinäre Zusammenarbeit und die rapide und qualitative Entwicklung von Fahrzeugen zu realisieren.

Die Knowledge-Graph-basierte Bordnetzentwicklung bildet eine gute Möglichkeit für die weitere Zusammenarbeit der Domänen Elektrotechnik und Informatik. Für die Domänenexperten der Informatik werden die Bordnetzdaten mit der neuen Vorgehensweise verständlicher, zugänglicher und direkt algorithmierbar. Anderseits sind die Daten und Algorithmen für die Experten der Domäne Bordnetz transparenter, so dass sich eine effektive Zusammenarbeit gestalten lässt. In der Domäne werden neue Optimierungsalgorithmen für das Bordnetz im Hinblick auf autonomes Fahren und die Gewährleistung der funktionalen Sicherheit erforderlich sein. Ein weiteres Ziel in der Domäne ist die automatische Generierung von Bordnetzen. Die semantischen Technologien bilden eine gute Basis, um intelligente Anwendungen zu realisieren. Nach der Umsetzung der Knowledge-Graph-basierten Bordnetzentwicklung werden einerseits die Bordnetzdaten semantisch angereichert sowie gleichzeitig durch die formalen Konsistenzprüfungen eine gute Datenqualität erreicht und andererseits das Wissen der Experten formalisiert. Auf dieser Basis können Algorithmen entworfen werden, die z. B. durch fallbasiertes Schließen das Wissen aus abgeschlossenen Projekten nutzen, um (teil-)automatisch ein neues Bordnetz zu generieren.

Abkürzungsverzeichnis

API	Application Programming Interface
ASIL	Automotive Safety Integrity Level
CAD	Computer Aided Design
CPM	Core Product Model
EMV	Elektromagnetischen Verträglichkeit
FIT	Failures In Time
FMEA	Failure Mode and Effects Analysis
FTA	Fault Tree Analysis
HTTP	Hypertext Transfer Protocol
JAXB	Java Architecture for XML Binding
KBL	Kabelbaumliste
KG	Knowledge Graph
LPD	Labeled Property Graph
OASIS	Organization for the Advancement of Structured Information Standards
OSLC	Open Services for Lifecycle Collaboration
PLM	Product Lifecycle Management
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
REST-API	Representational State Transfer - Application Programming Interface
SPARQL	Protocol And RDF Query Language
STEP	Standard for The Exchange of Product model data
SWRL	Semantic Web Rule Language
VDA	Verband der Automobileindustrie
VES-WF	Vehicle Electric Systems Workflow Forum
W3C	World Wide Web Consortium
WHAT	Wiring Harness Analysis Tool

XML	Extensible Markup Language

Abbildungsverzeichnis

Tabellenverzeichnis

Quelltextverzeichnis

Literaturverzeichnis

Angele, Jürgen, Erdmann, Michael und Wenke, Dirk. 2008. Ontology-Based Knowledge Management in Automotive Engineering Scenarios. [Buchverf.] Martin Hepp, et al. *Ontology Management: Semantic Web, Semantic Web Services, and Business Applications.* Boston, MA : Springer US, 2008, S. 245-264.

Apache Jena. 2012. A free and open source Java framework for building Semantic Web and Linked Data applications. [Online] The Apache Software Foundation, 2012. [Zitat vom: 23. 04 2018.] https://jena.apache.org/.

Apache TinkerPop. 2018. The Gremlin Graph Traversal Machine and Language. [Online] 2018. [Zitat vom: 23. 04 2018.] http://tinkerpop.apache.org/gremlin.html.

Arndt, Natanael, et al. 2019. Decentralized Collaborative Knowledge Management Using Git. *Journal of Web Semantics.* 2019, Bd. Volume 54.

Balandi, Oguzhan. 2018. VEO Project. [Online] 2018. https://github.com/obalandi/veo.

Barbau, Raphael, et al. 2012. OntoSTEP: Enriching product model data using ontologies. *Computer-Aided Design.* 2012, Bd. Volume 44 , Issue 6.

Becker, Johannes. 2018. Efficient Navigation on large XML Structures. [Online] 2018. [Zitat vom: 23. 05 2021.] https://www.4soft.de/blog/2018/navigation-on-large-xml-structures/.

Becker, Johannes und Pöschl, Martin. 2013. *The VEC - the past, the present and the future.* München : 4Soft GmbH, 2013.

Benavides, David, Segura, Sergio und Ruiz-Cortés, Antonio. 2010. Automated analysis of feature models 20 years later: A literature review. 2010, Bd. Volume 35 Issue 6.

Berners-Lee, Tim, et al. 1998. Uniform Resource Identifiers (URI): Generic Syntax. [Online] Internet Engineering Task Force, 1998. [Zitat vom: 05. 04 2021.] https://www.ietf.org/rfc/rfc2396.txt.

Berners-Lee, Timothy John und Cailliau, Robert. 1990. WorldWideWeb: Proposal for a HyperText Project. [Online] World Wide Web Consortium, 1990. [Zitat vom: 05. 04 2021.] https://www.w3.org/Proposal.html.

Betsche, Bernd und Lechner, Gisbert. 2004. *Zuverlässigkeit im Fahrzeug- und Maschinenbau.* Berlin Heidelberg : Springer-Verlag, 2004.

Birch, John, et al. 2013. Safety Cases and Their Role in ISO 26262 Functional Safety Assessment. [Buchverf.] Friedemann Bitsch, Jérémie Guiochet und Mohamed Kaâniche.

Computer Safety, Reliability, and Security. 32nd International Conference. Toulouse, France : Springer Berlin Heidelberg, 2013.

BMIR. 2019. Protégé Editor. [Online] Stanford Center for Biomedical Informatics Research (BMIR), 2019. [Zitat vom: 14. 11 2019.] https://protege.stanford.edu.

Bräuchle, Christoph, Leitner, Andrea und Wallner, Alfred. 2015. Tool Interoperabilität für durchgängiges modellbasiertes Systems Engineering. [Buchverf.] Sven-Olaf Schulze und Christian Tschirmer. *Tag des Systems Engineering. Verteiltes Arbeiten mit ganzheitlicher Kontrolle.* Ulm : Carl Hanser Verlag GmbH & Co. KG, 2015.

Chacon, Scott und Straub, Ben. 2014. *Pro Git.* Berkeley, CA : Apress, 2014.

COMSA. 2018. LDorado Vestigo. Visualisieren und Aufbereiten von KBL- und LDorado-Dokumenten. [Online] 2018. [Zitat vom: 24. Oktober 2019.] https://www.comsa.de/de/product/ldorado-vestigo/1057.

Conway, Melvin E. 1968. How Do Committees Invent? [Online] F. D. Thompson Publications, Inc., 1968. [Zitat vom: 06. 04 2021.] http://www.melconway.com/research/committees.html.

CRYSTAL-Projekt. 2016. CRYSTAL - CRITICAL SYSTEM ENGINEERING ACCELERATION. [Online] 2016. [Zitat vom: 17. 04 2021.] https://www.crystal-artemis.eu/.

DBpedia Association. 2014. DBpedia. [Online] 2014. [Zitat vom: 05. 04 2021.] https://www.dbpedia.org/.

Dimou , Anastasia und Sande, Miel Vander. 2020. RDF Mapping Language (RML). [Online] 2020. [Zitat vom: 05. 04 2021.] https://rml.io/specs/rml/.

DIN 40041. 1990. DIN 40041:1990-12 Zuverlässigkeit; Begriffe. [Online] 1990. [Zitat vom: 08. 04 2021.] https://www.beuth.de/de/norm/din-40041/1639558.

Eberlin, Stefan und Barbara, Hock. 2014. *Zuverlässigkeit und Verfügbarkeit technischer Systeme. Eine Einführung in die Praxis.* Wiesbaden : Springer Vieweg, 2014.

Eclipse GEF. 2013. Graphical Editing Framework. [Online] Eclipse Foundation, 2013. [Zitat vom: 18. 04 2021.] https://www.eclipse.org/gef/.

Eclipse IDE. 2013. Eclipse Kepler. [Online] Eclipse Foundation, 2013. [Zitat vom: 18. 04 2021.] https://www.eclipse.org/downloads/packages/release/kepler.

Eclipse RCP. 2013. Eclipse for RCP and RAP Developers. [Online] Eclipse Foundation, 2013. [Zitat vom: 18. 04 2021.] https://www.eclipse.org/downloads/packages/release/kepler/sr2/eclipse-rcp-and-rap-developers.

Ege, Börtecin. 2015. Marktstudie: Welche Standards und Tools werden in Unternehmen eingesetzt? [Buchverf.] Börtecin Ege, Bernhard Humm und Anatol Reibold. *Corporate Semantic Web: Wie semantische Anwendungen in Unternehmen Nutzen stiften.* Berlin Heidelberg : Springer Vieweg, 2015.

Ehrlinger, Lisa und Wöß, Wolfram. 2016. Towards a Definition of Knowledge Graphs. [Buchverf.] Michael Martin, Martí Cuquet und Erwin Folmer. *Joint Proceedings of the Posters and Demos Track of the 12th International Conference on Semantic Systems - SEMANTiCS2016 and the 1st International Workshop on Semantic Change & Evolving Semantics (SuCCESS'16).* Leipzig, Germany : CEUR Workshop Proceedings, 2016, Bde. Vol-1695.

Eigner, Martin und Stelzer, Ralph. 2009. *Product Lifecycle Management.* Berlin Heidelberg : Springer-Verlag, 2009.

El Hajjamy, Oussama, et al. 2016. Mapping UML to OWL 2 Ontology. *Journal of Theoretical and Applied Information Technology.* 2016, Bd. Vol. 90. No.1.

EU-Verordnung. 2014. Richtlinie 2014/30/EU (EMV-Richtlinie) Elektromagnetische Verträglichkeit von Elektro- und Elektronikprodukten – EMV. [Online] CE-Kennzeichnung, 2014. [Zitat vom: 06. 04 2021.] https://www.ce-richtlinien.eu/emv-richtlinie/.

FeatureIDE. 2007. An extensible framework for feature-oriented software development. [Online] FeatureIDE development team, 2007. [Zitat vom: 11. 04 2021.] https://featureide.github.io/.

Franz Inc. 2021. Gruff - Graph Visualization and Graphical Query Builder. [Online] Franz Inc. - Semantic Graph and Common Lisp Solutions, 2021. [Zitat vom: 05. 04 2021.] https://franz.com/agraph/gruff/.

Galindo, José A. , et al. 2018. Automated analysis of feature models: Quo vadis? *Computing.* 2018, Bd. Volume 101 Issue 5.

Gausemeier, Jürgen, et al. 2006. *Vernetzte Produktentwicklung - Der erfolgreiche Weg zum global engineering networking.* München, Berlin : Carl Hanser Verlag, 2006.

Gebhardt, Vera, et al. 2013. *Funktionale Sicherheit nach ISO 262622. Ein Praxisleitfaden zur Umsetzung.* Heidelberg : dpunkt.verlag, 2013.

Gemmerich, Ralf. 2008. *Eine Methode zur Generierung einer kostenoptimalen Bordnetzarchitektur Schwerpunkt: Energieversorgung.* Kassel : Kassel University Press, 2008.

Gemmerich, Ralf, et al. 2016. Methode zur Bestimmung der Bordnetzzuverlässigkeit. [Buchverf.] Carsten Hoff und Ottmar Sirch. *Elektrik/Elektronik in Hybrid- und*

Elektrofahrzeugen und elektrisches Energiemangement 7. Renningen : expert Verlag, 2016, Bd. Haus der Technik Fachbuch Band 142.

Gerlicher, Ansgar . 2008. Implementing XML data formats for the exchange and migration of electrical harness information. *In Proceedings of the ProSTEP iViP Symposium.* Berlin : ProSTEP iViP, 2008, S. 136-145.

Giunchiglia, Fausto und Zaihrayeu, Ilya. 2009. Lightweight Ontologies. [Buchverf.] Ling Liu und M. Tamer Özsu. *Encyclopedia of Database Systems.* Boston, MA : Springer, 2009.

Gräbel, Patrick. 2013. *Eine Testmethodik zur automatisierten Validierung von elektrischen Kfz.-Bordnetzen.* Kassel : Kassel University Press, 2013.

Grabowski, Hans und Anderl, Reiner. 1990. *Produktdatenaustausch und CAD-Normteile: für Konstruktionsleiter, Normstellenleiter, CAD-Projektleiter, DV-Leiter.* Böblingen : expert Verlag, 1990.

Grabowski, Hans, et al. 1993. *Integriertes Produktmodell.* Berlin, Wien, Zürich : Beuth, 1993.

GraphML Working Group. 2019. The GraphML File Format. [Online] Graph Drawing Steering Committee, 2019. [Zitat vom: 07. 04 2021.] http://graphml.graphdrawing.org.

Gutierrez, Claudio und Sequeda, Juan F. 2021. Knowledge Graphs. *Communications of the ACM. Association for Computing Machinery.* 2021, Bd. Vol. 64, No. 3.

Hart, Laura E. 2015. Introduction To Model-Based System Engineering (MBSE) and SysML. [Online] 30. Juli 2015. [Zitat vom: 24. 11 2019.] https://www.incose.org/docs/default-source/delaware-valley/mbse-overview-incose-30-july-2015.pdf.

Hartig, Olaf. 2017. Foundations of RDF⋆ and SPARQL⋆ (An Alternative Approach to Statement-Level Metadata in RDF). *Alberto Mendelzon Workshop on Foundations of Data Management and the Web.* Montevideo, Uruguay : CEUR Workshop Proceedings, 2017, Bd. Vol 1912.

Hauck, Uwe, et al. 2019. Thermosimulation für das High Power Charging (HPC) von Elektrofahrzeugen. [Buchverf.] Carsten Hoff und Ottmar Sirch. *Elektrik/Elektronik in Hybrid- und Elektrofahrzeugen und elektrisches Energiemanagement IX.* s.l. : expert verlag GmbH, 2019, Bd. Haus der Technik Fachbuch Band 148.

Hellmuth, Thomas W. 1994. *Datenmodellierung zur marktgerechten Führung der Produktionsbereiche.* Ulm : Vieweg+Teubner Verlag, 1994.

Herrmann, Raffael. 2012. Wissenspyramide. [Online] 12. 09 2012. [Zitat vom: 23. 05 2021.] https://derwirtschaftsinformatiker.de/2012/09/12/it-management/wissenspyramide-wiki/.

Hesse, Wolfgang. 2006. Modelle - Janusköpfe der Software-Entwicklung - oder: Mit Janus von der A- zur S-Klasse. [Buchverf.] Mayr Heinrich C. und Ruth Breu. *Modellierung 2006.* Bonn : Gesellschaft für Informatik e.V., 2006.

Hillenbrand, Martin. 2012. *Funktionale Sicherheit nach ISO 26262 in der Konzeptphase der Entwicklung von Elektrik / Elektronik Architekturen von Fahrzeugen.* Karlsruhe : KIT Scientific Publishing, 2012.

Hitzler, Pascal, et al. 2008. *Semantic Web.* Berlin Heidelberg : Springer-Verlag, 2008.

INCOSE. 1990. Systems Engineering. [Online] International Council on Systems Engineering, 1990. [Zitat vom: 11. 04 2021.] https://www.incose.org/about-systems-engineering/system-and-se-definition/systems-engineering-definition.

Inoue, Takuya, et al. 2000. *The Development of a Method to Estimate the Bending Reliability of Wiring Harness.* Detroit, Michigan : SAE 2000 World Congress SAE International, 2000.

ISO 10303. 2004. ISO 10303-11:2004 Industrial automation systems and integration — Product data representation and exchange — Part 11: Description methods: The EXPRESS language reference manual. [Online] International Organization for Standardization, 2004. https://www.iso.org/standard/38047.html.

ISO 26262. 2018. ISO 26262-1:2018 Road vehicles — Functional safety. [Online] 2018. https://www.iso.org/standard/68383.html.

JUnit. 2021. JUnit - Programmer-friendly testing framework for Java and the JVM. [Online] The JUnit Team, 2021. [Zitat vom: 17. 04 2021.]

Junk, Christopher, et al. 2015. Model-Based Variant Management with v.control. [Buchverf.] Richard Curran, et al. *Transdisciplinary Lifecycle Analysis of Systems.* Online : IOS Press, 2015, Bd. Advances in Transdisciplinary Engineering.

Kang, Kyo C., et al. 1990. *Feature-Oriented Domain Analysis (FODA). Feasibility Study.* Pittsburgh, PA : Carnegie-Mellon University. Software Engineering Institute., 1990.

Kirsch, Michael , Kleiner, Sven und Minkner, Oliver. 2019. Intelligent Information Management. Digitale Durchgängigkeit von 3D-Produktinformationen. *Produkt Daten Journal prostep ivip e.v.* 2019, 1.

Kitsios, Vasileios und Haslauer, Richard . 2014. *3D-Master. Zeichnungslose Produktbeschreibung mit CATIA V5.* München : Springer Vieweg, 2014.

Koneksys. 2019. RDF Version Control Project. *GitHub.* [Online] 2019. [Zitat vom: 22. 11 2019.] https://github.com/koneksys/Git4RDF.

Kontokostas, Dimitris, et al. 2014. Test-driven evaluation of linked data quality. *WWW '14 Proceedings of the 23rd international conference on World wide web.* Seoul, Korea : ACM New York, 2014.

Krima, Sylvere I. , et al. 2009. OntoSTEP: OWL-DL Ontology for STEP. [Online] 2009. [Zitat vom: 21. 12 2019.] https://www.nist.gov/publications/ontostep-owl-dl-ontology-step.

Kuhn, Marlene und Nguyen, Huong Giang. 2019. *The Future of Harness Development and Manufacturing. Results from an Expert Case Study.* Nürnberg : The Institute for Factory Automation and Production Systems. Friedrich-Alexander University Erlangen-Nuremberg, 2019.

Kuhn, Thomas. 2017. Digitaler Zwilling. *Informatik Spektrum.* 2017.

Kyriazis, Jorgos. 2019. Dokumentation KBL (VDA 4964) VOBES spezifische Erweiterungen zu KBL2.4 SR-1. [Online] 2019. https://ecad-wiki.prostep.org/specifications/kbl/dok_vobes-kbl-format_20190329.pdf.

—. 2016. Product Data Model for Vehicle Electric Systems. Landshut : WEKA Fachmedien GmbH, 2016. In Tagungsunterlagen Bordnetz Kongress 2016.

Matlabcontrol. 2013. A Java API to interact with MATLAB. [Online] Google Code, 2013. [Zitat vom: 18. 04 2021.] https://code.google.com/archive/p/matlabcontrol/.

MBSE Initiative. 2007. MBSE Wiki. [Online] 2007. [Zitat vom: 11. 04 2021.] https://www.omgwiki.org/MBSE/doku.php?id=start.

Meyna, Arno und Pauli, Bernhard. 2010. *Zuverlässigkeitstechnik. Quantitative Bewertungsverfahren.* München Wien : Carl Hanser Verlag, 2010.

NASA. 2013. Model-based Systems Engineering. [Online] 2013. [Zitat vom: 24. 11 2019.] https://trs.jpl.nasa.gov/bitstream/handle/2014/43989/13-0392_A1b.pdf?sequence=1&isAllowed=y.

Neckenich, Jonas. 2017. *3D-Master-Leitungssatz - Konzept zur Entwicklung von Leitungssätzen als 3D-Master in einem realistischen, vollständigen DMU-Modell.* Saarbrücken : Publikations-Server der Universität des Saarlandes, 2017.

OASIS. 2013. Lifecycle integration inspired by the web. [Online] Organization for the Advancement of Structured Information Standards, 2013. [Zitat vom: 17. 04 2021.] http://www.oasis-oslc.org/.

OAT. 2013. Ontology Action Team. [Online] 11. 06 2013. [Zitat vom: 25. 11 2019.] https://www.omgwiki.org/MBSE/doku.php?id=mbse:ontology.

OSLC. 2019. Open Services for Lifecycle Collaboration - Creating standard REST APIs to connect data. [Online] 2019. [Zitat vom: 17. 04 2021.] https://open-services.net/.

—. 2011. Quality Management Specification Version 2.0. [Online] Open Services for Lifecycle Collaboration, 2011. [Zitat vom: 17. 04 2021.] https://archive.open-services.net/bin/view/Main/QmSpecificationV2.html.

OWL Working Group. 2012. OWL 2 Web Ontology Language. [Online] World Wide Web Consortium, 2012. [Zitat vom: 17. 12 2019.] https://www.w3.org/TR/owl2-overview/.

—. 2012. OWL 2 Web Ontology Language Profiles (Second Edition). [Online] World Wide Web Consortium, 2012. [Zitat vom: 18. 12 2019.] https://www.w3.org/TR/owl2-profiles/.

Perzylo, Alexander, et al. 2015. An ontology for CAD data and geometric constraints as a link between product models and semantic robot task descriptions. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* Hamburg, Germany : IEEE, 2015.

Picard, André Charles Roger. 2015. *Integriertes Werkstückinformationsmodell zur Ausprägung werkstückindividueller Fertigungszustände.* Aachen : Shaker Verlag, 2015.

Poveda-Villalón, María. 2021. OOPS! (OntOlogy Pitfall Scanner!). [Online] 2021. [Zitat vom: 09. 04 2021.] http://oops.linkeddata.es/.

Poveda-Villalón, María, et al. 2015. OOPS! (OntOlogy Pitfall Scanner!): supporting ontology evaluation on-line. *Semantic Web – Interoperability, Usability, Applicability an IOS Press Journal.* 2015.

Poveda-Villalón, María, Suárez-Figueroa, Mari Carmen und Gómez-Pérez, Asunción. 2012. Validating Ontologies with OOPS! [Buchverf.] Annette Ten Teije, et al. *Knowledge Engineering and Knowledge Management.* 18th International Conference, EKAW 2012, Galway City, Ireland : Springer, 2012.

Project Group Car Electric. 2005. Harness Description List (KBL) 4964 V2. [Online] Verband der Automobilindustrie e.V., 01. 11 2005. [Zitat vom: 28. 04 2018.] https://www.vda.de/en/services/Publications/4964---harness-description-list.html.

prostep ivip. 2019. Code of PLM Openness. [Online] prostep ivip Verein, 2019. [Zitat vom: 18. 04 2021.] https://www.prostep.org/projekte/code-of-plm-openness/.

—. 1993. prostep ivip Verein. [Online] 1993. [Zitat vom: 06. 04 2021.] https://www.prostep.org/.

—. 2020. Vehicle Electric Container. [Online] 2020. [Zitat vom: 23. 05 2021.] https://ecad-wiki.prostep.org/specifications/vec/.

Quit Store Project. 2019. Quit Store Project. *GitHub.* [Online] 2019. [Zitat vom: 22. 11 2019.] https://github.com/AKSW/QuitStore.

RDB2RDF Working Group. 2012. R2RML: RDB to RDF Mapping Language. [Online] World Wide Web Consortium, 2012. [Zitat vom: 05. 04 2021.] https://www.w3.org/TR/r2rml/.

RDF Data Shapes Working Group. 2017. SHACL JavaScript Extensions. [Online] World Wide Web Consortium, 08. 06 2017. [Zitat vom: 18. 12 2019.] https://www.w3.org/TR/shacl-js/.

—. 2017. Shapes Constraint Language (SHACL). [Online] World Wide Web Consortium, 20. 06 2017. [Zitat vom: 23. 04 2018.] https://www.w3.org/TR/shacl/.

RDF Working Group. 2014. RDF 1.1 N-Quads. A line-based syntax for RDF datasets. [Online] World Wide Web Consortium, 25. 02 2014. [Zitat vom: 23. 11 2019.] https://www.w3.org/TR/n-quads/.

—. 2014. RDF Schema 1.1. [Online] World Wide Web Consortium, 2014. [Zitat vom: 17. 12 2019.] https://www.w3.org/TR/rdf-schema/.

—. 2014. Resource Description Framework. [Online] World Wide Web Consortium, 2014. [Zitat vom: 05. 04 2021.] https://www.w3.org/TR/rdf11-concepts/.

—. 2014. Terse RDF Triple Language. [Online] World Wide Web Consortium, 2014. [Zitat vom: 05. 04 2021.] https://www.w3.org/TR/turtle/.

Rehman, Zobia und Kifor, Claudiu Vasile. 2016. An Ontology to Support Semantic Management of FMEA Knowledge. *INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL.* 2016, Bd. Vol 11 No 4 .

Reif, Konrad. 2014. *Batterien, Bordnetze und Vernetzung.* Wiesbaden : Vieweg+Teubner, 2014.

Rigolet, Jean-Yves. 2011. From silos to agility: Rethinking enterprise software delivery with the Rational solution for Collaborative Lifecycle Management. [Online] 20. Juli 2011. [Zitat vom: 17. 04 2021.] https://jazz.net/library/article/648.

Rock, Georg. 2009. Variantenmanagement - Forschung und industrieller Einsatz. [Online] 2009. [Zitat vom: 05. 12 2019.] https://www.es.tu-darmstadt.de/fileadmin/download/lehre/ik/PROSTEP-IMP-Variantenmanagement-ASE-TUD-GRock.pdf.

Rosenplänter , Sylke, et al. 2016. Hat PLM im Zeitalter der Digitalisierung eine Zukunft? Freies Thesenpapier Future PLM. *prostep ivip Verein.* [Online] 2016. [Zitat vom: 22. 05 2021.] https://www.prostep.org/fileadmin/downloads/Thesen_PLM-Future_160725.pdf.

Ross, Hans-Leo. 2014. *Funktionale Sicherheit im Automobil.* München Wien : Carl Hanser Verlag, 2014.

Sabou, Marta und Fernandez, Miriam. 2012. Ontology (Network) Evaluation. [Buchverf.] Mari Carmen Suárez-Figueroa, et al. *Ontology Engineering in a Networked World.* Berlin : Springer, 2012.

Sabou, Marta, et al. 2017. Beiträge des Semantic Web zum Engineering für Industrie 4.0. [Buchverf.] Birgit Vogel-Heuser, Thomas Bauernhansl und Michael ten Hompel. *Handbuch Industrie 4.0 Bd.2.* Berlin, Heidelberg : Springer Vieweg, 2017.

Schekotihin, Konstantin, Rodler, Patrick und Schmid, Wolfgang. 2018. OntoDebug: Interactive Ontology Debugging Plug-in for Protégé. [Buchverf.] Flavio Ferrarotti und Stefan Woltran. *Foundations of Information and Knowledge Systems.* 10th International Symposium, FoIKS 2018, Budapest, Hungary : Springer, 2018, Bd. International Symposium on Foundations of Information and Knowledge Systems.

Schuhbauer, Heidi, Fuhr, Thomas und Wittmann, Susanne. 2014. Flexible Informationsstrukturen mit Ontologien. *HMD Praxis der Wirtschaftsinformatik, Springer.* 2014, 45.

Schwimmbeck, Stefan und Weissinger, Christoph. 2020. Virtuelle Energiebordnetz-Entwicklung im Zeitalter der Digitalisierung. Landshut : WEKA Fachmedien GmbH, 2020. In Tagungsunterlagen Bordnetz Kongress 2020.

Seibertz, Achim, Brandstätter, Markus und Schreiber, Kai. 2013. Kompositionales Variantenmanagement – Ganzheitlicher Ansatz zur Komplexitätsbeherrschung im Systems Engineering Umfeld. [Buchverf.] Maik Maurer und Sven-Olaf Schulze. *Tag des Systems Engineering. Zusammenhänge erkennen und gestalten.* Paderborn : Carl Hanser Verlag, 2013.

Simon, Klaus. 2008. *Modellierung einer EE-Architektur und qualitative Bewertung des Absicherungskozept (Masterarbeit).* Kassel : Universität Kassel, 2008.

Sindermann, Sebastian. 2014. Schnittstellen und Datenaustauschformate. [Buchverf.] Martin Eigner, Daniil Roubanov und Radoslav Zafirov. *Modellbasierte Virtuelle Produktentwicklung.* Berlin Heidelberg : Springer Vieweg, 2014.

Soylu, Ahmet , et al. 2016. Experiencing OptiqueVQS: a multi-paradigm and ontology-based visual query system for end users. *Universal Access in the Information Society.* 2016, Bd. Volume 15 Issue 1.

Soylu, Ahmet, et al. 2018. OptiqueVQS: A visual query system over ontologies for industry. *Semantic Web.* 2018, Bde. Vol. 9, No. 5.

SPARQL Working Group. 2013. SPARQL 1.1 Federated Query. [Online] World Wide Web Consortium, 2013. [Zitat vom: 02. 02 2020.] https://www.w3.org/TR/sparql11-federated-query/.

—. 2013. SPARQL 1.1 Overview. [Online] World Wide Web Consortium, 2013. [Zitat vom: 18. 12 2019.] https://www.w3.org/TR/sparql11-overview/.

Stardog. 2019. Adressing Model-Based System Engineering Challenges With Knowledge Graph. [Online] 2019. [Zitat vom: 24. 11 2019.] https://www.stardog.com/resources/addressing-model-based-system-engineering-challenges-with-knowledge-graphs/.

Steenwinckel, Bram, et al. 2018. Automated extraction of rules and knowledge from risk analyses: a ventilation unit demo. *ISWC-P&D-Industry-BlueSky.* Monterey, USA : CEUR Workshop Proceedings, 2018, Bd. Vol 2180.

Studer, Rudi, Benjamins, V.Richard und Dieter, Fensel. 1998. Knowledge engineering: Principles and methods. *Data & Knowledge Engineering.* 1998, Bde. Volume 25, Issues 1–2.

SWRL Working Group. 2004. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. [Online] World Wide Web Consortium, 21. 05 2004. [Zitat vom: 23. 04 2018.] https://www.w3.org/Submission/SWRL/.

Thüm, Thomas, et al. 2011. Abstract Features in Feature Modeling. *In Proceedings of the 15th International Software Product Line Conference (SPLC). Association for Computing Machinery.* 2011, Bd. 2.

Vargas, Hernan, Aranda, Carlos Buil und Hogan, Aidan. 2019. RDF Explorer: A Visual Query Builder for Semantic Web Knowledge Graphs. *ISWC2019-Satellites.* Auckland, New Zealand : CEUR Workshop Proceedings, 2019, Bd. Vol 2456.

Venceslau, Allan , et al. 2014. Ontology for computer-aided fault tree synthesis. *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA).* 2014.

VES-WF. 2019. Vehicle Electric Systems Workflow Forum. [Online] prostep ivip Verein, 2019. https://www.prostep.org/projekte/vehicle-electric-systems-workflow-forum-ves-wf/.

W3C. 2011. Category: OWL Reasoner. [Online] World Wide Web Consortium, 2011. [Zitat vom: 05. 04 2021.] https://www.w3.org/2001/sw/wiki/Category:OWL_Reasoner.

—. 2013. Category: Triple Store. [Online] World Wide Web Consortium , 2013. [Zitat vom: 05. 04 2021.] https://www.w3.org/2001/sw/wiki/Category:Triple_Store.

—. 2020. Large Triple Stores. [Online] World Wide Web Consortium, 2020. [Zitat vom: 05. 04 2021.] https://www.w3.org/wiki/LargeTripleStores.

—. 2018. RDF Store Benchmarking. [Online] World Wide Web Consortium, 2018. [Zitat vom: 18. 04 2021.] https://www.w3.org/wiki/RdfStoreBenchmarking.

—. 2009. Semantic Web Development Tools. [Online] World Wide Web Consortium, 03. 12 2009. [Zitat vom: 12. 11 2019.] https://www.w3.org/2001/sw/wiki/Category:Programming_Language.

Wang, Hai H., et al. 2007. Verifying feature models using OWL. *Journal of Web Semantics.* 2007, Bd. Volume 5 Issue 2.

Wappis, Johann und Jung, Berndt. 2010. *Taschenbuch Null-Fehler-Management: Umsetzung von Six Sigma.* Wien : Carl Hanser Verlag, 2010.

Weinrich, Ulrike. 2018. *Methoden zur Bestimmung der Ausfallraten von elektrischen und elektronischen Systemen am Beispiel der Lenkungselektronik.* Stuttgart, : Springer Vieweg, 2018.

Wikipedia. 2018. Betriebliches Informationssystem. *Wikipedia, Die freie Enzyklopädie.* [Online] 04. 11 2018. [Zitat vom: 23. 05 2021.] https://de.wikipedia.org/w/index.php?title=Betriebliches_Informationssystem&oldid=182454168.

Witte, Frank. 2019. *Testmanagement und Softwaretest.* Wiesbaden : Springer Vieweg, 2019.

Zedlitz, Jesper, Jörke, Jan und Luttenberger, Norbert. 2011. From UML to OWL 2. [Buchverf.] Dickson Lukose, Abdul Rahim Ahmad und Azizah Suliman. *Knowledge Technology.* Third Knowledge Technology Week, KTW 2011, Kajang, Malaysia : Springer, 2011.

Zhu, Zaoxu, van Tooren , Michael und La Rocca, G. 2012. A KBE Application for Automatic Aircraft Wire Harness Routing. *Structures, Structural Dynamics and Materials Conference.* s.l. : American Institute of Aeronautics and Astronautics, 2012.

Zündorf, Albert, et al. 2010. Using Graph Grammars for Modeling Wiring Harnesses - An Experience Report. *Graph Transformations and Model-Driven Engineering.* Berlin Heidelberg : Springer, 2010, S. 512-532.

Ein integriertes und erweiterbares Produktmodell, das als „Single-Source of Truth“ dient, ist nach wie vor ein Ziel der Bordnetzentwicklung. Ein solches Produktmodell ist notwendig, um ein Bordnetz ausführlich digital auszuwerten und in naher Zukunft nur noch digital zu entwickeln. Dabei ist die Integration der Systeme mit hoher Komplexität und mit Hindernissen verbunden. Diese Arbeit untersucht die Kernprobleme auf der Datenebene, die robuste und nachhaltige Integration verhindern. Als Lösungsansatz werden in dieser Arbeit Ontologien bzw. Knowledge-Graph-Technologien und Methoden des Software-Engineerings für die strukturierte und semantische Datenintegration und -verarbeitung auf Basis des standardisierten Datenmodells der Bordnetzdaten vorgestellt. Aufbauend wird gezeigt, wie in diesem Ansatz die normgerechte Zuverlässigkeitsanalyse auf der Grundlage der funktionalen Sicherheit effektiv realisiert werden kann.

ISBN 978-3-7376-1020-9

9 783737 610209 >