Thomas Gruber

Prozessintegrierte Dokumentation und optimierte Wiederverwendung von Simulationsmodellen der automobilen Funktionsabsicherung

Wissenschaftliche Schriftenreihe
EINGEBETTETE, SELBSTORGANISIERENDE SYSTEME
Band 15

Prof. Dr. Wolfram Hardt (Hrsg.)

Thomas Gruber

Prozessintegrierte Dokumentation und optimierte Wiederverwendung von Simulationsmodellen der automobilen Funktionsabsicherung

TECHNISCHE UNIVERSITÄT
CHEMNITZ

Universitätsverlag Chemnitz
2016

Impressum

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Angaben sind im Internet über http://dnb.d-nb.de abrufbar.

Cover-Illustration: Maria Feifarek
Satz/Layout: Thomas Gruber

Technische Universität Chemnitz/Universitätsbibliothek
Universitätsverlag Chemnitz
09107 Chemnitz
http://www.tu-chemnitz.de/ub/univerlag

Herstellung und Auslieferung
Verlagshaus Monsenstein und Vannerdat OHG
Am Hawerkamp 31
48155 Münster
http://www.mv-verlag.de

ISSN 2196-3932 (Druck) - ISSN 2196-4815 (online)
ISBN 978-3-944640-89-1

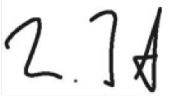
http://nbn-resolving.de/urn:nbn:de:bsz:ch1-qucosa-202845

Vorwort zur Wissenschaftlichen Schriftenreihe „Eingebettete, Selbstorganisierende Systeme“

Der vorliegende Band der wissenschaftlichen Schriftenreihe *Eingebettete, Selbstorganisierende Systeme* widmet sich der methodischen Unterstützung von Entwicklungsprozessen der automobilen Funktionsentwicklung. Mit dem neuen Umschlagdesign aktualisiert die Schriftenreihe das Erscheinungsbild und passt sich in das Corporate Design der Technischen Universität Chemnitz ein. Die Entwicklung und Absicherung moderner, hochvernetzter Fahrerassistenzsysteme stellt hohe Anforderungen an die Entwicklungsmethoden der Automobilhersteller. Sie sehen sich dabei im Spannungsfeld zwischen den zunehmend wichtigen gesetzlichen Nachweispflichten bezüglich sicherheitskritischer Fahrzeugfunktionen wie sie beispielsweise die ISO 26262 fordert und der Umsetzung effizienter und marktwirtschaftlich konkurrenzfähiger Entwicklungsprozesse. Herr Gruber definiert in seiner Arbeit eine Methodik zur Erfassung dokumentationsrelevanter Informationen in bestehenden Entwicklungsprozessen ohne in den Prozessablauf verändernd einzugreifen. Aufbauend auf dieser Dokumentation schlägt er Methoden vor, durch die eine Wiederverwendung von Entwicklungsgegenständen über den gesamten Entwicklungsprozess gefördert, Ressourcen gespart und gleichzeitig die Rückverfolgbarkeit sichergestellt werden kann.

Dabei sind Schwerpunkte der Arbeit die Untersuchung und Definition einer minimalen Informationsbasis durch eine prozessintegrierte Erfassung aus bestehenden Datenquellen sowie deren Weiterverarbeitung zu einer Wissensdatenbank, die eine zielgerichtete Wiederverwendung unterstützt. In der Phase der Informationserfassung nutzt ein modellbasierter Dokumentationsprozess die etablierte Kompetenz der modellbasierten Entwicklung des Entwicklungsingenieurs, um dessen Bewertung der Relevanz zielgerichtete einsetzen zu können. Anschließend zeigt die Arbeit eine auf der Graphtransformation aufbauende Methodik zur Generierung einer inferenzfähigen Wissensbasis auf, die zur strukturierten Speicherung und zur Nutzung in Suchmaschinen oder Empfehlungssystemen herangezogen werden kann.

Ich freue mich, Herrn Gruber für die Veröffentlichung seiner Arbeit in dieser wissenschaftlichen Schriftenreihe gewonnen zu haben und wünsche allen Lesern viel Freude und Nutzen bei der Lektüre.



Prof. Dr. Wolfram Hardt
Professur Technische Informatik
Dekan Fakultät für Informatik

TECHNISCHE UNIVERSITÄT
CHEMNITZ

Prozessintegrierte Dokumentation und optimierte Wiederverwendung von Simulationsmodellen der automobilen Funktionsabsicherung

Dissertation

zur
Erlangung des akademischen Grades
Dr.-Ing.

Fakultät für Informatik
Professur Technische Informatik

eingereicht von: Herrn Thomas Gruber geboren am 14.08.1985 in Karl-Marx-Stadt

Erstgutachter: Prof. Dr. Wolfram Hardt
Zweitgutachter: Prof. Dr. Reinhard German

Tag der Verteidigung: 04.05.2016

Kurzfassung

Die Schaffung, Wahrung und Nutzung von Wissen stellt heute eine wichtige Säule für die Konkurrenzfähigkeit von Unternehmen am Markt dar. Vor diesem Hintergrund steht insbesondere die moderne Funktionsentwicklung der Automobilindustrie vor der Herausforderung immer neue, hochgradig vernetzte Fahrzeugfunktionen zu entwickeln und in immer kürzerer Zeit und immer geringeren Kosten in den Markt zu bringen.

Um dieser Herausforderung gerecht zu werden, hat sich die modellbasierte Entwicklung mit dem Ziel der Beherrschung dieser steigenden Komplexität etabliert. Dadurch ist es möglich die Entwicklungsaufgaben auf unterschiedlichen Ebenen zu abstrahieren und eine verteilte, vernetzte Entwicklung zu realisieren. Die Entwicklung einer einzigen Funktion benötigt heute häufig mehrere hundert Personen, die in einen gemeinsamen Entwicklungsprozess integriert werden müssen. Hier fehlt es an Konzepten um den Informations- und Wissensfluss zwischen den Prozessbeteiligten sicherzustellen.

In diesem Kontext entwickelt die vorliegende Arbeit einen Ansatz zur prozessintegrierten Dokumentation der in modellbasierten Entwicklungsprozessen benötigten Entwicklungsartefakte. Der Ansatz betrachtet dabei den vollständigen Informationsfluss, von der Definition benötigter Informationen, über deren automatisierte Erfassung und Verarbeitung bis zur zielgerichteten Wiederverwendung.

Anschließend skizziert die Arbeit die Architektur eines Informationssystems, dass diese Durchgängigkeit in beliebigen, modellbasierten Entwicklungsprozessen ermöglicht und überträgt diese zur Validierung des Ansatzes auf einen konkreten Entwicklungsprozess der automobilen Funktionsentwicklung.

Der Fokus des Ansatzes liegt dabei insbesondere auf der Integration in bestehende Entwicklungsprozesse, ohne in diese verändernd einzugreifen. Dies wird einerseits durch eine modellbasierte Beschreibung des Informationsmodells, mit Methoden wie sie im Funktionsentwicklungsprozess Anwendung finden, erreicht. Prozessbeteiligte können dadurch das Informationsmodell selbst verstehen und bei Bedarf Anpassungen vornehmen, ohne auf geschulte Experten angewiesen zu sein. Andererseits erlaubt der architektonische Ansatz einen direkten Zugriff auf bestehende Entwicklungssysteme und darin enthaltenen dokumentationsrelevanten Informationen.

Abstract

Today, the creation, preservation and exploitation of knowledge represent key factors of the competitiveness of companies in the global market. In this context, the modern function development in the automotive industry faces challenges to bring new, highly interconnected vehicle functions at shorter time and lower cost to the market.

To meet these challenges and manage the growing compelity, a model-based development process has been established. Thus, it is possible to distribute development tasks to different levels of abstraction and enable a distributed, interconnected function development. This development involves up to several hundred persons per function, who have to be integrated in a common development process. Especially when it comes to managing the information and knowledge flow between the process participants, there is a lack of concepts to support this communication.

Based on this context, this work presents an approach for process integrated documentation of the necessary development artifacts in model-based development processes. This approach considers the complete information flow, from the definition of necessary information, over automatic acquisition and processing to its targeted reuse during the process.

Subsequently, this work sketches the architecture of an information system, which enables this continuous approach to be applied to any model-based development process. For validation purposes, the approach is then applied to an actual development process of the automotive function development.

The focus of the presented approach lies in the integration in existing development processes without changing them. On the one hand, this is achieved by applying a model-based description of the information model using methods, that can be found in the function development process today. Thus, process participants can understand the information model themselves and apply changes when they are required without the necessity of qualified experts. On the other hand, the architectural approach allows a direct access to existing development systems and documentation relevant information they contain.

Danksagung

Die vorliegende Arbeit ist während meiner Tätigkeit in der Abteilung *Prozesse / Methoden für die Technische Entwicklung im Bereich Eigenschaften / Funktionen* bei der AUDI AG in Ingolstadt und in Zusammenarbeit mit dem Lehrstuhl Technische Informatik der Technischen Universität Chemnitz von Herrn Prof. Dr. Wolfram Hardt entstanden.

Prof. Dr. Wolfram Hardt gilt mein Dank für die hervorragende Betreuung und die sehr interessanten und hilfreichen Diskussionen in Chemnitz und Laubusch. Mit vielen guten und anregenden Ideen hat er entscheidend zum Gelingen der Arbeit beigetragen.

Bei Prof. Dr. Reinhard German möchte ich mich sowohl für die Möglichkeit meine Arbeit an seinem Lehrstuhl vorzustellen als auch für das von ihm erstellte Gutachten bedanken.

Mein besonderer Dank gilt meinem Freund und Mentor vom ersten Tag an, Herrn Dr. Sebastian Thiel, der mich mit vielen hilfreichen Diskussionen, konstruktiver Kritik und stets einem offenen Ohr durch alle Berge und Täler der Entstehung dieser Arbeit begleitet und unterstützt hat.

Bei meinen Kollegen Matthias Wiedemann und Dr. Harald Wilhelm möchte ich mich für die stets anregenden Diskussionen und vielen hilfreichen Ratschläge bedanken.

Ganz herzlich bedanken möchte ich mich bei meiner Verlobten Maria Feifarek und meiner Mutter Helga Gruber, die mir stets Mut zugesprochen und mich in meiner Arbeit unterstützt haben.

Baar-Ebenhausen, den 10.05.2016
Thomas Gruber

Inhaltsverzeichnis

Abkürzungsverzeichnis

ABS	Antiblockiersystem
ACC	Adaptive Cruise Control
API	Application Programming Interface, eine Programmierschnittstelle
ARIS	Architektur integrierter Informationssysteme, ein Konzept für ein betriebliches Informationssystem
ASCII	American Standard Code for Information Interchange, ein Format zur Kodierung von Zeichen
BMBF	Bundesministerium für Bildung und Forschung
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAT	Computer Aided Testing
CDO	Connected Data Objects
DSL	Domain Specific Language
DTD	Document Type Definition
EMF	Eclipse Modelling Framework
EMV	Elektromagnetische Verträglichkeit
EnProVe	Softwareentwicklungsprozess für Fahrzeugfunktionen der AUDI AG
ESP	Elektronisches Stabilitätsprogramm
FCM	(Factor, Criteria, Metric)-Modell bildet die Grundlage eines Qualitätsmodells für Software
FMI	Functional Mockup Interface
HDL	Hardware Description Language
HIL	Hardware-in-the-Loop-Simulation
IP	Intellectual Property

IPCHL	Intellectual Property Characterization Language, eine Dokumentationssprache für Hardwarekomponenten
IPQ	Intellectual Property Qualification, Projekt zu Dokumentation und Austausch von Hardwarekomponenten
LOC	Lines of Code
MBT	Modell - Bauteil - Technik, ein IT-System der Fahrzeugentwicklung bei der AUDI AG
MIL	Model-in-the-Loop-Simulation
MIM	Modulares Informationsmanagementsystem
MOF	Meta Object Facility
MSB	Modularer Simulationsbaukasten (auch MDSim)
NLP	Natural Language Processing
OEM	Original Equipment Manufacturer (hier Automobilhersteller)
OMG	Object Management Group
OMIS	Organizational Memory Information System
OWL	Web Ontology Language
RDF(S)	Resource Description Framework (Schema)
SDM	Simulationsdatenmanagement
SIL	Software-in-the-Loop-Simulation
SQL	Structured Query Language
SQuaRE	Software product Quality Requirements and Evaluation
SVN	Subversion
SysML	Systems Modelling Language
UML	Unified Markup Language
UUID	Universally Unique Identifier
UUT	Unit Under Test
VHDL	Very High Speed Integrated Circuit Hardware Description Language
VW	Volkswagen
W3C	World Wide Web Consortium

XMI	XML Metadata Interchange
XML	eXtensible Markup Language
CAN	Controller Area Network, serielles Bussystem
XIL	X-in-the-Loop-Simulation, wobei das 'X' einen Platzhalter für die Simulationsverfahren Model-, Software- und Hardware-in-the-loop repräsentiert

1 Einleitung

Die schnelle Entwicklung der Informationstechnologien sowie deren tiefe Integration in das tägliche Leben stellt alle Wirtschaftszweige vor die Herausforderung dieser sich verändernden Welt und ihrem wachsenden Informationsbedarf gerecht zu werden. Heute spricht man von „Industrie 4.0", der vierten industriellen Revolution, die das Ziel der Schaffung einer intelligenten Fabrik verfolgt. Diese intelligente Fabrik ist gekennzeichnet von Anpassungsfähigkeit, Effizienz und der durchgegängigen Integration und Informationsversorgung von Ingenieur bis Kunde in Entwicklungs- und Herstellungsprozessen.

Vor diesem Hintergrund stellt die Schaffung, Wahrung und Nutzung von Wissen eine wichtige Säule für die Konkurrenzfähigkeit von Unternehmen am Markt dar. Noch vor wenigen Jahren waren die individuellen Leistungen eines einzelnen Mitarbeiters oder einer kleinen Gruppe für eine Entwicklungstätigkeit auf höchstem Niveau ausreichend. Heute erhöht sich die Komplexität von Entwicklungsprozessen, wirtschaftlichen Herausforderungen und technologischen Möglichkeiten so schnell, dass Unternehmen durch die Anpassung ihrer Entwicklungsstrukturen aktiv gegensteuern müssen, um diese in Zukunft beherrschen zu können. In diesem Zusammenhang werden neue Lösungen benötigt, die eine optimale Zusammenarbeit früherer, individuenzentrierter Organisationsstrukturen unterstützen.

Die Suche nach neuen Lösungen führt zum zunehmenden Einsatz von Informationssystemen in allen Industriezweigen. Wikis, Ticketing Systeme und neue Kollaborationswerkzeuge, wie sie früher lediglich in der Softwareentwickung zum Einsatz kamen, halten Einzug in die tägliche Arbeit. Dem gegenüber steht ein für den individuellen Nutzer nicht immer direkt erkennbarer Nutzen, der im Vergleich zum Mehraufwand für die Systempflege häufig zur Ablehnung führt. Hier verspricht der Einsatz von anwendungsoptimierten Wissensmanagementsystemen eine zielgerichtetere Unterstützung und damit eine höhere Akzeptanz der Nutzer.

Vor diesem Hintergrund steht die moderne Funktionsentwicklung der Automobilindustrie, die den Rahmen der vorliegenden Arbeit darstellt, vor der Herausforderung immer neue, hochgradig vernetzte Fahrzeugfunktionen zu entwickeln und in immer kürzerer Zeit und immer geringeren Kosten in den Markt zu bringen. Um dies in den Entwicklungsprozessen der Automobilhersteller abbilden zu können, kommt der virtuellen Entwicklung und Absicherung dieser Fahrzeugfunktionen in wachsendem Maße Bedeutung zu. Ehemals von kleinen Gruppen getriebene Entwicklungsprozesse und die Absicherung dieser Funktionen im realen Fahrversuch reichen für die Komplexität heutiger Funktionen nicht mehr aus.

Diese Entwicklungen führen zu Anpassungen der Funktionsentwicklungsprozesse weg von einer organisatorisch getrennten hin zu einer vernetzten Entwicklung, in der Wissenstransfer und Kommunikation über den Erfolg oder Misserfolg entscheiden. Für die automobile Funktionenwicklung hat sich hier ein durchgängig modellbasierter Entwicklungsprozess etabliert. Dieser beginnt mit der modellbasierten Softwareentwicklung und reicht über virtuelle Absicherungsmethoden von Model- über Software- bis Hardware-in-the-Loop-Simulation. Die beschriebene Transformation der Entwicklungsprozesse hin zur vernetzten Ent-

wicklung stellt aktuell hohe Anforderungen an die Funktionsentwicklungsabteilungen der Automobilindustrie. Insbesondere fehlt es an Methoden und Werkzeugen zur verteilten Entwicklung und den dafür benötigten Wissenstransfer. Hier setzt die vorliegende Arbeit an und schlägt einen neuen Ansatz vor, wie auf Grundlage etablierter Entwicklungsprozesse ein Informationsmanagement etabliert werden kann, das den Übergang zur kollaborativen Zusammenarbeit in verteilten, modellbasierten Entwicklungsprozessen unterstützt.

1.1 Umfeld

Die Untersuchungen und Entwicklungen der vorliegenden Arbeit wurden im Umfeld der automobilen Funktionsabsicherung durch X-in-the-Loop-Simulation der AUDI AG durchgeführt. Das „X" steht als Platzhalter für die Simulationverfahren Model-, Software- und Hardware-in-the-Loop.

In diesem Zusammenhang beschäftigt sich das Projekt *XIL-Datenmanagement* mit der Entwicklung einer Datenmanagementlösung für die genannten XIL-Simulationsverfahren. Diese Simulationen werden im Rahmen der funktionalen Absicherung von Fahrzeugen beispielsweise in den Domänen Elektronik, Fahrwerk und Antrieb eingesetzt. Die entsprechenden Fachabteilungen benötigen dabei in der Regel ähnliche Simulationsmodelle und Modellaufbauten, um ihre Absicherungsaufgaben durchführen zu können. Primär wird als Simulationssoftware MATLAB® Simulink®[Ang14] eingesetzt.

Das angestrebte Simulationsdatenmanagementsystem (SDM) soll einerseits als zentrale Datenhaltung der Simulationsmodelle dienen und andererseits Funktionalitäten einer Austauschplattform für die simulierenden Fachabteilungen zur Verfügung stellen. Je nach Simulationsverfahren und Entwicklungsdomäne existieren unterschiedliche fachliche und technische Anforderungen an die Modelle. Diese Anforderungen beschränken sich bei X-in-the-Loop-Simulationen nicht nur auf einzelne Simulationsmodelle, sondern gelten auch für die Verknüpfung dieser zu komplexeren Gesamtfahrzeugmodellen.

Eine besondere Herausforderung besteht in der Koordination der verteilten Entwicklung von Simulationsmodellen im Fahrzeugentwicklungsprozess. Der Entwicklungszeitraum erstreckt sich dabei von der frühen Konzeptionsphase bis weit nach der Einführung eines Fahrzeuges in den Markt. Heute liegt dieser Zeitraum bei mindestens vier Jahren. Als zentrale Datenmanagementplattform für Simulationsmodelle stellt dies hohe Anforderungen an die Dokumentations- und Rückverfolgbarkeitsinformationen von XIL-Datenmanagement. Um diese Anforderungen zu identifizieren und die damit verbundenen Herausforderungen wissenschaftlich zu untersuchen, wurde die vorliegende Arbeit im Kontext des Entwicklungsprojektes angesiedelt.

1.2 Aufgabenstellung und Ziele der Arbeit

Eine verbreitete Entwicklungsmethodik in vielen Industriezweigen stellt die modellbasierte Entwicklung dar. Im Rahmen der Forschung zur vorliegenden Arbeit sollen Möglichkeiten zur Unterstützung modellbasierter Entwicklungsprozesse durch zielgerichtetes und möglichst automatisiertes Wissensmanagement untersucht und ein durchgängiges Konzept

erstellt werden. Als Anwendungsfall wurde die automobile Funktionsentwicklung gewählt, in der sich ein modellbasierter Entwicklungsprozess bereits heute bei allen Herstellern etabliert hat (vgl. Abschnitt 1.1).

Die automobile Funktionsentwicklung ist stark geprägt durch das Vorgehen nach dem klassischen V-Modell [SZ13], das zunächst mit einer Dekompositionsphase beginnt und dabei eine Entwicklungsaufgabe nach dem Top-Down-Prinzip in kleine, beherrschbar Einzelaufgaben zerlegt. Die Einzellösungen werden später in einer Integrationsphase zunächst unabhängig voneinander abgesichert und anschließend zur finalen Funktion zusammengeführt. Strukturell bedeutet dieses Vorgehen für die Entwicklung eine zunehmende Spezialisierung der beteiligten Mitarbeiter in Ihrer Kernkompetenz je weiter unten im V-Modell sich ihre Arbeit ansiedelt. Selbst wenn von Mitarbeitern der sprichwörtliche „Blick über den Tellerrand" erwartet werden kann, so hält sich das detaillierte Wissen zur Arbeit anderer, nicht direkt in die eigene Arbeit involvierter Fachabteilungen sehr in Grenzen.

Ausgehend von der Idee, Simulationsmodelle über Fachbereichsgrenzen, Entwicklungsdomänen, -disziplinen und sogar -phasen hinweg auszutauschen, stellt sich die Frage, welche Informationen für einen solchen Datenaustausch mindestens notwendig sind, um eine Wiederverwendung von Simulationsmodellen zu ermöglichen und dadurch Mehrarbeit in den nutzenden Fachabteilungen zu vermeiden?

Darauf aufbauend werden im Rahmen der Arbeit Möglichkeiten der automatisierten funktionalen und qualitativen Bewertung von Simulationsmodellen und deren Zusammenschaltung zu komplexeren Modellen, z.B. Modellaufbauten, Mock-ups, untersucht. Ziel ist es der Herausforderung der mangelnden Dokumentation in fachspezifischen Prozessen zu begegnen. Grundlage ist die These, dass alle Informationen, die nötig sind, um Simulationsmodelle zielgerichtet zu suchen und eine erste Nutzenbewertung durchzuführen, bereits heute im Fahrzeugentwicklungsprozess vorhanden und dokumentiert sind. Ausgehend vom Stand der Technik sollen in den folgenden drei Arbeitsschwerpunkten Untersuchungen durchgeführt werden:

1. Definition einer minimalen Informationsbasis für den Austausch von Modellen
2. Entwicklung von Methoden zur automatischen Erfassung dieser minimalen Informationsbasis und Umsetzung dieser im Rahmen eines Demonstrators
3. Umsetzung einer Modellsuche sowie Entwicklung eines kriterienbasierten Modellempfehlungssystems

In einem **ersten Arbeitsschwerpunkt** zum Thema *Definition einer minimalen Informationsbasis* soll unter Berücksichtigung heterogener Entwicklungsstrukturen und Simulationsverfahren der X-in-the-Loop-Simulation ein Rahmen definiert werden, der die zur Suche und Bewertung von Simulationsmodellen notwendigen Daten zusammenfasst.

Im Rahmen des **zweiten Arbeitsschwerpunkts** *Untersuchung von Methoden zur automatischen Erfassung einer minimalen Informationsbasis* sollen Möglichkeiten der Datenbeschaffung in vorhandenen Softwaresystemstrukturen untersucht werden. Ziel ist die Entwicklung einer Softwarearchitektur, die es innerhalb einer heterogenen Landschaft von Entwicklungsprozessen ermöglicht, eine minimale Informationsbasis unabhängig von der expliziten Dokumentation zu generieren und prozessübergreifend nutzbar zu machen.

Im **dritten Arbeitsschwerpunkt** sollen die Untersuchungsergebnisse der ersten beiden Schwerpunkte in Form eines Such- und Empfehlungssystems für das XIL-Datenmanagement

umgesetzt werden. Ziel ist es Modelle bereits beim Einpflegen ins System zu untersuchen und mittels Attributen und Kennzahlen zu dokumentieren. Dazu soll jedem Modell ein Verwendungskontext zugeordnet werden, der Aussagen zu den bereits genannten Simulationskriterien zuordnet. Unter Nutzung dieser Kontextinformationen sollen anschließend bei der Aggregation der Modelle zu komplexeren Modellen, z.B. ein Gesamtfahrzeug oder Mock-up, entsprechend dem zweiten Arbeitsschwerpunkt Bewertungen der Modellaufbauten durchgeführt werden. Aufgrund der Kennzahlen sind unterschiedliche Modelle für den gleichen Sachverhalt vergleichbar und bewertbar. So können z.B. zwei Motormodelle mit unterschiedlicher Modellierungsdetaillierung in Form physikalischer Gleichungen oder physikalischer Annäherungen, verglichen werden. Dies erlaubt die Abschätzung von Unterschieden in der Simulierbarkeit des Gesamtmodells, Simulationslaufzeit und Ergebnisqualität und je nach Anwendungsfall Empfehlungen für die Anwendung des einen oder des anderen Simulationsmodells.

Die vorliegende Arbeit beschreibt eine neue Methodik zur Unterstützung der Funktionsentwicklung in der Automobilindustrie durch die weitestgehend automatische Erfassung von dokumentationsrelevanten Informationen für Simulationsmodelle aus dem X-in-the-Loop-Umfeld. Hauptziel ist es eine strukturierte Suche nach Entwicklungsartefakten zu ermöglichen und dabei den zielgerichteten Informationsfluss zu beschleunigen. Darüber hinaus wird eine Methodik zur standardisierten Aufbereitung von Modellinformationen vorgestellt, die das Modellverständnis fördert, um somit die Relevanz von Suchergebnissen für den individuellen Nutzer bewerten zu können. Im Rahmen der Arbeit werden eine Reihe von Nebenzielen bearbeitet

- die Sammlung von Anforderungs-, Modell- und Infrastrukturdaten, sowie die Aufbereitung dieser zu einem standardisierten Modellkontext
- die Definition und Berechnung von Kriterien zur Modellvergleichbarkeit
- die Schlagwort- und kontextbasierte Bewertung von Simulationsmodellen hinsichtlich strukturierter Suchanfragen

Die Korrektheit und Gültigkeit der getroffenen Aussagen in Bezug auf Attributierung und Kennwerte wird anhand eines prototypischen Informationsmanagementsystems durch Implementierung und Test in ausgewählten Szenarien nachgewiesen. Kennwerte können dabei sowohl konkrete Zahlenwerte, aber auch Abschätzungen oder Grenzwerte sein, die qualitative und quantative Aussagen über Simulationsmodelle erlauben. In der Arbeit wird der Stand der Technik der Bereiche Wissensmanagement in der Industrie, automatisierte Informationsgewinnung sowie der Qualitätsbewertung von Simulinkmodellen dargelegt. Darüber hinaus werden die Lösungswege zur automatisierten Gewinnung eines Modellkontext aufgezeigt und deren Gültigkeit gezeigt.

Abbildung 1.1 zeigt die Rahmenbedingungen der Arbeit auf. Es ist nicht das Ziel Veränderungen im Entwicklungsprozess zur Erfassung von Anforderung, Modellerstellung und Pflichtdokumentationen zu erarbeiten. Vielmehr dienen vorhandene Entwicklungsstrukturen als Grundlage zur Aufbereitung einer strukturierten Dokumentation. Auch ist es nicht das Ziel, die Ergebnisse einer Suche explizit für einen bestimmten Nutzungszweck zu bewerten oder diesem direkt zur Verfügung zu stellen. Vielmehr muss es einem potenziellen Nutzer auf Grundlage der Informationsbasis und deren Aufbereitung möglich sein, eine solche Bewertung durchzuführen.

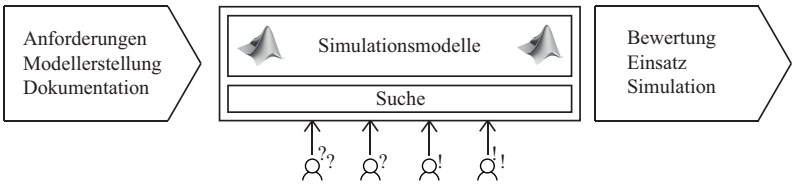


Abbildung 1.1: Rahmen der Arbeit

Unabhängig vom konkreten Anwendungsfall der automobilen Funktionsentwicklung soll das im Rahmen der Arbeit entwickelte Konzept als Referenz für ein anwendungsoptimiertes Wissensmanagementsystem für modellbasierte Entwicklungsprozesse dienen. Dabei steht insbesondere die Interaktion und das Wissen der Prozessbeteiligten im Vordergrund, um die Lernkurve und den Eingriff in etablierte Entwicklungsstrukturen zu minimieren und dennoch notwendige Informationsflüsse zu realisieren. Ziel ist es generische, wenig akzeptierte Lösungen abzulösen und die Qualität, sowie die Zukunftsfähigkeit der Prozesse durch zielgerichtete Unterstützung der Anwender sicherzustellen.

1.3 Aufbau der Arbeit

Kapitel 2 stellt zunächst die Motivation der vorliegenden Arbeit im Kontext der virtuellen Funktionsentwicklung der Automobilindustrie dar.

In Kapitel 3 werden die Grundlagen für das Verständnis der folgenden Kapitel der Arbeit beschrieben. Es folgt dabei einer Top-Down-Struktur. Zunächst werden die Methoden der modellbasierten Entwicklung allgemein und deren konkrete Ausprägung für die automobile Funktionsentwicklung im Besonderen dargestellt. Anschließend werden die X-in-the-Loop-Simulationen und damit verknüpften, relevanten Entwicklungsartefakte vorgestellt. Abschließend wird der Begriff des Modelldatenmanagement definiert und im beschriebenen Kontext eingebettet.

Kapitel 4 stellt den Stand der Technik in den an diese Arbeit angrenzenden Themengebieten Wissensmanagement, Modelldatenmanagement, Qualität von Simulationsmodelle sowie Suchen in strukturieren und unstrukturierten Daten vor.

In Kapitel 5 werden die Lösungskonzepte der drei Arbeitsschwerpunkte der vorliegenden Arbeit (vgl. Abschnitt 1.2) beschrieben. Abschnitt 5.1 bearbeitet dabei den ersten Schwerpunkt der Definition eines Modellkontextes. Die Abschnitte 5.2 und 5.3 stellen die Grundlagenarbeiten für die Lösungsansätze des zweiten Arbeitsschwerpunktes, der automatischen Erfassung des Modellkontextes, dar, welche in den Abschnitten 5.4 und 5.5 beschrieben wird. Der dritte Arbeitsschwerpunkt, der Nachweis der Korrektheit durch Umsetzung von Wiederverwendungskonzepten der Suche und Empfehlung, wird in den Abschnitten 5.6 und 5.7 bearbeitet. Das Kapitel schließt mit einer formalen Beschreibung der Informationsverarbeitungskonzepte in Abschnitt 5.8, welche über die Arbeitsschwerpunkte hinweg in Form eines Informationsmanagementsystem entwickelt wurden.

Das Kapitel 6 beschreibt die prototypische Umsetzung des im Rahmen der Arbeitsergebnisse entwickelten und in Kapitel 5 vorgestellten Informationsmanagementsystems. Es geht

dabei auf die gewählten Technologien sowie die konkrete Umsetzung ein. Anschließend geht Abschnitt 6.2 auf die geplante, produktive Nutzung der Arbeitsergebnisse im Rahmen des Projektes XIL-Datenmanagement ein.

Kapitel 7 dient der Validierung der Ergebnisse der vorliegenden Arbeit am praktischen Anwendungsfall. Es beschreibt die Übertragung der Konzepte auf den praktischen Anwendungsfall eines Modellentwicklungsprozesses der automobilen Funktionsabsicherung unter Verwendung der in Kapitel 6 vorgestellten prototypischen Umsetzung. In diesem Zusammenhang wird in Abschnitt 7.4 die Erreichung der in Abschnitt 1.2 definierten Ziele der Arbeit bewertet.

Abschließend fasst Kapitel 8 die Ergebnisse der vorliegenden Arbeit noch einmal zusammen und gibt einen Ausblick hinsichtlich der weiteren Nutzung sowie offener Themengebiete, die weiterführende Untersuchungen erfordern.

2 Motivation

In den letzten rund 10 Jahren haben sich Fahrzeugfunktionen zu einem Innovationstreiber der Automobilindustrie entwickelt, der heute als zentrales Werkzeug der Differenzierung von Fahrzeugherstellern dient [SZ13]. Ob im Bereich der Fahrerassistenz- oder der Infotainmentfunktionen - kein neues Fahrzeug kommt mehr ohne eine breite Palette von Funktionen aus. Getrieben wird diese Entwicklung vorwiegend durch steigende kundenspezifische Ansprüche nach mehr Mobilität und Komfort sowie durch gesetzliche Rahmenbedingungen, die in der Regel auf Reduzierung des Kraftstoffverbrauchs oder Erhöhung der Fahrzeugsicherheit zielen [Str12]. Daraus entstanden neben zulassungsrelevanten Fahrerassistenzfunktionen, wie ABS und ESP, hochkomplexe Funktionen, wie automatische Abstandsregelung (Adaptive Cruise Control - ACC) oder Start-Stop-Automatik, die bereits heute im Premiumsegment über die gesamte Fahrzeugpalette angeboten werden. Das in absehbarer Zukunft marktreife hochautomatisierte oder sogar autonome Fahren und die damit verbundene Vernetzung dieser Fahrzeugfunktionen wird die Komplexität des Gesamtsystems Fahrzeug erneut sprunghaft erhöhen.

Dieser Komplexität stehen immer kürzere Entwicklungszeiten bzw. Time-to-Market-Wünsche der Kunden gegenüber. Eine steigende Komplexität führt jedoch stets zu höheren Entwicklungskosten und längeren Entwicklungszeiten. Betrachtet man das magische Dreieck des Projektmanagements [pmb13], dann lässt sich folgern, dass ein Ausgleich dieser Faktoren zwangsläufig zu qualitativen Einbußen und weniger Kundenzufriedenheit führt. Um diesen Auswirkungen zu begegnen, werden neue Ansätze wie die virtuelle Entwicklung und die in dieser Arbeit präsentierten Konzepte benötigt.

Dieses Kapitel gibt zunächst einen Überblick über die Entwicklung von Fahrzeugfunktionen im Allgemeinen und geht anschließend auf die dafür immer wichtiger werdende virtuelle Entwicklung ein. Abschließend stellt das Kapitel die Herausforderungen dieser Form der Entwicklung dar, die den Ausgangspunkt für die vorliegende Arbeit bilden.

2.1 Funktionsentwicklung

Die Funktionsentwicklung befasst sich mit der Entwicklung eines breiten Spektrums an Fahrzeugfunktionen. Die Disziplin hat sich im Entwicklungsprozess erst in den letzten rund 25 Jahren neben der klassischen Eigenschaftsentwicklung etabliert. Während die Eigenschaftsentwicklung sich mit definierten Kennwerten eines Fahrzeugs, wie der Karosseriegeometrie, Gewicht und Verbrauch beschäftigt, dient die Funktionsentwicklung der Entwicklung von elektronischen bzw. mechatronischen Systemen, die definierte Aufgaben im Fahrzeug automatisieren.

Eine Fahrzeugfunktion kann als eine Regelstrecke bestehend aus Sollwertgeber, Regler, Aktuatoren, Regelstrecke und Sensoren (vgl. Abbildung 2.1) aufgefasst werden. Dabei wird durch äußere Einflüsse, z.B. dem Fahrer, ein Sollwert vorgegeben. Für ein ABS-System wäre

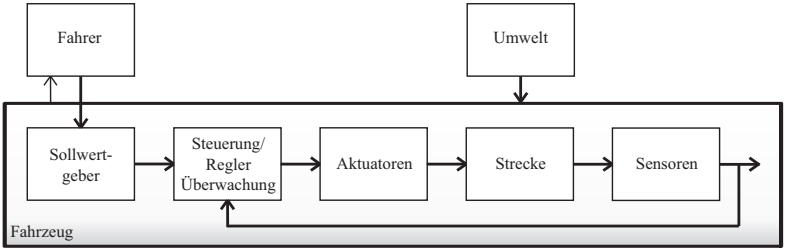


Abbildung 2.1: Logische Systemarchitektur von Regelungssystemen nach [SZ13]

dies beispielsweise die Anforderung einer bestimmten Bremsleistung durch Bremspedalbetätigung. Der Sollwertgeber dient als Eingangsgröße für den Regler. Dieser stellt heute in der Regel ein Steuergerät dar, auf dem die entsprechende Funktion ausgeführt wird. Im Falle des ABS wäre dies die Ansteuerung der Aktuatoren (Bremsen) die auf die Regelstrecke (Raddrehzahl) wirken und die Geschwindigkeit reduzieren. Durch Sensoren wird die Wirkung der Aktuatoren auf die Regelstrecke überprüft und die Ergebnisse wieder an den Regler zurückgeführt. Sollte es aufgrund äußerer Einflüsse, wie einer glatten Fahrbahn, zu einer Radblockade kommen und das Fahrzeug noch nicht stehen, wird die ABS-Fahrzeugfunktion im Steuergerät den Sollwert überschreiben und die Bremse lösen, bis wieder eine Raddrehzahl messbar ist. Somit kann durch den beschriebenen Kreislauf beispielsweise das ABS die Bremsleistung optimieren, indem die Aktuatoren auf Werte zwischen Null und der maximal angeforderten Bremsleistung geregelt werden, um maximale Verzögerung ohne blockierende Räder zu erhalten.

Da die genannten Einzelkomponenten, als Beispiel sei der Sensor für die Fahrzeuggeschwindigkeit angeführt, nicht für jede einzelne Funktion dediziert im Fahrzeug verbaut werden können - u.a. aus Gründen der Wirtschaftlichkeit und des Bauraumes - müssen diese als geteilte Ressourcen zur Verfügung stehen. Daraus resultieren, wie immer bei geteilten Ressourcen, Abhängigkeiten der Funktionen untereinander, wenn zwei Funktionen die gleiche Ressource zur gleichen Zeit benötigen.

Zu Beginn der Funktionsentwicklung waren Steuergeräte und die von ihnen realisierten Funktionen weitestgehend autonom und ohne gegenseitige Wechselwirkung. Dies erlaubte ein eindeutige Zuordnung zu den Entwicklungsdomänen: Antriebsstrang, Fahrwerk, Karosserie und Multimedia [SZ13]. Das beschriebene ABS-System zählte in seiner ursprünglichen Ausbaustufe zu diesen Funktionen und war eindeutig dem Fahrwerk zuzuordnen. Technologie- und Leistungssprünge in der Steuergerätehardware erlaubten die Realisierung immer leistungsfähigerer Funktionen allein in Software und damit die Ausführung mehrerer, unabhängiger Funktionen auf einem Steuergerät. Ab Ende der 90er Jahre wurde damit begonnen die elektronischen Systeme zu vernetzen und dadurch wesentlich komplexere Systeme zu realisieren [SZ13]. Durch die Vernetzung der Steuergerät mittels sogenannter Fahrzeugbusse war es möglich über Steuergerätegrenzen hinweg nachrichtenbasierte Kommunikation [KR08] zwischen Funktionen zu realisieren.

Heute sind in Oberklassefahrzeugen bereits bis zu einhundert Steuergeräte [Ric09] verbaut, die aufgrund der steigenden Leistungsfähigkeit der Hardware auch immer mehr Funktio-

nen je Steuergerät beherbergen. Dieses Wachstum und die zunehmende Vernetzung von Funktionen untereinander kann nur durch einen strukturierten Entwicklungsprozess beherrschbar umgesetzt werden. Etabliert hat sich hier das V-Modell [SZ13] als Vorgehensmodell für die Funktionsentwicklung. Das „V" repräsentiert dabei zwei klassische Vorgehensmodelle der Entwicklung. Während der linke Arm einen Top-Down-Ansatz zur Dekomposition der Problemstellung verfolgt und es dadurch in kleinere, beherrschbare Unteraufgaben zerlegt, repräsentiert der rechte Arm die Integration und den Test der Unteraufgaben zur Gesamtllösung über einen Bottom-Up-Ansatz. Abbildung 2.2 zeigt das genutzte V-Modell im Detail. Es ist zu erkennen, dass das V-Modell eine Systemebene und eine Softwareebene unterscheidet. Dies ist von zentraler Bedeutung, da es sich bei der Funktionsentwicklung um eine Hardware-Software-Codesign Aufgabe handelt, wobei es das optimale Verhältnis zwischen Realisierungen in Hardware und Software zu finden gilt [Har10].

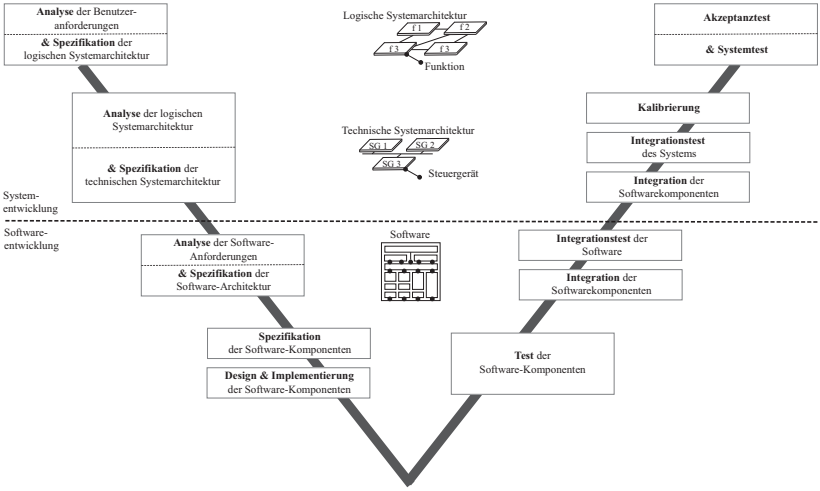


Abbildung 2.2: V-Modell der Funktionsentwicklung nach [SZ13]

Die Entwicklung einer neuen Funktion startet mit der Analyse der Anforderungen, die diese Funktion erfüllen soll. Ausgehend davon ergeben sich logische Ein- und Ausgaben, die äußeren Systemschnittstellen, welche in einer logischen Spezifikation der Systemarchitektur beschrieben werden. Darauf aufbauend wird im nächsten Schritt die technische Systemspezifikation abgeleitet, die das Zusammenspiel zwischen Hardwarekomponenten, wie Sensoren, Aktuatoren und der Steuergerätearchitektur beschreibt. Mit der technische Systemarchitektur sind die verfügbaren Ressourcen für die Funktionsrealisierung beschrieben, auf denen die zu entwickelnde Funktion umgesetzt wird. Hier setzt der Softwareentwicklungsprozess ein, der zunächst wieder mit einer Anforderungsanalyse auf Basis der Funktionsanforderungen und der gewählten Systemarchitektur beginnt. Das Ergebnis dieses Schritts ist eine Softwarearchitektur, die für alle Softwarekomponenten die Aufgaben und das Zusammenspiel auf Schnittstellenebene beschreibt. Die Softwarekomponenten werden anschließend im Einzelnen spezifiziert und den jeweils spezialisierten Entwicklungsabteilungen

übergeben, die die Entwicklung und Implementierung durchführen. In der Regel wird heute aufgrund der komplexen Funktionalitäten der Softwarekomponenten auch deren Entwicklung erneut in so genannte Module unterteilt. Abschnitt 3.1 wird diese Entwicklung aufgreifen und detaillierter betrachten. Mit dem Abschluss der Implementierung der Softwarekomponenten erfolgt der Aufstieg im rechten Arm des V-Modells. Zunächst werden die Softwarekomponenten einzeln getestet und anschließend zur Gesamtsoftware integriert. Nach der Durchführung von Integrationstest wird die Integration von Systemkomponenten und Systemsoftware durchgeführt und wiederum getestet. Nach erfolgreichem Abschluss der Systemintegration erfolgt die Kalibrierung der Funktionseigenschaften auf fahrzeugabhängige Charakteristika. Anschließend wird die Funktion in das Fahrzeug integriert und im Gesamtsystem auf Akzeptanz - insbesondere auch aus Kundensicht - getestet. Auf die für die vorliegende Arbeit zentralen Abschnitte des Absicherungsprozess wird in Abschnitt 3.2 detaillierter eingegangen. Für weiterführende Informationen zum V-Modell sei an dieser Stelle auf [SZ13] verwiesen.

Das beschriebene Vorgehen in der Funktionsentwicklung erlaubt eine baumartige, feingranulare Zerlegung der Aufgaben in kleine, beherrschbare Unteraufgaben und bedient sich des Prinzips „Teile und Herrsche" [Sed02], um die Komplexität heutiger Funktionen abbilden zu können. Sie führt zur Beteiligung vieler, hochspezialisierter Fachabteilungen und Fachspezialisten an der Entwicklung einer einzelnen Funktion und damit letztendlich zu qualitativ hochwertigen Funktionen. Der Beherrschbarkeit der Entwicklung stehen allerdings auch größere Aufwände bei der Integration zur finalen Funktion gegenüber. Je mehr Ebenen bei der Dekomposition eingeführt werden, desto mehr kritische Schnittstellen ergeben sich für den Integrationsprozess.

In der klassischen Fahrzeugentwicklung wurden Funktionen durch Versuche im realen Fahrzeug erprobt. Eine vollständige Testspezifikation sorgte dafür, dass alle Zustände der Funktion abgetestet wurden. Bei Betrachtung der Funktion als Programmcode eines Steuergerätes sind dies alle möglichen Eingangswerte und Programmpfade. Die zuvor beschriebene Komplexität heutiger Fahrzeugfunktionen erlaubt es hingegen nicht mehr, alle möglichen Zustände, insbesondere auch der Abhängigkeiten von Funktionen, im Rahmen einer vollständige Testspezifikation nachzubilden. Wie in der klassischen Software- und Hardwareentwicklung der Informatik ist daher auch hier eine Absicherungsmethodik entstanden, die insbesondere kritische Pfade und Grenzfälle der Funktionen absichert. Dabei kann nicht länger die vollständige Korrektheit der Funktion bzw. die Abwesenheit von Fehlern gezeigt werden, sondern lediglich durch sorgfältige Planungsmethoden die Anwesenheit bestimmter Fehler ausgeschlossen werden.

Eine weitere Herausforderung bei der Absicherung besteht in der Komplexität der notwendigen Testszenarien. Während die Absicherung einer ABS-Funktion im realen Fahrversuch durch die Durchführung von Bremsungen realisierbar und hier die Reproduzierbarkeit gegeben ist, stellen moderne Fahrerassistenzfunktionen größere Anforderungen an die Absicherungsmethoden der Fahrzeughersteller. Ein Beispiel ist eine Notbremsfunktion, die bei einem Hindernis auf der Fahrbahn eine Gefahrenbremsung ausführt. Hier würde ein Realversuch die Komplexität der Absicherung stark erhöhen. Welche Person würde sich freiwillig vor ein Fahrzeug stellen, um eine Notbremsfunktion zu testen? Unter welchen äußeren Bedingungen ist die Hinderniserkennung möglich? Regen, Schnee, Nebel sind schwer abbildbar und konkrete Situationen nicht reproduzierbar. Darüber hinaus führt jeder Fehlversuch zu Beschädigungen des Testfahrzeugs.

Um die beschriebenen Einflussfaktoren Anzahl, Vernetzung, hohe und zunehmende Anforderungen an die Zuverlässigkeit, Verfügbarkeit und Sicherheit von Funktionen bei der parallel wachsenden Anzahl von Fahrzeugvarianten beherrschen zu können, gewinnt die virtuelle Entwicklung zunehmend an Bedeutung.

2.2 Virtuelle Entwicklung

Die virtuelle Entwicklung bezeichnet die Unterstützung von Entwicklungsprozessen durch den Einsatz rechnergestützter Entwicklungsmethoden. Laut Bullinger [Bul02] besteht sie aus Datenerzeugung, Datenmanagement, System-Integration, Virtual-Engineering-Organisation und Anwenderzugriff, die sich im Idealfall in einer zentralen Kooperationsplattform bündeln. Im Zentrum stehen so genannte virtuelle Prototypen, die zunächst als Computermodell existieren und mit dem Ziel der Kostenvermeidung physikalischer Prototypen entwickelt werden. Die Idee der virtuellen Entwicklung ist eng damit verknüpft Fehlerfolgskosten möglichst gering zu halten. Diese werden umso höher, je „realer" ein Produkt bereits ist [Bul02].

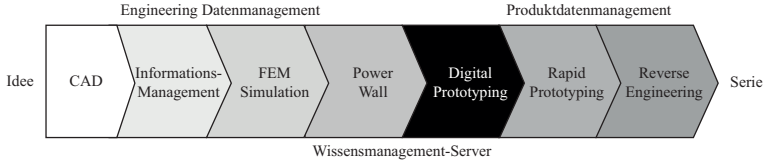


Abbildung 2.3: Rapid Product Development Process nach [Bul02]

Abbildung 2.3 zeigt den Rapid Product Development Process der virtuellen Entwicklung, wie ihn Bullinger [Bul02] beschreibt. Der Prozess beginnt in der virtuellen, geometrischen Gestaltung des Produktes. Während der Informationsmanagementphase werden die zukünftigen Eigenschaften des Produktes definiert und festgeschrieben. Darauf aufbauend erfolgt die Konstruktion und Absicherung dieser Eigenschaften mit Hilfe der FEM-Simulation. Eine sogenannte Power Wall dient im beschriebenen Prozess der kollaborativen Zusammenarbeit am Produkt mit dem Ziel der realitätsnahen, virtuellen Darstellung des Produktes. Es folgt die Phase des digitalen Prototypen, der neben Geometrie und Eigenschaften auch die Entwicklung und Absicherung Funktionen (vgl. Abschnitt 2.1) in geeigneten Simulationsformen beinhaltet. Nach dieser Phase ist das Produkt hinsichtlich seiner Gestaltung vollständig definiert, entwickelt und abgesichert. Anschließend folgt das Rapid Prototyping in dem das Produkt erstmalig vollständig oder teilweise mit realen (nicht virtuellen) Komponenten erstellt und abgesichert wird. Abschluss bildet die Phase des Reverse Engineering, wobei das finale Produkt gegenüber der initialen Spezifikation überprüft wird. Durch das beschriebene Vorgehen wird es möglich, die rein virtuelle Entwicklung bis kurz vor der Serienreife im Entwicklungsprozess zu verankern. Wichtige Unterstützungsprozesse stellen dabei das Daten- und das Wissensmanagement dar.

Neben den reinen Kosten der Entwicklung steht auch die Realisierbarkeit von Entwicklungsaufgaben im Fokus, da die klassische Entwicklung, wie in Abschnitt 2.1 am Beispiel der Reproduzierbarkeit von Absicherungsaufgaben beschrieben, diese vor dem Hintergrund

wachsender Komplexität nicht mehr sicherstellen kann. Bereits heute erfordert die Absicherung in der Fahrzeugentwicklung mehrere tausend Entwicklungsfahrzeuge, die einzig der Durchführung von Realversuchen dienen. Eine wachsende Modellpalette führt zunehmend zur Forderung der Einsparung solcher Entwicklungsfahrzeuge und deren Ablösung durch Methoden der virtuellen Entwicklung. Bei der AUDI AG wuchs die Zahl beispielsweise im Jahr 2014 auf über 60 Modelle.

Die virtuelle Entwicklung hat sich in verschiedenen Bereichen der Fahrzeugentwicklung bereits seit vielen Jahren etabliert. Ob in der Geometrieentwicklung in Form von Computer Aided Design (CAD), in der Eigenschaftsentwicklung als Computer Aided Engineering (CAE) oder im Fahrzeugtest als Computer Aided Testing (CAT) - in allen Disziplinen werden virtuelle Entwicklungsmethoden eingesetzt. Die oben beschriebenen Schichten der virtuellen Entwicklung sind hierbei jeweils mit unterstützenden Applikationen annähernd vollständig versorgt.

Dem gegenüber steht die Funktionsentwicklung als relativ junge Disziplin, die sich aufgrund ihrer heterogenen Aufgabenstellungen in den Entwicklungsdomänen unabhängig und damit sehr unterschiedlich entwickelt hat. Erst in den letzten Jahren, mit der zunehmenden Vernetzung der Funktionen und damit der Abhängigkeit der Domänen untereinander, wurde der Handlungsbedarf für die Schaffung einer durchgängigen, rechnergestützten Funktionsentwicklung deutlich. Das in Abschnitt 1.1 erläuterte Projekt XIL-Datenmanagement der AUDI AG befasst sich hierbei zentral mit der Schicht des Datenmanagements für die Funktionsabsicherung und den zugehörigen Prozessen der System-Integration und Organisation. Ziel ist die Entwicklung einer zentralen Datenablage und -austauschplattform [GT12]. Im Rahmen des Projektes wurde erkannt, dass die Ablage von Entwicklungsartefakten allein nicht ausreichend ist, um einen domänenübergreifenden Prozess zu etablieren, sondern auch die weiteren Ebenen der virtuellen Entwicklung betrachtet werden müssen.

Die vorliegende Arbeit greift diesen Bedarf für die modellbasierte Entwicklung auf und sucht Konzepte und Lösungswege, um eine durchgehende Datenversorgung über Domänen- und Fachbereichsgrenzen hinweg zu ermöglichen. Dabei wird ein Konzept entwickelt und umgesetzt, das die Systemintegrations- und Organisationsschicht der virtuellen Entwicklung für die Funktionsabsicherung durch Nutzung von heute verfügbaren Methoden der Informationsgewinnung und -aufbereitung unterstützt. Hintergrund ist es, durch die Verbesserung der Zusammenarbeit aller Prozessbeteiligten, die virtuelle Entwicklung mit Hilfe modellbasierter Entwicklungsprozesse zu stärken und die Leistbarkeit aktueller und zukünftiger Entwicklungsaufgaben sicherzustellen.

3 Grundlagen

Das vorliegende Kapitel führt Konzepte und Begriffe ein, die für das Verständnis des in dieser Arbeit vorgestellten Ansatzes und notwendig sind. Um die Zusammenhänge nachvollziehbar zu gestalten, folgt das Kapitel einer Top-Down-Strukturierung. Zunächst werden in Abschnitt 3.1 die wichtigsten Informationen zur modellbasierten Entwicklung und Absicherung als Vorgehensmodelle eines Entwicklungsprozesses erläutert. Anschließend wird die automobile Funktionsabsicherung in Abschnitt 3.2 als eine Anwendung der modellbasierten Absicherung vorgestellt. Für die Durchführung der modellbasierten Absicherung wird dabei die X-in-the-Loop-Simulation als eine zentrale Methodik identifiziert und im Anschluss in Abschnitt 3.3 im Detail erläutert. Abschnitt 3.4 führt daraufhin die wichtigsten Artefakte der modellbasierten Entwicklung ein und setzt diese strukturell, wie semantisch in Beziehung. Abschließend geht Abschnitt 3.5 auf die Datenmanagementaspekte ein, die die modellbasierte Entwicklung und ihre zugehörigen Entwicklungsartefakte bedingen.

3.1 Modellbasierte Entwicklung und Absicherung

Seit Jahrhunderten besteht die Aufgabe wissenschaftlicher Tätigkeit darin, Zusammenhänge des Universums in Modellen, auch als Theorien bezeichnet, zu fassen und somit für den Menschen nachvollziehbar zu repräsentieren. Die natürlichen Denkmuster sind stets von den individuellen Modellen eines jeden Menschen von sich selbst und seiner Umwelt geprägt. Hawking bezeichnet diese Tatsache als modellabhängigen Realismus [HM10]. Im Ergebnis stellen wissenschaftliche Erkenntnisse, die in allgemeingültige Modelle gefasst werden, eine modellbasierte Repräsentation in einer formalen Form dar, die diese unabhängig vom individuellen Modell des Menschen repräsentieren.

Ein Modell stellt nach Stachowiak [Sta73] ein beschränktes Bild der Wirklichkeit, mit mindestens folgenden drei Eigenschaften dar:

- **Abbildung:** Ein Modell ist stets eine Abbildung oder Repräsentation eines natürlichen oder eines künstlichen Originals, das selbst wieder Modell sein kann.
- **Verkürzung:** Ein Modell erfasst im Allgemeinen nicht alle Attribute des Originals, sondern nur diejenigen, die dem Modellschaffer bzw. Modellnutzer relevant erscheinen.
- **Pragmatismus:** Zwischen Modellen und ihren Originale existiert keine eineindeutig Zuordnung. In der Regel lässt sich ein Modell dem Original eindeutig zuordnen, aber nicht umgekehrt. Das Modell erfüllt eine Ersetzungsfunktion für bestimmte Einsatzszenarien, innerhalb bestimmter Zeitintervalle und unter Einschränkung auf definierte Operationen.

Die Informatik macht sich diese Erkenntnisse über menschliche Denkmuster bereits seit vielen Jahren in Form von modellbasierter Softwareentwicklung zu nutze. Eine weit verbreitete Repräsentation von Modellen für die Spezifikation, Konstruktion und Dokumentation

von Software ist die Unified Markup Language (UML), welche von der Object Management Group (OMG) entwickelt und in der ISO/IEC 19505 [Int12] standardisiert wurde. Die Standardisierung führte zu einer großen Entwicklergemeinde, die ein hohes Maß an Toolunterstützung mit sich brachte. Dabei können Zusammenhänge einer zu entwickelnden Software nicht nur modelliert werden, sondern auf Basis formaler Modelle auch direkt Quellcode generiert und diese damit in ausführbare Applikationen übersetzt werden.

Mit der Durchsetzung in der Softwareentwicklung wurde die modellbasierte Entwicklung für einen größeren Personenkreis sichtbar und für neue Anwendungsfelder nutzbar gemacht. Neue Modellierungssprachen für domänenspezifische Problemstellungen entstanden und werden unter dem Begriff der domänenspezifischen Sprachen (Domain Specific Language, DSL) [Sta07] zusammengefasst. Im Bereich der Modellierung von elektronsichen und mechatronischen Systemen ist die Systems Modelling Language (SysML) [Obj14] ein Vertreter dieser DSLs. Sie wurde als Ableitung von UML ebenfalls von der OMG entwickelt und dient der Spezifikation von elektronischen Systemen auf verschiedenen Ebenen - von Schnittstellen bis hin zur Systempartitionierung.

In der Automobilindustrie, die wie in Abschnitt 2.1 beschrieben, von einer sehr heterogene Entwicklungsgeschichte im Bereich der Funktionsentwicklung gekennzeichnet ist, haben die Einflüsse der modellbasierten Entwicklung eine ebenso uneinheitliche Prozesslandschaft geschaffen. Von Hersteller zu Hersteller und auch innerhalb der Konzerne existieren individuelle Ansätze, die alle in sich funktionieren und an dieser Stelle nicht näher beleuchtet werden sollen. Ein zentrales Werkzeug der modellbasierten Entwicklung ist jedoch industrieeinheitlich stark verbreitet - Simulink®.

Die Simulink®-Toolbox ist ein Teil der proprietären Toolsammlung MATLAB® (MATrix LABoratory) [Ang14], die 1984 von der Firma The Mathworks herausgebracht wurde. Ursprünglich als Hilfsmittel für Mathematikstudenten für numerische Berechnungen entwickelt, verbreitete es sich schnell als Hilfsmittel für diese Berechnungen in vielen verschiedenen Industrien - von der Medizin über Luft- und Raumfahrt bis in die Automobilindustrie.

Um sich wiederholende Berechnungen zu vereinfachen, entwickelten sich so genannte Toolboxen, die spezifische Berechnungsaufgaben in MATLAB® mit Hilfe grafischer Oberflächen abstrahierten und damit auch ohne tiefere Programmierkenntnisse für Ingenieure zugänglich machten. Die Simulink®-Toolbox ist darauf spezialisiert Systeme unterschiedlicher Art (elektronisch, physikalisch, medizinisch etc.) zu modellieren. Möglich wird dies über die hierarchische Modellierung durch Verknüpfung so genannter Blöcke. Ein Block kann dabei jeweils als ein Funktionstriple aus Eingabewerten, Verarbeitungsregel und Ausgabewerten betrachtet werden. Als Verarbeitungsregel können sowohl kontinuierliche, als auch diskrete Operationen abgebildet werden. Verbunden werden Blöcke in Form von gerichteten Graphen [Sed02] jeweils von einem Ausgang zu einem Eingang. Durch so genannte Blocksets werden Simulink® bereits mit der Auslieferung häufig benötigte Funktionsblöcke mitgeliefert. Darunter sind neben logischen Operatoren auch Blocksets für elektronische und Regelungssysteme. Von zentraler Bedeutung ist in der Funktionsentwicklung hier die Simulink Control Design Toolbox®.

Der Funktionsumfang für die modellbasierte Entwicklung mittels Simulink® reicht dabei von der Modellierung, über zeitdiskrete und kontinuierliche Simulation von Systemen, Funktionen für Test und Validierung bis hin zur Codegenerierung. Je nach beschriebenem System werden Codegenerierungen bspw. für C-Code oder VHDL unterstützt. Mit der Etablierung

von Simulink® im Funktionsentwicklungsprozess begannen auch andere Toolanbieter ihre Werkzeuge kompatibel zu Simulinkmodellen zu gestalten. So existieren für bestimmte Entwicklungsschritte heute z.B. spezielle Code-Generatoren und Simulationswerkzeuge.

Ausgehend von der Fähigkeit der Toolbox Simulink® Systeme auf unterschiedlichsten Ebenen modellieren zu können, eröffneten sich breite Einsatzfelder für die Funktionsentwicklung. Die Entwicklung einer Fahrzeugfunktion kann dabei allein durch Modellierungen in Simulink® wie folgt realisiert werden: Zunächst geht die Idee einer Funktion in der Regel auf eine Anforderung zurück, die das Verhalten der zukünftigen Fahrzeugfunktion beschreibt. Auf abstrakter Ebene kann man sich eine erste Verhaltensbeschreibung vorstellen, die auf Basis logischer Operationen die Funktion beschreibt. Durch Beaufschlagung von Eingabegrößen und die Beobachtung der Ausgangsgrößen, können mit den Simulationsmethoden von Simulink® erste Erkenntnisse zur Korrektheit der Modellierung gewonnen werden. Aufbauend auf diesem Modell muss in weiteren Verfeinerungsschritten eine Trennung zwischen physikalischen und elektronischen Schnittstellen erfolgen und das System von einer Verhaltensbeschreibung in eine Strukturbeschreibung überführt werden, die elektronische Komponenten (Hardware) und algorithmische Komponenten (Software) unterscheidet. Durch Codegenerierung können anschließend sowohl Hardwarebeschreibungen, als auch C-Code für die Software generiert werden. Für die Tests und Absicherung der entstandenen Funktion können mittels Simulink® wiederum die äußeren Einflüsse als Eingabegrößen modelliert und die Korrektheit durch Simulation gezeigt werden.

Eine virtuelle Entwicklung für Funktionen, wie sie im obigen Gedankenspiel durchlaufen wurde, ist im heutigen Entwicklungsprozess (vgl. Abschnitt 2.1) nicht umsetzbar. Die Vorentwicklung, Serienentwicklung und Absicherung von Funktionen findet unter Beteiligung von bis zu mehreren hundert Personen bei den OEMs und Zulieferern statt, die in ihren jeweiligen Toolketten unabhängig voneinander ihre Arbeit durchführen. Um näher an das Ideal einer durchgängigen Funktionsentwicklung zu gelangen, bedarf es eines zielgerichteten Daten- und Informationsflusses über das gesamte V-Modell. Für

3.2 Funktionsabsicherung

Die Absicherung von Funktionen auf den einzelnen Ebenen der Integrationsphase im rechten Arm des V-Modells, ist eine zentrale Aufgabe des Entwicklungsprozesses. Aus Sicht der Hersteller sollte sie möglichst kostengünstig, schnell, sicher (für Tester und Testobjekt), einfach und zuverlässig sein [Thi12]. Dabei werden Kerneigenschaften der elektronischen Komponenten, wie Temperaturfestigkeit, mechanische Belastbarkeit und elektromagnetische Verträglichkeit (EMV) getestet, die zunächst die Funktionsfähigkeit innerhalb der Fahrzeugumgebung sicherstellen. Hierbei haben sich Standardmethoden unter Nutzung von Dauerlauf- und EMV-Prüfständen sowie Erprobungen in realen Fahrzeug etabliert [Thi12].

Neben der reinen Absicherung der Hardwarekomponenten gegenüber äußeren Einflüssen gewinnt die Absicherung der eigentlichen Funktion insbesondere vor der Hintergrund der gestiegenen Anzahl und Komplexität an Bedeutung. Dabei müssen kritische Pfade und Rahmenbedingungen für die Funktionssoftware identifiziert und überprüft werden. Ein reiner Fahrversuch, der die identifizierten Szenarien abtestet, reicht dafür heute in der Regel nicht mehr aus. Aufgrund der Vielzahl und breiten Verfügbarkeit an Sensoren, die heute in die

Entscheidungswege der Funktionen einbezogen werden, ist es schwer Versuchsabläufe eindeutig zu rekonstruieren und damit wiederholbar zu gestalten.

Diese Komplexität in Kombination mit der Verkürzung von Entwicklungszeiten, den neuen Sicherheitsanforderungen an Fahrzeugfunktionen, wie der ISO-26262 [Int11b] und der zunehmenden Derivatisierung der Fahrzeugpalette führt zunehmend zum Einsatz virtueller Absicherungsmethoden. Im VW-Konzern hat sich die X-in-the-Loop-Simulation als Ergänzung zu bisherigen Verfahren etabliert [Thi12].

3.3 X-in-the-Loop-Simulation

Die X-in-the-Loop-Simulation ist eine Sammelbezeichnung, die Test- und Simulationswerkzeuge zur funktionalen Absicherung von Fahrzeugfunktionen zusammenfasst. Allen gemein ist, dass die zu testende Funktion, die so genannte Unit Under Test (UUT), in einen simulierten Kreislauf eingebettet wird, der gesamte Regelkreis (vgl. Abbildung 2.1) nachbildet. Man unterscheidet die Open- und die Closed-Loop-Simulation. Bei der Open-Loop-Simulation wird die Regelstrecke mit Testvektoren für die Eingänge angesteuert und das Ergebnis an den Ausgängen beobachtet und ausgewertet. Szenarien ergeben sich dabei durch die Aneinanderreihung von Testvektoren, ohne eine vollständige Rückkopplung der Sensoren über das Umfeld zu den Eingängen der UUT. Dem gegenüber steht die Closed-Loop-Simulation, die eine Simulation mit geschlossenem Regelkreis beinhaltet und die Wechselwirkungen zwischen Fahrzeug und Umfeld simuliert. Damit können beispielsweise reale Fahrszenarien, wie eine Autobahnfahrt, als Testszenario wiederholt unter immer gleichen Randbedingungen gefahren und bewertet werden.

Das „X" in X-in-the-Loop dient als Platzhalter für die jeweilige UUT. Man unterscheidet hier Model-, Software-, Processor-, Hardware- und Vehicle-in-the-Loop (vgl. Abschnitt 1.1). Im Folgenden werden die Simulationsmethoden Model-, Software- und Hardware-in-the-Loop näher betrachtet, da sie für das Umfeld der vorliegenden Arbeit von Bedeutung sind.

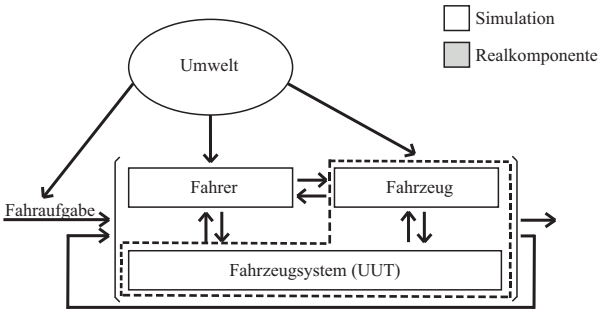


Abbildung 3.1: Aufbau eines MIL-Prüfstandes nach [Str12]

Model-in-the-Loop-Simulation Der funktionale Aufbau einer Model-in-the-Loop-Simulation (MIL) ist in Abbildung 3.1 dargestellt. Das Fahrzeugsystem, das in diesem Falle die

UUT darstellt, ist das Modell der Fahrzeugfunktion, dass wie zuvor beschrieben, in der Regel ein Simulinkmodell ist. Die Simulation findet direkt in der gewählten Simulationsumgebung, beispielsweise in Simulink, oder Co-Simulationsumgebung [BSMG12] statt, weshalb alle übrigen Komponenten der Simulation, der Fahrer, das Fahrzeug und das Umfeld ebenfalls als Simulationsmodelle sein müssen. Dies führt je nach Komplexität der angestrebten Testszenarien zu großem Aufwand für deren Modellierung. Ausführungssystem ist in der Regel ein Entwicklungs- oder Simulationsrechner. Simulations- und Analysewerkzeuge, die in der Simulationsumgebung integriert aber auch über Schnittstellen integrierte Tools sein können, erlauben die detaillierte Steuerung und Beobachtung der Simulation, sowie die Beeinflussung der Simulationsgeschwindigkeit. Dies ermöglicht einerseits die Szenarien langsam abzuarbeiten und dabei Funktionen der UUT zu beobachten. Andererseits können je nach Rechnerperformanz und Simulationskomplexität auch Berechnungen schneller als Echtzeit durchführt werden und damit über Testautomatisierungen ein breiteres Feld an Testszenarien durchgeführt werden. MIL wird heute vorwiegend in der Vorserien- und der frühen Serienentwicklung eingesetzt, um verschiedene Realiserungskonzepte für Funktionen auf einfache und kostengünstige Weise zu erproben.

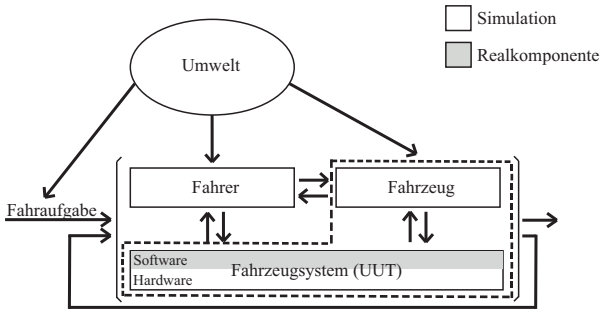


Abbildung 3.2: Aufbau eines SIL-Prüfstandes nach [Str12]

Software-in-the-Loop-Simulation Der funktionale Aufbau einer Software-in-the-Loop-Simulation (SIL) ist in Abbildung 3.2 dargestellt. Die UUT ist der Software-Code der Fahrzeugfunktion, der später auf dem realen Steuergerät im Fahrzeug ausgeführt werden soll. Primär wird die UUT in diesem Fall aus einem Modell generiert und der erzeugte Programmcode anschließend in die SIL-Simulation integriert. Die Ausführung des Codes erfolgt, wie auch die Model-in-the-Loop-Simulation auf Entwicklungsrechnern unter Nutzung spezieller Entwicklungsumgebungen, wie Simulink® oder auch ADTF [Voi08]. Ebenfalls analog zu MIL müssen die übrigen Simulationskomponenten als Modell vorliegen. Der Vorteil der SIL-Simulation liegt vor allem in der einfacheren Anbindung von Schnittstellen an für den Serieneinsatz entwickelten Software-Code. Die genutzten Entwicklungsumgebungen erlauben es, die Testausführung ebenso wie bei MIL zwischen langsamem Abarbeiten von Befehlen, als eine Form des Debugging bis hin zur testautomatisierten Abarbeitung zu variieren. Da für den SIL Test keine reale Hardware, insbesondere Steuergeräte, notwendig ist, ist es möglich bereits früh im Serienentwicklungsprozess Absicherungen durchzuführen und dadurch Fehler zu erkennen. Dabei eignet sich SIL insbesondere für die Erken-

nung arithmetischer Probleme [FS07] sowie die Realisierung von Code-Abdeckungstests zur Identifizierung noch nicht durchlaufener Programmteile [LB05].

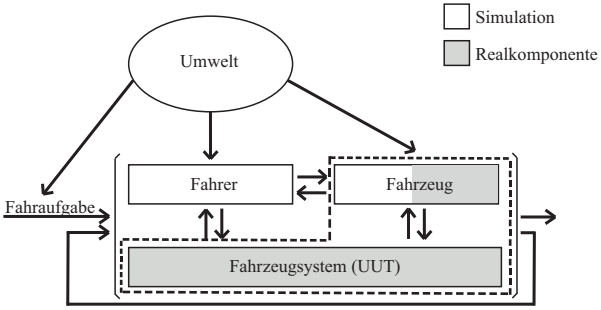


Abbildung 3.3: Aufbau eines HIL-Prüfstandes nach [Str12]

Hardware-in-the-Loop-Simulation Der funktionale Aufbau einer Hardware-in-the-Loop-Simulation (HIL) ist in Abbildung 3.3 dargestellt. Eine HIL-Simulation ist charakterisiert durch die Integration von Hardwarekomponenten in die Simulationsumgebung. Diese Hardwarekomponenten können Steuergeräte, aber auch Bauteile, wie Drosselklappen oder Bremskreisläufe sein. Die UUT kann hierbei als simuliertes Steuergerät, auch SoftECU bezeichnet, oder als reales Steuergerät vorliegen. Der Entwicklungsstand enthält in dieser Phase alle Schnittstellen, die auch in der Serie im Fahrzeug verbaut werden. Die Herausforderung dieser Simulation besteht insbesondere darin Interaktionen zwischen realen und simulierten Komponenten zu ermöglichen. Dazu werden je nach Testszenario am HIL verschiedene weitere Echtteile verschaltet und alle fehlenden Komponenten, die mit der UUT kommunizieren, simuliert. Dabei besteht die Komplexität dieser Simulationsform insbesondere in der realistischen Ansteuerung des Steuergerätes durch exakte Replikation der im Fahrzeug vorhandenen Schnittstellen. Neben der reinen Softwarefunktion, die auf dem Steuergerät realisiert ist, durchläuft die UUT bei der Anbindung an Fahrzeugbusse Diagnoseroutinen mit den für die Funktionsausführung notwendigen Ressourcen am Bus. Diese Kommunikationsprotokolle müssen vom HIL-Prüfstand vollständig, d.h. elektrisch ebenso, wie signaltechnisch plausibel [Har01] nachgebildet werden, da das Steuergerät ansonsten in einen nicht testbaren Sicherheitsmodus (Fail-Safe-Mode) wechselt. Die Realisierung von HIL-Prüfständen ist auf die Nutzung spezieller Rechensysteme angewiesen. Diese Rechnersysteme bestehen aus Spezialhardware, die insbesondere auf die Echtzeitausführung von Simulationen ausgerichtet ist, um die Kommunikation zwischen simulierten Komponenten und realer Hardware in harter Echtzeit [WB05] zu ermöglichen. Sie stellen darüber hinaus spezielle Anschlussmöglichkeiten für Echtteile zur Verfügung. Neben CAN- und anderen Busschnittstellen [LH02], stehen beispielsweise Leistungsschalter zur Verfügung, die es erlauben elektrische Signale in Strom und Spannung zwischen Simulation und Echtteil auszutauschen. Die signaltechnische Anbindung von realer Fahrzeughardware und die Modellierung des gesamten Restfahrzeugs zur Simulation auf den HIL-Rechnern, auch als Restbussimulation bezeichnet, führen zu einer sehr hohen Komplexität und Fehleranfälligkeit von HIL-Tests. Dem gegenüber steht der Nutzen durch die Realisierung von Komponenten-,

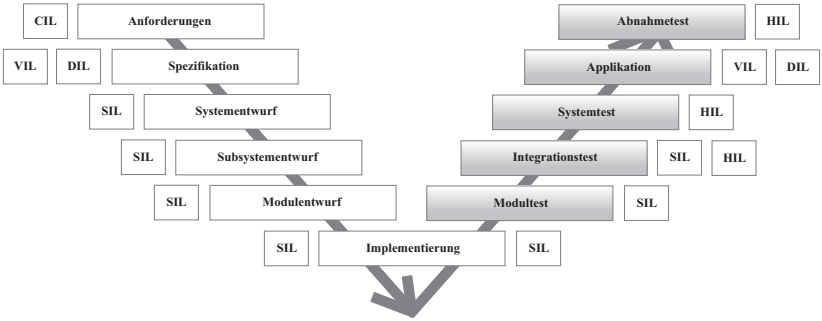


Abbildung 3.4: Einordnung der XIL-Simulation in den Entwicklungsprozess nach [vNC14]

Modul- und Steuergeräteverbundtest [EB07] sowie Tests des Diagnoseverhaltens, des Verhaltens im Fehlerfall und des Netzwerk- und Kommunikationsverhaltens [KH06]. Darüber hinaus ist es möglich Testszenarien an realer Hardware unter stets gleichen Bedingungen zu reproduzieren und somit einen Vorteil gegenüber der klassischen Erprobung im Fahrzeug zu generieren. Dieser besteht einerseits in der besseren Vergleichbarkeit einzelner Testergebnisse und andererseits in der Einsparung von Versuchsfahrzeugen.

Abbildung 3.4 verdeutlicht, wie sich die vorgestellten Simulationsmethoden heute in der Regel in das V-Modell der Funktionentwicklung einordnen[vNC14]. In der frühen Phase der Spezifikation und des Systementwurfs dient die Model-in-the-Loop-Simulation dem formalen Konzeptentwurf. Während der Entwicklung der Funktionssoftware vom Systementwurf zur Implementierung wird die Software mithilfe der Software-in-the-Loop-Simulation abgesichert. Anschließend wird, sobald die konkrete Ausführungsplattform, in der Regel Steuergeräte, verfügbar sind, die darauf integrierte Funktion mithilfe der Hardware-in-the-Loop-Simulation abgesichert. Für weiterführende sei auf [Str12] und [vNC14] verwiesen.

3.4 Artefakte der modellbasierten Entwicklung von Fahrzeugfunktionen

In Abschnitt 3.1 wurde die modellbasierte Entwicklung vorgestellt. Für das Verständnis der vorliegenden Arbeit ist es von zentraler Bedeutung, die für diese Form der Entwicklung relevanten Artefakte bzw. Datenbausteine im Detail zu verstehen. Der Fokus liegt dabei auf der Funktionsentwicklung und -absicherung unter Nutzung einer Entwicklungstoolkette, die sich auf die Simulationsumgebung von Simulink® stützt.

Modell Das zentrale Artefakt der modellbasierten Entwicklung ist das *Modell* - auch als Simulationsmodell oder im Speziellen Simulinkmodell bezeichnet. Es definiert die Bearbeitungsvorschriften, wie gegebene Eingangsdaten in Ausgangsdaten transformiert werden. Ein Simulinkmodell kann dabei auf Basis von so genannten Subsystemen hierarchisch

strukturiert sein. Je nach Art der Modellierung ist dabei der Modellbegriff auch rekursiv anwendbar. Angenommen, es gebe eine Modelldatei, die auf oberster Ebene nur Subsysteme enthält, ohne dass deren Schnittstellen miteinander verbunden sind. Dann stellt jedes dieser Subsysteme ein in sich geschlossenes Modell dar. Die Modelldatei selbst wird in diesem Fall als Modellbibliothek bezeichnet.

Signale Über definierte Schnittstellen werden zwischen Modellen oder zwischen Modell und Umgebung Daten ausgetauscht. Dieser Datenaustausch findet nachrichtenbasiert [KR08] statt. Die dabei ausgetauschten Nachrichten werden als *Signale* bezeichnet. Dieser Begriff stammt aus der Elektronik und verdeutlicht den Austausch von Daten über physische Medien. Diese Signale können verschiedene Qualitäten annehmen. Für den Automobilbau wird hier in der Regel zwischen physikalischen und elektrischen Signalen unterschieden. Elektrische Signale sind dabei meist Nachrichten, die im realen Fahrzeug über Fahrzeugbusse, wie CAN [LH02] oder FlexRay [LH02] zwischen Steuergeräten ausgetauscht würden. Physikalische Signale können beispielsweise Kräfte, wie Motordrehmoment oder Beschleunigungen, sein, aber auch thermische Größen, wie z.B. Reifen- oder Motortemperaturen. Allen Signalen ist jedoch gleich, dass sie mindestens einen Bezeichner (Namen), einen Datentyp und eine Einheit haben. Signale werden innerhalb von Simulinkmodellen über Signalpfade transportiert, die jeweils von einer Ausgangsschnittstelle eines Simulink®-Blocks zu einer Eingangsschnittstelle verlaufen. Diese können in Simulink®, in Analogie zum Fahrzeug, auch zu Bussen aggregiert werden.

Parameter Ein Modell ist eine Verarbeitungsvorschrift, die, genau wie ein Algorithmus, während der Ausführung in Struktur und Befehlsfolge nicht veränderbar ist. Die konkrete Ausprägung eines Algorithmus wird geregelt durch die Definition von Konstanten und Variablen. Bei Simulationsmodellen werden diese unter dem Begriff *Parameter* zusammengefasst. Ein Parameter hat dabei die Eigenschaften einer Variable. Das sind im Speziellen ein Name, einen Datentyp und einen Wert. Zusätzlich unterstützen Parameter noch Einheiten sowie Startwerte, die eine Variable mit diesem Wert initialisieren, noch bevor ihr ein Wert explizit zugewiesen wird. Die Anwendung bzw. Nutzung von Parametern zur Steuerung der konkreten Abarbeitung eines Modells wird als Bedatung oder Parametrierung bezeichnet. Eine Aggregation von Parametern in einer logischen Einheit, wie einem Skript zur vollständigen Bedatung eines Modells, bezeichnet man dabei als Parametersatz. Die Nutzung von Parametersätzen ist abhängig von der gewählten Simulationsumgebung. Zwei verbreitete Methoden sind die Initialisierung eines Simulationsworkspace, beispielsweise des MATLAB®-Workspace bei der Nutzung der Simulink®-Toolbox, oder so genannte Parameterports, die die Bedatung ähnlich zu Signalen extern über Modellschnittstellen durchführen lassen. Während der Einsatz der ersten Methode auf die jeweilige Simulationsumgebung beschränkt ist, erlaubt die Methode der Parameterports die Nutzung der gleichen Parametrierung über weitere Schritte im Entwicklungsprozess, beispielsweise nach einer Code-Generierung und Kompilierung für eine bestimmte Zielplattform.

Auf Grundlage der vorgestellten Entwicklungsartefakte sind weitere Betrachtungen, insbesondere ihrer Ausprägungen und Zusammenhänge im Rahmen der Funktionsabsicherung, notwendig.

Zusammenhänge der Entwicklungsartefakte Abbildung 3.5 zeigt die beschriebenen Entwicklungsartefakte und ihre Zusammenhänge für die konkrete Ausprägung eines Simulink®-Motor-Modells für einen Ottomotor [Hak13]. Dabei bildet das Modell selbst zunächst einen generischen 4-Zylinder Ottomotor ab. Auf einfachster Ebene könnte dieses Modell als ein Platzhalter einer Motorkennlinie interpretiert werden, die eine Leistungsanforderung auf der X-Achse auf eine Leistungsabgabe auf der Y-Achse abbildet. Die Kennlinie selbst ist dabei in der Regel nicht Teil des Simulinkmodells. Die Schnittstellen für die Leistungsanforderung und Leistungsabgabe sind Teil des Modells und werden je nach Fachabteilung ggf. noch zusätzlich in generischer Form, z.B. in XML, gespeichert. Die Schnittstellen sind dabei über Signalpfade mit den Blöcken der Berechnungsvorschrift, in diesem Fall der Motorkennlinie verbunden und die erlaubten Signale sind spezifiziert. Um ein solches generisches Motormodell für einen tatsächlichen Ottomotor einer bestimmten Leistungsstufe auszuprägen, werden Parametersätze verwendet. In diesem einfachen Fall würde ein Parametersatz einen strukturierten Datentyp enthalten, der eine im realen Motor gemessene Motorkennlinie als mathematische Funktion enthält. Dadurch wäre unter Nutzung unterschiedlicher Parametersätze die Ausprägung ein und desselben Modells beispielsweise als kleiner 1,4L Motor mit 122PS oder als großer, leistungsfähigerer Motor mit 1,8L Hubraum und 180PS denkbar.

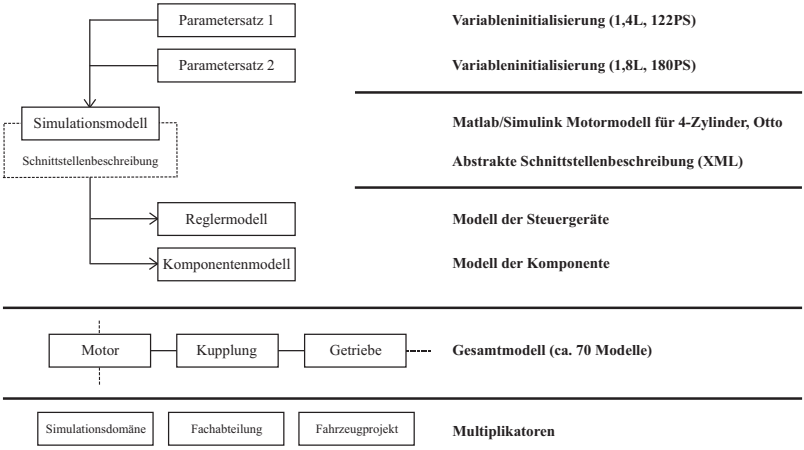


Abbildung 3.5: Struktur modellbasierter Entwicklungsartefakte am Beispiel der automotive Domäne

Kategorisierung von Modellen Ausgehend von den beschriebenen Artefakten wird die Ausprägung von Modellen für Fahrzeugfunktionen in der Regel nach Steuergeräte- und Komponentenmodellen unterschieden. Steuergerätemodelle bilden dabei den elektronischen Regler und die darauf laufenden Algorithmen ab. Eingangsgrößen dieser Modelle sind dabei die Sollwerte und die Sensorsignale. Ausgangsgrößen sind die Stellwerte der Aktuatoren. (vgl. Abbildung 2.1) Komponentenmodelle stellen in der Regel physikalische Strecken-

modelle dar, die Aktuatoren, Regelstrecke und Sensoren vereinen (im Sinne der rekursiven Definition von Simulinkmodellen, können Aktuator, Regelstrecke und Sensoren wieder als einzelne Modelle betrachtet werden). In Komponentenmodellen sind hierbei in der Regel keine elektrischen Eigenschaften und Algorithmen modelliert, sondern physikalische Gesetze und Übertragungsfunktionen.

Gesamtmodell Um die Simulation komplexer Zusammenhänge zu ermöglichen, werden diese in der Regel modular in einzelne Simulationsmodelle zerlegt und mithilfe definierter Schnittstellen zu Gesamtmodellen verschaltet. Die Zerlegung kann sich dabei beispielsweise an der Fahrzeugstruktur, wie in Abbildung 3.5 darstellt mit der Trennung in Motor-, Kupplung- und Getriebemodell, oder an der Funktionalität, beispielsweise elektrisches, thermisches und mechanisches Modell, orientieren. Modelle aktueller Fahrzeugprojekte erreichen dabei eine Größe von 70 oder mehr Einzelmodellen. Darüber hinaus reicht ein Gesamtmodell je Fahrzeugprojekt in der Regel nicht aus. Je nach Simulationsdomäne und fachspezifischen Anforderungen können unterschiedliche Gesamtmodelle des gleichen Umfangs für die Entwicklung notwendig sein.

Die Vielzahl an Einzelmodellen sowie die Ausbildung unterschiedlicher Modellvarianten mit Hilfe von Signalen und Parametern stellt die Automobilindustrie vor die Herausforderung diese in einer vernetzten Entwicklungsumgebung geordnet und möglichst zentral allen Prozessbeteiligten zugänglich zu machen. Dieser Herausforderung wird mithilfe von Modelldatenmanagementsystemen begegnet.

3.5 Modelldatenmanagement

Unter Modelldatenmanagement wird im Rahmen der vorliegenden Arbeit die Menge an Daten und Informationen verstanden, die ein Modell umfassend definieren und beschreiben. Die Beschreibung ist dabei durch das im modellbasierten Entwicklungsprozess notwendige Wissen abgegrenzt. Zum Management dieser Modelldaten gehören mindestens folgende Elemente:

- Das Modell selbst sowie die Datei, die das Modell enthält
- Schnittstellen-, Signal- und Parameterdokumente
- Meta-Daten zu Autor, Datum, Software, Lizenzen, Versionen
- Beschreibende Informationen zu Einsatzzweck, Qualität und Ergebnisgüte, Rahmenbedingungen des Einsatzes
- Beziehungen und Abhängigkeiten zu anderen Modellen

Das Modelldatenmanagement umfasst in der Regel ein oder mehrere Softwaresysteme, die sich nach Art der gehaltenen Daten sowie ihrer Rolle im Entwicklungsprozess klassifizieren lassen. Im Entwicklungsprozess unterscheidet man zwischen Datenhaltungssystemen und Autorenwerkzeugen. Datenhaltungssysteme nehmen definierte Arten von Daten auf, speichern diese in einer systemabhängigen Form - dies könnte beispielsweise eine stark strukturierte Datenbank oder ein weniger strukturiertes Dateisystem sein. Die Rolle der Datenhaltungssysteme im Prozess besteht anschließend in der dauerhaften oder zeitlich beschränkten Persistierung der Daten sowie der Bereitstellung für bestimmte Personenkreise oder Autorenwerkzeuge.

Autorenwerkzeuge dienen der Erzeugung, Bearbeitung bzw. Veränderung und der Nutzung von Daten. Dafür sind sie direkt oder indirekt (durch Zwischenschritte des Im- und Exports) mit Datenhaltungssystemen verbunden, um die dort verfügbaren Daten zu nutzen oder neue bzw. veränderte Daten in die Systeme einzupflegen.

Die Auswahl der Autorenwerkzeuge ist direkt von den Anforderungen des modellbasierten Entwicklungsprozesses abhängig. Die Durchführung bestimmter Entwicklungstätigkeiten und dafür gesetzter Rahmenbedingungen beschränkt dabei die Auswahl der am Markt verfügbaren Werkzeuge für modellbasierte Entwicklung. Die Auswahl der Autorenwerkzeuge definiert wiederum die notwendigen Datenhaltungssysteme, da sie die Entwicklungsartefakte vorgeben, die in diesen gehalten werden müssen.

Für die automobile Funktionsentwicklung und -absicherung sind hierbei folgende Besonderheiten zu erkennen. Wie in Abschnitt 3.1 erläutert wurde, stellt die modellbasierte Entwicklung im Automobilbau eine relativ junge, historisch gewachsene Entwicklungsform dar. Die unterschiedlichen Entwicklungsdomänen haben die Nutzung von Modellen im Entwicklungsprozess unabhängig voneinander etabliert und für ihren jeweiligen, fachspezifischen Prozess ausgeprägt. Entsprechend heterogen stellt sich das Datenmanagement dar. Die Analyse der Ist-Situation zu Beginn der Untersuchungen zur vorliegenden Arbeit, zeigte Anfang des Jahres 2012, dass die Entwicklungsabteilungen in ihren eigenen, wenig vernetzten Prozessen arbeiteten. Als Ablage für die Daten kommt heute in der Regel eine Auswahl der folgenden Datenhaltungssysteme zum Einsatz:

- **Fileshares** als Netzlaufwerke: die meist genutzte Datenablage sind klassische Ordnerstrukturen, die im Netzwerk freigegeben werden und in hierarchischen Strukturen eine gewisse Ordnung der Entwicklungsartefakte vorgeben. Der Grund für die verbreitete Nutzung liegt vorwiegend in der breiten Verfügbarkeit und der einfachen Bedienung dieser Ablageform.
- **Subversion** [PCSF08] als Versionierungssystem: die Verbreitung von Subversion (SVN) als Versionierungssystem ist in den Fachabteilungen bereits annähernd auf dem Niveau der Fileshares angekommen. Die hierarchisch organisierten Repositories erlauben es die Entwicklungsartefakte ähnlich wie in Ordnern auf einfache Art und Weise zu strukturieren. Darüber hinaus erlauben sie die dateibasierte Versionierung, d.h. die parallele Speicherung aller Änderungen an einzelnen Dateien und die Verfügbarkeit älterer Versionen. Subversion ist als zentrales Entwicklungswerkzeug bereits heute in viele Toolketten tief integriert, so dass Entwicklungswerkzeuge in diesen Toolketten meist direkt auf den Repositories arbeiten und die Daten zugreifbar sind. Die Prozessanalyse zur vorliegenden Arbeit zeigte jedoch auch, dass es klare Nachteile gibt. So existiert kein dateibasiertes Rollen- und Rechtemanagement. Freigaben werden daher in der Regel für ganze Zweige im Repository oder mindestens auf Ordnerebene durchgeführt. Ein weiterer Nachteil besteht darin, dass Subversion ursprünglich für die Verwaltung von Softwarecode im Klartext, d.h. ASCII-Dateien [Int02] mit konstanten Strukturen je Version ausgelegt ist. In diesen Dateien können Änderungen zeichengenau identifiziert werden und nur die Änderungen zur letzten Version, das so genannte „Diff" gespeichert werden. Bei Simulink®-Dateien handelt es sich zwar um ASCII-Dateien, allerdings sind die Strukturen nicht konstant. Am einfachsten wird dies anhand der in Abschnitt 3.1 beschriebenen Blockstrukuren von Simulinkmodellen deutlich. Jeder Block hat eine Block-ID, die diesen im Modell identifiziert. Zwischen zwei Speicherständen ein und desselben Modells können sich die-

se laut Formatspezifikation allerdings ändern, was eine „Diff"-Speicherung praktisch nicht möglich macht.

- **PTC Integrity** [PTC14] als Versionierungssystem und Prozesssteuerung: besonders für die Entwicklung sicherheitskritischer Funktionen findet das Datenmanagementsystem PTC Integrity Anwendung. Neben der reinen Versionsverwaltung ist hier auch das Rollen- und Rechtemanagement auf Dateiebene möglich. Das System wird auch als Life-Cycle-Management [Eig08] System vertrieben und bietet daher, über die Versionsverwaltung hinaus, die Anbindung von Prozesssteuerungstools sowie die Verschlagwortung von Artefakten. Da es sich jedoch um eine proprietäre Software der Firma PTC handelt, ist die breite Verfügbarkeit im Entwicklungsprozess stets durch die Verfügbarkeit von Schnittstellenanbindungen in vorhandenen Toolketten sowie die Verfügbarkeit von Lizenzen beschränkt.
- **Individuallösungen** Neben den beschriebenen Lösungen, die aus der reinen Softwareentwicklung stammen, existieren auch verschiedene Einzellösungen, die zumeist auf sehr spezifische Toolkettenbedarfe hin entwickelt wurden. Dabei können zwei Typen von Individuallösungen unterschieden werden. Zum einen handelt es sich um proprietäre Datenablagen, die im Rahmen von Auftragsentwicklungen speziell für bestimmte Fachbereiche entwickelt wurden. Zum anderen verbergen diese Lösungen die generischen Oberflächen der oben vorgestellten Lösungen und stellen diese in prozessangepassten Sichten zur Verfügung.

Neben der Speicherung und Versionierung von Daten kommen im Entwicklungsprozess weitere Unterstützungswerkzeuge zum Einsatz. Diese lassen sich nach dem Zweck unterteilen in Anforderungs-, Aufgaben-, Prozessmanagement und Dokumentation. Für das Anforderungsmanagement ist im VW-Konzern, wie auch bei den meisten anderen OEMs, das Tool IBM Rational Doors [IBM14] das zentrale Werkzeug zur Verwaltung von Anforderungen auf allen Ebenen der Fahrzeugentwicklung. Auch Funktionen werden zunächst in Doors definiert. Für die Absicherung von Funktionen müssen aus diesen Anforderungen Absicherungsaufgaben abgeleitet werden, die wiederum in Anforderungen für die Erstellung von Simulationsmodellen münden. In den letzten Jahren hat sich hier bei der AUDI AG das Tool Atlassian JIRA [Atl14] etabliert, das als reines Aufgabenmanagement vertrieben wird, aber aufgrund der relativ einfachen Erweiterbarkeit auf spezifische Prozessabläufe hin angepasst werden kann. In den im Rahmen dieser Arbeit betrachteten Entwicklungsprozessen kann JIRA als defacto Standard für die Verwaltung von Absicherungsaufgaben und Modellanforderungen angenommen werden.

Für die Dokumentation von Entwicklungsartefakten existiert aktuell kein einheitliches Vorgehen. So werden Informationen zum Verwendungszweck von Modellen beispielweise häufig in hierarchischen Ordnerstrukturen oder Dateinamen dokumentiert. Eine Erfassung von Metadaten, wie Ersteller, Datum, aber auch Versionsabhängigkeiten in Entwicklungswerkzeugen werden in der Regel nicht dokumentiert. Vereinzelt fnden sich bei der Prozessanalyse Microsoft Word-Dateien, die als Formatvorlage für die Definition von Testaufgaben oder Einzeldokumentationen genutzt werden. Zur Verwaltung von Signalen, Parametern und Schnittstellen ist der Einsatz von Microsoft Excel-Dateien ein verbreitetes Vorgehen, doch auch hier existieren keine einheitlichen Formate. Bei näherer Betrachtung wird deutlich, dass es auch für die Benennung der Artefakte keine einheitliche Vorgabe gibt und damit auf dieser Basis erstellte Modelle allein aufgrund uneinheitlicher Schnittstellen nicht kompatibel sind.

3.6 Zusammenfassung

im vorliegenden Kapitel 3 wurden die Konzepte und Begriffe eingeführt, die für das Verständnis des in dieser Arbeit vorgestellten Ansatzes und notwendig sind. Dabei wurden in Abschnitt 3.1 zunächst die wichtigsten Informationen zur modellbasierten Entwicklung und Absicherung als Vorgehensmodelle eines Entwicklungsprozesses erläutert. Darauf aufbauend wurde in Abschnitt 3.2 die automobile Funktionsabsicherung als eine Anwendung der modellbasierten Absicherung vorgestellt. Für die Durchführung dieser Absicherung wurde die X-in-the-Loop-Simulation als eine zentrale Methodik identifiziert und im Anschluss in Abschnitt 3.3 im Detail erläutert. In Abschnitt 3.4 wurden die dafür notwenigen Entwicklungsartefakte eingeführt, sowie strukturell und semantisch in Beziehung gesetzt. Abschließend wurden in Abschnitt 3.5 Datenmanagementaspekte und deren Abhängigkeiten zu den Entwicklungsartefakten erläutert.

Aufbauend auf den in diesem Kapitel dargestellten Grundlagen zur Einordnung des Themas in die übergeordneten Entwicklungsstrukturen wird in Kapitel 4 der Stand der Technik sowohl für die dafür heute genutzten, als auch für die in den Konzepten dieser Arbeit (vgl. Kapitel 5) benötigten Technologien beschrieben.

4 Stand der Technik

Die vorliegende Arbeit beschreibt ein Konzept, das unter anderem zur strukturierten, automatisierten Dokumentation von Simulationsmodellen geeignet ist. Dabei ist es das Ziel, vorhandenes Wissen aus Entwicklungsprozessen automatisiert zu extrahieren und in einer Wissensbasis abzubilden. Um diese Thematik angehen zu können, werden in Abschnitt 4.1 zunächst die Begriffe Daten, Information und Wissen erläutert. In den Abschnitten 4.2 und 4.3 werden die heute dafür genutzten Technologien im Bereich des Modelldatenmanagement im Automobilbau und anderen Industriezweigen vorgestellt. Im Anschluss daran werden in den Abschnitten 4.4 und 4.5 die nach heutigem Stand wichtigen und für die Arbeit relevanten Ansätze der Informatik zur Gewinnung, Speicherung und Verarbeitung von Wissen vorgestellt.

4.1 Wissensmanagement

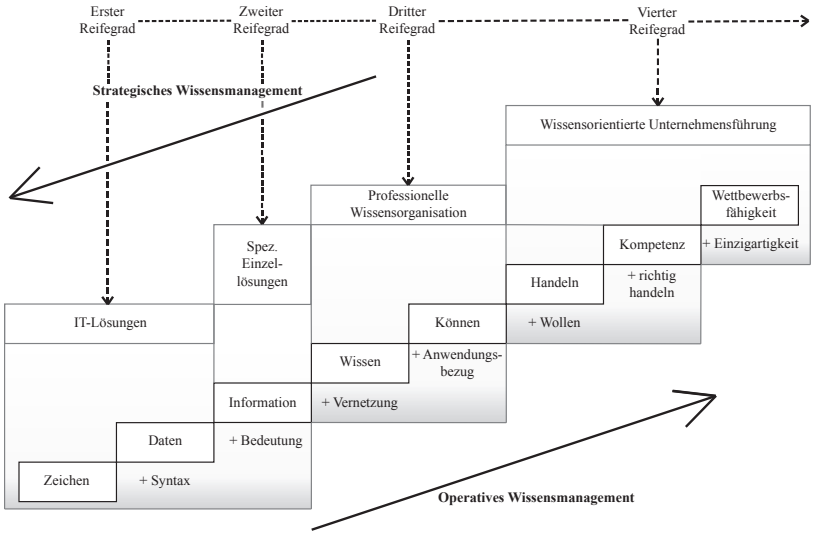


Abbildung 4.1: Wissenstreppe nach North [Nor02]

Das Schlagwort Wissensmanagment wird heute in vielen Kontexten für die strukturierte Speicherung von Daten verschiedenster Form genutzt. Eine definierte Ordnung auf Daten allein ist jedoch kein Merkmal von Wissen. Um den Zusammenhang von Daten zu Wissen zu verstehen, gehen wir von der Wissenstreppe nach North [Nor02] aus, die in Abbildung 4.1 dargestellt ist. Das kleinste Datenelement stellt das Zeichen dar, d.h. Buchstaben, Ziffern oder Sonderzeichen. Werden diese Zeichen unter Nutzung von strukturellen Regeln, so genannter Syntax, zu Zeichenketten verknüpft, werden diese als Daten bezeichnet. Daher ist streng genommen jede logische Verknüpfung von Zeichen, die einer gewissen Regel folgt, ein Datum. Als Beispiel sei die Zeichenkette „A3" angeführt. Der Begriff Information ist eine Zusammensetzung aus Daten mit der Zuordnung einer Bedeutung, der so genannten Semantik. Ordnet man der Zeichenkette „A3" wahlweise die Semantik „Fahrzeugmodell" und „Papier" zu, ergibt sich eine grundlegend unterschiedliche Bedeutung des gleichen Datums - das Ergebnis ist eine Information. Durch Vernetzung von Informationen, d.h. durch die Definition von Beziehungen, entsteht Wissen. Im gewählten Beispiel sei das Datum „A3" ein Fahrzeugmodell und das Datum „TDI" eine Motortechnologie. Definiert man auf diesen Informationen die Beziehung, dass ein Fahrzeug einen Motor besitzt („Fahrzeug besitzt Motor"), so ergibt sich das Wissen, dass die Kombination „A3 TDI" in dieser Wissensbasis sinnvoll ist, als logische Schlussfolgerung, sofern keine andere Beziehung existiert, die dieser Schlussfolgerung widerspricht (vgl. Abschnitt 4.1.2). Die übrigen Stufen der Wissenstreppe umfassen den Einsatz und die Randbedinungen von Wissen in einer wissensorientierten Unternehmensführung. Für weiterführende Informationen sei auf [Nor02] und [HKRS08] verwiesen.

4.1.1 XML-Repräsentationen von Wissen

Die eXtensible Markup Language (XML), ins Deutsche übersetzt „erweiterbare Auszeichnungssprache", wurde 2008 als Spezifikation vom World Wide Web Consortium (W3C) [Con14] herausgegeben. XML ist eine Metasprache, die der Darstellung hierarchisch strukturierter Daten in Form von ASCII-Dateien dient. „Meta" bezeichnet die Fähigkeit anwendungsspezifische Sprachen zu definieren. Eine spezifische Sprache wird dabei durch Einschränkung der Sprachmittel durch so genannte Document Type Definitions (DTD) oder XML-Schemata definiert. Für weiterführende Informationen zu XML sei auf [HKRS08] verwiesen.

Der Grund, warum XML im Zusammenhang mit Wissensmanagement an dieser Stelle von Bedeutung ist, ist die Mächtigkeit der Metasprache und die daraus resultierende starke Verbreitung in diesem Forschungsgebiet. Beispielhaft sei dazu auf [VLKH04] verwiesen. Wie in den folgenden Abschnitten gezeigt wird, ist XML sowohl für den Semantic Web Ansatz, als auch für den modellbasierten Ansatz in der Praxis relevant.

In Algorithmus 4.1 wird das Beispiel aus Abschnitt 4.1 aufgegriffen und in einer frei gewählten, aber strukturell und semantisch für Menschen lesbaren Form in XML repräsentiert. Es wird deutlich, dass auf diese Weise bereits Wissen explizit abgebildet, gespeichert und transportiert werden kann. In der Regel wird dabei Wissen in Form von gerichteten, gewichteten Graphen (im Folgenden als Graph bezeichnet) [Die06] repräsentiert. Ein Graph ist dabei definiert als $G = \{V, E\}$ eine Menge von Knoten V und eine Menge von Kanten E mit $E = V \times V$. Dabei repräsentieren die Knoten V Daten, denen entweder durch Attribute oder andere Knoten eine gewisse Bedeutung zugeordnet wird. Die Menge der Kanten E re-

Algorithmus 4.1 Einfache Wissensrepräsentation in XML

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Wissensbasis>
        <Fahrzeugtypen>
                <Fahrzeugtyp>A3</Fahrzeugtyp>
                <Fahrzeugtyp>A4</Fahrzeugtyp>
                <Fahrzeugtyp>...</Fahrzeugtyp>
        </Fahrzeugtypen>
        <Motorentechnologie>
                <Motor>TFSI</Motor>
                <Motor>TDI</Motor>
        </Motorentechnologie>
        <Relationen>
                <Relation>
                        <Subject>A3</Subject>
                        <Prädikat>hat</Prädikat>
                        <Objekt>TDI</Objekt>
                </Relation>
        </Relationen>
</Wissensbasis>
```

präsentiert gerichtete Beziehung - Relationen - von einem Knoten $v_1 \in V$ zu einem Knoten $v_2 \in V$, so dass gilt $e \in E = v_1 \rightarrow v_2$. Dabei enthält jede Kante $e \in E$ einen Wert ω, der die Beziehung zweier Knoten, die mit dieser Kante verbunden sind, definiert. Die Beziehung zweier Knoten wird wie folgt bezeichnet: $e_\omega = v_1 \rightarrow v_2$.

Der Graph ist eine geeignete Form, um Daten Bedeutungen und Informationen Beziehungen zuzuordnen. Er bildet direkt die natürlichsprachliche Satzbildung ab, die der Grammatik romanischer Sprachen zu eigen ist. Die Aussage einer Beziehung zwischen zwei Objekten bildet sich aus der Struktur „Subjekt Prädikat Objekt", in der Form, dass ein Objekt (das Subjekt) eine qualifizierte Beziehung (das Prädikat) zu einem anderen Objekt (das Objekt) hat. Die Benennung der Relation des Beispiels in Algorithmus 4.1 verdeutlicht dies noch einmal. Für die Graphrepräsentation bedeutet dies, ein Subjekt $s \in V$ und ein Objekt $o \in V$ haben die durch das Prädikat p qualifizierte Beziehung $e_p = s \rightarrow o$. Man bezeichnet eine Aussage dieser Form aus Subjekt, Prädikat und Objekt als Triple. Im Rahmen dieser Arbeit werden alle Triple in folgender Schreibweise dargestellt: $\{s, e_p, o\}$

4.1.2 Semantic Web Ansatz

Der Semantic Web Ansatz zur Speicherung und Verarbeitung von Wissen stammt ursprünglich aus dem Umfeld des World Wide Web (WWW) [Con14] - dem Teil des Internets, der von Tim Berners-Lee [BL89] erstmals beschrieben wurde, als ein Netz mittels so genannte Hyper-Texts verknüpfter Dokumente, um logische Strukturen und Verweise für den Menschen verständlich abbilden zu können. Mit der Verbreitung des Internets und der rapiden Zunahme an Dokumenten im World Wide Web, entstand der Bedarf das dort vorhandene

Wissen indizieren und maschinell verarbeiten zu können, da es für einen einzelnen Menschen nicht mehr beherrschbar war. Die Herausforderung bei dieser Aufgabe besteht im Speziellen darin, die vorhandenen, meist unstrukturierten Dokumente für Computer maschinenverarbeitbar aufzubereiten, um beispielsweise Fragen gegen diese Wissensbasis zu stellen und passende Antworten zu erhalten.

Aus heutiger Sicht sind für dieses Problem zwei Lösungen denkbar und Gegenstand aktueller Forschungen. Einerseits werden Methoden der künstlichen Intelligenz untersucht, um die kognitiven Aufgaben des Menschen bei der Verarbeitung der Informationen zu übernehmen und diese in formale, maschinenverarbeitbare Daten zu verwandeln. Zwar existieren in diesem Forschungsfeld aktuell noch Aktivitäten, doch die bisherigen Ergebnisse lassen eine praktische Anwendung, zumindest in der näheren Zukunft, nicht erwarten [Den12].

Der zweite Ansatz, das Semantic Web, folgt der Idee Informationen initial, d.h. bei deren Generierung, in einer Form zur Verfügung zu stellen, die die Verarbeitung durch Maschinen erlaubt. Dieser Ansatz hat sich nach aktuellem Stand als erfolgversprechend herausgestellt und soll daher im Folgenden näher vorgestellt werden.

Skizziert wurde die Idee eines Semantic Web erstmals von Tim Berners-Lee in [BLHL01] als eine Erweiterung des World Wide Web, die es ermöglicht, das vorhandene Wissen in einer Form zu annotieren und damit algorithmisch zu verarbeiten. Um diesen Funktionsumfang zu ermöglichen existieren zwei Grundvoraussetzungen. Zunächst ist es nötig, dass offene Standards für die Beschreibung der Informationen vereinbart werden, die es unabhängig von Plattformen und Anwendungen ermöglichen diese zu einander in Beziehungen zu setzen. Darüber hinaus müssen Methoden geschaffen werden, die das Schlussfolgern von neuem Wissen auf vorhandenem erlauben. Konkret ist die Umsetzung einer formalen Logik notwendig, die nicht nur explizit spezifiziertes Wissen verarbeitet, sondern auch implizites Wissen schlussfolgern kann [Den12].

Die Spezifikationen der Standards des Semantic Web werden derzeit vom World Wide Web Consortion (W3C) [Con14] vorangetrieben. Die grundlegenden Technologien dieser Standards sind neben XML die Ontologiesprachen RDF(S)[Bec04] und OWL [PSMP12, Den12]. Eine Ontologie ist in diesem Zusammenhang eine Wissensbasis, die das Wissen einer konkreten Fach- oder Anwendungsdomäne modelliert. An dieser Stelle wird auf eine detailliertere Beschreibung der einzelnen Sprachen verzichtet - für beide Sprachen existieren XML-Serialisierungen [Bec04, PSMP12], die die Repräsentation des Wissens äquivalent zu den Erläuterungen in Abschnitt 4.1.1 vornehmen.

Ontologien sind in diesem Zusammenhang definiert als explizite, formale Spezifikation der Konzeptualisierung eines abgegrenzten Diskursbereiches zu einem definierten Zweck, auf die sich eine Gruppe von Akteuren geeinigt hat [SS09]. Der Diskursbereich beschreibt einen abgegrenzten Informationsbereich, in welchem sich Akteure, d.h. Individuen, die in irgendeiner Form eine Beziehung zu diesem Informationsbereich besitzen, auf eine bestimmte Form der Kommunikation geeinigt haben. Die Konzeptualisierung bezeichnet die Begriffe und Zusammenhänge, die diesen Informationsbereich strukturieren. Die Konzeptualisierung besteht primär aus sogenannten Konzepten, Eigenschaften und Beziehungen. Konzepte beschreiben in der Regel Begriffe, die im Informationsbereich von Bedeutung sind. Eigenschaften konkretisieren die Ausprägungen von Konzepten. Beziehungen beschreiben das Verhältnis bzw. die Relationen der Konzepte im Informationsraum zueinander. Diesen Beziehungen können mathematische Eigenschaften, wie Symmetrie, Transitivität, Reflexivi-

tät oder Inversität zugeordnet werden. Formal ergeben sich folgende Strukturen für Ontologien:

- Sei C die Konzeptualisierung eines Diskursbereiches D mit den Beziehungen (Relationen) R definiert als: $C = (D, R)$.
- Sei der Diskusbereich D definiert als Menge von Begriffen d: $D = \{d_1, d_2, \ldots, d_n\}$
- Seien die Beziehungen R definiert als Menge von Beziehungen r: $R = \{r_1(d_{x_1} \in D, d_{y_1} \in D), r_2(d_{x_2} \in D, d_{y_2} \in D), \ldots, r_n(d_{x_n} \in D, d_{y_n} \in D)\}$

Ausgehend von einer standardisierten Repräsentation von Domänenwissen als Tripel in Ontologien wurden Methoden zur Schlussfolgerung von Wissen, d.h. neuen Tripeln, auf vorhandenem Wissen entwickelt. Diese basieren auf formaler Logik, der den relativ frei definierbaren Beziehungen eine mathematisch-logische Relation zuordnet. Dadurch können Beziehungen beispielsweise als transitiv, reflexiv oder symmetrisch modelliert werden [Hed12]. Übertragen auf das Beispiel aus Abschnitt 4.1 könnte die Wissensbasis die Triple {A3, ist_ein, PKW} und {PKW, ist_ein, Kraftfahrzeug} enthalten. Sei „ist_ein" definiert als eine transitive Relation, so folgt aus dieser Wissensbasis das neue Wissen {A3, ist_ein, Kraftfahrzeug}. Die verfügbaren Relationen sind dabei abhängig von der genutzten Methode der Schlussfolgerung. Eine Schlussfolgerung, wie die eben demonstrierte, wird als Inferenz, die softwaretechnische Umsetzung zur automatischen Schlussfolgerung auf Ontologien als Inferenzmaschine (Engl. „inference engine") bezeichnet.

Die Inferenz auf Ontologien unterscheidet zwei zentrale Konzepte, die die Ergebnisse einer Inferenzmaschine beeinflussen. Diese sind als Closed-World-Assumption und Open-World-Assumption bekannt und beschreiben die Gültigkeit von Beziehungen innerhalb der Ontologien. Die Closed-World-Assumption bedeutet für eine Wissensbasis, dass alles, was nicht auf Basis der Wissensbasis beweisbar, d.h. inferrierbar, ist, als falsch angenommen wird. Im Gegensatz dazu geht die Open-World-Assumption davon aus, dass alles Wissen, das nicht explizit spezifiziert ist, als unbekannt angenommen werden muss. In obiger Wissensbasis ist spezifiziert, dass ein A3 ein Kraftfahrzeug ist. Auf die Frage: „Ist ein A4 ein Kraftfahrzeug?", führt die Closed-World-Assumption dazu, dass die Inferenzmaschine ein „Nein" zurückliefert, da es kein Triple gibt, das diese Aussage stützt. Auf Basis der Open-World-Assumption ergibt sich die Antwort „Unbekannt", da zwar auch hier kein Triple existiert, dieses Fehlen jedoch auf eine unvollständige Wissensbasis zurückführbar sein kann.

Für weiterführende Informationen zu den Technologien des Semantic Web sei an dieser Stelle auf [Den12, HKRS08] verwiesen.

4.1.3 Modellbasierte Ansätze

Das modellbasierte Wissensmanagement hat sich in der Industrie in allen Bereichen als Methode der geschäftlichen Wissensorganisation etabliert. Entsprechend der in Abschnitt 3.1 erläuterten Denkmuster des Menschen, ist es sinnvoll Organisationsstrukturen und Geschäftsprozesse in Modellen zu repräsentieren [Kra07]. Um das vorhandene Wissen eines Unternehmens strategisch nutzen zu können, ist es notwendig dieses in einer geeigneten Dokumentation und Darstellung zu erfassen (vgl. Allweyer in [Sch98]). Verbreitet wird hier nach dem ARIS-Konzept nach Scheer[Sch98] vorgegangen, um vorhandene Prozesse zu beschreiben und Anpassungen zu planen und zu steuern.

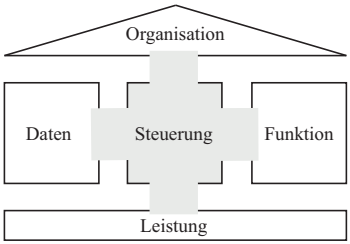


Abbildung 4.2: Sichten der Prozessmodellierung nach dem ARIS-Konzept [Sch98]

Das ARIS-Konzept sieht die Gestaltung des Modells eines Geschäftsprozesses vor, die fünf Sichten (vgl. Abbildung 4.2) oder auch Facetten der Prozessmodellierung berücksichtigt und dadurch die Aufbereitung vereinfacht. Die erste Sicht ist die *Funktionssicht*, die die Tätigkeiten des Unternehmens gruppiert und in hierarchische Beziehungen setzt, und die Ziele der Tätigkeiten definiert. Die *Organisationssicht* beschreibt alle Ressourcen, Arbeitskräfte ebenso, wie materielle Arbeitsmittel, wie Maschinen oder Rechner und setzt diese in Beziehung. Die *Datensicht* erfasst alle Tätigkeiten, die Daten generieren und beschreibt die Datenflüsse im Unternehmen. Die *Leistungssicht* beschreibt alle Dienst-, Sach- und finanziellen Leistungen. Abschließend fasst die *Steuerungssicht* alle anderen vier Sichten in einen zeitlich-logische Ablaufplan zusammen.[KH06]

Nach der initialen Erstellung des Modells dient es im modellbasierten Wissensmanagement als übergeordnetes Regelwerk, an dem sich prozessuale Veränderungen orientieren und planen lassen. Ein Beispiel dafür ist das von Allweyer vorgeschlagene Vorgehensmodell zur Prozess-/Modelländerung, das er als „Knowledge Process Redesign" bezeichnet [Sch98, Kra07]. Zusammenfassend bedeutet dies, dass das Modell im Wissensmanagement stets eine Einschränkung der Freiheitsgrade bei wissensbasierten Entscheidungen ermöglicht und damit die zielgerichtete Steuerung von Tätigkeiten unterstützt.

Einen weiteren Ansatz für modellbasiertes Wissensmanagement bietet EMF [Ste09], welches in der modellbasierten Softwareentwicklung eingesetzt wird und auch für die Strukturierung von Wissen genutzt werden kann. Die generische Definition von MOF erlaubt die Beschreibung von Graphen in der in Abschnitt 4.1.1 beschriebenen Form, so dass eine Wissensrepräsentation möglich wird. Das EMF-Modell bildet dabei das Domänenwissen ab, wie es die Ontologien bei semantischen Technologien übernehmen. Anschließend kann je nach Anwendung das vorhandene Wissen im Rahmen der im Modell beschriebenen Strukturen abgelegt werden. Eine detaillierter Beschreibung zu EMF wird in Abschnitt 4.3.1 gegeben.

Wie in Abschnitt 4.1.2 erläutert, ist darüber hinaus auch eine Methode zur Wiederverwendung bzw. Abfrage des gespeicherten Wissens notwendig. Für das modellbasierte Wissensmanagement gibt es hier verschiedene Ansätze. Einerseits kann mittels Codegenerierung ein API (Application Programming Interface, eine Programmierschnittstelle) generierte werden, das den strukturierten Zugriff auf das Modell und darin abgelegten Informationen ermöglicht. Diese Lösung ist besonders bei Modellen mit seltenen bis keinen Änderungen am Modell geeignet, da sich dadurch Änderungen am generierten Code und damit auch der API ergeben. Dem gegenüber steht die Möglichkeit der Nutzung von Abfragesprachen. Die-

se erlauben es, strukturierte Anfragen, vergleichbar zu SQL bei relationalen Datenbanken, gegen die Datenbasis zu stellen. Die zurückgelieferten Ergebnisse einer solchen Abfrage können anschließend algorithmisch verarbeitet werden. Diese Vorgehensweise erlaubt eine einfachere Anpassung des Modells, da die Struktur der Suchergebnisse mit der Anfrage beschrieben wird (vgl. Select-Anteil SQL[Kre01]). Dadurch können Änderungen zwar dazu führen, dass die Ergebnisse beeinflusst werden, für die nutzenden Algorithmen werden jedoch stets interpretierbare Daten zurückgeliefert. Beispielhaft sei hier EMF-IncQuery [EI15] als Abfragesprache für EMF genannt.

4.1.4 Zusammenfassung und Bemerkungen

In den vorherigen Abschnitten 4.1.2 und 4.1.3 wurden die heute genutzten semantischen und modellbasierten Ansätze für ein Wissensmanagement vorgestellt. Im Folgenden werden diese zunächst noch einmal vergleichend für bestimmte Anwendungsfälle betrachtet, da dies für den im Rahmen dieser Arbeit gewählten Lösungsweg wichtig ist. Anschließend werden weitere Elemente des Wissensmanagement vorgestellt, die heute in der Regel unterstützend für die vorgestellten Technologien eingesetzt werden.

Die Nutzung von Ontologien und insbesondere die Nutzung der Standardtechnologien RDF und OWL ermöglichen es Domänenwissen abzubilden. Eine Kernfunktionalität dieser Technologien besteht darin, dass es nicht notwendig ist, das gesamte Domänenwissen in einer einzigen Ontologie abzubilden, sondern durch Bildung von Verknüpfungen, mehrere einzelne Wissensbasen zu einer großen zu verknüpfen. Diese Verknüpfung muss nicht einmal explizit gestaltet sein, sondern kann unter Nutzung einer Inferenzmaschine implizit mit der Anfrage erfolgen. Dies ermöglicht es, die Wissensbasis dynamisch mit neu hinzukommendem Wissen zu erweitern. Für das modellbasierte Wissensmanagement muss dafür das Modell angepasst werden, also eine explizite Verknüpfung inklusive aller Seiteneffekte für die angebundenen Applikationen durchgeführt werden.

Mit der Freiheit der einfachen Verknüpfung von Wissensbasen, ergibt sich jedoch die Herausforderung, dass Ergebnisse von Anfragen gegen Ontologien schwer zum Zeitpunkt der Erstellung einer nutzenden Applikation vorhersehbar sind. Je nach gewählter „Assumption" für die logische Folgerung können sich Abfrageergebnisse grundlegend verändern. Darüber hinaus muss berücksichtigt werden, dass je Anfrage gegen eine Wissensbasis stets nur eine der beiden „Assumptions" Anwendung finden kann. Für große Ontologien kann man sich leicht vorstellen, dass Teile nach Open-Loop- und andere Teile nach Closed-Loop-Assumption behandelt werden müssen. Im Ergebnis ist die Nutzung von semantischen Technologien bei stark heterogenen logischen Strukturen schwierig. Dem gegenüber erlaubt die modellbasierte Wissensrepräsentation logische Beziehungen zwischen einzelnen Informationen explizit auszudrücken. Damit können bis auf Daten-Ebene einzelne Beziehungen eindeutig repräsentiert und bei der Nutzung der Wissensbasis auch korrekt interpretiert werden. Dieser Vorteil wird allerdings mit erhöhter Komplexität bei der Erstellung der Domänenmodelle erkauft - je vollständiger und detaillierter ein Modell das reale Domänenwissen abbildet, desto exakter werden die Ergebnisse gegenüber dem Einsatz von semantischen Technologien sein.

Eine weitere Herausforderung für die Wissensmanagementkonzepte stellen numerische Beziehungen zwischen Daten dar. Die logische Verknüpfung von Wissen, basiert auf logischen Operatoren (z.B. AND, OR, NOT), auf der Mengenlehre (z.B. Schnittmenge, Teilmen-

ge), sowie hierarchischen Beziehungen (z.B. Ist-Ein). Numerische Beziehungen, die logische Beziehungen in Abhängigkeit von konkreten Datenwerten (z.B. „>", „<") darstellen, sind insbesondere in semantischen Technologien nur schwer abbildbar. Um diese Herausforderung genauer zu verstehen, stelle man sich eine Schnittstelle zwischen einem Motor und einem Getriebe vor (vgl. [Hak13]). Der Motor überträgt ein gewisses Drehmoment auf die Eingangsachse des Getriebes (Kupplung geschlossen). Dieses Drehmoment muss einerseits stark genug sein, um das Getriebe anzutreiben, aber nicht so stark, dass das Getriebe zerstört wird. Setzt man für das Getriebe Leistunggrenzen von Minimal 100Nm und Maximal 450Nm Drehmoment, formal bezeichnet mit $G\,[100Nm, 450Nm]$, und soll zwischen zwei Motoren $M_1\,[150Nm, 400Nm]$ und $M_2\,[150Nm, 680Nm]$ wählen, dann wird durch einen einfachen Vergleich der Grenzwerte deutlich, dass nur Motor M_1 mit diesem Getriebe gekoppelt werden darf. Die logischen Methoden heutiger Inferenzmaschinen ermöglichen diese Form der Inferenz nicht, so dass auf Basis des obigen Wissen keine Inferenz in der Form $\{M_1, istGeeignetFür, G\}$ und $\{M_2, istNichtGeeignetFür, G\}$ durchführbar ist. EMF hingegen ermöglicht es, definierte Beziehungen auf Basis von konkreten Datenwerten anzuwenden und somit die oben dargestellten Wissenbeziehungen als boolsche Werte abzufragen. Hierbei muss jedoch das unterschiedliche Vorgehen beachtet werden. Die in EMF definierten Beziehungen werden nicht mittels Inferenz auf Daten gefolgert und somit dauerhaft in der Wissensbasis verankert, sondern durch Abfrage an die Daten und Überprüfung der numerischen Regeln gefiltert. Letztendlich ist die Entscheidung für einen der beiden Wissensmanagementansätze von einer Reihe von Faktoren abhängig, die im Wesentlichen die Struktur der Anwendungsdomäne betreffen.

Für beide Ansätze existieren darüber hinaus unterstützende Technologien, die für beide gleichermaßen geeignet sind. Stellvertretend und aufgrund ihrer Bedeutung für die vorliegende Arbeit werden im Folgenden Thesauri (vom altgriechischen Thesaurus - Schatz, Schatzhaus) näher erläutert. Ein Thesaurus ist ein kontrolliertes Vokabular, dass aus thematisch verwandten Begriffen einer bestimmten Domäne besteht, die über Relationen miteinander in Beziehung gesetzt werden. [NIS05] Daher werden Thesauri auch Wortnetz bezeichnet. Als Beziehungen werden dabei in erster Linie Synonyme, Ober- und Unterbegriffe aufgenommen. Der Einsatz von Thesauri ist im Bereich der Informationserschließung (Information Retrieval) verbreitet. Sprachen ermöglichen es ein und dieselbe Information in verschiedenen Formen auszudrücken. So ist die Information, die „A3 Familie bekommt einen neuen Motor", wie es vielleicht eine Fachzeitschrift formulieren würde, gleichbedeutend mit der Aussage „der A_12X bekommt einen neuen Motor", wie es ein Entwickler bei Audi ausdrücken würden. Für eine automatische Analyse ohne zusätzliche Informationen ist dieses Information jedoch vollkommen unterschiedlich. Ein Thresauruseintrag für den A3 könnte daher wie folgt aussehen: A3(Synonyme: A_12X; Oberbegriffe: A-Reihe, Automobil; Unterbegriffe: A3 Sportback, A3 Limousine, A3 Cabrio). Es ist vorstellbar, dass die in Thesauri abgelegten Informationen zwar domänenspezifisch sind (vgl. Abschnitt 4.1.1 A3 als Papierformat), jedoch unabhängig vom Wissen der Domäne existieren können. Um die Definition von Thesauri zu standardisieren und somit wiederverwendbar zu machen, wurden in der ISO-25964 [Int11a] Regeln zu deren Erstellung in Form von Empfehlungen dokumentiert. Dabei werden neben einem Datenmodell auch Formate zum Austausch, Im- und Export von Thesaurusdaten spezifiziert.

4.2 Modelldatenmanagement im Automobilbau

In Abschnitt 3.5 wurden die aktuell etablierten Methoden des Modelldatenmanagement am Beispiel eines Entwicklungsprozesses der AUDI AG vorgestellt. Das sich dieses Bild auch auf andere OEMs übertragen lässt, zeigen die gemeinsamen Aktivitäten der Hersteller, insbesondere für den Bereich der Schnittstellen und einheitlichen Architektur in der modellbasierten Funktionsentwicklung. Für das Thema Schnittstellen wird der Standard des Functional Mockup Interface (FMI) [BO12] vorangetrieben. Dieser ist bereits in Version 3 in Entwicklung. Aufgrund fehlender Toolunterstützung und notwendiger Anpassungen in den Entwicklungsprozessen konnte er sich noch nicht durchsetzen. Im Bereich der Architektur, insbesondere der Steuergerätearchitektur, zeichnet sich hier ein anderes Bild. Mit AUTOSAR [KF09] hat ein Konsortium, an dem sich alle großen OEMs beteiligen, hier einen Standard mit breiter tooltechnischer Unterstützung entwickelt. Dabei gibt der Standard die Architektur der Steuergeräte vor und liefert eine Basisinfrastruktur, die die Entwicklung und den Einsatz von Funktionssoftware von der Hardware des Steuergeräts abstrahiert und damit die Softwareentwicklung von der Hardware entkoppelt. Für weiterführende Informationen zur Steuergeräteentwicklung und -absicherung mit AUTOSAR sei auf [DHM12] und [EMH14] verwiesen.

Auch im Bereich der HIL-Simulation existieren ähnliche Standardisierungsgremien. Mit dem HIL-Bus [BS14] wurde eine Kommunikationsmethode für HIL-Prüfstände unabhängig vom jeweiligen Hersteller des Prüfstands entwickelt. Dies ermöglicht es Komponenten-HILs, die für einzelne Steuergeräteabsicherungen ausgelegt sind, zu komplexeren, so genannte Verbund-HILs zu vernetzen und damit Integrationstests über verschiedene Steuergeräte hinweg durchzuführen.

Die oben aufgezeigten Herausforderungen wurden bereits frühzeitig von den OEMs erkannt und im Rahmen von Prozessoptimierungsprojekten angegangen. Bei der AUDI AG existieren hierbei aktuell nach Entwicklungsdomäne und Simulationsdisziplin getrennte Standardisierungsprojekte, die es sich zum Ziel gesetzt haben, zunächst in den Einzeldomänen eine Durchgängigkeit von Daten und Informationen zu ermöglichen. Für die Softwareeigenentwicklung schafft das Projekt EnProVe [KB13] die Grundlagen, indem Prozess- und Methodenvorgaben die entwickelnden Fachabteilungen unterstützen und dadurch die gemeinsame, vernetzte Entwicklung der Funktionen ermöglichen. Im Bereich MIL wurde durch das Projekt MSB (Modularer Simulationsbaukasten) [BSMG12] für die Simulationen Thermomanagement und Fahrleistung/Verbrauch ein durchgängiger Prozess zur Vernetzung der beteiligten Entwicklungsdomänen geschaffen. Dieser erlaubt es Schnittstellen von Modellen abstrakt, in Form von XML-Repräsentationen, zu dokumentieren und über entsprechende Toolunterstützung die Integration entsprechender Modelle zu unterstützen. Auch im Bereich HIL wurden im Rahmen des Projekts XIL-Datenmanagement Benennungsvorschriften für Schnittstellen entwickelt, die es ermöglichen Schnittstellenbezeichner auf Basis der von der jeweiligen Schnittstelle angebundenen Daten, eindeutige Bezeichner zu erzeugen, die in den beteiligten Fachabteilungen unabhängig voneinander stets zu gleichen Bezeichnern führen (vgl. [GTH13]).

Bei Volkswagen PKW wurde im Jahr 2008 die Integrationsplattform MeMo [Bre09] für die modellbasierte Entwicklung im Bereich Mechatronik entwickelt und produktiv eingeführt. Das System bietet eine datenbankgestützt Verwaltung und Dokumentation von in Subversion verwalteten Entwicklungsartefakten. MeMo erlaubt dabei hierarchische Strukturierung

sowie die Steuerung der nutzerbezogenen Rechte der Entwicklungsartefakte. Trotz der Auslegung als Integrationsplattform mit dem Ziel der zentralen Verwaltung von Entwicklungsartefakten in der modellbasierten Entwicklung konnte sich MeMo nie über eine Individuallösung hinaus entwickeln. Die Hauptgründe werden von den aktuellen Nutzern unter anderem in der notwendigen Anpassung der Entwicklungsprozesse, der Notwendigkeit einer Migration bestehender Datenbestände sowie dem damit verknüpften Aufwand einer formalen Dokumentation beziehungsweise Nachdokumentation gesehen.

Aktuell existieren alle beschriebenen Prozessoptimierungen überwiegend unabhängig nebeneinander. Es fehlt eine zentrale Plattform, die es ermöglicht Entwicklungsartefakte über Domänen und Disziplinen hinweg einheitlich auszutauschen. Die Entwicklung einer solchen Plattform ist Aufgabe des Projekts XIL-Datenmanagement (vgl. [GT12]). Eine Marktanalyse hinsichtlich einer Datenmanagementlösung, die vorhandene Prozesse abbilden und Entwicklungsartefakte strukturiert verwalten kann, um diese anschließend für eine Wiederverwendung zur Verfügung zu stellen, ergab, dass eine hinreichend geeignete Lösung aktuell nicht existiert.

Erste Ansätze, die beschriebene Plattform durch ein Softwareprodukt zu realisieren, finden sich bei der Firma dSPACE [dG14a], die mit dem Produkt SYNECT eine Datenmanagementplattform für die Funktionsentwicklung anstrebt. Der Lösungsansatz konzentriert sich dabei aktuell auf den Bereich HIL, was darauf zurück zu führen ist, dass dSPACE marktführender Lieferant von HIL-Prüfständen und zugehörigen Softwarelösungen ist. Vielversprechend ist der Ansatz eines baumbasierten Variantenmanagements [dG14b] für Simulationsmodelle, das es erlaubt eine Ordnungsstruktur auf Basis der modellierten Inhalte bzw. des Einsatzzwecks zu definieren.

4.3 Modelldatenmanagement in anderen Industriezweigen

Die modellbasierte Entwicklung, wie sie in der Automobilindustrie Anwendung findet, ist am ehesten vergleichbar mit den Bereichen der Soft- und Hardwareentwicklung, die wie in Abschnitt 3.1 ausgeführt wurde, auch Ursprung dieser Entwicklungmethodik waren und damit auch die weiteste Entwicklung des Datenmanagements aufweisen. Aus diesem Grund werden im folgenden die grundlegenden Technologien dieses Wissenszweigs dargestellt.

4.3.1 Software

In der Softwarentwicklung war die Standardisierung von UML (vgl. Abschnitt 3.1) Grundlage für die Verbreitung der modellbasierten Entwicklung. Die modellbasierte Softwareentwicklung verfolgt das Ziel ein Softwaresystem auf Basis formaler Modelle, beispielsweise modelliert mittels UML, zu beschreiben und anschließend durch generative Methoden in Programmcode zu übersetzen. In der Regel erlauben die genutzten Modellierungssprachen die Beschreibung der Softwaresysteme auf unterschiedlichen Granularitätsebenen. Diese reichen von der Softwarearchitektur bis hin zur Modellierung der notwendigen Algorithmen. Mittels sogenannter Codegeneratoren wird auf Basis der Modelle zunächst ein Softwarecode in einer vom Codegenerator abhängigen Programmiersprache erzeugt. Verbreitet

sind hier C oder Java. Anschließend folgt in der Regel der Kompilationsprozess des Softwarecodes, wie er bei der klassischen Handkodierung Anwendung findet. Abbildung 4.3 stellt das beschriebene Vorgehensmodell noch einmal zusammenfassend dar.

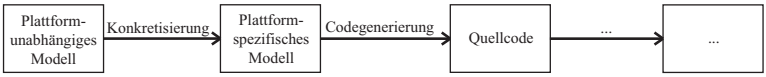


Abbildung 4.3: Vorgehensmodell der modellbasierten Softwarentwicklung

Bezüglich des aktuellen Standes der Technik im Bereich Modelldatenmanagement soll hier das Connected Data Objects (CDO) Model Repository [Rep15] als Beispiel für das Management von EMF-Modellen (Eclipse Modelling Framework) und zugehörigen Daten dienen. Doch bevor näher auf die Architektur des Repositories eingegangen werden kann, ist ein grundlegendes Verständnis für das Eclipse Modelling Framework [Ste09] notwendig. EMF ist die Open Source Implementierung der Meta Object Facility (MOF) Spezifikation der Object Management Group [Fac14]. Dabei handelt es sich um ein so genanntes Meta-Meta-Modell, als Meta-Modell zur Spezifikation von neuer Meta-Modellen genutzt wird. Wie in Abschnitt 3.1 erläutert, ist es notwendig Modelle unabhängig vom jeweiligen geistigen Modell eines Menschen in einer gemeingültigen Form zu repräsentieren. UML stellt eines dieser Meta-Modelle dar, das diese Aufgabe realisiert. Davon ausgehend repräsentiert MOF eine Metadaten-Architektur , die dazu geeignet ist Meta-Modelle, u.a. UML, zu spezifizieren. Als wichtigstes Mittel der Wahl zur Beschreibung dieser Meta-Meta-Modelle ist das XMI-Format (XML Metadata Interchange) [Ste09] zu nennen, dass auch die Basis für EMF darstellt. Für weiterführende Informationen zum MOF Metamodell sei auf [Ste09] verwiesen.

Wie beschrieben, ist EMF die Implementierung von MOF für die Eclipse Plattform. Das CDO Model Repository erlaubt es mittels EMF spezifizierte Meta-Modelle als Datenstrukturen einer Datenbanklösung zu definieren. Es bildet dabei eine Persistenzschicht [Rep15], die zum Client hin Operationen zur Speicherung von vollständigen Modellen auf Basis der spezifizierten Meta-Modelle zur Verfügung stellt und diese zum Datenbankserver hin wahlweise auf verschiedene Datenbanktechnologien, wie relationale [Kre01] oder graphbasierte [Hun14] Datenbanken, mappt.

Abbildung 4.4 zeigt die Architektur des CDO-Servers. Es ist zu erkennen, dass zum Client hin eine Reihe von Komponenten zur Verfügung stehen, die eine Nutzung des Repositories in verschiedensten Zugriffsszenarien erlauben. Zunächst stehen Basisoperationen zur Verfügung, die Modelle vergleichbar zu dateibasierten Repositories wie Subversion (vgl. Abschnitt 4.2) verwalten. So können diese in mehreren Versionen durch explizite Commit-Opterationen gespeichert, für parallele Bearbeitung zugelassen oder gesperrt (Lock-Mechanismen), sowie in verschiedene Entwicklungspfade gebrancht werden. Zusätzlich können Modelle als des Client in Form von Sessions persistent gehalten werden, so dass jede Änderung des Modells in der Clientanwendung direkt auf das Modell der Datenbank propagiert wird. Darüber hinaus erlaubt es CDO mittels einer zu SQL [Kre01] vergleichbaren Abfragesprache die Modelle zu durchsuchen. Grundlage dafür ist ein weiteres Open Source Projekt aus dem Eclipseumfeld mit dem Namen Model Query [Rep15].

Wie in Abschnitt 3.4 ausgeführt, gehören zum Modellmanagement neben dem reinen Modell auch beschreibende Informationen, wie Meta-Daten oder Abhängigkeiten. Der Einsatz

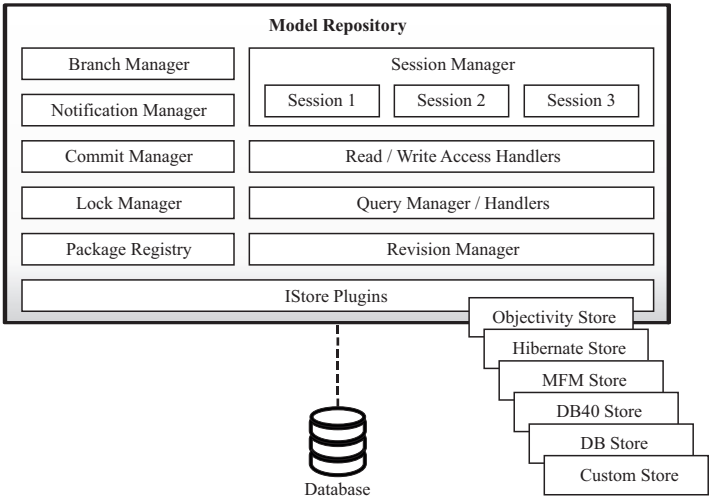


Abbildung 4.4: CDO-Server Architektur [Rep15]

MOF-basierter Meta-Modelle zur Verwaltung von Modellen erlaubt es, je nach Anwendungsfall, über die reinen Modellstrukturen hinaus Attribute und Beziehungen zwischen Modellen zu modellieren und damit auch im Repository zu hinterlegen.

Die Generierung dieser Meta-Informationen ist dabei in der Regel Teil des Entwicklungsprozesses. Im Bereich der Softwareentwicklung haben sich dafür Standardmethoden, wie Quellcode-Annotationen mit JavaDoc [Sch14] oder auch UML-Diagramme etabliert. Durch diese strukturierte Repräsentation ist bereits heute eine Unterstützung beim Suchen und bei Bewertungen von Quellcodes möglich. Der Grad an tatsächlicher Dokumentation ist in der Softwareentwicklung natürlich immer vom Entwicklungsprozess und seinen Vorgaben abhängig.

4.3.2 Hardware

Nachdem der Stand der Technik im Bereich der Softwareentwicklung exemplarisch dargestellt wurde, wird im Folgenden auf die Hardwareentwicklung eingegangen. Dafür werden heute vorwiegend so genannte Hardwarebeschreibungssprachen (HDL, Hardware Description Language) verwendet, die eine zielplattformunabhängige Beschreibung von Hardware ermöglichen. Ein wichtiger Vertreter ist hier beispielsweise VHDL (Very High Speed Integrated Circuit Hardware Description Language). Die HDL ermöglicht es, ähnlich einer Programmiersprache, digitale Systeme textbasiert zu beschreiben. Vergleichbar zu Modellen erlaubt VHDL Entwicklung auf unterschiedlichen Abstraktionsebenen. So können Verhaltensmodelle die Hardware auf logischer Ebene beschreiben und damit unabhängig von konkreten Halbleiterbauelementen eine gewisse Funktionalität realisieren. Andererseits ist

es möglich durch Strukturbeschreibungen konkrete Schaltungen durch VHDL zu beschreiben. Durch den modularen Aufbau von VHDL können die dadurch erzeugten Hardwarebeschreibungen in größeren und unterschiedlich komplexen Systemen integriert werden, was zur Wiederverwendung und in diesem Zusammenhang auch zum Handel mit Hardwarebeschreibungen führte. Eine Harwarebeschreibung wird dabei als Intellectual Property (IP) bzw. IP-Core bezeichnet. Die heute wohl bekannteste IP-Hardwarebeschreibung auf dem Markt ist der ARM-Kern [Ltd14] bzw. die ARM-Architektur. Diese Mircoprozessorarchitektur wird von der gleichnamigen Firma rein als Hardwarebeschreibung entwickelt und an Mircoprozessorhersteller zur Herstellung der physischen Hardware lizenziert.

Abgesehen von diesem konkreten IP-Core, dessen Architektur sehr gut und ausführlich dokumentiert ist, existieren verschiedenen Plattformen, die IP-Cores für unterschiedlichste Hardware vertreiben. Vergleichbar mit der Herausforderung des Modellaustauschs in der automobilen Funktionsentwicklung, stellt sich hier die Frage, welche Informationen notwendig sind, um diese hochkomplexen Hardwarebeschreibungen für potenzielle Einsetzer so verständlich zu dokumentieren, dass dieser in der Lage ist zu entscheiden, ob er den IP-Core lizenzieren möchte oder nicht.

Hier setzt das Projekt IPQ - IP Qualification for Efficient Design Reuse - an, das es sich als Ziel gesetzt hat, eine standardisierte Form der Qualifikation von IP-Cores zu erarbeiten. Das Projekt wurde vom Bundesministerium für Bildung und Forschung (BMBF) und dem europäischen Medea Projekt ToolIP [Voe03] - Tools and Methods for IP - gefördert. Ziel war dabei neben der Unterstützung von IP-Entwicklungsingeneuren mit Methoden und Tools zur IP Qualifikation, auch den Austausch von IP über organisatorische Grenzen hinweg zu ermöglichen. [Har02]

Ausgangsbasis für eine standardisierte Qualifikation, die im weiteren Sinne als Dokumentation oder Wissensbasis bezeichnet werden kann, sind hier so genannte Organizational Memory Information Systems (OMIS), die aus dem Forschungsbereich Wissensmanagement stammen. OMIS ermöglichen es, intelligente Informationssysteme zu entwickeln, die die Erstellung, Akkumulation, gemeinsame Nutzung, Wiederverwendung und Weiterentwicklung von explizitem Wissen einer Organisation unterstützten [ABH+98].

Ein OMIS setzt sich, wie in Abbildung 4.5 dargestellt, zusammen aus einer expliziten Abbildung des Domänenwissens in Form einer Ontologie (vgl. Abschnitt 4.1.2), den mit der Ontologie zu beschreiben Inhalten sowie einer Abbildung des Wissens auf die Inhalte. Das Forschungsprojekt IPQ setzte dieses generische Konzept, wie ebenfalls in der Abbildung 4.5 dargestellt, auf die Dokumenation von IP-Cores um. Dabei repräsentiert eine Ontologie das Domänenwissen zur Qualifikation von IPs im Rahmen einer definierten Institution beispielsweise einer Forschungseinrichtung. Dieses Wissen wird unter Verwendung der im Projekt entwickelten IP Characterization Language (IPCHL) [Har02] als Eigenschaften, so genannte IP Propterties, den IP-Cores zugewiesen. Damit ist der erste Schritt, die Dokumentation innerhalb einer Organisationseinheit abgeschlossen.

Das Projekt geht jedoch weiter und definiert mittels IPCHL einen Sprachraum zum Austausch von IP-Dokumentationen über verschiedene Domänen und damit Organisationseinheiten hinweg. Ermöglicht wird dies durch ein Transferformat, das die Verknüpfung des Wissens der jeweiligen Ontologien ermöglicht. Im Ergebnis spezifiziert IPQ ein so genanntes IPQ-Format, dass neben der mittels IPCHL beschriebenen Dokumentation auch den IP-Content transportiert und damit den IP-Core und dessen zugeordnete Qualifikation transportiert.

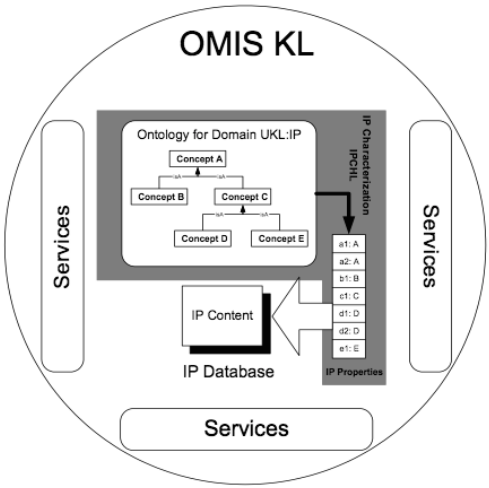


Abbildung 4.5: OMIS im Kontext Intellectual Property Qualification IPQ [Har02]

4.4 Kriterien für Messung, Bewertung und den Vergleich von Simulationsmodellen

4.4.1 Qualität von Algorithmen in der Informatik

Ein Algorithmus bezeichnet eine Menge von Regeln für ein Verfahren, um aus gegebenen Eingabegrößen bestimmte Ausgangsgrößen herzuleiten. Dabei müssen die Bedingungen Finitheit der Beschreibung, Effektivität im Sinne der Ausführbarkeit, Terminiertheit der Ausführung sowie Determiniertheit erfüllt sein. [Köc08] Diese allgemeine Definition trifft keine Aussagen darüber, ob und wann ein Algorithmus gut oder schlecht ist. Der Qualitätsbegriff (lat. qualitas - Beschaffenheit, Merkmal, Eigenschaft, Zustand), der dies übernimmt, ist in der Literatur unterschiedlich definiert. Die ISO 9000:2005 [Int09] definiert Qualität als „Grad, in dem ein Satz inhärenter Merkmale Anforderungen erfüllt". Grundlegend zu unterscheiden sind hier der neutrale Begriff, der die Summe aller Eigenschaften eines Objektes beschreibt sowie der bewertete Begriff, der die Güte aller Eigenschaften eines Objektes beschreibt.

Für die Qualität von Algorithmen wird in der Regel von bewertender Qualität ausgegangen, die die Erfüllung definierter Merkmale relativ zu einem Idealzustand bewertet. Die ISO-9126 spezifiziert die Qualitätsmerkmale von Softwaresystemen. Sie geht dabei von einem standardisierten Qualitätsmodell, das in Abbildung 4.6 schematisch dargestellt ist, aus. Das Maß für die Softwarequalität wird hierarchisch definiert. Zur Gesamtqualität tragen die so genannten Faktoren bei, die durch Kriterien weiter verfeinert werden können. Auf unterster Ebene werden diesen Kriterien Metriken zugewiesen, die den konkreten, messbaren

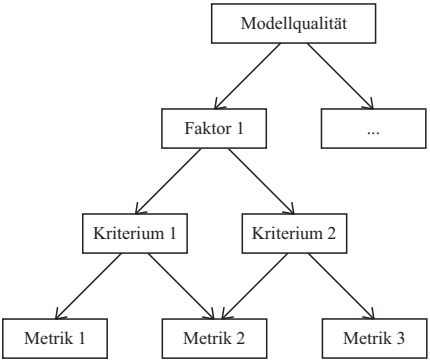


Abbildung 4.6: FCM-Modell zur Messung von Qualität

Wert repräsentieren, der das betrachtete Softwaresystem charakterisiert. Durch Aggregation der Ebenen in der Hierarchie nach oben, ergibt sich ein Gesamtmaß für die Qualität von Software. Das Qualitätsmodell wird in der Regel als FCM-Modell (F-Factor, C-Criteria, M-Metric) bezeichnet.

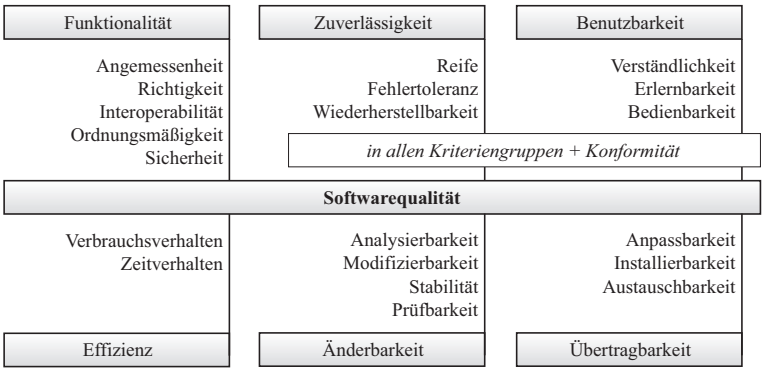


Abbildung 4.7: Softwarequalitätsmodell nach ISO-9126

Neben dem Qualitätsmodell spezifiziert die ISO-9126 darüber hinaus die wichtigsten Softwaremerkmale. Abbildung 4.7 zeigt die vorgeschlagene Instanziierung des beschriebenen Qualitätsmodells, die generisch für alle Softwaresysteme geeignet ist. Als wichtigste Merkmale (Faktoren) für die Softwarequalität gelten demnach:

- Funktionalität: Besitzt die Software die geforderten Funktionen? Erfüllen diese festgelegte Anforderungen?
- Zuverlässigkeit: Kann die Software ein definiertes Leistungsniveau unter definierten

Bedingungen über einen definierten Zeitraum aufrechterhalten?

- Benutzbarkeit: Welchen Aufwand fordert der Einsatz der Software von den Benutzern und wie wird er von diesen beurteilt?
- Effizienz: Wie ist das Verhältnis zwischen dem Leistungsniveau der Software und den genutzten Betriebsmitteln?
- Wartbarkeit: Welchen Aufwand erfordert die Durchführung vorgegebener Änderungen an der Software?
- Interoperabilität: Wie leicht lässt sich die Software in eine andere Umgebung übertragen?

Die spezifizierten Merkmale und ihre jeweiligen Kriterien geben dem Begriff Softwarequalität einen Rahmen, der aber in seiner konkreten Ausprägung vom betrachteten Softwaresystem abhängig ist. Bei Betrachtung der vorgeschlagenen Faktoren und Kriterien zeigt sich, dass der Großteil dieser Merkmale entweder nur subjektiv oder erst anhand zuvor spezifizierter Rahmenbedingungen messbar wird. Die Verständlichkeit oder Erlernbarkeit einer Software ist in Abhängigkeit von der Benutzergruppe oder sogar individuellen Fähigkeiten nur eingeschränkt greifbar. Die Reife und Fehlertoleranz einer Software kann nur durch individuelle Vorgaben für eine Software als Vergleich zu Referenzwerten gemessen werden. Aus diesem Grund spezifiziert die ISO-9126 keine konkreten Messverfahren oder Metriken der Kriterien. Das generische Qualitätsmodell muss für jeden Anwendungsfall individuell instanziiert werden, um die Qualität einer Software messen zu können.

Neben den erwähnten, eher subjektiv geprägten Merkmalen, existieren auch objektiv messbare Kriterien, die mit in der Informatik entwickelten Methoden bestimmt werden können. Diese Methoden werden unter dem Begriff Softwaremetriken zusammengefasst. Eine Softwaremetrik ist dabei eine Funktion, die eine Software-Einheit in einen Zahlenwert abbildet, welcher als Erfüllungsgrad einer Qualitätseigenschaft der Software-Einheit interpretierbar ist [Sch12]. Softwaremetriken werden dabei nach Zielgruppe und Anwendungsgebiet klassifiziert. Zielgruppen werden in der Regel nach Entwickler, Kunden und Management unterteilt. Anwendungsgebiete sind unter anderem Prozess-, Produkt-, Aufwands-, Projekt-, Komplexitäts- und Anwendungsmetriken [Sch12]. Zur Bewertung der Aussagefähigkeit einer Softwaremetrik für die jeweilige Zielgruppe und das Anwendungsgebiet werden Gütekriterien eingesetzt, die die Klassifikation von Metriken erlauben. Die wichtigsten Gütekriterien nach [Sch12] sind:

- Objektivität: Vermeidung subjektive Einflüsse durch den Messenden
- Zuverlässigkeit: Wiederholbarkeit der Messung
- Normierung: Messergebnis- / Vergleichsbarkeitsskala
- Ökonomie: Kosten der Messung
- Nützlichkeit: Erfüllung praktischer Bedürfnisse
- Validität: Schließen auf andere Kenngrößen

Neben dem vorgestellten Qualitätsmodell der ISO-9126 existieren weitere Qualitätsmodelle, wie das Modell von Cavano und McCall [CM78] oder das Modell von Boehm [BBL76]. Einen Überblick der Gemeinsamkeiten und Unterschiede dieser Herangehensweisen liefert [Sch12]. Abbildung 4.8 zeigt die jeweils betrachteten Faktoren des Qualitätsmodells im

Überblick. Zusammengefasst werden diese Modelle in der Standardsammlung ISO-25000 [Int14], die als SQuaRE (Software product Quality Requirements and Evaluation) bezeichnet wird und unter anderem die ISO-9126 integrierte.

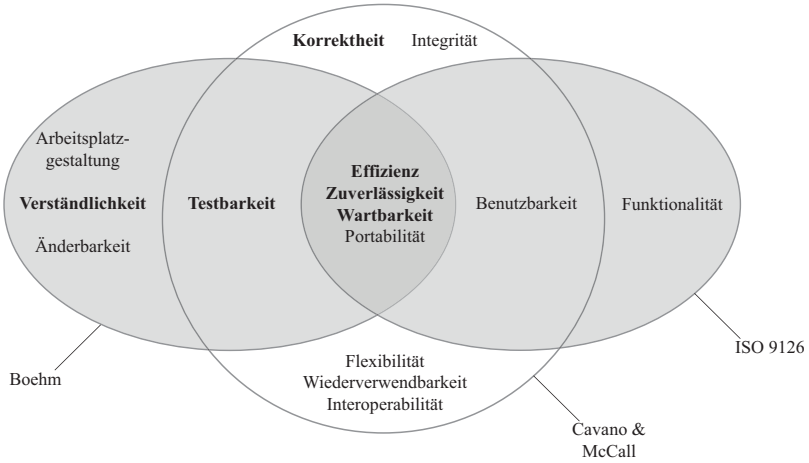


Abbildung 4.8: Vergleich der Qualitätsmodelle nach Scheible [Sch12]

Stellvertretend für Softwaremetriken und wegen ihrer Relevanz für das Verständnis der vorliegenden Arbeit, werden an dieser Stelle einige wichtige Metriken vorgestellt. Die „Lines of Code"-Metrik (kurz: LOC) misst die Anzahl an Codezeilen einer Software-Einheit, da diese einerseits ein Maß für die Komplexität der Software bezüglich Einarbeitung und Wartbarkeit darstellt und andererseits eine Referenz für die Anzahl der Rechenoperationen und damit auch der Laufzeit darstellt. Die Metrik LOC erlaubt keine Aussagen über die Effizienz der Programmierung oder der Entwickler, da die Anzahl der Zeilen sehr stark von Entwicklungsentscheidungen, wie der Softwarearchitektur, der gewählten Lösung, der Formatierung des Quellcodes, der Programmiersprache und den Fähigkeiten des Programmierers abhängig sind. Dennoch stellt diese Metrik ein Maß für die Produktivität dar, da es einen Zusammenhang zwischen den durchschnittlichen Codezeilen pro Mitarbeiter und Tag gibt, so dass im individuellen Qualitätsmodell diese Maßzahl als Relation zwischen Ist- und Soll-Werten im Entwicklungsprojekt bestimmt werden kann.

Eine weitere Softwaremetrik ist die McCabe-Metrik [McC76], die nach ihrem Entwickler Thomas J. McCabe benannt wurde. Sie berechnet die so genannte zyklomatische Komplexität einer Software-Einheit. Die Idee hinter dieser Metrik ist, dass eine Software-Einheit ab einer bestimmten Komplexität nicht mehr begreifbar für den Menschen ist. McCabe definiert die zyklomatische Komplexität einer Software-Einheit als die Anzahl linear unabhängiger Pfade des Kontrollflussgraphen. Für die Bewertung der Korrektheit einer Software-Einheit ist es notwendig, alle linear unabhängigen Pfade zu durchlaufen und zu testen. Die zyklomatische Komplexität stellt damit eine oberer Grenze für die Anzahl notwendiger Testfälle dar, um eine vollständige Zweigüberdeckung des Kontrollflussgraphen zu erreichen. Laut

McCabe soll für in sich geschlossene Software-Einheiten eine zyklomatische Komplexität von 10 nicht überschritten werden, um diese für den Menschen verständlich zu halten.

Die Halstead-Metrik vertieft den Begriff der Softwarekomplexität, da sie die genutzten Befehle einer Software-Einheit direkt analysiert. Es handelt sich dabei um ein so genanntes statisch-analytisches Verfahren, das gezielt Informationen über den Softwarecode sammelt, ohne diesen zur Ausführung zu bringen. Die Metrik geht von der Annahme aus, dass alle Programmteile aus Operanden (z.B. Variablen und Konstanten) und Operatoren (z.B. Schlüsselwörter, logische oder Vergleichsoperatoren) aufgebaut sind. Anschließend wird durch Codeanalyse die Implementierungslänge (N) als Anzahl insgesamt verwendeter Operatoren (N_1) und Operanden (N_2) , sowie die Vokabulargröße (η) als Anzahl unterschiedlicher verwendeter Operatoren (η_1) und Operanden (η_2) bestimmt. Auf Basis dieser Maßzahlen lassen sich folgende Kennzahlen berechnen [Sch12]:

- Schwierigkeit den Softwarecode zu schreiben oder zu verstehen: $D = \frac{\eta_1}{2} \times \frac{N_2}{\eta_2}$
- Aufwand: $E = D \times V$, mit dem Halstead-Volumen $V = N \cdot \log_2 \eta$
- Implementierungszeit: $T = \frac{E}{18}\,[s]$

Auf Basis bekannter Metriken werden immer wieder neue Metriken entwickelt. So existieren neben der Pfadüberdeckung der McCabe-Metrik heute weitere kontrollflussorientierte Metriken, wie beispielsweise die Bedingungsüberdeckung, Anweisungsüberdeckung und Zweigüberdeckung. Für weiterführende Informationen sei an dieser Stelle auf [BE08] und [SL07] verwiesen.

4.4.2 Messbarkeit der Qualität von Modellen

Mit der Zielstellung der vorliegenden Arbeit, eine weitestgehend automatisierte Dokumentation von Entwicklungsartefakten der modellbasierten Entwicklung durchzuführen, stellt sich die Frage nach der Bewertung qualitativer Eigenschaften der Simulationsmodelle. Mit der Konzentration auf die X-in-the-Loop-Simulation sind damit insbesondere Simulinkmodell von Bedeutung. Wie aus Abschnitt 4.4.1 hervorgeht, ist es dafür notwendig ein Qualitätsmodell für Simulinkmodelle zu definieren.

Die Qualitätssicherung von Simulinkmodellen ist auf unterschiedlichen Ebenen bereits heute in Entwicklungsprozessen etabliert. Bei den eingesetzten Methoden kommen sowohl manuelle, als auch automatisierte Messverfahren zum Einsatz, um bestimmte Eigenschaften von Modellen zu überprüfen. Die heute eingesetzten Methoden lassen sich dabei, wie folgt klassifizieren:

- **Modellierungsrichtlinien:** Ein Regelwerk schreibt für die Modellierung einzuhaltende Rahmenbedingungen (vgl.[CDF+05, SDP08]) vor. Diese Regeln können beispielsweise Farbschemata für Blöcke oder die Platzierung von Ein- und Ausgangsblöcken im Modell betreffen. Durch formale Beschreibung der Regeln ist es heute möglich die Einhaltung mit Toolunterstützung, wie MXAM der Firma MES [SDP08], zu überprüfen und teilweise automatisiert zu beheben.
- **Modellmetriken:** Vergleichbar mit Softwaremetriken (vgl. Abschnitt 4.4.1) bilden Modellmetriken Eigenschaften von Simulinkmodellen auf Messwerte ab. Tatsächlich zum Einsatz kommen hier jedoch nur sehr wenige Metriken, wie beispielsweise die Anzahl der Blöcke oder Subsysteme in einem Simulinkmodell [Rau01].

- **Komplexität:** Für die Berechnung der Komplexität von Modellen wurde ebenfalls von der Firma MES [SPR10] eine angepasste Variante der Halstead-Metrik (vgl. Abschnitt 4.4.1) in Form des Tools M-XRAY entwickelt.

- **Modelchecking:** Modelchecking bezeichnet die Definition und Überprüfung von Bedingungen an Simulinkmodelle. Dies könnten Wertebereiche von Signalen oder die Erreichbarkeit von Zuständen sein. Diese Überprüfung kann automatisiert, beispielsweise mit den in Simulink® verfügbaren Mitteln in Form von Testbenches, durchgeführt werden - allerdings müssen die Bedingungen zunächst von Hand definiert werden.

- **Modell-Reviews:** Modell-Reviews bezeichnen in der Regel die manuelle, strukturierte Durchsicht von Modellen. Dabei werden Protokolle erstellt, die Auffälligkeiten im Modell, wie Verstöße gegen Modellierungsrichtlinien oder Probleme bei der Verständlichkeit, sowie nicht oder falsch umgesetzte Anforderungen dokumentieren [CDF+05].

- **Tests:** Beim Testen werden für Simulinkmodelle individuell notwendige Testfälle zur Absicherung des gewünschten Verhaltens eingesetzt. Dabei werden auf Basis der Anforderungen an das Modell, Testspezifikationen erstellt und anschließend in speziellen Testsystemen implementiert. Während diese Implementierung manuell erfolgen muss, wird die anschließende Ausführung der Testfälle heute in der Regel durch Testautomatisierungslösungen unterstützt. [CFDY99, CDSS02, CFS05].

Nachdem der aktuelle Stand bei der Bewertung der Qualität von Modellen im praktischen Entwicklungseinsatz betrachtet wurde, wird im Folgenden der Stand der Forschung dargelegt. Einen umfassenden Ansatz für ein Qualitätsmodell für Simulinkmodelle liefert Scheible in [Sch12]. Ausgehend vom Qualitätsmodell der Softwareentwicklung nach ISO-25000 [Int14] und bereits vorhandenen Metriken, wie sie vorstehend beschrieben wurden, überträgt er die dort etablierten Definitionen und Methoden in den Kontext der modellbasierten Entwicklung.

Abbildung 4.9 zeigt den schematischen Aufbau des Qualitätsmodells nach [Sch12]. Die Ähnlichkeit zum vorgestellten Ansatz der Softwareentwicklung (vgl. Abbildung 4.7) ist dabei klar erkennbar. Faktoren beschreiben in diesem Fall die gewünschten Eigenschaften für ein qualitativ hochwertiges Simulinkmodell. Kriterien sind Indikatoren für diese Eigenschaften. Metriken messen wiederum die Erfüllung der Kriterien. Das Qualitätsmodell definiert dabei neben der Empfehlung von Faktoren und Kriterien auch Metriken, wie diese Kriterien messbar sind.

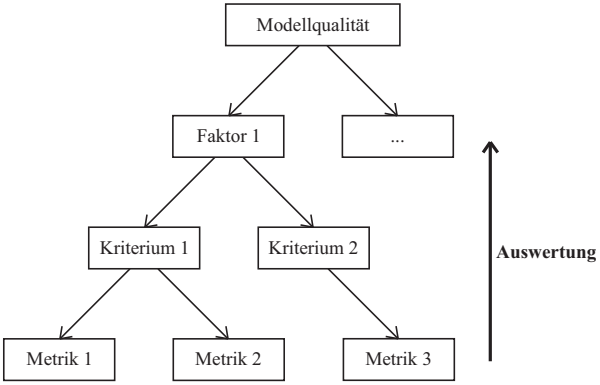


Abbildung 4.9: Schematischer Aufbau des Qualitätsmodells nach Scheible [Sch12]

Da Qualität in der Entwicklung stets ein relatives Maß darstellt (vgl. Abschnitt 4.4.1), schlägt das Qualitätsmodell ebenfalls Methoden für den Aufbau eines Referenzmodells für Vergleichswerte und damit ein Mittel der Normierung für die gemessenen Werte vor. Somit schließt der Ansatz den Kreis bis hin zu einer standardisierten Bewertung von Simulinkmodellen und ist damit ein umfassendes Qualitätsmodell für Simulinkmodelle.

4.5 Suchen in strukturierten und unstrukturierten Daten

Die vorliegende Arbeit grenzt an die Forschungsgebiete Wissensmanagement (vgl. Abschnitt 4.1), Informationsextraktion, sowie Suchverfahren in Daten unterschiedlicher Qualitäten an. Darüber hinaus beschreibt das im Rahmen der Arbeit entwickelte Konzept eine durchgängige Methode, wie die gewonnenen Informationen und das darauf abgebildete Wissen bei der Suche nach und bei der Wiederverwendung von Simulationsmodellen genutzt werden kann. Aus diesem Grund befassen sich die folgenden Abschnitte mit dem aktuellen Stand der Wissenschaft für die Suche in strukturierten und unstrukturierten Daten. Die Analyse des in den Abschnitten 4.5.1 und 4.5.2 vorgestellten Standes der Technik wurde in Zusammenarbeit mit dem Forschungsprojekt „FIND - Flexible Daten- und Informationsvernetzung" [FINte] erarbeitet, welches unter Beteiligung der AUDI AG seit 2012 am Forschungsinstitut „Das virtuelle Fahrzeug" in Graz angesiedelt ist.

4.5.1 Einführung grundlegender Konzepte

Die Suche in einem definierten Datenbestand muss sich stets an dessen Struktur orientieren. Man unterscheidet dabei so genannte Strukturgrade nach strukturierten, semi-strukturierten und unstrukturierten Daten.

- **Unstrukturierte Daten** haben kein festes Schema. Sie bestehen aus beliebigen Freiformtexten. Beispiele sind E-Mails, Webseiten oder Textdokumente.

- **Strukturierte Daten** haben eine strikte Form. Sie folgen einem Schema, das den Aufbau und die Formatierung der Daten vorschreibt. Beispiele sind XML-Schemata oder Datenbankeinträge.
- **Semi-strukturierte Daten** sind eine Mischung aus unstrukturierten und strukturierten Daten. Sie enthalten in der Regel strukturierte Meta-Daten oder Ordnungsstrukturen, die Informationen über den Inhalt der Daten liefern. Die eigentlichen Daten hingegen sind zumeist Freiformtexte. Ein Beispiel wäre die Sortierung von Dateien in einer hierarchischen Ordnerstruktur, wobei die Strukturierungsebenen und die Ordnernamen eine Ordnung auf den Daten definieren.

Software und die dahinter liegenden Algorithmen, die die Suche in einem Datenbestand durchführen, werden unter dem Begriff Suchmaschinen zusammengefasst. Dabei unterscheiden sich die konkreten Suchmaschinen stark in ihrer Leistungsfähigkeit und müssen daher für den jeweiligen Anwendungsfall gewählt werden. Daten, die von einer Suchmaschine verarbeitet werden, werden als Artefakte bezeichnet. Die Ablageorte dieser Artefakte werden als Quellen bezeichnet. Quellen können z.B. Dateisysteme, Datenbanken aber auch Wikis sein. Mehrere Quellen unterschiedlichen Typs werden als heterogene Quellen bezeichnet und von dafür geeigneten Suchmaschinen in der Regel über so genannte Konnektoren angebunden.

Nach der Anbindung einer Suchmaschine folgt die Phase der Informationsextraktion. Dabei werden relevante Daten innerhalb der Artefakte gesucht und extrahiert. Hierbei werden verschiedene Verfahren der Extraktion unterschieden, die wiederum vom Quellentyp abhängig sind.

- Die **unstrukturierte Extraktion** wird für unstrukturierte Daten eingesetzt. Die Artefakte werden dafür aus ihrem Ursprungsformat, z.B. einem Office Dokument oder PDF, in Volltext konvertiert. Anschließend werden heute in der Regel statistische Modell (so genannte lernenden Verfahren), Regelwerke oder Nachschlagelisten, wie z.B. Thesauri (vgl. Abschnitt 4.1.4) genutzt, um relevante Daten zu identifizieren.
- Die **strukturierte Extraktion** wird für strukturierte Daten eingesetzt. Die bereits vorhandene Struktur wird dabei in die für die Suchmaschine standardisierte Form konvertiert. Dieser Vorgang wird auch als Normalisierung bezeichnet.

Zu den relevanten Daten zählen neben anwendungsfallspezifischen Daten, in der Regel Orte, Personen, Organisationen, Datumsangaben, Parameter oder Autoren. Darüber hinaus können Daten aus den Quellen der Artefakte relevant sein - z.B. der Quellenname, Modifikationsdatum oder die Historie der Daten. Die extrahierten Daten werden auch als Facetten bezeichnet.

Der Phase der Informationsextraktion folgt die Informationsvernetzung. Dabei werden die Artefakte auf Basis ihrer Facetten miteinander in Verbindung gebracht. Gemeinsamkeiten von Artefakten, wie Autor, Quelle oder ähnliche Inhalte des Volltexts, werden dabei identifiziert und als neue Facetten zu diesen hinzugefügt. Eine Informationsvernetzung ist jedoch nur möglich, wenn die Facetten über alle Artefakttypen und Quellen definiert werden und normalisiert sind, um eine Vergleichbarkeit zu erreichen.

Nachdem alle erkennbaren Facetten extrahiert wurden, folgt die so genannte Indizierung. Dabei werden die Facetten und gesammelte Informationen aus den Volltexten in einen Index eingepflegt, der als Ablagestruktur für Suchmaschinen dient. Dieser kann als eine Art

Inhaltsverzeichnis aufgefasst werden, das die gesammelten Informationen für die Suche optimiert abspeichert. Die Leistungsfähigkeit einer Suchmaschine ist dabei direkt von genutzten Index abhängig. Dies soll anhand eines Beispiels verdeutlicht werden. Betrachtet man die Facetten als Schlüsselwörter und speichert sie in einer Liste der Länge n ohne eine bestimmte Ordnung darauf zu definieren, so müssen maximal alle n Elemente der Liste durchlaufen werden, um ein passendes Element zu finden. Für die Laufzeitkomplexität bedeutet dies eine Laufzeit von $O(n)$. Schon durch eine alphabetische Sortierung der Liste bei der Erstellung ergibt sich die Möglichkeit eine binäre Suche zu verwenden, die mit jedem Suchschritt die Hälfte der noch vorhandenen Elemente ausschließt. Es ergibt sich eine Laufzeit von $O(\log n)$. Heutige Indizes sind jedoch wesentlich komplexer aufgebaut. In der Informatik wird hierbei einheitlich der Index des Apache Lucene Projektes als Stand der Technik bei frei verfügbaren Indizes betrachtet, der in Abschnitt 4.5.3 noch näher vorgestellt wird.

Nach Abschluss der Indizierung kann die Suchmaschine zur Suche genutzt werden. Dabei kann der Nutzer je nach Suchmaschine über Facetten aber auch Volltexte suchen. Hierbei werden verschiedenen Arten der Suche unterschieden. Neben der Schlagwortsuche, bei der eine Zeichenkette als Suchbegriff dient, wären hier die boolesche Suche sowie die Facettierung zu nennen. Bei der booleschen Suche formuliert die Sucheingabe eine logische Gleichung, deren Erfüllung in der Suche überprüft wird. Ein Beispiel wäre die Frage, ob ein Eintrag zum Audi A3 existiert. Die Facettierung beschreibt die Eingrenzung der Suche entlang von Facetten im Index, die explizit ausgewählt werden. Im Gegensatz zu einer Schlagwortsuche, führt die Facettierung stets zu den Ergebnissen, die der ausgewählten Facette im Suchindex zugeordnet sind. Neben diesen klassischen Sucharten, werden heute häufig unterstützende Funktionen bei der Suche eingesetzt. Dabei werden beispielsweise Suchvorschläge bei der Eingabe einer Zeichenkette der Schlagwortsuche angezeigt oder der Suchbegriff auf Basis des Index mit einer „Meinten Sie"-Empfehlung verbessert. Darüber hinaus werden beispielsweise mit den Suchergebnissen verwandte Schlagworte vorgeschlagen oder ausgewertet und als zu den Suchergebnissen passend mit in den Suchergebnissen zurückgegeben.

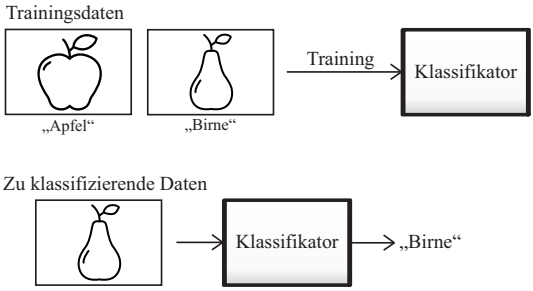


Abbildung 4.10: Beispiel für überwachte, lernende Suchverfahren [FINte]

Neben diesen *nutzerzentrieten Maßnahmen* zur Verbesserung der Suchergebnisse werden heute zunehmend so genannte lernende Verfahren eingesetzt, die unabhängig vom individuellen Nutzer eine Qualitätssteigerung anstreben. Sie werden dabei beispielsweise für die In-

formationsextraktion, die Ergebnisbewertung (das so genannte Ranking), sowie die oben beschriebenen Unterstützungsverfahren eingesetzt. Lernende Verfahren werden nach überwachtem und unüberwachtem Lernen unterschieden. Beim *überwachten Lernen* werden manuell aufbereitete Trainingsdaten eingesetzt, die mithilfe eines Trainingsalgorithmus ein Modell herleiten auf Basis dessen die Suchmaschine neue Beispiele klassifiziert. Abbildung 4.10 verdeutlich dies am klassischen Beispiel des Vergleichs von Äpfeln und Birnen. Überwachte Verfahren erreichen dabei im modernen Systemen eine Genauigkeit von bis zu 96% [FINte]. Dem gegenüber steht der Aufwand in ausreichendem Umfang (in der Regel mehrere tausend Datensätze) Trainingsdaten manuell zu erstellen. Zur weiteren Verbesserung der Resultate, sowie der Verringerung notwendiger Trainingsdaten wird Relevanzfeedback eingesetzt. Dabei werden beispielsweise die in den Suchergebnissen ausgewählten Einträge registriert oder aktiv eine Bewertung nach „relevant" oder „nicht relevant" vom Benutzer gefordert. Diese Unterstützung führt einerseits zu einer Verbesserung der Suchergebnisse. Andererseits können die nutzerzentrierten Angaben auch zu fehlerhaften Annahmen führen, wenn sich beispielsweise der Fokus des Benutzers ändert.

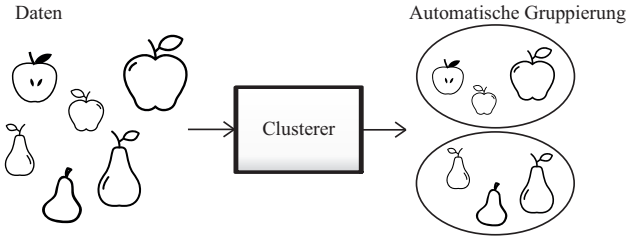


Abbildung 4.11: Beispiel für unüberwachte, lernende Suchverfahren [FINte]

Das *unüberwachte Lernen* wird dagegen häufig zur Gruppierung, dem so genannten Clustering, verwendet. Im Gegensatz zu überwachtem Lernen geht dem Einsatz keine Trainingsphase voraus. Ähnlichkeitsmaße der Artefakte bilden die Grundlage für diese Verfahren. Diese Maße müssen automatisch aus den Artefakten extrahierbar sein, um den Einsatz dieser Verfahren zu ermöglichen. Das Clustering erkennt anschließend Gemeinsamkeiten und ordnet die Artefakte darauf aufbauend Gruppen zu. Abbildung 4.11 verdeutlicht dies im Vergleich zu überwachten Verfahren. Eine Aufteilung kann hierbei hierarchisch, aber auch partitionierend erfolgen. Eine Gruppierung kann überlappend oder ausschließend sein und auch Mehrfachzuordnungen sind möglich. Der Einsatz unüberwachter Verfahren beschränkt sich heute in der Regel auf Empfehlungsdienste, so genannte Recommender, die Suchergebnisse nicht direkt beeinflussen, sondern den Nutzer bei der Spezifizierung seiner Suchanfrage oder der Einschränkung der Suchergebnisse unterstützen. Dies ist darauf zurückzuführen, dass die Ergebnisse einer automatischen Klassifikation meist grob und die Qualität des Einsatzes damit stark schwankend ist.

Nach der Vorstellung der grundlegende Konzepte von Suchmaschinen werden im folgenden zwei Suchmaschinen vorgestellt, die aufgrund ihrer neuartigen Ansätze im Rahmen des dieser Arbeit zugrunde liegenden Entwicklungsprojektes untersucht wurden.

4.5.2 Kommerzielle Suchansätze

In einer informationsorientierten Gesellschaft stellt das Suchen und Finden von Informationen eine der grundlegenden Anforderungen dar, um sich im Wettbewerb durchsetzen zu können. Auch aus diesem Grund sind die Algorithmen und Methoden, die hinter kommerziellen Angeboten von Suchmaschinen stehen die Geheimnisse der jeweiligen Unternehmen. Ein gutes Beispiel dafür ist die Google Search Appliance [Goo14]. Mit diesem Angebot richtet sich die Firma Google an Unternehmen, um deren Datenbestände ähnlich zu seiner Internetsuchmaschine zugänglich zu machen. Um sein geistiges Eigentum zu schützen geht der Anbieter dabei soweit, dass er die Software und die dafür notwendige Hardware als „Black Box" zur Verfügung stellt, die von den Mitarbeitern für das jeweilige Einsatzszenario konfiguriert wird und dann ohne Zugriff der Administratoren der Kunden seine Dienste anbietet.

Dieses sicherlich extreme Beispiel verdeutlicht die Schwierigkeit hier einen detaillierteren Einblick in die Technik aufzuzeigen, da unter anderem keine Informationen über den Marktführer unter den Suchmaschinenanbietern verfügbar sind. Dennoch sollen an dieser Stelle zwei grundlegend unterschiedliche Suchansätze unter den kommerziellen Lösungen vorgestellt werden.

Die Suchmaschine iFinder der Firma IntraFind Software AG [AG12] ist eine klassische Unternehmenssuchlösung, die die Technologien nutzt, wie sie in Abschnitt 4.5.1 vorgestellt wurden. Die Zielstellung der Suchmaschine ist das Suchen in bisher nicht erschlossenen Datenquellen. Fokus ist vorwiegend die Anwendung für Dateisysteme, also unstrukturierte oder semistrukturierte Daten. Methodisch handelt es sich bei iFinder um eine textbasierte Suche, die auf Basis von Häufigkeiten, Ähnlichkeiten und statistischen Annahmen Suchergebnisse bewertet. Eine explizite Informationsvernetzung findet nicht statt. Zur Indizierung verwendet das System ein Textanalyse. Dieser dienen Thesauri (vgl. Abschnitt 4.1.4) sowie Fachwörterbücher zur Erkennung fachlich Begriffe. Zur Erkennung allgemeiner Begriffe, wie Namen, Bezeichnungen oder Datumsangaben dient ein so genannter Tagging-Service aus dem Bereich des „Natural Language Processing" [Gur13], einem Fachgebiet der Linguistik. Dabei werden die Textbausteine zunächst in Sätze zerlegt und mit Hilfe von Sprachmodellen die grammatikalische Bedeutung der Wörter identifiziert. Dies erlaubt unter anderem die Klassifizierung von so genannten „Named Entities", also Bezeichnern die für das Verständnis wichtige Objekte beschreiben. Die mit diesen Verfahren erkannten Facetten lassen sich im iFinder anschließend durch den Administrator der Kundenunternehmen in Facettierungsregeln nutzen, um eine Kategorisierung des Ergebnisraums zu erreichen. Dadurch können Suchergebnisse nach anwendungsspezifischen Begriffen eingeschränkt werden.

Eine weitere Suchmaschine ist Conweaver, welches von der gleichnamigen Firma Conweaver GmbH entwickelt wird. Die hier verwendete Technologie unterscheidet sich grundlegend von der klassischer Suchmaschinen, da sie durch den Einsatz semantischer Technologien (vgl. Abschnitt 4.1.2) unterstützt wird. Daher wird Conweaver auch als Wissensmanagementsystem vertrieben. Ziel ist es dabei Informationen in Form definierter Sichten auf Daten zu einem definierten Thema zu aggregieren. Dafür wird zunächst ein bedarfs- und analysebasiertes Informationsnetz (eine Ontologie) aufgebaut, das anschließend für die Analyse von Beziehungswissen zwischen Informationen genutzt wird. Das Konzept folgt den in Abschnitt 4.1 beschriebenen Denkmustern des Menschen, wonach die Interpretation von Daten stets vom individuellen geistigen Modell abhängig ist. Das Informationsnetz bildet

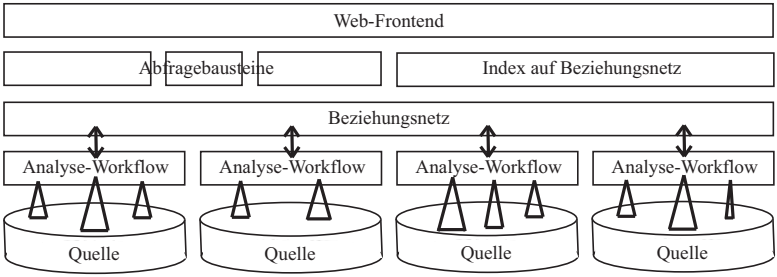


Abbildung 4.12: Architektur von Conweaver [FINte]

das abstrakt beschriebene Wissen des Nutzungskontexts, das so genannte Expertenwissen, ab und nutzt dieses, um Zusammenhänge zwischen analysierten Daten herzustellen. Abbildung 4.12 verdeutlicht die Architektur von Conweaver. Ausgangsbasis sind die vorhandenen Daten, die sich über verschiedene Quellsysteme erstrecken können. Darauf aufbauend werden Analyse-Workflows nach dem Adapter-Prinzip verwendet, um die Artefakte zu klassifizieren und für die Bewertung standardisiert aufzubereiten. Die Artefakte werden dann auf Basis des Beziehungsnetzes in einen logischen Zusammenhang gebracht und vernetzt. Das Beziehungsnetz - ein semantisches Netz - dient dabei als Abbildung des Domänenwissens, das als Administrationsaufgabe dauerhaft gepflegt werden muss. Nach der Vernetzung setzt Conweaver auf sogenannte Abfragebausteine, die bei der anwendungsfallspezifischen Sichtenbildung auf der Wissensbasis unterstützen. Die tatsächliche Nutzersicht bildet abschließend ein Web-Frontend.

4.5.3 Open Source Suchansätze

Der Stand der Technik bei Open Source Suchansätzen lässt sich im Vergleich zu kommerziellen Suchansätzen exakter eingrenzen. Das Projekt Lucene Core, das in der Apache Software Foundation betrieben wird, entwickelt mit dem Lucene Index den Suchindex, der insbesondere aufgrund seiner Performanz und Erweiterbarkeit in der Regel für Open Source Volltext-Suchmaschinen zum Einsatz kommt. Während frühere Volltext-Indizes häufig auf Basis von Bäumen realisiert wurden [RTK] stellt Lucene einen dokumentenzentrierten Ansatz dar. Abbildung 4.13 zeigt einen Beispielindex auf Basis dreier Dokumente. Dabei werden die Terme der Dokumente in eine alphabetisch sortierte Liste eingepflegt, in der jeder einzelne Term nur einziges Mal vorkommt. Darüber hinaus erhält jeder Term und jedes Dokument eine Identifikationsnummer. Zu jedem Term werden die zugehörigen Dokumente als Liste verwaltet. Somit entsteht eine Liste von Termen mit einem Mapping zu den Dokumenten, in denen sie vorkommen. Mit dem entstanden Index können dadurch für einen definierten Term alle relevanten Dokumente erreicht werden. Darüber hinaus pflegt Lucene einen umgekehrten Index, der die Identifikationsnummern der Terme in einer Liste an den Dokumenten pflegt. Dadurch ist es möglich ausgehend von einem gefundenen Dokument über die zugeordneten Terme eine Suche nach ähnlichen Dokumenten durchzuführen. Für weiterführende Informationen sei auf [RTK] verwiesen.

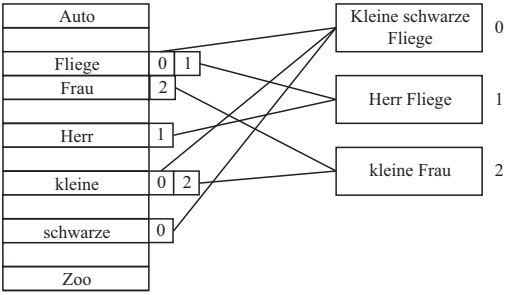


Abbildung 4.13: Aufbau eines Lucene Index

Stellvertretend für Open Source Suchmaschinen werden im Folgenden die in der Softwareentwicklung heute am weitesten Verbreiteten Projekte Apache Solr [Sol14] und Elastic Search [Sea14] vorgestellt. Bei sind Volltext-Suchmaschinen und basieren auf dem Lucene Core.

Die Entwicklung von Apache Solr begann 2004 als Projekt der Firma CNET Networks und wurde im Jahr 2006 unter der Apache Open Source Lizenz veröffentlicht. Die Softwarebasis ist in Java entwickelt und wird als Standalone- und eingebettete Version zur Verfügung gestellt. Insbesondere die eingebettete Version, die sich direkt in Java Applikationen integrieren lässt, trug zum breiten Einsatz in Softwareprojekten bei. Als wichtige Funktionen gelten neben der Volltext-Suche die facettierte Suche (vgl. Abschnitt 4.5.1), die Unterstützung von strukturieren Abfragesprachen über die reine Volltext-Suche hinaus, sowie die Unterstützung von verteilten Suchclustern, bei denen individuell gepflegte Indizes in einzelnen Instanzen des Solr-Servers gepflegt werden und für die Suche als ein gemeinsamer Index zusammengefasst werden können. Darüber hinaus liegt der Fokus von Solr auf der Interoperabilität mit standardisierten Protokollen. Unter anderem existiert dedizierte Indexschnittstellen für JSON, XML, PHP, Ruby, Python und XSLT.

Elastic Search wurde 2010 in der ersten Version ebenfalls unter der Apache Lizenz veröffentlicht. Hinsichtlich des Funktionsumfangs in Bezug auf die reine Volltext-Suche unterscheiden sich Solr und Elastic Search nicht. Auch Elastic Search setzt auf dem Lucene Core auf. Der Fokus liegt dabei einerseits auf der Einfachheit der Bedienung, so beschränkt sich die Indexschnittstelle auf JSON. Andererseits bietet der Ansatz hochverteiltes Suchen mit einer Ausrichtung auf Big Data Anwendungen [Sea14]. Abbildung 4.14 zeigt die Architektur eines Elastic Search Servers. Es ist zu erkennen, dass Master und Data Nodes unterschieden werden. Dies können, müssen aber nicht, einzelne Rechner im Cluster sein. Die Aufteilung dient lediglich der Strukturierung von Elastic Search Serverinstanzen hinsichtlich ihrer Funktionalität. Über Masternodes erfolgt die Analyse neuer Dokumente für den Index sowie deren Indizierung. Anschließend pflegen diese die Indexaktualisierung in die Data Nodes ein. Eine Suche wird stets gegen die Data Nodes ausgeführt. Dieses Vorgehen stellt sicher, dass auch während der Analyse und Erweiterung des Suchindex dieser stets konsistent und nutzbar zur Verfügung steht. Für die Suche stellt Elastic Search eine integrierte Lastverteilung zur Verfügung, die wiederum durch eine entsprechende konfigurierte Instanz des Elastic Search Servers realisiert werden, welche die Suchanfragen auf die Data

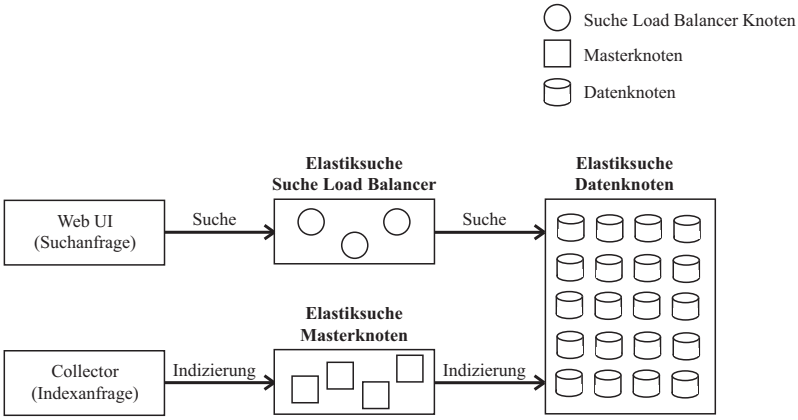


Abbildung 4.14: Elasic Search Architektur

Nodes verteilen. Anwendung findet Elastic Search aufgrund dieser Big Data Optimierung insbesondere in Webportalen, wie beispielsweise eBay, XING oder GitHub.

4.6 Zusammenfassung

In Kapitel 4 wurde der Stand der Technik in den an die Konzepte der vorliegenden Arbeit angrenzenden Wissenschaftsgebieten vorgestellt. Dabei wurden in Abschnitt 4.1 zunächst die Begriffe Daten, Information und Wissen erläutert und anschließend die heute relevante Konzepte der Wissensrepräsentation vorgestellt. In den Abschnitten 4.2 und 4.3 wurden darauf aufbauend die heute dafür genutzten Technologien im Bereich des Modelldatenmanagement im Automobilbau und anderen Industriezweigen präsentiert. Im Anschluss daran wurden in den Abschnitten 4.4 und 4.5 die nach heutigem Stand wichtigen und für die Arbeit besonders relevanten Ansätze der Informatik zur Gewinnung und Verarbeitung von Wissen vorgestellt. Abschnitt 4.4 verdeutlicht den Stand der Technik in Bezug auf die Extraktion und Repräsentation von Informationen für modellbasierte Entwicklungsprozesse mit besonderem Fokus auf die Qualität der genutzten Artefakte, während sich Abschnitt 4.5 mit der Aufbereitung der Informationen für eine zielgerichtete Wiederverwendung befasste. Im Folgenden Kapitel 5 werden ausgehend vom Stand der Technik die im Rahmen der vorliegenden Arbeit entwickelten Konzepte vorgestellt.

5 Konzepte der prozessintegrierten Dokumentation und optimierten Wiederverwendung

Aufbauend auf dem in Kapitel 4 vorgestellten Stand der Technik, zeigt sich insbesondere im Bereich des Modelldatenmanagements, dass heutige Ansätze eher das individuelle Entwicklungsartefakt betrachten und eine explizite, händische Dokumentation erfordern. Die Ansätze des Wissensmanagements sind in diesem Bereich noch nicht vertreten. Dies ist unter anderem auf die Aufwände zur Erstellung und Pflege geeigneter Informationsdatenmodelle zurückzuführen, die in der Regel eine besondere Qualifikation des Erstellers erfordern. In der Folge kann auch die Suche und Wiederverwendung nicht mit semantischen Informationen unterstützt werden, was den Einsatz von Suchmaschinen auf die reine Schlagwortsuche beschränkt. Hier bedarf es neuer Konzepte, die eine Dokumentation von Entwicklungsartefakten aus Kontext des Entwicklungsprozesses erlauben und mittels eines geeigneten Informationsmodells die Suche und Wiederverwendung unterstützen.

Das vorliegende Kapitel beschreibt das im Rahmen der Arbeit entwickelte Konzept zur automatischen Dokumentation von Artefakten im modellbasierten Entwicklungprozess sowie das darauf aufbauende Konzept zur Unterstützung der Wiederverwendung mit Hilfe eines Empfehlungssystems. Es geht dabei unter anderem auf das Vorgehen bei der Entwicklung sowie die Rahmenbedingungen einer Anwendung des Konzeptes ein. In Abschnitt 5.1 wird zunächst das Vorgehen zur Bestimmung des so genannten Modellkontexts als minimale Wissensbasis eines Entwicklungsartefaktes einer Anwendungsdomäne vorgestellt und der Lösungsansatz zum ersten Arbeitsschwerpunkt (vgl. Abschnitt 1.2), der Definition eines Modellkontextes, beschrieben. Darauf aufbauend wird in Abschnitt 5.2 eine Analyse von Entwicklungsartefakten am Beispiel der Automobildomäne durchgeführt, um die davon extrahierbaren Daten zu identifizieren.

Im Anschluss an die Erschließung der im Entwicklungsprozess verfügbaren Daten wird in Abschnitt 5.3 die im Rahmen dieser Arbeit entwickelte Architektur einer Lösung zur Extraktion der identifizierten Daten und deren Weiterverarbeitung zu einer Wissensbasis vorgestellt. Die Abschnitte 5.4, 5.5 und 5.6 gehen anschließend auf die einzelnen Verarbeitungsschritte der Architektur ein. Damit stellen diese Abschnitte den Lösungsansatz für den zweiten Arbeitsschwerpunkt der vorliegenden Arbeit (vgl. Abschnitt 1.2), die automatische Erfassung des Modellkontextes, vor.

Abschließend erläutert Abschnitt 5.6 die passive Nutzung der generierten Wissensbasis zur Suche von Entwicklungsartefakten und Abschnitt 5.7 stellt das ebenfalls im Rahmen der Arbeit entwickelte Konzept einer kriterienbasierten Modellbewertung vor, die den Entwicklungsprozess durch aktive Vorschläge von geeigneter Artefakte unterstütz.t Damit stellen diese Abschnitte den Lösungsansatz für den dritten Arbeitsschwerpunkt (vgl. Abschnitt 1.2), des Nachweises der Korrektheit des Ansatzes durch Umsetzung von Wiederverwendungskonzepten, vor.

5.1 Definition eines Modellkontext

Die Spezifikation einer Wissensbasis für eine Domäne erfordert es zunächst den Bedarf für die jeweiligen Einsatzszenarien zu bestimmen. Dies ist notwendig, da die vorliegende Arbeit von der These ausgeht, dass alle notwenigen Informationen zur automatischen Generierung einer Wissensbasis für Simulationsmodelle im Automobilbau bereits heute im Entwicklungsprozess vorhanden sind. Da diese These die Grundlage der weiteren Untersuchungen darstellt, wird die Gültigkeit dieser im ersten Schritt untersucht. Der Wissensrahmen eines einzelnen Artefaktes wird im Rahmen dieser Arbeit unter dem Begriff Modellkontext zusammengefasst.

Definition. Ein Modellkontext umfasst alle Informationen die ein Simulationsmodell beschreiben oder es zu anderen Simulationsmodellen in Beziehung setzen.

Als erster Ansatz zur Definition eines Modellkontexts hat sich die Anforderungsaufnahme in Form von Interviews über alle im Prozess beteiligten Rollen als geeignet erwiesen. Wie bereits aus der Definition des Modellkontextes hervorgeht, sind für die Beurteilung von Entwicklungsartefakten zwei Typen von Informationen notwendig - Beschreibungen und Beziehungen.

Diese Informationstypen lassen sich des Weiteren in Klassen von Informationen zu einem Artefakt unterteilen. Ein Modellkontext enthält dabei mindestens folgende Klassen von Informationen:

- **Meta-Daten**: Meta-Daten bezeichnen im Allgemeinen Daten, die Informationen zu anderen Daten, definieren. Unter Meta-Daten im konkreten Fall werden die einem Objekt direkt zugeordneten, beschreibenden Attribute verstanden. Dies können beispielweise Bezeichner, Herkunfts- oder Zustandsangaben sein.
- **Architektur**: Die Klasse der Architekturinformationen beschreibt strukturelle Gegebenheiten von Objekten. Dies können Informationen zum internen und externen Aufbau, Schnittstellen sowie Nutzungsinformationen sein.
- **Organisation**: Organisatorische Daten bezeichnen Informationen die einem Objekt über Organisationsstrukturen der Verwendungsumfeldes zugeordnet werden. Dazu gehören unter anderem Zuordnungen zu betrieblichen Prozessen, Projekten sowie Regeln und Beschränkungen der Nutzung.
- **Qualität**: Die Klasse der Qualitätsinformationen umfasst die Daten, die qualitative Aussagen zu den Eigenschaften eines Objektes erlauben. Grundlage hierfür stellt in der Regel das Vorhandensein eines Qualitätsmodells (vgl. Abschnitt 4.4.2) dar.
- **Verschlagwortung**: Durch die so genannte Verschlagwortung ist es möglich Objekten individuelle Schlüsselbegriffe zuzuordnen, um deren Suche oder Ordnung den jeweiligen Bedürfnissen anzupassen. Hierdurch wird eine unvollständige Datenbasis generiert, die es erlaubt, bei Verfügbarkeit entsprechender Schlagworte, Aussagen zu einem Objekt zu treffen. Bei fehlenden Schlagworten gilt jedoch nicht der Umkehrschluss der Aussagen.

Nachdem vorstehend der Begriff des Modellkontextes eingeführt wurde, wird dieser im Folgende auf die betrachtete Anwendungsdomäne dieser Arbeit, die automobile, modellbasierte Entwicklung übertragen. Aus der durchgeführten Anforderungsaufnahme wurde ein

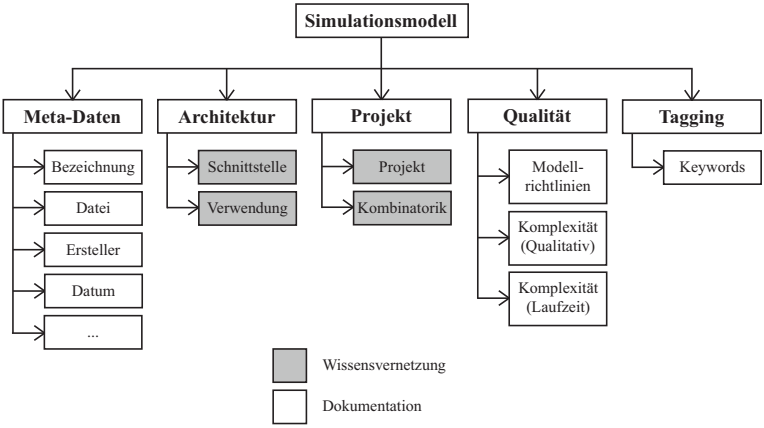


Abbildung 5.1: Modellkontext der modellbasierten Simulation

erster Entwurf einer minimalen Wissensbasis abgeleitet. Diese ist in Abbildung 5.1 für ein Simulationsmodell dargestellt. Es ist zu erkennen, dass die Meta-Daten von klassischen Dateiattributen geprägt sind. Neben der Bezeichnung, dem Dateipfad, Erstellerinformationen sind hier auch Versionsinformationen notwendig, um Änderungen an Simulationsmodellen verfolgen zu können. Bezüglich der Architektur sind die Schnittstellen der Modelle sowie deren frühere Verwendung von Bedeutung. Als organisatorische Informationen sind insbesondere Zuordnungen zu übergeordneten Regelwerken und betrieblichen Strukturen notwendig. Dies können im konkreten Fall einerseits Entwicklungsprojekte sein und andererseits Vorgaben, welche Modelle kombiniert werden dürfen, weil sich daraus in einem größeren Zusammenhang, beispielsweise einem konkreten Produkt, eine sinnvolle Verschaltung der realen Komponenten dieser Modelle ergibt. Bezüglich der Qualität besteht der Bedarf eines Qualitätsmodells, das eine Vergleichbarkeit der Simulationsmodelle erlaubt. Ein solches ist heute nicht etabliert. Der Ansatz der vorliegenden Arbeit ist es, hierfür die Nutzung des Qualitätsmodells nach Scheible (vgl. Abschnitt 4.4.2) zu prüfen. Abschließend dient die Verschlagwortung der Anpassbarkeit des Modellkontextes durch anwendungsspezifische Schlüsselworte (Tags).

Abbildung 5.1 stellt dabei zwei Kategorien von Informationen innerhalb des Modellkontextes heraus - die Dokumentation und die Wissensvernetzung. Die Dokumentation beschreibt Informationen, die einem Simulationsmodell direkt zugeordnet sind und es im Sinne eine Attributierung näher beschreiben. Informationen der Wissensvernetzung stellen darüber hinaus Zusammenhänge mit organisatorischen Strukturen dar, die auf Beziehungen zwischen mehreren Simulationsmodellen schließen lassen. Sie vernetzen damit die einzelnen Modellkontexte und damit die Simulationsmodelle miteinander.

Ausgehend vom vorgestellten ersten Entwurf einer minimalen Wissensbasis für die modellbasierte Simulation werden im folgenden Abschnitt 5.2 die Entwicklungsartefakte dieser Anwendungsdomäne vorgestellt und die damit verbundenen, in der Regel im Entwicklungsprozess zur Verfügung stehenden Informationen, analysiert.

5.2 Strukturanalyse des verfügbaren Modellwissens

In Abschnitt 5.1 wurde der grundlegende Informationsbedarf in modellbasierten Entwicklungsprozessen Simulation vorgestellt. Auf Basis dieser Anforderungen werden in diesem Abschnitt die Entwicklungsartefakte (vgl. Abschnitt 3.4) detailliert analysiert, um die Verfügbarkeit der Informationen bewerten zu können. Die dabei getroffenen Aussagen, welche Artefakte welche Form von Informationen enthalten können, erfolgt auf Basis einer Analyse von aktiv genutzten Modellierungsformen und Ablagestrukturen. Es muss davon ausgegangen werden, das in Abhängigkeit eines konkreten Anwendungsfalls Daten in anderen Formen und Formaten vorliegen. Der in dieser Arbeit vorgestellte Lösungsansatz berücksichtigt diese Einschränkungen, geht jedoch auch davon aus, dass diese Daten vital für die Durchführung eines modellbasierten Entwicklungsprozesses sind und daher verfügbar sind.

5.2.1 Modell

Das Simulationsmodell als zentrales Element der modellbasierten Simulation ist der erste Analyseschwerpunkt. Daten, die anhand eines Modells gewonnen werden können, sind stark abhängig von der Art des Modells sowie den genutzten Ablagestrukturen. Bei der Art des Modells kann einerseits nach Modelltypen, wie Simulinkmodellen oder UML-Modellen und andererseits nach deren Verarbeitungszustand unterschieden werden. Dabei kann ein Simulinkmodell als Quelldatei (vgl. Abschnitt 3.4) vorliegen, durch Code-Generierung als C-Code oder auch als Kompilat in binär Code. Es ist offensichtlich, dass je nach Verarbeitungszustand die Zugänglichkeit zu menschenlesbaren Dokumentationen eingeschränkter möglich ist. Es stellt sich daher die Frage, welche Informationen aus den verschiedenen Modellarten extrahiert werden können?

Zunächst wird die Quelldatei für den Anwendungsfall der Simulinkmodelle betrachtet. Man unterscheidet hier zwei Formen der Modellierung - Einzelmodelle und Modellbibliotheken. Ein Einzelmodell enthält auf oberster Ebene Blöcke, die über Kanten bzw. Signalpfade verbunden sind. Hierbei enthält die oberste Ebene Input- und Output-Blöcke, die die externen Schnittstellen eines Einzelmodells repräsentieren und diese auf die internen Blöcke verteilen. Eine Modellbibliothek enthält auf oberster Modellierungsebene nur Subsysteme, um die Modelle (vgl. Definition Abschnitt 3.4) zu gruppieren. Auf Basis dieser rein strukturellen Unterscheidbarkeit, kann bei entsprechendem algorithmischen Zugriff eine automatisierte Klassifikation durchgeführt werden.

Modelle bauen sich stets aus Blöcken auf. Jeder Block beinhaltet ihm zugeordnete Attribute, die seine Eigenschaften innerhalb des Modells beeinflussen. Diese Attribute, insbesondere Bezeichner und numerische Werte, die die Funktionalität steuern sind hier von Bedeutung. Informationen, die sich hieraus ergeben könnten, sind z.B. Modellbezeichnungen, gewählte Einstellungen und Konstanten, erwartete Datentypen oder Einheiten. Ein für die Dokumentation besonders relevanter Block ist der Text-Block. Er erlaubt es innerhalb der Modelle textuelle Beschreibungen zu hinterlegen, die keinen Einfluss auf die vom Modell realisierten Funktionalitäten haben. Sie erlauben damit die Durchführung von Dokumentationen im Modell, was vergleichbar zu Kommentaren im Quellcode bei der klassischen Programmierung ist.

5.2.2 Signale und Parameter

Wie in Abschnitt 3.4 vorgestellt wurde, sind Simulationsmodelle vergleichbar mit Klassen der objektorientierten Programmierung. Eine Instanziierung für einen konkreten Anwendungsfall benötigt stets die Zuordnung von Signalen und Parametern. Beide Elemente geben Aufschluss auf die Funktionen und Einsatzmöglichkeiten des jeweiligen Simulationsmodells.

Signale werden heute je nach Entwicklungsprozess unterschiedlich verwaltet. Ausgangsbasis ist in der Automobilentwicklung stets die so genannte Kommunikationsmatrix (kurz K-Matrix), die sämtliche in einem Fahrzeug genutzten Signale mit Quelle und Senke des Signals beinhaltet. Im Entwicklungsprozess werden diese Kommunikationsmatrizen häufig im Microsoft Excel Format exportiert und für die Konfiguration der Simulationsmodelle genutzt. Unabhängig vom tatsächlichen Format einer Signalkonfiguration, erlaubt diese Aussagen über den Realitätsgrad eines Modells. Wie im Abschnitt 3.3 erläutert wurde, sind die Anforderungen der Simulationsdomänen stark heterogen bezüglich Rechengenauigkeit und Realitätsgrad des Modellverhaltens. Implementiert ein Modell nur die Signale, die für Ausführung der Funktionalität an den Schnittstellen notwendig sind, so ist es in der Regel lediglich für die Offline-Simulation geeignet. Werden jedoch alle in der K-Matrix spezifizierten In- und Output-Signale implementiert, so kann das Modell, unter der Voraussetzung einer entsprechenden Modellierung, als realitätsnaher Ersatz auch mit echten Steuergeräten verschaltet werden. Darüber hinaus lassen sich durch detaillierter Analyse der Signale zusätzliche Informationen gewinnen. Über die Bezeichner und Einheiten einzelner Signale, lässt sich beispielsweise bestimmen, ob es sich um elektrische oder mechanische Signale handelt, was wiederum Rückschlüsse auf die Modellierung ermöglicht. Ein Steuergerätemodell enthält beispielsweise keine mechanischen Signale. Darüber hinaus können Signalen Grenzwerte zugeordnet werden, die gültige Werte für diese vorgeben. Die Analyse dieser Grenzwerte erlaubt es die Rahmenbedingungen des Einsatzes eines solchen Modelle zu extrahieren.

Parameter beeinflussen die Abarbeitung von Funktionen des Modells vergleichbar mit Variablen (vgl. Abschnitt 3.4). Es existieren verschiedene Formen der Parametrierung und die tatsächliche Aussagefähigkeit von Parametern ist stark vom Nutzungsverhalten abhängig. Wie bei Variablen kann das Drehmoment eines Motors in der Einheit Nm am Ausgang bezeichnet werden als „Torque_Out_Nm" oder schlicht als „var_a". Es ist offensichtlich, dass das die Dokumentationsaussage der Bezeichner qualitativ stark unterschiedlich sind. Eine verlässliche Quelle für Modellinformationen sind Parameter demnach nur in Verbindung mit Richtlinien für deren Benennung. Darüber hinaus kann eine Parametrierung intern und extern des Modells erfolgen. Bei interner Parametrierung, werden die Parameter innerhalb der Modelldatei deklariert und sind stets zusammen mit dem Modell verfügbar. Bei externer Parametrierung werden Parameter in Form von Skripten gespeichert und initialisieren die Ausführungsumgebung erst zur Modellnutzung.

5.2.3 Modellierungszielstellungen

Die Eigenschaften eines Modells hängen hauptsächlich mit der Zielstellung der Modellierung zusammen. Man unterscheidet Modelle hier nach Modellierungsstilen und -detaillierungsgraden. Ein Modell kann dabei als Verhaltensmodell oder als physikalisches Modell

entwickelt sein. Während Ersteres durch Algorithmen das Verhalten eines realen Objektes rekonstruiert, bildet das physikalische Modell reale Strukturen nach, um näher an das reale Verhalten zu gelangen. Ein Verhaltensmodell kann dabei beispielsweise eine mathematische Approximation des angestrebten Verhaltens sein. Ein physikalisches Modell kann beispielsweise eine Tabelle realer Messwerte, wie ein Kennlinie, sein, auf Basis derer das Modell in Abhängigkeit der Eingabewerte die Ausgabewerte konstruiert. Dies wird auch als „Look-Up-Table" bezeichnet.

Darüber hinaus werden Modelle nach Regler- und Komponentenmodell (vgl. Abschnitt 3.4) unterschieden. Reglermodelle bilden das Verhalten von Steuergeräten nach, während Komponentenmodelle das reale, physikalische Verhalten von Komponenten, d.h. die Regelstrecke, nachbilden.

5.2.4 Modellablagestrukturen

Wie in Abschnitt 3.5 erläutert, ist die Verfügbarkeit insbesondere von Meta-Daten stark abhängig von den genutzten Modellablagestrukturen. Es ist leicht zu erkennen, dass spezialisierte Ablagestrukturen für Entwicklungsartefakte die zielgerichtete Dokumentation dieser unterstützen. Häufig sind die eingesetzten Lösungen reine Netzlaufwerken so genannte Fileshares, mit hierarchischen Ordnungsstrukturen und gezielter Benennung von Strukturierungsknoten, so genannten Ordnern. Diese Strukturen erlauben den Zugriff auf klassische Dateiattribute, wie Name bzw. Pfad, Dateigröße, letzte Bearbeitung, Autor, Rechte etc. Neben Fileshares kommen Versionsverwaltungssysteme zum Einsatz. Hier kommen zu den Dateiattributen noch Versionsinformationen, teilweise Bezeichner für bestimmte Versionsstände), sowie in der Regel Freitext-Daten in Form von Änderungskommentaren zu Versionsständen. Darüber hinaus erlauben es einige Versionsverwaltungssysteme Rechte für Sichtbarkeit, Lesen und Schreiben personen- oder gruppenspezifisch zu vergeben. Zusätzlich zu diesen Modellablagen werden zunehmend anwendungsspezifische Lösungen entwickelt, wie sie in den Abschnitten 3.5 und 4.2 vorgestellt wurden. Diese spezialisierten Datenmanagementsysteme erlauben es, den Artefakte strukturierte Daten und Attribute zuzuweisen und somit eine explizite Dokumentation durchzuführen. Somit lassen sich je nach Vorgaben des Entwicklungsprozesses die notwendigen Informationen durch manuelle Erfassung speichern. Für die Qualität der Ergebnisse der automatisierten Dokumentation mit den Methoden der vorliegenden Arbeit wäre eine strukturierte Quelle für Dokumentationen förderlich. Auf der anderen Seite ist es das Ziel die Modelle über organisatorische Grenzen hinweg auszutauschen, was eine explizite Dokumentation, für alle möglichen Anwendungsfälle eines Modells gegenüber den rein intraorganisatorischen Bedürfnissen verkompliziert.

5.2.5 Vernetzung von Simulationsmodellen

Die Nutzbarkeit eines Simulationsmodells für definierte Anwendungsfälle ist von einer Reihe von Rahmenbedingungen abhängig. Neben rein architektonischen Anforderungen, wie der Kompatibilität von Schnittstellen, sind dies insbesondere Modellierungs- und Organisationskriterien. Für die Modellierung ist vor allem die berechnungstechnische Umsetzung als diskrete oder kontinuierliche Auswertung von Bedeutung, da diese bei der Berechungsausführung zu berücksichtigen ist. Ein weiteres Kriterium ist die Geschwindigkeit der Ausfüh-

rung einer Berechnung insbesondere bei zeitkritischen Modellen. Benötigt beispielsweise eine Modell die Ergebnisse einer vorherigen Berechnung alle 10ms, die Berechnung benötigt jedoch 20ms, so kommt es zu so genannten numerischen Instabilitäten und damit zu nicht nutzbaren Ergebnissen. Die automatische Auswertung bzw. Bestimmung dieser Ausführungszeiten stellt allerdings im Rahmen der Dokumentation eine Herausforderung dar, da sie einerseits von der verwendeten Hardware und andererseits von der Komplexität der Gesamtsimulation bestehend aus allen genutzten Modellen abhängig ist.

Organisatorische Rahmenbedingungen beschränken die Nutzung von Simulationsmodellen, die rein technisch simulierbar wären. Diese Regeln werden heute meist aus indirekten Quellen über das implizite Wissen von Menschen berücksichtigt. Sind diese Regeln zugreifbar in den indirekten Quellen dokumentiert, so müssen diese Quellen bei einer automatischen Dokumentation berücksichtigt und das darin enthaltene Wissen logisch verknüpft werden.

5.2.6 Lifecycle Management für Simulationsmodelle

Die bisher beschrieben Attribute und vernetzenden Informationen tragen zum Verständnis der Funktionalität und der Einsatzmöglichkeiten von Simulationsmodellen bei. Darüber hinaus sind Informationen zum Reifegrad, Rechten und dem Einsatz im Entwicklungsprozess relevant. Zur Verfolgung dieser Informationen werden heute verbreiten Lifecycle Management Systeme eingesetzt, um die Prozesse softwaretechnisch abbilden und der Ausführung verfolgen zu können.

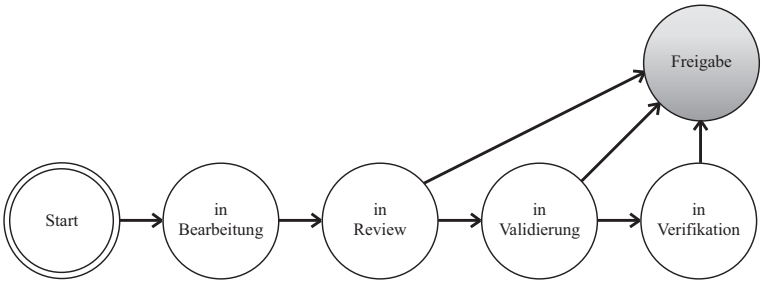


Abbildung 5.2: Exemplarischer Zustandsgraph für Life-Cycle-Managmentsysteme

Für die Dokumentation sind dabei insbesondere Anforderungsmanagement- sowie Workflow-Managementsysteme von Bedeutung. Die Informationen aus Anforderungsmanagementsystem für die Modellentwicklung lassen Rückschlüsse auf Implementierungsdetails und umgesetzte Funktionalitäten zu, da in der Regel davon ausgegangen werden kann, dass ein Simulationsmodell, dessen Entwicklung abgeschlossen ist, die gestellten Anforderungen erfüllt. Um den Grad der Umsetzung bewerten zu können, ist wiederum die Analyse der Workflow-Managementsysteme notwendig. Unter eine Workflow-Managementsystem wird an dieser Stelle eine Software zur Verwaltung von Zustandsgraphen verstanden, wobei jeder Knoten einen Entwicklungszustand eines Entwicklungsartefaktes, hier in der Regel

eines Simulationsmodells, repräsentiert. Abbildung 5.2 zeigt beispielhaft einen solchen Zustandsgraphen für Simulationsmodelle. Dabei werden in der Regel Entwicklungs- („in Bearbeitung") sowie Absicherungszustände verschiedener Qualitäten (z.B. „in Review", „in Verifikation" oder „in Validierung") durchlaufen. Diese Zustandsgraphen können, insbesondere in verschiedenen Domänen, unterschiedlich ausgeprägt sein, was eine statische Abbildung innerhalb eines Softwaresystems nicht praktikabel macht.

5.3 Architektur des modularen Informationsmanagementsystems

Basierend auf der Anforderungs- und Bedarfsanalyse heute eingesetzter Simulationsverfahren und -prozesse in der modellbasierten Entwicklung wurden im Rahmen der Forschungs- und Entwicklungstätigkeiten der vorliegenden Arbeit die Ansätze und die Architektur des so genannten „modularen Informationsmanagementsystems" (MIM) entwickelt.

Ziel des Systems ist es, durch leichte Anpassbarkeit an sich verändernde, heterogene und organsisationsübergreifende Entwicklungsprozesse, eine automatisierte, prozessintegrierte Dokumentation durchzuführen, strukturiert zu speichern und mit den gesammelten Informationen die zielgerichteten Wiederverwendung von Entwicklungsartefakten zu unterstützen.

Dabei werden heute state-of-the-art Technologien genutzt und zu einer neuartigen Informationsmanagementsystem verknüpft. Das System unterteilt sich architektonisch in drei Ebenen, deren Zielstellung an dieser Stelle kurz vorgestellt und deren Konzeption und Umsetzung in den folgenden Abschnitten in der Tiefe erläutert wird:

- **Datenzugriff**: Die Datenzugriffsschicht hat das Ziel beliebige Datenquellen direkt anzubinden, die sich ohne Beeinflussung des Entwicklungsprozess in diesen zu integrieren und dessen Daten der Informationsanalyse strukturiert zuzuführen. Dafür wurde ein Adapter-Konzept entwickelt, dessen Fokus insbesondere auf der dynamischen Erweiterbarkeit sowohl in Richtung der Datenquellen, als auch der Datenstrukturen der Informationsanalyse liegt. Die Konzepte der Datenzugriffschicht werden im Abschnitt 5.5 näher betrachtet.
- **Informationsverarbeitung und Wissensmanagement**: Ausgehend von der strukturierten Zuführung von Daten aus der Datenzugriffsschicht erzeugt und verwaltet die Informationsverarbeitung- und Wissensmanagementschicht diese, durch Aggregation in informationsspezifischen Datenstrukturen. In Abhängigkeit des Ergebnistyps der Informationsverarbeitung werden diese in unterschiedliche Wissensmanagementlösungen eingepflegt. Die detaillierten Konzepte werden in Abschnitt 5.4 näher betrachtet.
- **Suche und Wiederverwendung**: Auf Basis der entstandenen Wissensbasis in der darunter liegenden Informationsverabreitungs- und Wissensmanagementschicht, stellt die Such- und Wiederverwendungsschicht die Schnittstelle zur Wissensnutzung dar. Dafür wird eine Abstraktionsschicht zum Zugriff auf die Datenbasen zur Verfügung gestellt, die in der Umsetzung als Programmierschnittstelle (API) den rollen- und rechtespezifischen Zugriff auf steuern soll. Darüber hinaus stellt die Schicht zur Verprobung des neu entwickelten modularen Informationsmanagementsystems Konzepte

und Implementierungen für eine volltextbasierte Suche, sowie ein wissensbasiertes Modellempfehlungssystem zur Verfügung. Die detaillierten Konzepte werden in den Abschnitten 5.6 und 5.7 näher betrachtet.

Einen ersten Überblick zur Architektur bietet Abbildung 5.3. Dargestellt sind die einzelnen Ebenen sowie die darin enthaltenen Konzeptbestandteile. Aus didaktischen Gründen wird im Folgenden zunächst die Informationsverarbeitungs- und Wissensmanagementschicht vorgestellt, da sowohl die darüber, als auch die darunter liegende Schicht stark an deren Schnittstellen orientiert sind und durch vorherige Kenntnis des inneren Aufbaus das Verständnis vereinfacht wird.

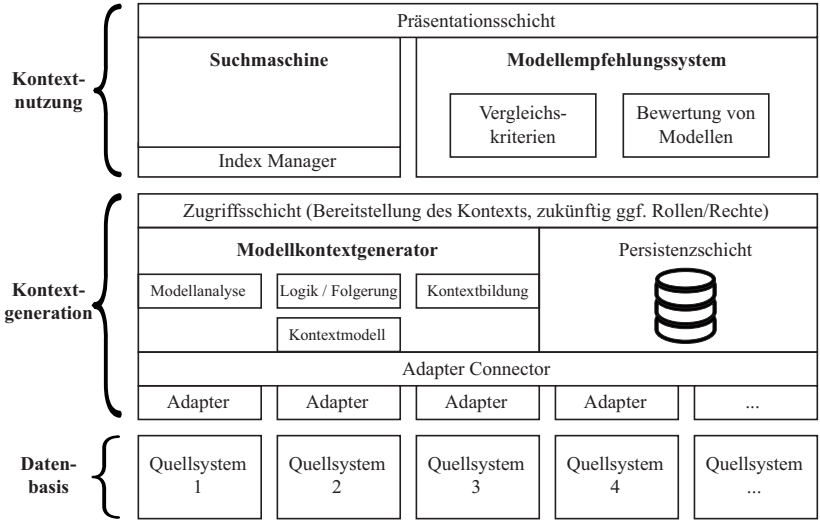


Abbildung 5.3: Überblick der Architektur des modularen Informationsmanagementsystems

5.4 Untersuchung einer Wissensrepräsentation für Simulationsmodelle

Die Wissensrepräsentation bildet die Basis eines Informationsmanagementsystems. In Abschnitt 4.1 wurden bereits die heute verbreiten Konzepte zur Verwaltung von Informationen vorgestellt. Diese wurden im Rahmen der Forschungsarbeiten der vorliegenden Arbeit untersucht und hinsichtlich ihrer Eignung für das Konzept des modularen Informationsmanagementsystems bewertet. Die Untersuchungsergebnisse werden in Abschnitt 5.4.2 dargelegt und das darauf aufbauend entwickelte Konzept in Abschnitt 5.4.3 erläutert. Zuvor wird die Strategie bei der Informations- und Wissensextraktion vorgestellt, um die Anforderungen und damit die Auswahl der gewählten Datenstrukturen nachvollziehen zu können.

5.4.1 Vom Datum zur Information

Bevor konkrete Lösungsansätze der Informationsextraktion und -repräsentation betrachtet werden können, muss eine Strategie zur Wissensgewinnung vorliegen. Eine reine Datenextraktion ohne die zielgerichtete Verarbeitung führt zwar zu einer Datensammlung über die Entwicklungsartefakte, ist allerdings ohne eine semantische Spezifikation nicht als Information nutzbar.

Wie in Abschnitt 4.1 erläutert wurde, werden aus Daten erst dann Informationen, wenn ihnen eine semantische Bedeutung zugeordnet wird. Wissen entsteht aus Informationen, wenn diese über Beziehungen verknüpft werden, die die Schlussfolgerung von Zusammenhängen ermöglichen. An dieser Reihenfolge zur Generierung von Wissen aus Daten orientiert sich auch die Strategie der vorliegenden Arbeit.

In einem ersten Schritt werden Daten aus dem Prozess extrahiert und in strukturierter, einheitlicher Form der Informationsverarbeitungsschicht zur Verfügung gestellt. Die Informationsverarbeitungsschicht enthält und pflegt eine Domänenbeschreibung, die die Zuordnung einer Bedeutung zu den Daten erlaubt und diese dadurch zu Informationen transformiert. Eine Domänenbeschreibung ist notwendig, da sie die Referenz für die Interpretation von Daten bildet. Ohne eine solche Referenz, würden Missinterpretationen, wie sie in Abschnitt 4.1.4 vorgestellt wurden, auftreten. Nach aktuellem Stand der Technik ist kein Informationsmanagementsystem ohne explizit zu definierende Domänenbeschreibung verfügbar. Dennoch wird in den folgenden Abschnitten erläutert, dass diese Domänenbeschreibung weder statisch noch vollständig sein muss, um zu hinreichend genauen Resultaten zu führen. In einem dritten Schritt erfolgt die Vernetzung der Informationen zu Wissen, indem zusätzliche Datenquellen für die Generierung von Beziehungen eingesetzt werden. Diese Datenquellen sind in der Regel nur mittelbar mit der eigentlichen Information verknüpft. Sie repräsentieren Regeln, Vorgaben und Erfahrungen, die ein Mensch bei der Betrachtung eines Entwicklungsartefaktes wissen sollte, um fundierte Entscheidungen zu treffen. Durch die Konzentration dieser Beziehungen in einer einzigen Wissensbasis, kann anschließend der Entscheidungsprozess verkürzt und Entscheidungssicherheit erhöht werden.

5.4.2 Vorstellung untersuchter Lösungsansätze der Wissensrepräsentation

In Abschnitt 4.1 wurden bereits verschiedene Ansätze für die Wissenrepräsentation vorgestellt. Um ein möglichst leicht anpassbares und erweiteresbares Informationsmanagementsystem zu schaffen, wurden diese Ansätze anhand definierter Kriterien untersucht und hinsichtlich ihrer Eignung bewertet. Diese Kriterien wurden auf Basis der Analyse aktueller Entwicklungsprozesse abgeleitet, da die gewählte Wissensrepräsentation eine heterogene Prozesslandschaft abbilden muss.

Als zentrale Kriterien einer Wissensrepräsentation wurden definiert:

- Datenpersistenz: Das System besitzt die Fähigkeit zur strukturierten Persistierung von Daten.
- Informationspersistenz: Das System kann ein übergeordnetes Datenschema (semantische Referenz) mit Daten zu Informationseinheiten verknüpfen und diese persistieren.

- Wissenspersistenz: Das System kann zwischen Informationseinheiten beliebige Typen von Beziehungen repräsentieren und die resultierenden Wissenseinheiten (Tripel, vgl. Abschnitt 4.1) persistieren.
- Dynamische semantische Referenz: Das System unterstützt die (inkrementelle) Veränderung der semantischen Referenz ohne dabei vorhandene Informationseinheiten zu beeinflussen.
- Dynamische Typisierung von Beziehungen: Das System unterstützt die Definition beliebiger Typen von Beziehungen zur Laufzeit, ohne das diese zuvor in einem Datenmodell hinterlegt sind. Dieses nicht offensichtliche Kriterium ergibt sich aus der nicht Vorhersehbarkeit von Wissensverknüfungen. Die Anforderung der Anpassbarkeit an heterogene Prozesslandschaften führt zu einer offenen Menge an Beziehungstypen, die prozessspezifischen Wissen repräsentieren.
- Strukturierte Abfragesprache (Query Language): Das System unterstützt eine strukturierte Abfragesprache, die beliebig feingranulare Anfragen sowie vorhersehbare Ergebnissätze für die weitere Verarbeitung ermöglicht.
- Einfache Erweiterbarkeit: Das System unterstützt mit geeigneten Methoden die einfache, nachvollziehbare Anpassung und Erweiterung von Datenmodellen im Betrieb. Die Implikationen von Änderungen müssen dabei weitestgehend lokal auf die Änderungen beschränkt sein. Diese Anforderung ist im Gegensatz zu den vorherigen optional, wirkt sich jedoch auf die Nutzbarkeit (Useablitiy) des Gesamtsystems aus.

Für die Untersuchungen anhand der beschriebenen Kriterien wurden folgende Systeme betrachtet:

- Relationale Datenbankmanagementsysteme (RDBMS) [Kre01]
- Semantic Web Technologien (SemTech) (vgl. Abschnitt 4.1.2)
- Modellbasierte Repräsentation (MR) (vgl. Abschnitt 4.3.1)
- Reine Graphdatenbanken (GDB) [Hun14]

Diese Systeme wurden anhand ihrer technologischen und funktionalen Beschreibungen, sowie repräsentativer Testszenarien bezüglich ihrer Eignung für die Wissensrepräsentation mit Fokus auf modellbasierte Entwicklungsprozesse bewertet. Die Testszenarien basierten dabei auf den obigen Kriteriendefinitionen und waren jeweils auf die Bewertung eines einzelnen Kriteriums abgestimmt. Dabei wurden jeweils gleiche Daten-, Informations- und Wissenseinheiten in diesen Systemen hinterlegt, die Auswirkungen von Datenstrukturänderungen und die verfügbaren Abfragesprachen betrachtet. Das Kriterium der einfachen Erweiterbarkeit kann lediglich subjektiv auf Basis von Erfahrungswerten bewertet werden.

Tabelle 5.1 zeigt die Ergebnisse der Untersuchungen. Es ist zu erkennen, dass alle vier betrachteten Systeme zur Repräsentation von Wissen grundsätzlich geeignet sind und daher auch dessen Vorstufen Daten und Informationen aufnehmen können. Grund hierfür ist die Möglichkeit der Repräsentation von gerichteten, gewichteten Graphen, wie sie nach Abschnitt 4.1.1 zur Repräsentation von Wissen notwendig sind. Während relationale Datenbankmanagementsysteme dafür Relationen zwischen Tabellen nutzen, können alle übrigen Systeme Graphen direkt verarbeiten.

RDBMS unterstützen keine dynamische semantische Referenz. Das heißt, es existiert kein übergeordnetes System, um semantische Beziehungen zu interpretieren und zur Laufzeit

	RDBMS	SemTec	MR	GDB
Datenpersistenz	Ja	Ja	Ja	Ja
Informationspersistenz	Ja	Ja	Ja	Ja
Wissenspersistenz	Ja	Ja	Ja	Ja
Dynamische Semantische Referenz	Nein	Ja	Ja	Ja
Dynamische Typisierung von Beziehungen	(Ja)	Nein	Nein	Ja
Strukturierte Abfragesprache	SQL	SPARQL	z.B. ModelQuery	z.B. Cypher
Einfache Erweiterbarkeit	Nein	Nein	Ja	Nein
Summe der positiven Bewertungen	5/7	5/7	6/7	6/7

Tabelle 5.1: Bewertung von Wissensrepräsentationen

zu aktualisieren. Das Datenbankschema, das einer RDBMS Datenbank zugrunde liegt, ist zwar prinzipiell veränderbar - in der Praxis führt dies aber zu starken Änderungen an den Tabellenstrukturen, was eine automatische Anpassung anfällig für Fehler macht. Bei Semantic Web Technologien dienen Ontologien (vgl. Abschnitt 4.1.2) als dynamische semantische Referenz. Das Hinzufügen von neuen Informationen führt dabei in der Regel nicht zu Veränderungen in vorhandenem Wissen, sondern erweitert dieses, um bisher nicht bekannte Informationen. Ähnlich verhält es sich bei modellbasierten Repräsentationen. Das Modell als zentrale Referenz kann zur Laufzeit erweitert werden, was die Speicherung neuere Wissensstrukturen ermöglicht, ohne vorhandene Strukturen zu beeinflussen. Reine Graphdatenbanken besitzen keine zentrale Referenz im eigentlichen Sinne. Die Bezeichnung von Beziehungen ist für jedes einzelne Tripel frei wählbar und die Interpretation ist von der nutzenden Instanz zu realisieren. Dadurch ergibt sich ein Höchstmaß an Flexibilität, was jedoch zu Lasten höherer Komplexität bei der Nutzung erreicht wird.

Ähnliches gilt für die dynamische Typisierung von Beziehungen. Während der Typ bzw. die semantisch definierte Bezeichnung einer Beziehung für semantische Web Technologien und modellbasierte Repräsentationen in der semantischen Referenz fest definiert sein muss und damit die flexible Erweiterung um neue Typen von Beziehungen nur eingeschränkt möglich ist, ist dies bei reinen Graphdatenbanken nicht der Fall. Hier können neue Beziehungen ohne vorherige Anpassungen eingepflegt werden. Dies ist auch bei RDBMS Datenbanken der Fall, da Beziehungen hier über Referenzen in Tabellenzellen typisiert werden. Dadurch kann ein Tabelleneintrag frei gewählt und ein neuer Beziehungstyp generiert werden. Dies ist jedoch nicht gleich zu setzen mit Graphdatenbanken, da die Auswertung dieser Beziehungen nicht durch das Wissensmanagementsystem vorgenommen werden kann, sondern hierfür die explizite Erweiterung von nutzenden Applikationen notwendig ist.

Da es sich bei allen untersuchten Systemen im Kern um Datenmanagementlösungen handelt, stellt jede der Lösungen eine strukturierte Abfragesprache für den Zugriff auf gespeicherte Daten zur Verfügung. Während SQL die explizite Beschreibung von Suchort, Suchbedingung und Struktur des Ergebnisdatensatzes erfordert, sind die übrigen Abfragesprachen eher an der Struktur von Wissen orientiert. Dabei werden Subjekt, Prädikat und Objekt einer Wissenseinheit beschrieben, wobei jedes der Elemente auch durch generischer Platzhalter ersetzt werden kann. Eine Suche liefert dabei in der Regel die Ergebnisse für diese

generischen Platzhalter. So könnte eine Anfrage lauten: (Fahrzeug, lieferbarInFarbe, Grün) - das Ergebnisse wäre anschließend eine boolesche Aussage Ja oder Nein. Alternativ wäre die Anfrage möglich: (Fahrzeug, lieferbarInFarbe,?), wobei „?" einen generischen Platzhalter darstellt. Der Ergebnisdatensatz bestände in diesem Fall aus einer Liste aller Farben, in der ein Fahrzeug lieferbar ist.

Die Beurteilung der einfachen Erweiterbarkeit der Systeme unterscheidet sich je nach Einsatzgebiet. Die hier durchgeführte Untersuchung stützt sich auf voraussetzbares Wissen im Bereich der Funktionsentwicklung im Automobilbau. Die Untersuchungen und Erfahrungen aus der Tätigkeit in diesem Bereich zeigen, dass es insbesondere als schwierig gesehen wird, Ontologien dauerhaft durch Ingenieure pflegen zu lassen, da hierfür fundiertes Fachwissen und vorherige Weiterbildungen notwendig sind. Diese Anpassungen sind beim Einsatz von Systemen mit semantischen Technologien jedoch heute Stand der Technik (vgl. Abschnitt 4.1.2). Ähnlich verhält es sich bei RDBMS und Graphdatenbanken, da die freie Strukturgestaltung dieser Ansätze das Wissen über das Datenbankschema bzw. das dahinter liegende Datenmodell voraussetzt. Die modellbasierte Repräsentation erlaubt es Ingenieuren Anpassungen am Datenmodell in für sie nachvollziehbarer Form durchzuführen. Die Gründe hierfür liegen dabei im modellbasierten Entwicklungsprozess der Funktionsentwicklung (vgl. Abschnitt 3.1) und der damit verbundenen Ähnlichkeit der Modellierungsmethoden. Die Arbeit mit Modellen, wie UML oder Ecore (vgl. Abschnitt 4.3.1), ist zentraler Bestandteil der Arbeit. in modellbasierten Entwicklungsprozessen. Somit kann die Bedeutung vorhandener Strukturen erschlossen und deren gezielte Erweiterung durchgeführt werden.

5.4.3 Konzept der Wissensrepräsentation

Ausgehend von den Untersuchungen in Abschnitt 5.4.2 wurde im Rahmen dieser Arbeit ein Konzept entwickelt, das unterschiedliche Qualitäten von Wissensstrukturen berücksichtigt. Wie in Abschnitt 5.2 am Beispiel von automobiler Funktionentwicklung beschrieben, können Informationen direkt mit Modellen in Verbindung stehen oder über indirekte Beziehungen hergeleitet werden. Die Analyse beider Informationstypen zeigte, dass direkte Informationen in der Regel klare Strukturen im Bezug auf Quellen, Beziehungen und Verfügbarkeit aufweisen, während indirekte Informationen unscharf, bedingt und unvollständig verfügbar sind.

Direkte Informationen werden heute in Form von manueller Dokumentation im Prozess erfasst und bilden somit den direkten Kontext eines Entwicklungsartefaktes ab. Das hierfür entwickelte Konzept nutzt die modellbasierte Repräsentation von Wissensstrukturen, um einen minimalen Modellkontext im Sinne des Abschnitt 5.1 zu konstruieren. Dieser minimale Modellkontext ändert sich in der Regel selten, nämlich dann, wenn sich der konkrete Entwicklungsprozess eines Artefaktes ändert.

Die Gründe der Auswahl der modellbasierten Repräsentation liegen zunächst in der einfachen Instanziierbarkeit definierter Wissensstrukturen für einzelne Entwicklungsartefakte. Dadurch lassen sich stets gleiche Datenstrukturen je Entwicklungsartefakt auf Basis eines vorbereiteten Kontextmodells generieren und befüllen. Dieses Kontextmodell wird mit Hilfe einer Metasprache (vgl. Abschnitt 4.3.1) beschrieben und gepflegt. Prinzipiell kann dabei für jedes Entwicklungsartefakt ein eigenes Kontextmodell erstellt werden. Als sinnvoll hat sich jedoch die Nutzung eines globalen Modellkontext-Modells erwiesen, da dadurch die Dokumentation standardisiert werden kann. Ein weiterer Grund liegt in der einfachen

Erweiterbarkeit dieser Repräsentationsform. Die vorliegende Arbeit konzentriert sich auf die Unterstützung modellbasierter Entwicklungsprozesse. Dadurch kann für die Anwender ein Grundwissen über die Erstellung, Nutzung und Manipulation von modellbasierten Repräsentationen, seien es beispielsweise UML-, EMF- oder Simulinkmodelle, vorausgesetzt werden. Es ist leicht erkennbar, dass sich beim Einsatz einer modellbasierter Repräsentation für einen minimalen Modellkontext für die Anwender eine direkte Nutzbarkeit oder wenigstens eine kurze, steile Lernkurve ergibt. Dadurch können die Anwender einerseits den Informationsfluss in der Wissensrepräsentation nachvollziehen und andererseits notwendige Anpassungen des Modellkontext selbstständig durchführen, ohne Wissen über die Implementierungsdetails der Wissensbasis zu benötigen. Ein weiterer wichtiger Grund der modellbasierten Repräsentation liegt in der Verfügbarkeit von Code-Generierung auf Basis des Modells (vgl. Abschnitt 4.3.1). Diese wird im Folgenden genutzt, um das Adapterkonzept zur Informationsextraktion aus dem Prozess zu ermöglichen, welches in den nächsten Abschnitten erläutert wird. Dadurch ergibt sich eine einfache Erweiterbarkeit und Anpassbarkeit an sich ändernde Entwicklungsprozesse.

Abbildung 5.4: Metasprache minimaler Modellkontext

Abbildung 5.4 verdeutlicht die Nutzung einer modellbasierten Repräsentation für die Abbildung des minimalen Modellkontext. Voraussetzung dafür ist eine Metasprache, die es erlaubt Klassen von Informationen zu definieren. Dargestellt sind diese als Rechtecke. Innerhalb dieser Klassen, die durch einen Bezeichner identifiziert ist, sind Attribute zu spezifizieren, die die Klasse näher definieren. Dabei besteht ein Attribut stets aus einem Bezeichner und einem Datentyp. Mit der Instanziierung der Klasse wird das Attribut mit einem Wert entsprechend des gewählten Datentyps instanziiert. Jede Klasse enthält stets ein Attribut mit der Bezeichnung „UUID", den so genannten „Universally Unique Identifier", das zur Definitionszeit der Klasse einen im Gesamtsystem eindeutigen alphanumerischen Wert zugewiesen bekommt. Diese eindeutige Identifikation wird im Folgenden genutzt, um Informationsklassen zu identifizieren. Darüber hinaus muss die Metasprache die Definition von Beziehungen zwischen Klassen ermöglichen. Diese werden als Kanten (gerichtete Verbindungslinien) zwischen den Klassen dargestellt und besitzen einen Bezeichner, eine Richtung (eine Beziehung von einer Klasse A zu einer Klasse B ist mit der Pfeilspitze bei Klasse B definiert) sowie eine Kardinalität. Eine Besonderheit bei der modellbasierten Repräsentation von Entwicklungsartefakten ist die überwiegend hierarchische Strukturierung der Informationsklassen. Ein Modellkontext weißt, wie bereits aus Abschnitt 5.1 ersichtlich, in der Regel eine Baumstruktur auf, die mit Hilfe eines dedizierten Wurzelknotens die semantische Verknüpfung zwischen dem Entwicklungsartefakt und seinem Kontext herstellt. Wie im Folgenden näher vertieft wird, existieren im Baum jedoch auch Querbeziehungen zwischen den Ästen. Somit kann die Baumstruktur lediglich als hierarchische Ordnung der Knoten betrachtet werden.

Mit der Beschreibung eines Modells unter Nutzung einer Metasprache mit den oben genannten Eigenschaften lassen sich die direkten Informationen von Entwicklungsartefakten

abbilden. Für indirekte Informationen wird das vorgestellte Konzept um eine Graphdatenbank ergänzt. Wie bereits erläutert, kann die Verfügbarkeit von indirekten Informationen stark schwanken. Dies kann zeitliche oder prozessuale Gründe haben oder auch vom jeweiligen Charakter der nutzenden Anwendung abhängig sein. Reicht die Nutzung eines Entwicklungsartefaktes beispielsweise von einer frühen Phase im Entwicklungsprozess bis zu dessen Ende, ist es möglich, dass Informationen, die für die spätere Verwendung relevant sind, in der frühen Phase noch gar nicht zur Verfügung stehen. Darüber hinaus ist es nicht möglich alle Informationen, die Entwicklungsartefakte beschreiben können, als Klasse oder in Form einer semantischen Referenz vor deren Auftreten zu definieren, da Informationen gegebenenfalls auch erst durch Änderungen von Rahmenbedingungen relevant werden können.

Die in Abschnitt 5.4.2 vorgestellte Analyse zeigte, dass für dynamische Wissensstrukturen ohne die Notwendigkeit einer semantischen Referenz insbesondere reine Graphdatenbanken geeignet sind. Ausgehend von einem minimalen Modellkontext auf Basis von modellbasierter Repräsentation und der Notwendigkeit der Vernetzung von indirekter Information, entstand das hybride Wissensrepräsentationskonzept dieser Arbeit. Im Folgenden wird der Begriff Graphdatenbank stellvertretend für das entsprechende Konzept der Architektur genutzt. Die Graphdatenbank des Konzeptes bildet das Gesamtwissen des modularen Informationsmanagementsystems ab. Sie beinhaltet dabei nicht nur die indirekten Informationen, sondern nutzt die vorgegeben, strikten Strukturen der modellbasierten Repräsentation als semantische Referenz des direkten Wissens

Für die Erläuterung des Aufbaus der Gesamtwissensbasis für ein Entwicklungsartefakt wird zunächst vorausgesetzt, dass gemäß der obigen Erläuterungen ein minimaler Modellkontext bestehend aus einem allgemeinen Informationsmodell und einer Instanziierung des Modells für das entsprechende Entwicklungsartefakt existiert. Im ersten Schritt wird dieser minimale Modellkontext in die Graphdatenbank eingefügt. Dabei wird eine Graphtransformation in Anlehnung an die Arbeit von [BK14] durchgeführt. Das dort vorgeschlagene Vorgehen bei der Transformation einer modellbasierten Repräsentation in eine Graphdatenbank bildet nicht nur die reine Information, sondern auch die semantische Referenz ab. Dabei werden zunächst die Klassen der modellbasierten Repräsentation (vgl. Abbildung 5.4) in Knoten des Graphen umgewandelt. Jeder Knoten wird durch die vergebene UUID eindeutig identifiziert und erhält auch die übrigen Attribute der jeweiligen Klasse. Anschließend existiert für jede Informationsklasse ein repräsentierender Knoten im Graph. Um später redundante Informationen im Graph zu vermeiden, werden die Beziehungen zwischen Informationsklassen nicht transformiert, da diese in einer Graphrepräsentation für jeden Knoten individuell erzeugt werden müssen. Der Grund ist die fehlende Fähigkeit zusammengehörige Instanzen von Knoten in Graphen zu kapseln. Konkret bedeutet dies, dass bei der Instanziierung eines Modells in der modellbasierten Repräsentation alle Instanzen der Informationsklassen zu dieser Instanz des Modell zugeordnet werden können. Die Klassenbeziehungen sind in diesem Fall eindeutig den Instanzen zuzuordnen. In der Graphrepräsentation einer Informationsklasse stehen alle Instanzen dieser Klasse unabhängig von ihrer Modellinstanz in direkter Beziehung zum Klassenknoten. Somit muss die Zusammengehörigkeit der Instanzen zu einer Modellinstanz durch die jeweiligen Kanten zwischen den Instanzen der Klassenknoten repräsentiert werden. Die Transformation der Instanzen der modellbasierten Repräsentation wird in ähnlicher Form, wie die der Klassen durchgeführt. Zunächst werden die Instanzen der Informationsklassen in Graphknoten transformiert, die, wie ihre zugehörigen Klassen, eine UUID sowie ihre jeweiligen Attribute enthalten. Darüber hin-

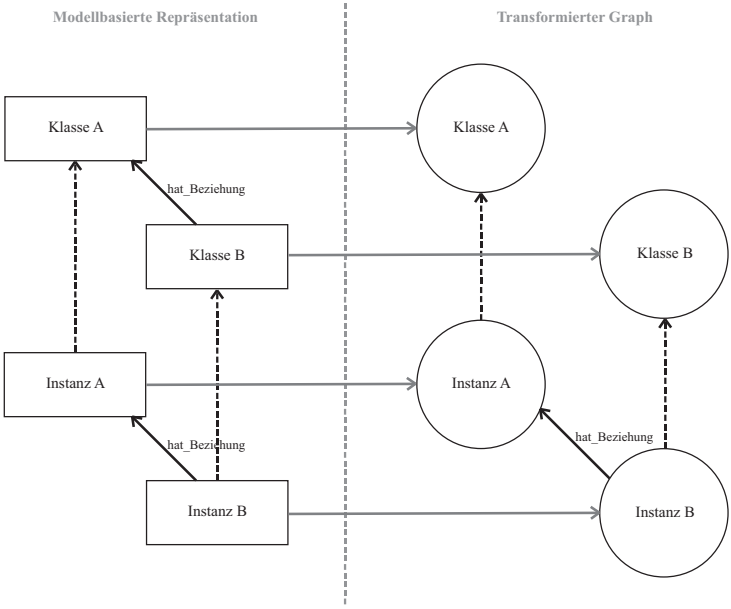


Abbildung 5.5: Graphtransformation des minimalen Modellkontext

aus wird eine Instanzbeziehung mit der Bezeichnung „is_a" - zu deutsch „ist eine Instanz der Klasse" - zum zugehörigen Klassenknoten dieser Instanz erzeugt. Anschließend werden die Beziehungen der Modellinstanzen auf Beziehungen zwischen den Instanzenknoten transformiert. Dabei wird Richtung, Bezeichnung und Kardinalität der Beziehung berücksichtigt. Abbildung 5.5 verdeutlich die Schritte der Transformation schematisch.

Nach Abschluss der Transformation enthält der Graph den minimalen Modellkontext zusammen mit der semantischen Referenz des dahinter liegenden Informationsmodells. Er bildet somit für ein einzelnes Entwicklungsartefakt eine vollständige Wissensbasis bezogen auf den minimalen Modellkontext. Es ist daher leicht erkennbar, dass diese Transformation im Bedarfsfall auch umkehrbar ist. Dafür sei an dieser Stelle auf [BK14] verwiesen. Die Transformation in eine vollständige, graphbasierte Wissensbasis ermöglicht, wie bereits aus den Untersuchungen in Abschnitt 5.4.2 hervorgeht, die einfache Erweiterbarkeit der Wissensbasis. Wie im Folgenden gezeigt wird, gilt dies nicht nur in Bezug auf indirekte Informationen, sondern insbesondere auch für die direkten Informationen.

Die semantische Referenz der modellbasierten Repräsentation des minimalen Modellkontext ändert sich, wie bereits erwähnt, nur selten. Mit Hilfe der Graphtransformation des Wissensmanagementkonzeptes der vorliegenden Arbeit wirkt sich eine Änderung jedoch nicht auf die Integrität der vorhanden Modellkontexte aus. Eine Erweiterung des zugrunde liegenden Modells um neue Informationsklassen führt zur Generierung neuer Klassenkno-

ten im Graph, die für neu hinzukommende Modellkontexte instanziierbar sind. Da jedoch Beziehungen zwischen Klassen nicht transformiert werden, wirken sich diese Änderungen nicht auf bestehende Strukturen aus. Ähnlich verhält es sich bei der Entfernung von Informationsklassen. Für vorhandene Modellkontexte werden diese im Graph behalten und für neue Modellkontexte nicht weiter instanziiert. Dies erlaubt die implizite Versionierbarkeit von semantischen Referenzen je Modellkontext. Die Unabhängigkeit der Modellkontexte verdeutlicht Abbildung 5.6. Modellkontext A stellt dabei die Ausgangssituation dar, während Modellkontext B eine Reduzierung und Modellkontext C eine Erweiterung des ursprünglichen Modells zeigen. Es ist erkennbar, dass die damit einhergehenden Änderungen keine Auswirkung auf den Modellkontext A implizieren.

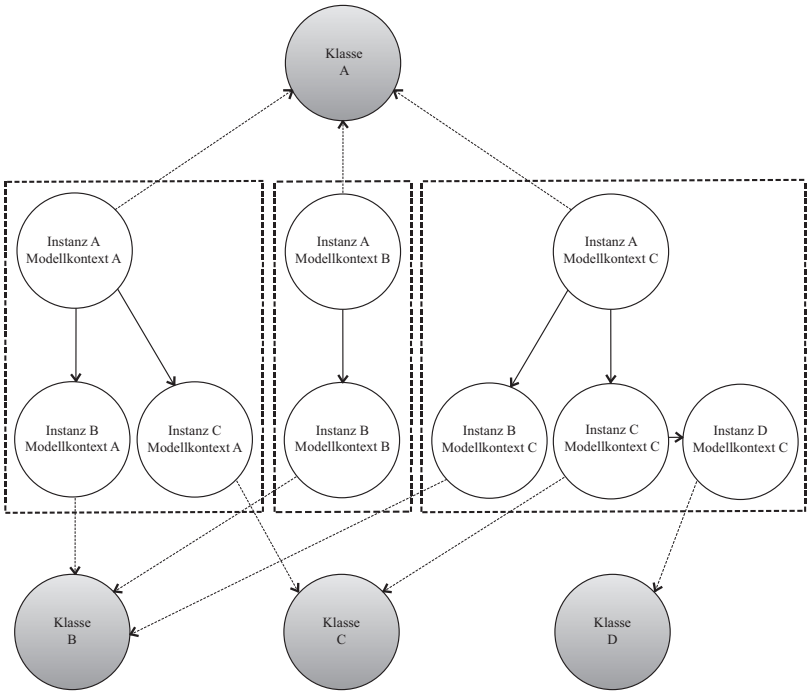


Abbildung 5.6: Wissensintegrität bei Modelländerungen

Die Erweiterung des minimalen Modellkontext um indirekte Informationen stellte eine der Herausforderungen der vorliegenden Arbeit dar. Das entwickelte Wissensmanagementkonzept nutzt dabei die Möglichkeiten der graphbasierten Wissensrepräsentation ohne verfügbare semantische Referenz. Dafür ist es notwendig Voraussetzungen für das Einpflegen neuer Informationen zu vereinbaren, um semantisch nicht explizit beschriebene Informationen wiederverwenden zu können. Folgende Voraussetzungen sind für die Erweiterung der Wissensbasis notwendig:

- Es existiert ein dedizierter Wurzelknoten einer jeden Modellkontextinstanz, die die Verknüpfung zum eigentlichen Entwicklungsartefakt herstellt.
- Semantisch stellt der Wurzelknoten stets das Subjekt einer jeden Information dar, die mit dem Entwicklungsartefakt direkt verknüpft werden soll. Zu beachten ist an dieser Stelle, dass indirekte Informationen durchaus in sich graphstrukturen enthalten können - die hier definierte Vereinbarung gilt lediglich für die gerichtete Beziehung vom Modellkontext zur neu verknüpften Information.
- Die Interpretation einer Beziehung qualifiziert stets eine Eigenschaft des Entwicklungsartefaktes.

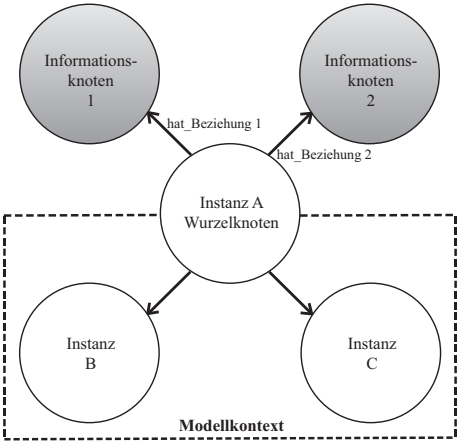


Abbildung 5.7: Vernetzung indirekter Information

Die Einhaltung dieser Voraussetzungen erlaubt es, sowohl das Hinzufügen neuer Informationen als auch für deren Nutzung zu automatisieren. Die dafür notwendigen Konzepte werden in den Folgeabschnitten näher erläutert. Schematisch fasst Abbildung 5.7 die Vernetzung von indirekten Informationen mit vorhandenen Modellkontexten noch einmal zusammen.

5.5 Informationsextraktion im Entwicklungsprozess

Modellbasierte Entwicklungsprozesse finden heute in vielen Industriezweigen Anwendung (vgl. Abschnitt 4.3). Die steigende Nutzerzahl und die zunehmende Komplexität der Entwicklungsaufgaben führt zum Einsatz verschiedenster Werkzeuge zur Unterstützung und Absicherung eines zielgerichteten Entwicklungsprozesses. Diese Werkzeuge enthalten die für die Ausführung des Prozesses notwendigen Informationen. Deren Nutzung beschränkt sich in der Regel auf die konkreten Arbeitsaufgaben, während ihr potenzieller Nutzen für andere oder parallele Entwicklungsaufgaben nicht erkannt wird.

Eine der zentralen Aufgaben des im Rahmen dieser Arbeit vorgestellten Informationsmanagementsystems ist die Extraktion von Informationen genau da, wo sie im Entwicklungsprozess entstehen. Um diese Informationen zielgerichtet abzugreifen und der Wissensbasis zuzuführen, wurde ein Adapterkonzept für die Informationsextraktion entwickelt. Das vorliegende Kapitel stellt die dafür notwendigen Untersuchungen sowie das daraus abgeleitete Adapterkonzept vor.

5.5.1 Adapterkonzepte für die Datenanbindung

Das Entwurfsmuster der Adapter ist eines der verbreitetsten Softwarepattern, die von [Gam95] beschrieben wurden. Es kommt immer dann zum Einsatz, wenn Schnittstellen unterschiedlicher Softwaresysteme verbunden werden müssen. Ein Adapter fungiert dabei als als Übersetzer zwischen zwei oder mehr Schnittstellen. Häufigster Einsatzzweck ist die Anbindung mehrerer Softwaresysteme an ein integrierendes System. In der Regel stellt dieses integrierende System eine im Rahmen der Anwendung als standardisiert zu betrachtende Schnittstelle zur Verfügung, die von Adaptern bedient werden soll. Adapter übersetzen anschließend die individuellen Schnittstellen der anzubindenden Systeme auf die Standardschnittstelle.

Die Komplexität eines Adapters hängt von den genutzten Systemen sowie den verfügbaren und benötigen Datenqualitäten ab. Im einfachsten Fall handelt es sich bei einem Adapter um das Routing von Daten zwischen zwei Schnittstellen. Unterscheiden sich beispielsweise nur die Bezeichnungen für Daten zwischen Quell- und Zielsystem, so übernimmt der Adapter die Umbenennung und die Zuführung der Daten. Unterscheidet sich hingegen die erwartete Qualität des Zielsystems vom Quellsystem, so können einzelne Adapter durchaus komplexe Algorithmen implementieren.

Ein konkretes Beispiel hierfür ist die Erwartung strukturierter Daten im Zielsystem und die Verfügbarkeit unstrukturierter Daten im Quellsystem (vgl. Abschnitt 4.5.1). Dies könnten im Quellsystem Fließtexte sein, wobei für das Zielsystem lediglich qualifizierte Informationen, wie Datumsangaben, Personen oder Bezeichnungen relevant sind. Hier kommt dem Adapter die zusätzliche Aufgabe der Datenextraktion und -aufbereitung zu.

Für die Nutzung im Rahmen des modularen Informationsmanagementsystems sind eher komplexe Adapter zu erwarten, da die Quellsysteme der Entwicklungsprozesse heute in der Regel keine Nutzung von Dokumentationsinformation durch externe Systeme berücksichtigen. Aus diesem Grund wurden im Rahmen der Entwicklung der Konzepte Methoden der Informationsextraktion für unterschiedliche Typen von Quellen untersucht und hinsichtlich der Eignung bewertet.

5.5.2 Untersuchte Lösungsansätze der Informationsextraktion

Die Untersuchung von Lösungsansätzen der Informationsextraktion aus datenhaltenden Systemen in modellbasierten Entwicklungsprozessen zeigte, dass sich die Typen der Informationsextraktion grundlegend in drei Kategorien einteilen lassen - Transformation, Filterung und Aggregation. Jede der Kategorien benötigt unterschiedliche Lösungsansätze die im Folgenden vorgestellt werden.

Transformation Die Transformation setzt eine strukturierte Datenquelle voraus, auf deren Daten über eine definierte Schnittstelle zugegriffen werden kann. Aus dieser Datenquelle sollen einzelne Daten zielgerichtet exportiert und in ein Datenformat transformiert werden, welches das Zielsystem aufnehmen kann. Die Transformation kann dabei die Anpassung von Bezeichnungen, Formatierungen oder auch die Umwandlung von Datentypen sein. Beispielsweise könnte im Quellsystem ein Zeitstempel als Datumsformat (Tag.Monat.Jahr - Stunde:Minute:Sekunde) im Datentyp String vorliegen. Ein Zielsystem könnte ihn im Unix-Timestamp Format als Integer erwarten. Das Quelldatenformat kann dabei als unveränderlich angesehen werden. Eine Datenumwandlung, wie die Anpassung des Datenformates, lässt sich unter dieser Voraussetzung durch algorithmische Implementierung der Transformation umsetzen. Anschließend hat sich die Überführung der Daten in ein quell- und zielsystemneutrales Transportformat, wie beispielsweise XML, als praktikabel erwiesen. Dies erlaubt die implementierungsneutrale Beschreibung des gewünschten Datenformates inklusive der gewünschten Datentypen. Dieses Transportformat kann anschließend an das Zielsystem übergeben werden und muss von diesem interpretiert werden.

Filterung Die Filterung von Datenquellen ist in der Regel notwendig, wenn es sich um wenig oder gar nicht strukturierte Daten handelt. Insbesondere bei Fließtexten in Quellsystemen wird eine Filterung nach bestimmten Inhalten notwendig, wenn daraus strukturierte Informationen extrahiert werden sollen. Die Filterung selbst lässt sich in Schlagwortsuche, Mustersuche und Faktenextraktion einteilen.

Eine Schlagwortsuche nutzt eine Wortbasis, die gewünschte Begriffe beinhaltet, nach denen gesucht werden soll. Diese Wortbasis kann eine einfache Listung der Schlagworte oder auch strukturiert und mit semantischen Informationen angereichert sein. Letzteres wird beispielsweise durch die Nutzung von Thesauri (vgl. Abschnitt 4.1.4) unterstützt. Die Schlagwortsuche selbst besteht anschließend aus einem linearen Durchlaufen der Wörter eines Fließtextes und der Ausführung eines String-Vergleichs mit den Elementen der Schlagwortbasis. Dieser Vorgang wird als „Parsing" bezeichnet. Wird ein Element der Schlagwortbasis gefunden, so kann der Text mit diesem Schlagwort in Verbindung gebracht und dadurch inhaltlich genauer klassifiziert werden.

Eine Mustersuche nutzt Musterbeschreibungssprachen (Pattern Languages), um Zeichenketten bzw. deren Strukturen abstrakt zu beschreiben. Gibt man beispielsweise im deutschen Sprachraum die Zeichenkette TT.MM.JJJJ an, so ist es den meisten Menschen möglich zu schlussfolgern, dass es sich um einen Platzhalter für ein Datum mit der entsprechenden Formatierung (4-stelliger Jahreszahl) handelt. Ähnlich ist das Vorgehen bei der Mustersuche in Fließtexten. Als Standardmethode der Musterbeschreibung haben sich heute sogenannte reguläre Ausdrücke (Regular Expressions) durchgesetzt. In der Industrie und insbesondere in Entwicklungsprozessen sind an verschiedenen Stellen Standards oder etablierte Formate für Bezeichnungen zu finden, die sich mit regulären Ausdrücken klassifizieren lassen. Neben Datumsangaben können dies Projektbezeichnungen, Abteilungsbezeichnungen, Teilenummern usw. sein. Sind die regulären Ausdrücke zur Textklassifikation identifiziert, verläuft die Mustersuche äquivalent zur Schlagwortsuche, indem der Fließtext durchlaufen und die Anwendbarkeit der regulären Ausdrücke auf die Zeichenketten überprüft wird. Wie leicht zu erkennen ist, lassen sich dadurch zielgerichtet Informationen aus Texten extrahieren.

Die Faktenextraktion unterscheidet sich grundlegend von der Schlagwort- und der Mustersuche. Ziel ist es durch die Analyse von Texten semantisch korrekte Aussagen zu extrahieren, welche Fakten genannt werden. Zum Einsatz kommen hierbei Methoden des Forschungsgebietes „Natural Language Processing", das sich mit der Analyse und Verarbeitung natürlichsprachlicher Texte befasst. Der Vorteil gegenüber Schlagworten und Mustern besteht darin, dass es möglich ist auch relevante Fakten aus Texten zu erhalten, die zuvor nicht bekannt und in einer semantischen Basis hinterlegt waren. Um die damit verbundenen Erwartungen für die Informationsextraktion zu verstehen, werden diese zunächst anhand eines Beispiels erläutert. Dafür wird der folgende Textausschnitt „Ingenieur A ist verantwortlich für die Umsetzung von Projekt X, während Ingenieur B sich nicht um Projekt X, sondern Y kümmert" betrachtet. Bei hinreichender Standardisierung der Formatierung von Namens- und Projektbezeichnungen, lässt sich der Text mit Hilfe der Mustersuche als relevant für die Ingenieure A und B sowie die Projekte X und Y klassifizieren. Wie diese jedoch in Verbindung stehen, nämlich durch die jeweiligen Verantwortlichkeiten, lässt sich auf diese Weise nicht herausfinden. Das Vorgehen bei der Faktenextraktion hat zum Ziel vollständig qualifizierte Informationen (vgl. Abschnitt 4.5.1) zu extrahieren. Als Analyseergebnis könnten hier beispielsweise die folgenden Fakten zur Wissensbasis hinzugefügt werden: (A, verantwortet, X), (B, verantwortet, Y), (B, verantwortet nicht, X). Da die mit dieser Methodik verbundene Mächtigkeit bei der Analyse von Datenquellen für automatische Dokumentation von Entwicklungsartefakten von besonderer Bedeutung ist, wurde die Anwendbarkeit im Rahmen der Untersuchungen dieser Arbeit eingehend betrachtet. Das Vorgehen sowie die Ergebnisse werden im Folgenden vorgestellt.

Die Methoden des Natural Language Processing (NLP) [Gur13] benötigen heute eine mehrstufige, sequentielle Verarbeitung der Eingabetexte. Die Ausführung dieser Schritte durch Hintereinanderschaltung der benötigten Algorithmen wird als Pipeline bezeichnet. Abbildung 5.8 zeigt den Aufbau einer solchen Pipeline exemplarisch, wie er im Rahmen der vorliegenden Arbeit untersucht wurde. Je nach Anwendungsfall kann die Pipeline mehr oder auch weniger spezifische Stufen enthalten. Die Verarbeitung eines Eingabetextes beginnt stets mit dem sogenannten Preprocessing. Dabei werden die Texte mit Hilfe von Tags, syntaktischen Hilfsmitteln die als Codeworte in den Text eingefügt werden, annotiert. Die Tokenisation durchläuft den Text und markiert jedes Wort des Textes als zusammenhängende Zeichenkette. Damit sind für spätere Phasen Worte, d.h. zusammenhängende Zeichenketten, als sogenannte Token identifiziert. Es folgt das Sentence Splitting. Dabei wird der Text in einzelne Sätze zerlegt, die in der Regel anhand von Satzzeichen getrennt werden. Als letzter Schritt des Preprocessing folgt das sogannte POS-Tagging. POS steht dabei für „Part-of-Speech" und bezeichnet die Identifikation der syntaktischen Rolle einzelner Token. Dadurch werden beispielsweise Subjekt, Prädikat und Objekt eines Satzes identifiziert. Für die folgenden Stufen werden sogenannte Sprachmodelle benötigt, die in der Regel in Form von Ontologien (vgl. Abschnitt 4.1.2) vorliegen. Diese Sprachmodell repräsentieren typische Worte, Satzmuster und Textstrukturen einer Sprache. Diese Sprachmodelle unterscheiden sich jedoch auch innerhalb einer Sprache nach Anwendungsdomänen. Texte mit technischem Hintergrund, medizinische Texte oder literarische Texte unterscheiden sich hier deutlich. Aktuell existieren vorwiegend Modell im medizinische Umfeld. Im technischen Umfeld, in dem sich diese Arbeit eingliedert, liegen aktuell keine bekannten Sprachmodelle vor. Aus diesem Grund diente ein generisches Modell der deutschen Sprache [Dar14] hier als Grundlage.

Die im Preprocessing annotierten Texte dienen als Eingangsdaten der nächsten Pipelinestufe Grammatik. Diese identifiziert so genannte „Named Entities", d.h. Objekte die einen Eigennamen besitzen, wie Straßen, Personen, Tätigkeiten. Mit Hilfe der „Fact Recognition" werden Subjekt-Prädikat-Objekt-Beziehungen identifiziert und im Text annotiert. Im Ergebnis können Fakten auf die reinen Basisinformationen reduziert werden. Beispielsweise lässt sich dadurch aus dem Text „Der Mitarbeiter Mustermann der Fachabteilung A ist verantwortlich für die Entwicklung des Steuergerätes . Er benötigt dafür 14 Tage." der Fakt extrahieren [Mustermann, ist verantwortlich, Steuergerät B]. Es ist leicht zu erkennen, dass dieses Format dem Aufbau von Informationen (vgl. Abschnitt 4.1) entspricht. Aus identifizierten Fakten eines Textes müssen daher anschließend die Fakten mit relevanten Informationen gefunden werden. Dafür folgt in der Pipeline zunächst die Stufe der Normalisierung. Diese nutzt ebenfalls das Sprachmodell, um Doppeldeutigkeiten und Abkürzungen. Dabei werden unter anderem Worte gleicher Bedeutung auf eine definierte Bedeutung angepasst. Dabei kann beispielsweise das Prädikat „ist verantwortliche" des Beispielsatzes geändert werden in „verantwortet". Somit lässt sich für eine spätere algorithmische Verarbeitung der Fakten eine Normalisierung des Wortschatzes durchführen. Der Normalisierung folgt die Phase der „Coreference Resolution". Darunter wird die Auflösung unterschiedlicher Bezeichner für das gleiche Individuum verstanden. Aus dem Beispieltext lässt sich ebenfalls der Fakt [Er, benötigt, 14 Tage] identifizieren. Die „Coreference Resolution" löst die Gleichheit zwischen „Mustermann" und „Er" auf, so dass der extrahierte Fakt anschließend [Mustermann, benötigt, 14Tage] ist. Mit Abschluss der Phase folgt das sogenannte Grounding, die algorithmische Verarbeitung der extrahierten Daten und das Einpflegen in eine zentrale Wissensbasis. Die Standardisierung des Wortschatzes mit Hilfe der Informationen einer zentralen Wissensbasis, erlaubt anschließend die Verknüpfung der Fakten mit bereits in der Wissensbasis befindlichen Informationen. Befinden sich beispielsweise bereits Informationen über Herrn Mustermann in der Wissenbasis, wie andere Projekte, die er verantwortet, kann als neue Information die Verantwortung für Steuergerät B eingepflegt werden.

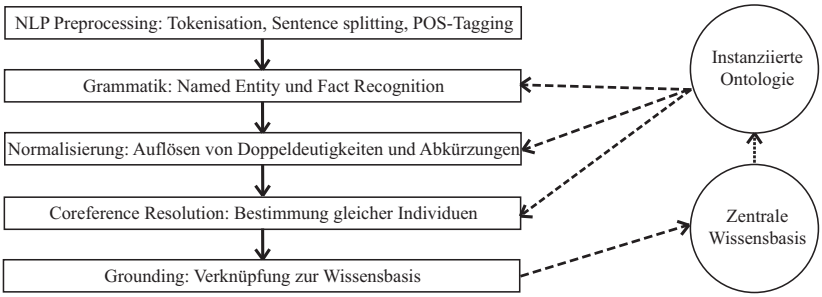


Abbildung 5.8: Natural Language Processing Pipeline

Das beschriebene Verfahren wurde im Rahmen der Arbeit insbesondere für Datenquellen mit Prosatexten untersucht. Zentraler Anwendungsfall waren Anforderungstexte für Entwicklungsartefakte. Als besonders hilfreich erwies sich die Named Entity Recognition, da diese auch mit dem generischen Sprachmodell zuverlässig beispielsweise Entwicklungsartefakt- und Projektbezeichnungen identifizierte. Als weniger geeignet erwies sich das genutz-

te Sprachmodell für die „Fact Recognition" sowie die „Coreference Resolution", da für diese Analysen insbesondere fachspezifische Anteile eines Sprachmodells notwendig sind. Im Einzelfall konnten auch hier positive Ergebnisse für die Informationsgewinnung erzielt werden. Die Entwicklung eines Sprachmodells für das technische Umfeld führt zu einer Verbesserung der Analyseergebnisse und damit auch der automatisierten Informationsextraktion. Dies ist jedoch nicht Teil der vorliegenden Arbeit und stellt einen Ansatzpunkt für weiterführende Untersuchungen dar.

Aggregation Die Aggregation von Informationen im Zuge der Extraktion bildet einen Sonderfall. Dabei werden in der Regel eine Vielzahl von Einzeldaten, die aus verschiedenen Datenquellen stammen können, zu einem neuen Datum oder einer Information zusammengefasst. Die jeweils genutzten Algorithmen hängen dabei stark vom Anwendungsfall ab. Im Fall des MIM findet diese Form der Informationsextraktion in der Qualitätsbewertung der Entwicklungsartefakte Anwendung. Allgemein lässt sich festhalten, dass die Aggregation immer dann notwendig ist, wenn Informationen aus einer Vielzahl von Daten zusammengefasst oder die vorhandenen Daten zur Erzeugung eines Mehrwertes für das Informationssystem vorverarbeitet werden müssen.

5.5.3 Datenabstraktion zum Informationsmodell

In Abschnitt 5.4.3 wurde das neuartige Konzept der Wissensrepräsentation durch die hybride Anwendung von modellbasierter und graphbasierter Repräsentation vorgestellt. In den Abschnitten 5.5.1 und 5.5.2 wurden Lösungsansätze für die Extraktion von Daten und Informationen aus unterschiedlichen Datenquellen erläutert. Damit sind sowohl die Gewinnung von Informationen, als auch das Zielsystem spezifiziert. Im Folgenden werden die entwickelten Konzepte zur Überführung der Informationen in die Wissensbasis erläutert. Die komplexe Herausforderung des modularen Informationsmanagementsystems liegt in der Anforderung der Anpassbarkeit an verschiedene und insbesondere an veränderliche Entwicklungsprozessstrukturen. Diese Anpassbarkeit bezieht sich nicht nur auf die strukturelle Anbindung von Datenquellen, sondern auch auf das dahinterliegende Datenmodell - die Wissensbasis.

Die Anpassbarkeit in Richtung der Datenquellen im Entwicklungsprozess wird durch die Verfügbarkeit entsprechender Adapterkomponenten für die in Abschnitt 5.5.2 beschriebenen Formen der Informationsextraktion gewährleistet. Ändern sich die Datenquellen, so ist ggf. die Entwicklung einer neuen Adapterkomponente notwendig. Wie in Abschnitt 5.4.3 bereits für die Anpassbarkeit des modellbasierten Datenmodells erläutert wurde, soll dessen Anpassung jedoch durch die Nutzer des Systems möglich sein. Somit muss eine Möglichkeit gefunden werden, wie diese Anpassung durch die Adapter mit Informationen versorgt werden kann. Eine weitere Herausforderung bestand in der Konzeption einer Schnittstelle für die Graphrepräsentation, die die Einhaltung der Abschnitt 5.4.3 vereinbarten Rahmenbedingungen bei gleichzeitig maximaler Flexibilität gegenüber den einzupflegenden Informationen sicherstellt.

Das unter den gegebenen Rahmenbedingungen entwickelte Konzept besteht aus einer zweiteiligen API. Sie unterteilt sich in eine dynamische und eine statische API. Ziel ist es, mittels

der dynamischen API dem sich ändernden Datenmodell der modellbasierten Repräsentation Rechnung zu tragen, während die statische API insbesondere für das strukturierte Einpflegen in den Graphen zuständig ist.

Dynamische API Die dynamische API bedient sich der Unterstützung von Code-Generierung auf Basis von Modellen im Rahmen der modellbasierten Softwareentwicklung. Jede Informationsklasse ist durch ihre zugehörige UUID (vgl. Abschnitt 5.4.3) eindeutig identifiziert. Ebenso verhält es sich mit der Identifikation der Instanz des Modellkontext eines Entwicklungsartefaktes. Voraussetzung ist die Existenz einer modellbasierte Repräsentation des minimalen Modellkontext. Mit Hilfe von Code-Generierung wird jede Informationsklasse zusammen mit ihren Attributen und Methoden für die definierten Klassenbeziehungen in das Datenformat der genutzten, als objektorientiert vorausgesetzte Basisplattform überführt. Konkret bedeutet dies die Erzeugung einer Klasse je Informationsklasse. Jede dieser Klassen ist wiederum über die gleiche UUID, wie seine Modellklasse identifiziert.

An dieser Stelle wird vorausgesetzt, dass mit Hilfe eines Adapters ein Informationsartefakt eines Entwicklungsartefaktes aus einer Datenquelle extrahiert wurde, das im Rahmen des minimalen Modellkontext relevant ist. Es ist leicht nachvollziehbar, dass die Zuordnung des Informationsartefaktes bei Kenntnis der UUID des Entwicklungsartefaktes sowie der UUID der Informationsklasse eindeutig die Zieldatenstruktur identifiziert. Dies nutzt die dynamische API, indem die Schnittstelle auf die Anfrage mit beiden UUIDs eine Instanz der Informationsklasse des Entwicklungsartefaktes zurückliefert. Diese Instanz enthält vorausgefüllt bereits vorhandene Informationsartefakte. Aufgabe des Adapters ist es anschließend sein Informationsartefakt in die Instanz einzupflegen und diese über die API zurückzuschreiben.

Die Identifikation über eindeutige Bezeichner erlaubt die Realisierung der als dynamisch bezeichneten Aspekte der API. Die Änderung des Datenmodells durch Anpassung der modellbasierten Repräsentation wird von der API vollständig transparent weitergegeben. Mit jeder Anpassung - sei es das Hinzufügen oder Entfernen einer Informationsklasse oder die Änderung der Attribute einer vorhandenen Klasse - geht ein Durchlauf der beschriebenen Code-Generierung einher. Dadurch ändert sich die Datenklasse der Zielplattform sowie deren Instanzen für neu generierte Modellkontexte. Die API bzw. die darüber kommunizierten Instanzen passt sich dem geänderten Datenmodell an. Abbildung 5.9 fasst die Funktionsweise der dynamischen API noch einmal zusammen.

Statische API Die statische API definiert die Schnittstelle zur graphbasierten Repräsentation der Wissensbasis. Wie in Abschnitt 5.4.3 beschrieben, muss diese die Erweiterung des minimalen Modellkontext um beliebige Beziehungen ermöglichen. Die API erlaubt die Abfrage vorhandener Beziehungen zwischen einem durch seine UUID identifizierten Modellkontext und den damit verknüpften, indirekten Informationen. Dies erlaubt es, bereits vorhandene Informationen hinsichtlich ihrer Prädikat-Objekt Beziehung zu bearbeiten. Darüber hinaus können Prädikat-Objekt Beziehungen neu generiert werden, um bisher nicht vorhandenen Informationsstrukturen abzubilden. In beiden Fällen wird die Prädikat-Objekt Beziehung durch die API zur Wissensbasis des durch seine UUID identifizierten Modellkontext hinzugefügt.

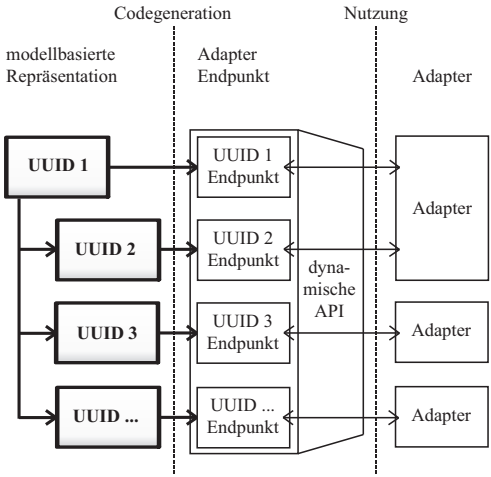


Abbildung 5.9: Funktionsweise der dynamischen API

5.6 Konzepte für die Nutzung der Wissensbasis

Dokumentation hat, wie bereits in Abschnitt 3.5 dargelegt, häufig den Charakter einer Datensenke, deren Nutzen für den Ersteller nicht unmittelbar sichtbar wird und daher häufig zu den weniger geschätzten Tätigkeiten zählt. Ein wichtiger Aspekt bei der Konzeption des modularen Informationsmanagementsystems war daher die Durchgängigkeit von der Generierung von Informationen zur Nutzung der entstandenen Wissensbasis. Dafür wurde das zu erwartende Nutzungsverhalten untersucht und die wichtigsten Anwendungsfälle identifiziert.

Die folgenden Abschnitte legen zunächst diese Anwendungsfälle dar und beschreiben anschließend die dadurch abgeleiteten Konzepte für die Wiederverwendung der in der Wissenbasis dokumentierten Informationen. Im Kern handelt es sich um zwei klassische Suchkonzepte - die Volltextsuche mit der Einschränkung über so genannte Facetten, sowie die Empfehlung von Suchergebnissen anhand definierter Kriterien, die zur Bewertung genutzt werden.

5.6.1 Identifikation der Anwendungsfälle

Die Informationsversorgung ist für Entwicklungsprozesse stets eine der wichtigsten Aufgaben, da diese die Grundvoraussetzung für die Durchführbarkeit darstellt. Für jede Information stellt sich hierbei die Frage der Bereitstellung. Es wird in der Regel das „Push"- und das „Pull"-Verfahren unterschieden.

Das „Push"-Verfahren bedeutet, dass ein Informationsnutzer (Senke) eine Information vom Informationsproduzenten (Quelle) zugeführt bekommt, sobald diese bei der Quelle verfüg-

bar ist. Dieses Verfahren ist zunächst an eine Liefervereinbarung zwischen Quelle und Senke gebunden. Für die Quelle bedeutet dies einen konstantem Aufwand für die Zuführung von Informationen, auch wenn diese gegebenenfalls von der Senke nicht oder noch nicht benötigt werden. Damit entsteht auf beiden Seiten Aufwand im Entwicklungsprozess, obwohl die Relevanz der Information nicht sichergestellt ist. Dieses Verfahren findet daher nur in strikten Prozessen statt, wie dies beispielsweise für die in Abschnitt 2.2 beschriebenen CAD-Prozesse der Automobilindustrie der Fall ist.

Dem gegenüber steht das „Pull"-Verfahren, in dem die Senke eine Information explizit bei der Quelle anfragt, sobald diese im eigenen Prozess benötigt wird. Damit verursacht dieses Verfahren lediglich den tatsächlich notwendigen Aufwand für Quelle und Senke. Für die Senke ergibt sich jedoch die Notwendigkeit des Wissens, von welcher Quelle sie eine bestimmte Information erhalten kann. Die Untersuchung modellbasierter Entwicklungsprozesse - am Beispiel der automobilen Funktionsentwicklung - zeigte, dass diese Informationsbeschaffungsaufgabe heute vorwiegend durch individuelle Kommunikation der Prozessbeteiligten möglich ist und nur dann effizient funktioniert, wenn die Informationsquelle bereits bekannt ist.

Die diesbezüglichen Untersuchungen wurden im Bereich der automobilen Funktionsabsicherung durchgeführt. Das Umfeld bestand aus sieben fachlich und räumlich getrennten, aber methodisch ähnlichen Entwicklungsbereichen mit insgesamt rund 250 Prozessbeteiligten. Der Informationsaustausch ist dabei fast ausschließlich über das „Pull"-Verfahren realisiert. Die Identifikation von Informationsquellen funktioniert nach heutigem Stand über persönliche Kontakte oder mittels „Try-and-Error"-Anfragen über klassische Kommunikationsmedien, wie Telefon oder E-Mail. Eine der größten genannten Herausforderungen besteht dabei in der Vollständigkeit der Quellenabdeckung. Es ist demnach nicht möglich zu folgern, dass eine Information im Unternehmen nicht vorhanden ist, wenn sie bei einer Quelle nicht gefunden wurden. Grundsätzlich müssten daher alle Quellen angefragt werden, solange bis keine Quelle mehr existiert oder die Information gefunden wurde. Da dies nachvollziehbar ein zu großer Zusatzaufwand für den Entwicklungsprozess wäre, war der einheitlich gewählte Ausweg, die Eigenbeschaffung der Informationen, was bedeutet, dass der Informationsnutzer zum Produzenten wird, auch wenn dies nicht zu seinen Kernaufgaben gehört.

Zusammenfassend lässt sich schlussfolgern, dass es an einer zentralen Informationsstelle für die Prozessbeteiligten fehlt, an der sie eine Information oder zumindest einen Verweis auf eine mögliche Quelle erhalten. Sollte dies nicht der Fall sein, so muss dies darüber hinaus der Aussage gleich kommen, dass diese Information noch nicht existiert.

Hier setzt das modulare Informationsmanagementsystem an. Es übernimmt die Rolle der zentralen Wissensbasis, die die Entwicklungsartefakte und damit die Arbeit der einzelnen Entwicklungsbereiche dokumentiert und anschließend potenziellen Nutzern (Senken) zur Verfügung stellt, um deren Entwicklungsprozess zu unterstützen. In Abschnitt 5.1 wurde eine minimale Informationsbasis für Entwicklungsartefakte der modellbasierten Entwicklung definiert. Diese Informationsbasis wurde aufgrund der benötigten Informationen von Prozessbeteiligten identifiziert und stellt somit die Grundlage für die oben beschriebene zentrale Informationsstelle dar.

5.6.2 Suchindexgenerierung aus der Wissensbasis

Ausgehend von der Existenz einer entsprechenden Wissensbasis in Form einer Graphrepräsentation entsprechend Abschnitt 5.4.3, wird im Folgenden die Generierung eines Suchindex (vgl. Abschnitt 4.5.1) beschrieben. Der Suchindex ist notwendig, um einen suchmaschinenbasierten Zugriff auf das vorhandene Wissen zu ermöglichen. Das Konzept einer Suchmaschine wurde im Rahmen der vorliegenden Arbeit gewählt, da es bei einer breiten Anwenderschaft als Nutzungskonzept vorausgesetzt werden kann und damit die Akzeptanz der Lösung unterstützt. Laut statistischem Bundesamt nutzten beispielsweise 2012 35 Mio. Menschen in Deutschland die Suchmaschine „Google". Ausgehend von rund 82 Mio. Einwohnern kann unter Abzug von Menschen über 70 und unter 16 Jahren eine Kenntnis des Nutzungskonzeptes bei mindestens 50% der Einwohner abgeschätzt werden. Im industriellen Umfeld wird diese Zahl eher wesentlich höher sein.

Ziel des Konzeptes einer Suchmaschine ist es, den identifizierten Anwendungsfall einer fehlenden, zentralen Informationsstelle anzugehen. Ein Suchindex kann unterschiedliche Formate annehmen (vgl. Abschnitt 4.5.1). Im Rahmen der Arbeit wurde die Nutzung einer dokumentorientierten Baumstruktur als passende Lösung identifiziert. Der Baum besteht dabei aus hierarchisch sortierten Schlüssel-Wert-Paaren, wobei jedes Blatt ein Suchergebnis und jeder innere Knoten einen Ergebnisraum repräsentiert. Die Datenstruktur des Baumes erlaubt eine direkte Transformation der Informationen einer graphbasierten Wissensstruktur.

Die Erzeugung des beschriebenen Suchindexes erfolgt zweistufig. Zunächst wird der Wissensgraph unter Nutzung der modellbasierten Repräsentation von Entwicklungsartefakten entlang deren Kanten traversiert. Die Knotenattribute werden dabei in Schlüssel-Wert-Paare überführt und sind damit ebenfalls im Suchindex enthalten. Nach Abschluss dieser ersten Phase enthält der Suchindex Einträge für jedes in der Wissensbasis beschriebene Entwicklungsartefakt. In der zweiten Phase werden die dynamisch eingepflegten Beziehungen (vgl. dynamische API Abschnitt 5.5.3) vom Wurzelknoten der Entwicklungsartefakte aus traversiert und als Schlüssel-Wert-Paare in den Suchindex eingepflegt.

Bei diesem Vorgehen werden Modellbeziehungen, d.h. die semantische Zusammengehörigkeit zwischen zwei Entwicklungsartefakten, welche über dynamische Beziehungen gerichtet zwischen zwei Wurzelknoten abgebildet werden können, nicht berücksichtigt. Grund dafür ist, das diese Form der Beziehungen in der Regel zu Graphstrukturen führen, die sich nicht mehr ohne Informationsverlust in Bäume transformieren lassen. In Graphen lassen sich beispielsweise Kontexte für Modelle repräsentieren. So kann die Abhängigkeit der Nutzbarkeit eines Motormodells mit einem Getriebemodell beispielsweise von einem Land abhängig sein.

5.6.3 Erläuterung berücksichtigter Suchkonzepte

Die Nutzung eines klassischen Suchmaschinenansatzes geht heute in der Regel einher mit zwei Suchkonzepten - der Volltext- und der facettierten Suche. Die Volltextsuche erlaubt dabei die Eingabe einer Zeichenkette, den sogenannten Suchstring, der als Referenz für eine vergleichende Suche genutzt wird. Meist handelt es sich dabei um Schlagworte, d.h. prägnante Wörter, die direkt mit dem gesuchten Element in semantischer Verbindung stehen. Die Suche wird daher auch als Schlagwortsuche bezeichnet. Darüber hinaus erlaubt die

Volltextsuche in der Regel die semantische Anreicherung von Suchstrings durch eine Menge an Suchoperatoren. Die häufigsten Suchoperatoren sind dabei:

- **„A B"** - die Suchergebnisse enthalten die Zeichenketten A und B in exakt der vorgegebenen Form (inklusive Formatierung, Reihenfolge etc.)
- **A + B** - die Suchergebnisse enthalten sowohl die Zeichenkette A, als auch die Zeichenkette B
- **A B** - die Suchergebnisse enthalten die Zeichenkette A oder die Zeichenkette B (logisches Oder - Ergebnisse mit beiden Zeichenketten sind in der Regel zu priorisieren)
- **!A** - die Suchergebnisse enthalten die Zeichenkette A nicht

Die Volltextsuche erlaubt es den Suchraum einzuschränken, ohne Kenntnis der in der Wissensbasis vorhandenen Elementen zu benötigen. Letztendlich handelt es sich jedoch um einen String-Vergleich, der lediglich bei korrekter Schreibweise und Wortwahl zu den erwünschten Ergebnissen führt. Wie bereits in Abschnitt 5.5.2 erläutert, wird die Generierung der Wissensbasis unter Nutzung von fachspezifischen Thesauri unterstützt. Das Volltextsuchkonzept der vorliegenden Arbeit nutzt daher die Thesauri, um den Suchraum auf Basis semantisch gleicher oder einschränkenderer Schlagworte einzuschränken. Als Beispiel sei die Suche nach allen Ergebnissen zu einem bestimmten Fahrzeugprojekt angeführt. Der Nutzer gibt den Suchstring „Projekt A4" vor. Der Projektbezeichnungsthesaurus wird dabei den Teilstring „A4" als Synonym für weitere Produktbezeichnungen des gleichen Produktes „A_1XY" identifizieren. Durch hierarchische Ordnung der Fahrzeugprojekte, kann so über den Ordnungsbuchstaben X das Fahrzeugprojekt A4 und über den Ordnungsbuchstaben Y die Derivate des Fahrzeugprojektes erreicht werden. Davon ausgehend, das „A_11Y" das Fahrzeugprojekt A4 identifiziert. Dann könnte Y folgende Bedeutungen besitzen:

- Y = 1: Limousine
- Y = 2: Avant
- Y = 3: Cabriolet

Unter Berücksichtigung dieser hierarchischen Ordnung ist es möglich den Suchstring „Projekt A4" um die Unterprojekte und Suchstrings „A_111", „A_112" und „A_113" zu erweitern, um dadurch bessere Ergebnisse des Stringvergleichs zu erzielen.

Das Suchkonzept der facettierten Suche erlaubt die Einschränkung des Suchraumes nach vorgegebenen Schlagworten. Diese Schlagworte werden als Facetten, aus dem Französischen am Besten als Teilaspekt übersetzt, bezeichnet. Eine Facette teilt den Suchraum in zwei oder mehr disjunkte Teilmengen und schränkt den Suchraum auf die möglichen Suchergebnisse der gewählten Teilmenge ein. Aspekte können dabei Aufzählungen, aber auch Grenzwerte sein. Ein Beispiel einer Aufzählung wäre das oben genutzte Fahrzeugprojekt. So könnte der Nutzer die Möglichkeit haben, aus einer Liste der Fahrzeugprojekte die für ihn relevanten auszuwählen. Die zugehörigen Suchergebnisse ergeben sich wiederum nach dem in der Volltextsuche beschriebenen Verfahren durch Stringvergleich. Grenzwerte bedingen in der Regel numerische Vergleiche anhand bestimmter Eigenschaften der Suchergebnisse. So kann beispielweise das Alter eines Sucheintrags eingeschränkt werden nach „< 2012", „2012-2014", „>2014". In diesem Fall wäre eine algorithmische Prüfung der Datumsangaben notwendig. Facetten werden dem Nutzer in der Regel über die Nutzeroberfläche vorgegeben und sind auf spezifische, häufig gesuchte Aspekte der Wissensbasis hin optimiert. Aufgrund der anwendungsspezifischen Ausprägung lassen sich hierzu im Rahmen

der vorliegenden Arbeit keine absoluten Aussagen zu geeigneten Facetten treffen. Es zeigte sich jedoch im Rahmen der Untersuchungen, das in jedem Fall die Informationsklassen der modellbasierten Repräsentation (vgl. Abschnitt 5.4.3) von besonderer Bedeutung sind, da diese den minimalen Informationsbedarf der Anwendungsdomäne repräsentieren.

Grundlegend sind beide Konzepte, die Volltext- und facettierte Suche, für eine zentrale Informationsbasis von Bedeutung. Während die Volltextsuche dem Anwender die Freiheit lässt, auf Basis selbst gewählter Suchbegriffe die Wissensbasis zu durchsuchen und damit auch eher unscharfe Suchen zulässt, erlaubt die facettierte Suche eine zielgerichtete Exploration des Suchraumes und somit die schnelle Einschränkung auf die gewünschten Ergebnisse. In der Regel bestehen Suchen heute aus einer Kombination beider Suchverfahren. Zunächst erfolgt eine Suchraumdefinition durch die Volltextsuche. Anschließend werden die für diesen Suchraum gültigen Facetten, d.h. solche, die unter der Vorbedingung der Volltextsuche nicht zu einer leeren Menge an Suchergebnissen führen, angezeigt und der erhaltene Suchraum lässt sich zielgerichtet einschränken.

5.7 Entwicklung einer Modellbewertungssystematik für kriterienbasierte Modellempfehlung

Neben der reinen Informationssuche, die in Abschnitt 4.5.1 erläutert wurde, stellt die Beherrschung der Komplexität der Entwicklungsartefakte im Entwicklungsprozess eine der größten Herausforderungen dar. Wie in Abschnitt 5.2.1 erläutert, bestehen Modelle in der modellbasierten Entwicklung und Absicherung bereits heute aus einer großen Zahl von Einzelmodellen, die zu komplexeren Gesamtmodellen verknüpft werden. Die Komplexität dieser Modelle liegt neben der Schnittstellen-, der numerischen und der zeitlichen Abhängigkeit auch in der semantischen Gültigkeit von Modellkombinationen. Ein reines Suchmaschinenkonzept ist hier nicht ausreichend, da die genannten Abhängigkeiten einerseits nur schwer im Suchindex abgebildet werden können und andererseits eine Suche alle Abhängigkeiten als Parameter berücksichtigen müsste.

Um die Komplexität des Entwicklungsprozesses nicht in der Suche zu reproduzieren, wurden im Rahmen der Arbeit Untersuchungen durchgeführt, wie die Recherche nach Entwicklungsartefakten zielgerichteter unterstützt werden kann. Dabei entsteht eine Bewertungssystematik, die es erlaubt Modelle anhand definierter Kriterien zu filtern und zu priorisieren. Dadurch ist es möglich relevante Modelle zu empfehlen. Der Begriff der Empfehlung wurde hier gewählt, da auf Basis von Kriterien, wie im Folgenden näher ausgeführt wird, lediglich relative Aussagen zu Modelle getroffen werden können - die absolute Entscheidung obliegt letztlich dem Anwender.

5.7.1 Explorative und kontextgebundene Suchform

Im Rahmen der vorliegenden Arbeit wird die Suche nach Informationen gegliedert in eine explorative und eine kontextgebundene Suchform. Eine explorative Suche dient der Orientierung in einem Datenbestand ohne einen konkreten Anwendungsfall zu berücksichtigen. Für diese Suchform ist eine Volltext oder facettierte Suche (vgl. Abschnitt 4.5.1) ein geeigneter Lösungsansatz, da mit Hilfe von Schlagworten ein Suchraum eingeschränkt werden

kann, der anschließend weiter verfeinert wird, um ein Überblick zu passenden Artefakten zu erhalten. Diese Suchform wird bereits heute durch entsprechende Softwaresysteme (vgl. Abschnitt 4.5.2 und 4.5.3) hinreichend unterstützt.

Die kontextgebundene Suche von Entwicklungsartefakten bezeichnet die Analyse bezüglich ihrer Eignung für einen konkreten Anwendungsfall. Dies könnte beispielsweise eine bestimmte Realisierungsform, wie Echtzeitfähigkeit, eine qualitative Eigenschaft, wie die Einhaltung bestimmter Modellerierungsrichtlinen oder eine bestimmte Schnittstellenkonfiguration sein.

Um eine Bewertung von Entwicklungsartefakten im Rahmen einer kontextgebundenen Suche durchzuführen ist stets eine formale, dem Anwendungsfall entsprechende Umfeldbeschreibung notwendig. Im Folgenden wird diese als Referenz bezeichnet. Eine Referenz kann beispielsweise ein Qualitätsmodell (vgl. Abschnitt 4.4.1), eine formale Beschreibung des Simulationsaufbaus oder auch eine bereits getroffene Auswahl an Entwicklungsartefakten sein.

Ausgehend von der Existenz einer Referenz ist es notwendig ein Entwicklungsartefakte gegenüber dieser zu bewerten. Dafür ist die Definition geeigneter Metriken (vgl. Abschnitt 4.4.1) notwendig, um die An- oder Abwesenheit definierter Eigenschaften des Entwicklungsartefaktes gegenüber der Referenz zu prüfen. An dieser Stelle wird deutlich, dass es sich bei einer kontextgebundenen Suche stets um eine relative Bewertung gegenüber einer Referenz handelt. Ein Suchergebnis kann demnach nur so gut sein, wie die zugrunde liegende Referenz. Daher werden diese Suchergebnisse im Rahmen der vorliegenden Arbeit lediglich als Empfehlung bezeichnet. Darüber hinaus dient die Modellbewertungssystematik dieser Arbeit der Beschreibung einer Methodik zur kontextgebundenen Suche nach Entwicklungsartefakten. Die genutzten Referenzen sind für jeden Anwendungsfall separat zu definieren.

5.7.2 Erläuterung berücksichtigter Suchmuster

Wie in Abschnitt 5.7.1 erläutert wurde, sind die Referenzen einer kontextgebundenen Suche eng mit dem konkreten Anwendungsfall verknüpft. Mit dem Fokus der vorliegenden Arbeit auf modellbasierte Entwicklungsprozesse konnte für diese eine Kategorisierung häufiger Suchmuster durchgeführt werden. Unter Suchmustern wird an dieser Stelle das methodische Vorgehen des Anwenders bei der Suche zur Erreichung seines Suchziels, das Auffinden von relevanten Suchergebnissen, verstanden.

Es zeigte sich, dass bei der Suche nach Entwicklungsartefakten der modellbasierten Entwicklung grundlegend drei Suchmuster identifiziert werden können. Die Anwender beschrieben ihr Suchvorgehen dabei als „projektbezogen", „vorauswahlbezogen" oder „simulationsbezogen". Die projektbezogene Suche hat das Ziel passende Entwicklungsartefakte eines Projektkontextes zu finden. Ein Projekt kann in diesem Zusammenhang jeder thematisch gruppierende Kontext eines Entwicklungsprozesses sein. In der Regel meint Projekte hier die organisatorischen Konstrukte in Unternehmen zur Gliederung ihrer Tätigkeiten. Eine vorauswahlbezogene Suche nutzt bereits getroffene Entscheidungen als Ausgangsbasis. Wie in Abschnitt 5.2.1 erläutert benötigt beispielsweise die XIL-Simulation eine große Anzahl an Simulationsmodellen, die zu einem Gesamtmodell verschaltet werden. Sind bereits Simulationsmodelle ausgewählt, so sucht der Anwender nur noch passende Modelle, die

mit dieser Vorauswahl verschaltet werden können. Die simulationsbezogene oder, allgemeiner formuliert, die anwendungsspezifische Suche bezieht Rahmenbedingungen des Umfeldes des Anwenders in die Suche ein. Diese Rahmenbedingungen können beispielsweise aus Organisationsstrukturen, wie HIL-Simulation (vgl. Abschnitt 3.3), oder Aufgabengebieten, wie Antriebsstrangentwicklung, hergeleitet werden. Um diese Suchmuster zu unterstützten wurden drei Basisreferenzmodelle identifiziert. Unter Basisreferenzmodell wird an dieser Stelle ein Modell verstanden, das einer kontextgebundenen Suche als Referenz für einen Basisanwendungsfall dienen kann. Diese Basisreferenzmodelle „Qualität", „Selektion" und „Umfeld" werden im Folgenden erläutert.

Basisreferenzmodell „Qualität" Die Qualität eines Entwicklungsartefaktes der modellbasierten Softwareentwicklung wurde bereits in Abschnitt 4.4.2 vorgestellt. Das dort genutzte Qualitätsmodell findet auch im Basisreferenzmodell für Qualität Anwendung. Dafür werden die beschriebenen Metriken zur Bewertung der Entwicklungsartefakte genutzt und zu Qualitätskriterien aggregiert. Damit ist die erste Voraussetzung aus Abschnitt 5.7.1, die Existenz von geeigneten Metriken, erfüllt. Wie in Abschnitt 4.4.1 erläutert wurde, ist Qualität stets ein relativer Wert, der nur im Vergleich mit einer Referenz zu einer bewertbaren Aussage führt. Diese Referenz lässt sich für die Messungen der Qualität auf unterschiedlichen Wegen herleiten. Eine mögliche Methode ist die Definition von absoluten Best- und Worst-Case-Werten für jede Metrik, so dass ein Referenzmodell mit Qualitätsaussagen von 0% (schlechte Qualität) bis 100% (beste Qualität) entsteht. Diese absolute Methode berücksichtigt allerdings nicht die unterschiedlichen Modellierungsmethoden (vgl. Abschnitt 5.2.1) für Entwicklungsartefakte, die sich insbesondere in einzelnen Metriken und damit auch der Wertigkeit der Aussagen dieser Metriken unterscheiden. Als geeignete Methode erwies sich die Definition von Referenzmodellen auf Basis vorhandener Entwicklungsartefakte. Hierbei wird zunächst eine Klassifikation der Entwicklungsartefakte hinsichtlich ihrer Vergleichbarkeit vorgenommen. Das Kriterium der Vergleichbarkeit ist abhängig von der Anwendungsdomäne. Als Klassifizierungsmerkmal könnten beispielsweise Modellierungsarten, wie Verhaltens- und Strukturmodellierung, oder Modellierungsinhalte dienen. Für jede Klasse der Entwicklungsartefakte wird anschließend ein Referenzmodell definiert. Dafür wird auf allen verfügbaren Entwicklungsartefakten eine Messung aller Metriken durchgeführt und aus den Ergebnissen der Durchschnitt gebildet. Für Metriken mit nicht numerischen Messwerten, in der Regel binäre Metriken mit „Ja"- oder „Nein"-Ergebnis, muss das gewünschte Ergebnis manuell definiert werden. Das auf diese Weise erhaltene Referenzmodell erlaubt zunächst Aussagen, ob die Qualität eines Entwicklungsartefakte oberhalb oder unterhalb des Durchschnitts seiner Klasse liegt. Eine weitere Verbesserung der Aussagenqualität lässt sich durch Definition von Grenzwerten für numerische Metriken erzielen. Diese Grenzwerte definieren den Bereich guter bzw. schlechter Entwicklungsartefakte. Für Methoden der Bestimmung und Definition solcher Grenzwerte sei auf [Sch12] verwiesen. Die Summe aller Referenzmodelle der Entwicklungsartefaktklassen ergibt das Basisreferenzmodell „Qualität".

Basisreferenzmodell „Selektion" Das Basisreferenzmodell „Selektion" repräsentiert den Zustand eines vom Anwender genutzten Recherchesystems, beispielsweise einer Suche, und persistiert alle getätigten Auswahlen (Selektionen). Die betrifft insbesondere genutzt oder ausgewählte Schlagworte sowie die Eigenschaften von Entwicklungsartefakten, die

innerhalb des Recherchesystems in einer bestimmten Form zur weiteren Verarbeitung vorgemerkt wurden. Für die Suche nach Simulationsmodellen für eine Modellverschaltung (vgl. Abschnitt 5.2.1) könnte dies beispielsweise eine bereits vorausgewählte Menge von Modellen sein. Relevante Eigenschaften der bereits gewählten Entwicklungsartefakte sind die Attribute innerhalb der Wissensbasis, alle zum Auffinden innerhalb der Wissenbasis durchlaufenen Beziehungen (Kanten des Graphen), sowie strukturelle Eigenschaften, wie Schnittstellen. Diese Sammlung aus Schlagworten, Beziehungen und strukturellen Eigenschaften bilden den Suchraum des Anwenders ab und stellen damit das Basisreferenzmodell der „Selektion" dar. Die gesammelten Selektionen können anschließend für eine Filterung der Suche genutzt werden. Schlagworte können beispielsweise Entwicklungsartefakte anhand ihrer Attribute identifizieren. Beziehungen erlauben es durch Verfolgung der Graphkanten weitere Entwicklungsartefakte, beispielsweise anhand ihrer Zuordnung zu gleichen Projektstrukturen zu identifizieren. Schnittstellen ermöglichen die Suche nach schnittstellenkompatiblen Entwicklungsartefakten.

Basisreferenzmodell „Umfeld" Das Basisreferenzmodell „Umfeld" repräsentiert den Anwender eines Recherchesystem in seinem Anwendungskontext. Dafür ist die Sammlung organisatorischer Informationen notwendig, die in der Regel nicht direkt aus Entwicklungsartefakten und dem zugehörigen Prozess extrahiert werden können. Zentrale Informationen, die eine Einschränkung des Suchraumes zulassen, sind beispielsweise Projektzugehörigkeiten, betriebliche Strukturvorgaben, wie Abteilungszugehörigkeit, sowie Tätigkeitsbeschreibungen. Alle Informationen sind direkt mit dem Anwender verknüpft und müssen daher in Form eines anwenderspezifischen Referenzmodells hinterlegt werden. Das Basisreferenzmodell „Umfeld" umfasst alle berücksichtigten Information aus dem Umfeld der Anwender und beschreibt diese in abstrakter Form. Ein anwenderspezifisches Referenzmodell stellt anschließend eine Instanz des Basisreferenzmodells dar.

5.7.3 Erläuterung des Bewertungskonzepts

Für die Bewertung von Entwicklungsartefakten gegenüber den Abschnitt 5.7.2 beschriebenen Basisreferenzmodellen muss zunächst der Begriff „Ähnlichkeit" für diese Artefakte definiert sein. Die Ähnlichkeit bezieht sich stets auf ein oder mehr vergleichbare Eigenschaften zweier Entwicklungsartefakte. Vergleichbar bedeutet, dass diese in einem gewissen Kontext als inhaltlich gleich betrachtet werden können. Eigenschaften können dabei gleich oder kompatibel sein. Gleichheit bedeutet, dass der der Eigenschaft zugeordnete Wert für beide Entwicklungsartefakte identisch ist. Kompatibel bedeutet, dass sich die Eigenschaften für beide Entwicklungsartefakte ergänzen.

Grundsätzlich können für die Ähnlichkeit von Entwicklungsartefakten zwei Formen unterschieden werden - die strukturelle und die semantische Ähnlichkeit. Die strukturelle Ähnlichkeit bezieht sich auf den Aufbau von Entwicklungsartefakten und bezeichnet somit eine direkt zugeordnete Eigenschaft, die für den Vergleich auf Ähnlichkeit herangezogen wird. Ein Beispiel hierfür wären Schnittstellen eines Entwicklungsartefaktes. Als ähnlich wären in diesem Fall sowohl gleiche Schnittstellen, aber auch kompatible Schnittstellen, d.h. ein Artefakt hat eine Eingangsschnittstelle und das Zweite eine passende Ausgangsschnittstelle, zu betrachten. Semantische Ähnlichkeit bezeichnet die Gemeinsamkeiten bezüglich

indirekt zugeordneter Eigenschaften. Diese werden dem Entwicklungsartefakt über eine Informationsquelle zugeordnet und können nicht aus diesem hergeleitet werden.

Die Bewertung eines Entwicklungsartefaktes wird im Rahmen dieser Arbeit als Quantifizierung der Ähnlichkeit eines Entwicklungsartefaktes mit einer geeigneten Referenz verstanden. Eine geeignete Referenz meint dabei eine für den Anwender bewertbare Referenz, die ihm eine relative Aussage über ein Entwicklungsartefakt in Bezug auf einen für ihn bekannten Standard erlaubt. Eine absolute Aussage, wie „Ein Entwicklungsartefakt ist gut", kann nicht automatisiert hergeleitet werden, da die Bewertung „gut" vom jeweiligen Umfeld des Anwenders abhängig ist. Eine Bewertung gegenüber einer standardisierten Referenz erlaubt die Aussage „Ein Entwicklungsartefakt ist besser als die Referenz", wobei sich für Eigenschaften mit zugeordneten Werten auch die Eigenschaft „besser" gegenüber der Referenz quantifizieren lässt.

Für die einheitliche Definition der beschriebenen Referenzen kommen in der Bewertungsmethodik dieser Arbeit die in Abschnitt 5.7.2 vorgestellten Basisreferenzmodelle „Qualität", „Selektion" sowie „Umfeld" zum Einsatz. Sie erlauben die quantifizierte Bewertung von Entwicklungsartefakten gegenüber bereits bekannten Eingaben des Anwenders. Dies erlaubt die Filterung der Wissensbasis nach ähnlichen Entwicklungsartefakten und damit die Empfehlung dieser Artefakte für den Anwender.

Eine zentrale Herausforderung bei der Umsetzung des beschriebenen Bewertungskonzepts besteht in der Standardisierung der Basisreferenzmodelle sowie der in ihnen repräsentierten Eigenschaften. Sowohl strukturelle, als auch semantische Ähnlichkeit lässt sich nur auf der Grundlage standardisierter Wertebereich und Eigenschaftsbezeichnungen erreichen.

5.8 Strukturierte Informationsverarbeitung des modularen Informationsmanagementsystem

Im vorliegenden Abschnitt werden die vorgestellten Konzepte der Abschnitte 5.3, 5.4 und 5.5 in eine formale Beschreibung gefasst

Die Informationsklassen I des Informationssystems müssen a priori als Elemente des Informationsdatenmodells Φ definiert sein, da sie die Grundlage der Informationsrepräsentation darstellen. Eine Informationsklasse I wird im Rahmen der vorliegenden Arbeit definiert, als Triple $I = (u, E, R)$ bestehend aus einem eindeutigen Bezeichner u (vgl. UUID der Informationsklasse Abschnitt 5.4.3), einer Menge von Attributen bzw. Eigenschaften E sowie einer Menge von typisierten, gerichteten Relationen R. Der eindeutige Bezeichner u identifiziert die Informationsklasse innerhalb des Informationsdatenmodells .

Die Menge der Eigenschaften E besteht aus Eigenschaften $e \in E$, welche die Informationsklasse inhaltlich definieren. Jede Eigenschaft $e = (k, t) \in E$ besteht aus einem Schlüssel k als syntaktischer und semantischer Bezeichner, sowie einem diesem Schlüssel k zugeordneten Datentyp t.

Die Menge der Relationen R einer Informationsklasse I repräsentieren gerichtete, typisierte Graphkanten deren Quelle per Definition stets die Informationsklasse I ist, welche die Relation $r \in R$ enthält. Jede Relation besteht aus einer Typisierung T , dem semantischen Bezeichner der Relation, sowie einer Informationsklasse $S \in \Phi$. Somit gilt $r = (T, S)$.

Wie bereits in Abschnitt 5.4.3 erläutert, folgt die Definition eines Informationsdatenmodells Φ dem Paradigma der modellbasierten Entwicklung. Für die Definition von Informationsklassen I eignet sich insbesondere die modellbasierte Repräsentation als Klassendiagramm. Für die Modellierung sind dabei folgende Richtlinien zu beachten:

- Jede Informationsklasse I enthält einen Bezeichner u, der diese im gesamten Informationsdatenmodell eindeutig identifiziert.
- Jede Relation $r \epsilon R$ einer Informationsklasse I besitzt eine Typisierung T, den semantischen Bezeichner. Bezeichner, deren Bedeutung semantisch äquivalent ist, sind identisch zu wählen.
- Jedes Informationsdatenmodells Φ enthält eine Informationsklasse I, die stellvertretende für den Modellkontext (vgl. Abschnitt 5.1) des repräsentierten Entwicklungsartefakt steht. Diese Informationsklasse I wird als Wurzelelement W bezeichnet. Für das Wurzelelement W gilt:
  - Für jedes $I \in \Phi / \{W\}$ existiert eine Menge $X := \{I_1, \ldots, I_n\}$ mit $n \in \mathbb{N}$
  - $I_1 := I$,
  - $I_i := (u_i, E_i, R_i), \forall i = 1, \ldots, n$ und
  - $\exists r_i = (t_i, s_i) \in R_i$ mit $s_i = \begin{cases} I_1, für & i < n \\ W, & sonst \end{cases}$

Eine Instanz i einer Informationsklasse I sei definiert, als $i = \{b, I, f(k)\}$. Dabei gibt der Bezeichner b den eindeutigen Bezeichner der Instanz an (vgl. UUID der Instanz einer Informationsklasse Abschnitt 5.4.3). Die Informationsklasse I identifiziert die Informationsklasse, deren Instanz i ist. Die Funktion $f(k)$ gibt für jede Eigenschaft $e = (k, t) \in E$ der Informationsklasse I für den Schlüsselwert k einen Wert des Typs t zurück.

Eine vollständige Instanziierung aller Informationsklassen I des Informationsdatenmodells Φ wird als Modellkontext m bezeichnet. Ein Modellkontext m ist durch die definierte Instanz $w \in m$ von W eindeutig einem Entwicklungsartefakt zugeordnet. Alle Modellkontexte werden durch den Modellkontextgenerator (vgl. Abschnitt 5.3) verwaltet.

Ein Adapter A enthält eine Menge $Z \subset \Phi$ an Informationsklassen I für die er hinsichtlich der Qualifikation von Daten zuständig ist. Sei D der Datenbestand eines Quellsystems mit potenziell relevanten Daten für das Informationssystem. Dann bezeichnet $d \epsilon D$ ein atomares Datum des Datenbestandes D, auf das unter Nutzung eines Adapters A (vgl. Abschnitt 5.3) zugegriffen werden kann. Der Adapter A enthält dafür eine Qualifizierungsfunktion $q(w, d, Z)$, die die Relevanz eines Datums d für die Informationsklassen $I \in Z$ in Bezug auf den durch w identifizierten Modellkontext m bewertet.

Gegeben sei im Folgenden ein Informationsdatenmodell Φ, das unter Berücksichtigung der beschriebenen Richtlinien erstellt wurde. Für die dynamische API gilt für die Datenakquise in Quellsystem folgender Ablauf:

Algorithmus 5.1 Adapter-Datenakquise im Quellsystem

Gegeben sei ein Datenbestand D sowie ein Adapter A für diesen Datenbestand.
Für alle Wurzelelemente w der Modellkontexte m des Modellkontextgenerators führe aus:
Für alle Informationsklassen $I \in Z$ der Zuständigkeit des Adapters A führe aus:
Für jedes Datum $d \in D$ führe aus:

1. Rufe die Qualifizierungfunktion $q(w, d, I)$ auf und führe sie aus:
   a) Rufe die Instanz i der Informationklasse I identifiziert durch w mittels des Modellkontextgenerators ab.
   b) Qualifizierte das Datum d gegenüber der Instanz der Informationsklasse i.
   c) Wenn Qualifizierung positiv, dann passe die Funktion $f(k)$ der Informationsklasse so an, dass der Aufruf $f(k)$ das Datum d zurückgibt.
   d) Gibt die Instanz der Informationsklasse i zurück.
2. Übergib i an den Modellkontextgenerator.

Der Algorithmus 5.1 führt für jedes Datum d eines Datenbestandes D die Qualifizierungsfunktion q aus. Die Qualifizierungsfunktion q ruft über die dynamische API (vgl. Abschnitt 5.5.3) eine Instanz i der Informationsklasse I identifiziert durch w ab. Diese Instanz i enthält alle bereits mit dem durch$w$ identifizierten Entwicklungsartefakt verknüpften Eigenschaften E sowie alle Relationen R. Anschließend speichert die Qualifizierungsfunktion q bei positivem Ergebnis der Qualifizierung das Datum d in der Instanz$i$ und gibt diese zurück. Das Ergebnis der Qualifizierung wird über die dynamische API identifiziert mittel w an den Modellkontextgenerator übergeben. Die Qualifizierung eines Datums d ist abhängig vom Datenbestand D und unterscheidet sich damit von Adapter zu Adapter. Vergleiche hierzu Abbildung 5.9.

Nach Ausführung des Algorithmus 5.1 für alle Adapter A ist die Phase der Datenakquise abgeschlossen. Es folgt für alle Modellkontexte m die Analyse ihrer zugehörigen Entwicklungsartefakte. Im Rahmen dieser Arbeit wird diese Phase als Modellanalyse bezeichnet. Voraussetzung ist, dass mindestens ein Attribut $e \in E$ innerhalb der Informationsklassen I in Φ den tatsächlichen Ort der Ablage des durch m beschriebenen Entwicklungsartefaktes enthält. Die Modellanalyse ist Teil der Modellkontextimplementierung und abhängig vom Entwicklungsartefakttyp. Zu den zentrale Aufgaben der Modellanalyse zählen die Durchführung von Qualitätsmessungen (vgl. Abschnitt 4.4.2), das Auslesen von Schnittstelleninformationen sowie weiterer, gegebenenfalls relevanter, entwicklungsartefaktabhängiger Daten.

Nach der obigen Beschreibung wird für jedes in das Informationssystem eingepflegte Entwicklungsartefakt ein Modellkontext m entsprechend der modellbasierten Repräsentation erzeugt. Wie in Abschnitt 5.4.3 erläutert, ist diese modellbasierte Repräsentation für die dynamische Erweiterung sowie die Wiederverwendung nur bedingt geeignet. Daher folgt die Transformation in eine generische Graphstruktur nach Algorithmus 5.2.

Algorithmus 5.2 Graphtransformation der modellbasierten Repräsentation

Gegeben sei ein Informationsdatenmodell Φ sowie eine Menge M von Modellkontexten m.
Schritt 1: Transformation der Informationsklassen
Für jede Informationsklasse $I = (u, E, R) \in \Phi$ führe aus:

1. Erzeuge einen Knoten n in G
2. Setze den eindeutigen Bezeichner b des Knoten n mit dem eindeutigen Bezeichner u der Informationsklasse I als $b := u$
3. Für alle Eigenschaften $e = (k_e, t_e) \in E$ von I erzeuge ein Attribut $a = (k, t, v)$ als $a := (k_e, t_e, \varnothing)$

Schritt 2: Transformation der Instanzen der Informationklassen
Für jeden Modellkontext m des Modellkontextgenerators als Instanzen von Φ und $\forall i \in m$ als Instanzen der Informationklassen von Φ führe aus:

1. Erzeuge einen Knoten n in G
2. Setze den eindeutigen Bezeichner b des Knotens n mit dem eindeutigen Bezeichner u der Instanz i als $b := u$
3. Für alle Eigenschaften $e = (k_e, t_e) \in E$ der Instanz i erzeuge ein Attribut a am Knoten n als $a := (k_e, t_e, f(k_e))$
4. Für alle $r \in R$ mit $r = (t_r, s_r)$ von i erzeuge eine Kante $k = () \in K$ von n als $k := (i, s_r, t_r)$

Schritt 3: Kante zum Knoten der Informationsklasse mit Kantenbezeichnung 'instance_of'

1. Erzeuge eine Kante vom Instanzknoten n zum Klassenknoten x, der in Schritt 1 für die Informationsklasse von n erzeugt wurde, d.h. erzeuge eine Kante $k \in K$ zur Knoten x im Graphen G, wobei x die Informationklasse I von m repräsentiert, als: $k := (i, x, \text{'instance\_of'})$

Dabei gelten für die generische Graphstruktur die folgende Bezeichnungen. Sei $G = (N, K)$ ein gerichteter, typisierter Graph mit N der Menge aller Knoten des Graphen G und $K \subset N \times N \times T$ die Menge aller gerichteten Kanten $k = (n_q, n_s, t)$ des Graphen G mit dem Quellknoten n_q, dem Senkenknoten n_s und einer zugeordneten Typisierung t. Sei ein Knoten $n \in N$ definiert als $n = (b, A)$ mit b als eindeutigem Bezeichner des Knoten n und A als Menge der Attribute $a \in A$ die diesem Knoten zugeordnet sind. Attribute $a = (k, t, v)$ sind Eigenschaften, die in einem Knoten n des Graphen G gespeichert werden. Dabei besteht ein Attribut a stets aus einem Schlüssel k, der das Attribut a eindeutig in einem Knoten n eindeutig identifiziert, einem Typ t, der den Datentyp des Wertes v beschreibt, und dem Wert v selbst. Nach Ausführung des Algorithmus 5.2 sind sowohl das Informationsdatenmodell Φ als auch alle Modellkontexte von Φ in einen Graphen G transformiert (vgl. Abbildung 5.5).

5.9 Zusammenfassung

In Kapitel 5 wurde das im Rahmen der Arbeit entwickelte Konzept zur automatischen, prozessintegrierten Dokumentation von Artefakten im modellbasierten Entwicklungprozess sowie das darauf aufbauende Konzept zur Unterstützung der Wiederverwendung mit Hilfe eines kriterienbasierten Empfehlungssystems erläutert. Es geht dabei unter anderem auf das Vorgehen bei der Entwicklung, sowie die Rahmenbedingungen einer Anwendung des Konzeptes ein. In Abschnitt 5.1 wurde zunächst das Vorgehen zur Bestimmung des so genannten Modellkontexts als minimale Wissensbasis eines Entwicklungsartefaktes einer Anwendungsdomäne vorgestellt und der Lösungsansatz zum ersten Arbeitsschwerpunkt (vgl. Abschnitt 1.2), der Definition eines Modellkontextes, beschrieben. Darauf aufbauend wurde in Abschnitt 5.2 eine Analyse von Entwicklungsartefakten am Beispiel der automotive Domäne durchgeführt, um die davon extrahierbaren Daten zu identifizieren.

Im Anschluss an die Erschließung der im Entwicklungsprozess verfügbaren Daten wurde in Abschnitt 5.3 die im Rahmen dieser Arbeit entwickelte Architektur einer Lösung zur Extraktion der identifizierten Daten und deren Weiterverarbeitung zu einer Wissensbasis vorgestellt. In den Abschnitten 5.4, 5.5 und 5.6 wurden anschließend die einzelnen Verarbeitungsschritte der Architektur erläutert. Diese Abschnitte stellen den Lösungsansatz für den zweiten Arbeitsschwerpunkt der vorliegenden Arbeit (vgl. Abschnitt 1.2), die automatische Erfassung des Modellkontextes, vor.

Abschließend wurden in den Abschnitten 5.6 und 5.7 Konzepte für die Wiederverwendung der in der Wissensbasis gespeicherten Informationen vorgestellt. Dabei geht Abschnitt 5.6 auf die Nutzung im Rahmen einer Suche von Entwicklungsartefakten ein, während Abschnitt 5.7 das Konzept einer kriterienbasierten Empfehlung präsentiert. Damit stellen diese Abschnitte den Lösungsansatz für den dritten Arbeitsschwerpunkt (vgl. Abschnitt 1.2), des Nachweises der Korrektheit des Ansatzes durch Umsetzung von Wiederverwendungskonzepten, vor.

Das Folgenden Kapitel 6 beschreibt die Umsetzung der im Rahmen des Kapitel 5 vorgestellten Konzepte in Form eines Demonstrators, welcher für deren Validierung am konkreten Anwendungsfall genutzt wird.

6 Umsetzung

Nach der Vorstellung der Lösungsansätze und Konzepte für die Zielstellungen der vorliegenden Arbeit und des darauf aufbauenden modularen Informationsmanagementsystems in Kapitel 5, beschreibt dieses Kapitel deren Nutzung für die Entwicklung einer prototypischen Implementierung. Diese verfolgt eine Reihe von Zielen. Zum einen dient sie der praktischen Bewertung der Ergebnisse dieser Arbeit, auf die in Kapitel 7 näher eingegangen wird. Darüber hinaus stellt sie eine Referenzimplementierung dar, die als Basis eines produktiven Einsatzes im Rahmen des bereits vorgestellten Entwicklungsprojektes XIL-Datenmanagement (vgl. Abschnitt 1.1) dienen soll. Abschnitt 6.1 beschreibt die Entwicklung des Demonstrators, geht auf die getroffenen technologischen Entscheidungen ein und stellt das Ergebnis der Entwicklung vor. Im Anschluss daran erläutert Abschnitt 6.2 den aktuellen Stand der Integration der Ergebnisse dieser Arbeit in das XIL-Datenmanagement.

6.1 Demonstrator zu Entwicklungszwecken

Der vorliegende Abschnitt beschreibt das Vorgehen bei der Entwicklung einer prototypischen Implementierung des in dieser Arbeit entwickelten modularen Informationsmanagementsystems. Die prototypische Implementierung des in Kapitel 5 vorgestellten modularen Informationsmanagementsystems in Form eines Demonstrators verfolgt einerseits das Ziel der Anwendung der Ergebnisse zur Bestätigung der Korrektheit und Eignung der Lösung für modellbasierte Entwicklungsprozesse. Andererseits dient sie als Referenzimplementierung für einen produktiven Einsatz innerhalb der betrachteten modellbasierten Entwicklungsprozesse. Als Rahmenbedingung für die Auswahl der eingesetzten Entwicklungstechnologien waren aus diesem Grund neben dem Einsatz von Technologien auf dem aktuellen Stand der Technik auch die vorgegebenen Beschränkungen eines produktiven Einsatzes zu berücksichtigen. Dies wurde insbesondere bei der Auswahl der umgesetzten Adapter für die Datenquellen berücksichtigt. Daraus ergeben sich wiederum Vorteile insbesondere für den Nachweis der Anwendbarkeit der Ergebnisse der vorliegenden Arbeit und den damit verknüpften Praxisbezug. Im Rahmen der Ergebnisdiskussion der Arbeit in Kapitel 7 kann dadurch die Anwendbarkeit an Daten tatsächlicher, praxisrelevanter Entwicklungsprojekte der automotive Domäne nachgewiesen werden. Der folgende Abschnitt stellt die auf Basis dieser Prämissen gewählten Entwicklungstechnologien näher vor.

6.1.1 Vorstellung der gewählten Entwicklungsumgebung

Eclipse Integrated Development Environment (IDE) Als Programmiersprache der Umsetzung wurde Oracle Java [Ind14] in der aktuellen Version 8 gewählt. Die Gründe hierfür liegen insbesondere in der Interoperabilität auf unterschiedlichen Betriebssystemen. Eine

zentrale Aufgabe der Programmiersprache des modularen Informationsmanagementsystems besteht im Zugriff auf und der Analyse von Entwicklungsartefakten, die in Dateisystemen unter Microsoft Windows oder einem Linux Derivat hinterlegt sind. Dateisystemabhängigkeiten sind eine Herausforderung für alle Programmiersprachen, da sie in der Regel zur Einbindung betriebssystemabhängiger Bibliotheken und damit zur Rekompilation führen. Mit Java 8 wurde die NIO-Bibliothek [Ind14] (engl. „New Input Output"-Library) in die Standardbibliothek der virtuellen Machine übernommen. Diese erlaubt es, ohne den Wechsel von Bibliotheken zur Ausführungszeit zwischen unterschiedlichen Dateisystemen zu wechseln. Darüber hinaus erlaubt es die NIO-Bibliothek über sogenannte „WatchServices", Änderungen auch in hochverzweigten Verzeichnisstrukturen zu überwachen, ohne die Performanz der Anwendung messbar zu beeinflussen.

Die Auswahl der Entwicklungsumgebung orientierte sich an der Auswahl der Programmiersprache. Java 8 war zu Beginn der Entwicklungsarbeiten noch nicht fertiggestellt. Das Eclipse Integrated Development Environment [Teu11] war die einzige verfügbare Entwicklungsumgebung, die die volle Unterstützung bereits umgesetzt hatte. Ein weiterer Grund war die Nutzung der Eclipse Rich Client Platform, als Basisarchitektur der Entwicklung.

Eclipse Rich Client Platform (RCP) Die Eclipse Rich Client Platform (RCP) stellt eine Basisarchitektur für Softwareentwicklungsprojekte zur Verfügung, die im Kern auf die Erweiterbarkeit und Austauschbarkeit von Programmteilen in Form von sogenannten Plugins unterstützt. Zentrum der Platform ist ein Plugin Lifecycle Management, welches die Installation, das Aktivieren bzw. Laden und das Deaktivieren von Plugins regelt. Plugins sind in sich geschlossene Softwarekomponenten, die innerhalb der RCP über sogenannte Extention Point miteinander lose gekoppelt kommunizieren. Lose gekoppelt bedeutet in diesem Zusammenhang, dass einerseits die Verfügbarkeit von Plugins als optional für die Softwarefunktion gestaltet werden kann, andererseits aber die Plugins zur Laufzeit gegen andere Plugins getauscht werden können, solange die über die Extention Points definierte Schnittstelle befriedigt ist. Der Vorteil im konkreten Anwendungsfall dieser Arbeit, liegt in der einfachen Erweiterbarkeit der Softwarelösung durch zusätzliche Adapter für Datenquellen in Form von Plugins [Teu11].

JavaFX Benutzerschnittstelle Als Präsentationskomponente kommt die JavaFX-Bibliothek [Ind14] zum Einsatz. Diese ist ab Java 8 die Standard Benutzerschnittstellenbibliothek. Für die Umsetzung der Informationsextraktion sowie des Informationsdatenmodells des modularen Informationsmanagementsystem sind keine Benutzerschnittstellen notwendig, da es sich um eine automatische Dokumentation von Entwicklungsartefakten handelt. Für die Wiederverwendung und die Überprüfung der Ergebnisse wird im Rahmen dieser Arbeit jedoch eine Referenzimplementierung für die Benutzerschnittstelle realisiert. Die Umsetzung in JavaFX, das angelehnt an HTML5 die strukturelle Beschreibung der Oberfläche von der Gestaltung trennt, ist es möglich diese Referenzimplementierung im Anschluss wahlweise als Rich Client Implementierung oder als Web-Applikation zu exportieren.

Eclipse Modeling Framework (EMF) Eine zentrale Komponente des modularen Informationsmanagementsystems stellt die modellbasierte Repräsentation der Wissensstrukturen

dar (vgl. Abschnitt 5.4). Für die Referenzimplementierung kommt hier das Eclipse Modeling Framework (vgl. Abschnitt 4.3.1) zum Einsatz. Gründe hierfür sind die Tiefe Integration in die Eclipse Rich Client Platform [Teu11], die hohen syntaktische Ähnlichkeit zu UML, die Open-Source Verfügbarkeit von graphischen Editoren und für die dynamische API nutzbare Code-Generatoren.

Neo4j Graphdatenbank Wie in Abschnitt 5.4.2 erläutert wurde, eigenen sich reine Graphdatenbanken besonders für die Wissensrepräsentation in Form semantischer Datenstrukturen. Bei der Evaluierung der unterschiedlichen Datenbanktechnologien für Abschnitt 5.4.2 erwies sich die Graphdatenbank Neo4j [Hun14] als besonders geeignet. Insbesondere zwei Eigenschaften trugen zur Auswahl von Neo4J bei. Zum einen bietet die Graphdatenbank die Abfragesprache „Cypher" [Hun14] als Schnittstelle, die speziell für Graphdatenbanken entwickelt wurde, und sich aufgrund der sprachlichen Konstrukte besonders für eine Erweiterung des Datenmodells im Rahmen der statische API (vgl. Abschnitt 5.5.3) eignet. Darüber hinaus stehen hier als Alleinstellungsmerkmal mit der Auslieferung Oberflächen zur graphischen Navigation durch den Wissensgraphen zur Verfügung. Vor dem Hintergrund der wissenschaftlichen Analyse und Ergebnispräsentation der vorliegenden Arbeit ermöglichen sie eine gute Präsentierbarkeit und Nachvollziehbarkeit der internen Datenflüsse des modularen Informationsmanagementsystems.

Elastic Search Für den Nachweis des Nutzens des modularen Informationsmanagementsystems dient unter anderem der Anwendungsfall der Suche und Wiederverwendung von Entwicklungsartefakten. Dafür wird als Suchmaschine Elastic Search (vgl. Abschnitt 4.5.3) genutzt. Wie bereits erläutert wurde, stellt die Open-Source Suchmaschine den aktuellen Stand der Technik dar. Darüber hinaus ist es möglich den Index anhand anwendungsspezifischer JSON-Strukturen zu gestalten. Dies erlaubt eine direkte Überführung der Wissensstrukturen der Graphdatenbank in den Index der Suchmaschine.

Adapter Die Auswahl der Adapter orientiert sich, wie in Abschnitt 5.5.1 beschrieben, am praktischen Anwendungsfall, auf den in Kapitel 7 näher eingegangen wird. Zum Einsatz kommen hier Schnittstellenbibliotheken für die notwendigen Quellsysteme. Für den Zugriff auf Simulinkmodelle wird die Conqat-Bibliothek [Dei14] der Technischen Universität München genutzt, die unter der Open-Source-Lizenz Apache 2.0 verfügbar ist. Der Zugriff auf SVN-Datenquellen wird über die Bibliothek SVNKit [SVN14] realisiert. Der Zugriff auf das System Atlassian JIRA wird über die öffentliche REST API [Atl14] des Produktes realisiert. Der Zugriff auf Microsoft Exchange Datenquellen sowie auf Microsoft Office Dokumente wird über die Bibliothek Apache POI [POI14] umgesetzt.

6.1.2 Beschreibung der Umsetzung

Die Architektur des im Rahmen dieser Arbeit entwickelten Konzeptes zur prozessintegrierten Dokumentation und Wiederverwendung von Entwicklungsartefakten ist zum Überblick in Abbildung 6.1 dargestellt. Es ist zu erkennen, dass sich die Gesamtarchitektur in drei

Ebenen teilt. Sie sind von unten nach oben zu lesen. Die Datenbasis bezeichnet alle Quellsysteme die potenziell relevante Daten für die Generierung eines Modellkontextes für Entwicklungsartefakte enthalten. Die Art des Quellsystems ist dabei nicht von Bedeutung, es kann jedes Softwaresystem sein, das in einem Entwicklungsprozess Daten speichert. Die einzig beschränkende Eigenschaft für Quellsysteme ist, dass die Daten von externen Softwaresystemen zugreifbar sein müssen. Die in der Abbildung 6.1 dargestellten Quellsysteme der prototypischen Umsetzung dienen der Validierung und werden bezüglich ihres Informationsgehalts im Rahmen des Kapitels 7 näher erläutert.

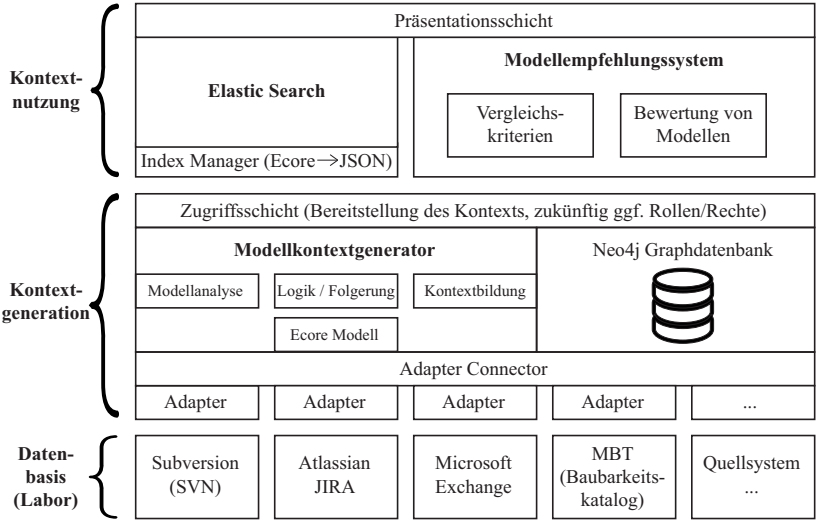


Abbildung 6.1: Gesamtarchitektur der Umsetzung

Die Ebene der Kontextgeneration greift über ein Adaptersystem bestehend aus einem Adapter Connector und einem jeweils quellsystemspezifischen Adapter auf die darunter liegenden Quellsysteme zu. Der Adapter Connector stellt die Verbindung zwischen den Informationsklassen des Kontextmodells bzw. Informationsdatenmodells und den datenliefernden Adaptern her. Das Kontextmodell ist die modellbasierte Repräsentation des Informationsdatenmodells, wie es in Abschnitt 5.4.3 erläutert wurde. Die Modellierung erfolgt in Klassendiagrammen des EMF als Ecore-Modell (vgl. Abschnitt 4.3.1) unter Beachtung der definierten Modellierungsrichtlinien (vgl. Abschnitt 5.4.3 und 5.8). Dadurch sind die einzelnen Informationsklassen anhand ihrer UUID eindeutig zu identifizieren. Darüber hinaus erhält jede Kante eine semantische Bezeichnung, die die gerichtete Beziehung der Klassen zueinander definiert. Abbildung 6.2 zeigt beispielhaft einen Ausschnitt des Kontextmodells des Demonstrators.

Die Umsetzung des Modellkontextgenerators entspricht algorithmisch den in Abschnitt 5.8 erläuterten Konzepten. Die Komponente Modellanalyse setzt die Konzepte der Analyse der

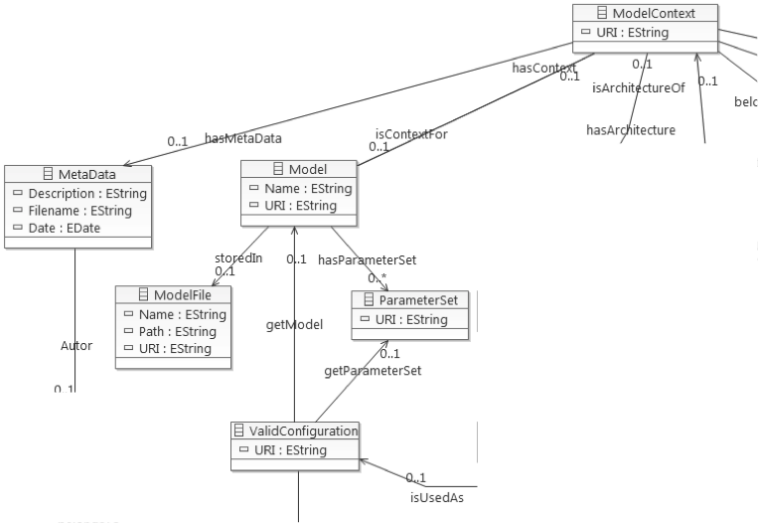


Abbildung 6.2: Ausschnitt des Kontextmodells

Entwicklungsartefakte bezüglich Schnittstellen, Qualität und möglichen Dokumentationselementen, wie Kommentaren, um. Der Bereich Logik und Folgerung bezeichnet das Auflösen von Beziehungen innerhalb der modellbasierten Repräsentation. Ein konkreter Anwendungsfall wäre beispielsweise die Verknüpfung eines Simulationsmodells und eines zugehörigen Parametersatzes, die eine bestimmte Modellkonfiguration ergeben, mit einem tatsächlichen Fahrzeugprojekt auf Basis der jeweiligen Meta-Daten. Die Komponente Kontextbildung verwaltet die Konsistenz der Modellkontexte als Instanzen des Ecore-Modells. Sie stellt sicher, dass Änderungen der Modellkontexte an einer zentralen Stelle stattfinden und somit keine widersprüchlichen Änderungen durchgeführt werden.

Als Graphdatenbank der prototypischen Umsetzung kommt Neo4j zum Einsatz. In Abschnitt 5.4.2 wurden unterschiedliche Datenbanktypen hinsichtlich ihrer Eignung für die Speicherung von Wissensstrukturen untersucht. Dabei wurden reine Graphdatenbanken als besonders geeignet identifiziert. Die Entscheidung für die Nutzung von Neo4j Version 2.0 als Graphdatenbank wurde aufgrund der breiten Unterstützung von strukturierten Abfragesprachen getroffen. Neben SPARQL [HKRS08] als Ontologieabfragesprache und Inferenzfunktionen, werden Gremlin als imperative und ab Version 2.0 Cypher [Hun14] als native deklarative Abfragesprache unterstützt. Für die Umfänge der vorliegenden Arbeit wurde durchgängig Cypher gewählt, da sich die deklarative Manipulation für die Transformationsschritte von der modellbasierten Repräsentation zur Graphrepräsentation als geeignet erwies. Auch die Transformation in den Suchindex, sowie die Erweiterung der Graphstrukturen durch die statische API (vgl. Abschnitt 5.5.3) konnten darüber zielgerichtet abgebildet werden.

Der Adapter Connector sowie die Zugriffsschicht wurden in Java im Rahmen der Arbeit eigenentwickelt. Für die Umsetzung der Adapter und die damit verbunden datenquellenspezifischen Informationsextraktionsmethoden (vgl. Abschnitt 5.7.2) werden unterschiedlichen Bibliotheken (vgl. Abschnitt 6.1.1) genutzt, die in der Regel vom Hersteller der Datenquellen in Form zur Verfügung gestellt werden. Für den Zugriff auf Simulinkmodelle kommt die Conqat-Bibliothek zum Einsatz. Sie erlaubt den objektorientierten Zugriff auf Simulink-Dateien. Dadurch konnte die Extraktion von Schnittstellen, sowie die Informationssammlung zur Qualitätsbewertung durch spezielle Zugriffsmuster, hier insbesondere das Visitor-Pattern [Gam95], beschleunigt werden. Hervorzuheben ist darüber hinaus die Bibliothek Apache POI [POI14] für die Zugriff auf Office Dokumente. Viele Informationen werden heute in Word- und Excel-Dokumenten innerhalb der Datenablagen zu den jeweiligen Entwicklungsartefakten abgelegt. Insbesondere in Verbindung mit dem Natural Language Processing (NLP) Ansatz lassen sich durch diese Artefakte Informationen extrahieren. Als NLP-Framework wird die Unstructures Information Mangement Architecture (UIMA) im Form von DKPro [Dar14] genutzt. DKPro steht dabei für Darmstadt Knowledge Processing und stellt Methoden zum Aufbau von NLP Pipelines zur Verfügung. Umgesetzt wird dabei konkret die in Abschnitt 5.5.2 erläuterte Pipeline.

Zur Demonstration der Wiederverwendung kommt in der Kontextnutzung die Suchmaschine Elastic Search (vgl. Abschnitt 4.5.3) zum Einsatz. Die Indexgenerierung in Elastic Search erfolgt auf Basis frei definierbarer JSON-Strukturen. Ziel der JSON-Strukturen ist es eine hierarchische Repräsentation der zu durchsuchenden Datenbestände zu erzeugen. Da die Wissensbasis als Graph vorliegt war es notwendig eine Methodik der Traversierung zu entwickeln. Die hierarchische JSON-Struktur wird durch eine kreisfreie Breitensuche im Graphen ausgehend von jedem Modellkontextknoten erzeugt. Durch die Breitensuche werden die ehemals in der modellbasierten Repräsentation modellierten hierarchischen Beziehungen auch in der Suchmaschine wieder aufgebaut, so dass zusammengehörige Modellkontext unabhängig von ihrer Vernetzung innerhalb der Graphstruktur in gleiche Zweige der JSON-Struktur transformiert werden. Die entspricht dem in Abschnitt 5.6.2 vorgestellten Konzept.

6.2 Integration in XIL-Datenmanagement

Das Projekt XIL-Datenmanagement der AUDI AG wurde bereits in Abschnitt 1.1 vorgestellt. Das dabei entwickelte Softwaresystem dient zukünftig als zentrale Datenplattform für die Funktionsentwicklung und -absicherung. Es befindet sich aktuell in der Umsetzungsphase. Die grobe Architektur der Umsetzung ist in Abbildung 6.3 dargestellt. Die Softwarelösung ist in die Schichten XIL-Kern und XIL-Prozess unterteilt. Der XIL-Kern dient der Abstraktion beliebiger Datenhaltungssysteme für Entwicklungsartefakte hin zu einer zentralen API für die spezifischen Fachprozesse. In der Regel handelt es sich bei den Datenhaltungssystemen um Versionierungssysteme, wie Subversion (vgl. Abschnitt 3.5) und PTC Integrity [PTC14]. Der XIL-Kern speichert darüber hinaus im Rahmen einer Meta-Daten-Verwaltung Attribute von und Verknüpfungen zwischen Entwicklungsartefakten, wie beispielsweise Beziehungen von Simulationsmodellen, Signalen, Parameter und Konfigurationen (vgl. Abschnitt 3.4). Die Speicherung dieser Beziehungen erfolgt auf Basis relationaler Datenbanken. Der XIL-Prozess baut auf dieser zentralen Datenablage auf und stellt fachspezifische Lösungen

auf einheitlichen Prozessartefakten sicher. Das heißt, es können je nach Entwicklungsdomäne unterschiedliche Softwarelösungen an den XIL-Kern angebunden werden. Es wird jedoch sichergestellt, dass der individuelle Entwicklungsprozess, sowie damit verknüpfte Rollen und Rechte, eines Entwicklungsartefaktes über alle angebundenen Systeme konsistent gehalten wird. Konkret bedeutet dies, dass beispielsweise ein Simulationsmodell, welches in einem Entwicklungsprozess A entwickelt wird und sich dort in einem definierten Prozesszustand aus Rollen, Rechten und Informationen befindet, in einem Entwicklungsprozess B genutzt werden kann, ohne den Kontext zum originalen Prozess zu verlieren. Dadurch ist insbesondere die Rückverfolgbarkeit von Informationen zu Entwicklungsartefakt durchgängig sichergestellt.

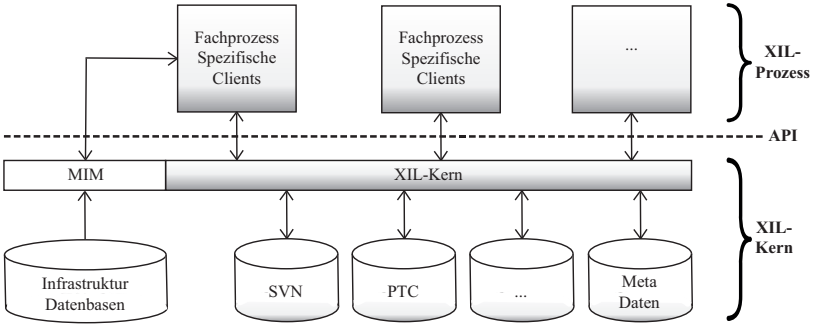


Abbildung 6.3: Integration der Konzepte in XIL-Datenmanagement

In Abbildung 6.3 ist darüber hinaus schematisch die Integration des modularen Informationsmanagementsystems auf XIL-Kern-Ebene dargestellt. Dies erlaubt eine direkte Anbindung an die Datenbasen des XIL-Kern und somit den Zugriff und die Pflege der dort abgelegten Daten. Darüber hinaus können auf dieser Ebene weitere Datenbasen aus dem Bereich der Infrastruktur angebunden werden, die beispielsweise den Einbezug indirekter Informationen (vgl. Abschnitt 5.2) in die Analyseprozesse ermöglichen. In Richtung XIL-Prozess erlaubt diese Form der Integration die Erweiterung der API-Funktionalitäten für fachprozessspezifische Clients. Dadurch wird eine bidirektionale Kopplung geschaffen, in der Such- und Bewertungsfunktionen des modularen Informationsmanagementsystem im Client genutzt werden können. Andererseits existiert eine direkte Rückkopplung von User-Eingaben und beispielsweise die in Abschnitt 5.7.2 beschriebenen Basisreferenzmodelle der kriterienbasierten Bewertung zu pflegen.

Für das im Rahmen der vorliegenden Arbeit entwickelte Konzept der prozessintegrierten Dokumentation existieren im Projekt XIL-Datenmanagement zwei konkrete Anwendungsfälle, die Migration existierender Prozesse in das neu geschaffene System, sowie die dauerhafte Dokumentation in der Produktivphase des Datenmanagementsystems. Die aktuelle Planung des Entwicklungsprojektes sieht eine erste Ausbaustufe für den Produktiveinsatz im ersten Quartal 2016 vor. In Abschnitt 4.2 wurde der Stand der Technik in heutigen Prozessen aufgezeigt. Die heterogenen Datenablagestrukturen würden ohne die Konzepte des modularen Informationsmanagementsystems dieser Arbeit zu einem hohen manuellen Do-

kumentationsaufwand führen, da entweder vorhandene Dokumentationen in das neue System übertragen werden müssten oder eine Nachdokumentation notwendig wäre. In der Migrationsphase unterstützt das modulare Informationsmanagementsystem den Aufbau der Modellkontexte der Entwicklungsartefakte durch automatisierte Extraktion definierter Dokumentationsumfänge aus vorhandenen Entwicklungsprozessen und deren Übertragung in das neue XIL-Kern-System. Nach der produktiven Einführung des neuen Systems unterstützen die Konzepte der vorliegenden Arbeit die dauerhafte Pflege und damit die Qualitätssicherung der verwalteten Entwicklungsartefakte. Durch die Standardisierung der Ablagestrukturen und des damit verknüpften Informationsraumes an Meta-Daten, welche im XIL-Kern gehalten werden, ergeben sich darüber hinaus Vorteile für die Pflege der modellbasierten Repräsentation des modularen Informationsmanagementsystems. Mit der Abbildung der Meta-Daten-Struktur im System sind bereits Teile des Modellkontext über alle Anwender definiert, was in der Folge lediglich geringe individuelle Anpassungen des Modellkontext erwarten lässt.

6.3 Zusammenfassung

In diesem Kapitel wurden die in Kapitel 5 vorgestellten Konzepte im Rahmen einer prototypischen Implementierung umgesetzt und die dabei getroffenen technologischen Entscheidungen vorgestellt. Zum einen stellt der entstandene Demonstrator eine Referenzimplementierung dar, die als Basis eines produktiven Einsatzes im Rahmen des bereits vorgestellten Entwicklungsprojektes XIL-Datenmanagement (vgl. Abschnitt 1.1) dienen soll. Darüber hinaus dient er der praktischen Bewertung der Ergebnisse dieser Arbeit, auf die im Folgenden Kapitel 7 näher eingegangen wird

7 Validierung am praktischen Anwendungsfall

Der Nutzen und die Gültigkeit der getroffenen Annahmen der im Rahmen dieser Arbeit entwickelten Architektur zur prozessintegrierten Dokumentation von Entwicklungsartefakten in modellbasierten Entwicklungsprozess soll im Folgenden anhand eines praktischen Beispiels in einem modellbasierten Entwicklungsprozess der automobilen Funktionsabsicherung demonstriert werden.

7.1 Beschreibung des Anwendungsfalls

Als Anwendungsfall wurde der Modellentwicklungsprozess von Hardware-in-the-Loop-Prüfstandsmodellen gewählt. Er ist besonders geeignet, da hier einerseits eine große Vielfalt an Simulationsmodellen kollaborativ erstellt werden müssen und damit der Dokumentations- und Austauschbedarf entsprechend hoch ist. Auf der anderen Seite sind die genutzten Informationssysteme im Prozess sehr vielfältig und erlauben somit die breite Validierung der Ergebnisse der vorliegenden Arbeit. Ein weiterer Vorteil besteht in der Anlehnung der vorhandenen Modellstrukturen an die tatsächlichen Fahrzeugsysteme. Dadurch kann die Dokumentation der Entwicklungsartefakte neben der prozessintegrierten Erfassung auch durch Informationen aus prozessübergreifenden Systemen ergänzt und bisher nicht systematisch erfasste Zusammenhänge aufgezeigt werden.

Die Hardware-in-the-Loop-Simulation wurde in Abschnitt 3.3 vorgestellt. Für die Simulation an einem HIL-Prüfstand ist stets ein sogenanntes Gesamtfahrzeugmodell notwendig. Dieses Gesamtfahrzeugmodell muss dabei einerseits die Schnittstellen der am HIL geprüften Steuergeräte ansprechen können, andererseits muss es bezüglich der Berechnung harten Echtzeitkriterien genügen. Ein Gesamtfahrzeugmodell setzt sich heute aus vielen Einzelmodellen zusammen, die unabhängig anhand von Modellspezifikationen in unterschiedlichen Teams erarbeitet werden. Aktuelle Gesamtfahrzeugmodelle bestehen aus 70 oder mehr Einzelmodellen. Die Entwicklung dieser Einzelmodelle und deren Integration in ein Gesamtfahrzeugmodell folgt in der Regel einem Entwicklungsprozess nach dem V-Modell (vgl. Abschnitt 2.2).

Im Folgenden wird der Modellentwicklungsprozess des Anwendungsfalls der vorliegenden Arbeit beschrieben. Zur Orientierung ist dieser in Abbildung 7.1 dargestellt. Ein Prozessdurchlauf des Modellentwicklungsprozesses startet stets mit einer Absicherungsanforderung eines übergeordneten Entwicklungsprozesses, in der Regel aus einem Fahrzeug- oder Funktionsentwicklungsprojekt. Aus dieser Absicherungsanforderung werden im Rahmen einer Anforderungsanalyse unter anderem die Anforderungen an ein dafür notwendiges Gesamtfahrzeugmodell abgeleitet. Im konkreten Anwendungsfall werden diese Anforderungen in Atlassian JIRA [Atl14] eingepflegt und ihre Abarbeitung überwacht. Auf Basis der Anforderungen erfolgt die Planung eines Gesamtfahrzeugmodells, das in der Regel auf bereits existierenden Modellen aufbaut.

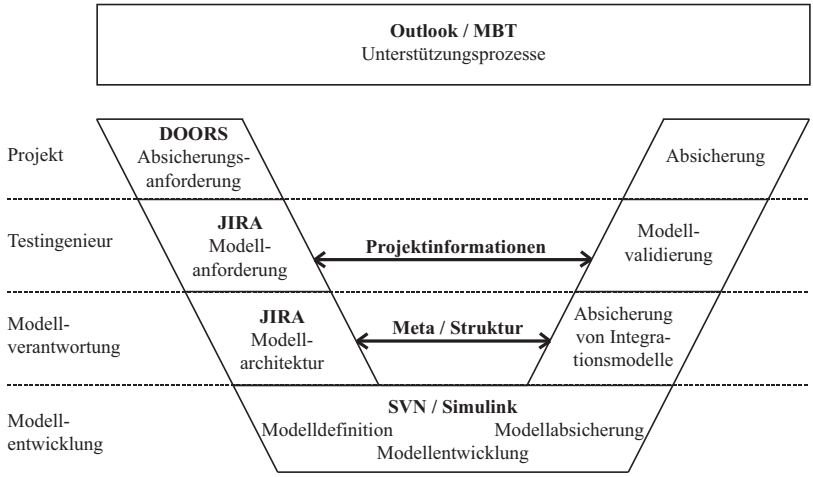


Abbildung 7.1: Entwicklungsprozess des Anwendungsfalls

Zentrale Aufgabe in dieser Phase ist die Erstellung von Arbeitsaufgaben, Modellanpassungen und Modellerstellungen, um ein der Absicherungsaufgabe genügendes Gesamtfahrzeugmodell zu erstellen. Zunächst erfolgt die Spezifikation des Gesamtfahrzeugmodells sowie der notwendigen Teilmodelle als Anforderungen in JIRA. Anschließend erfolgt die Zuweisung der Teilmodellanforderungen zu den sogenannten Modellverantwortlichen. Der Modellverantwortliche verantwortet im Entwicklungsprozess Modelle für wiederkehrende Teilstrukturen von Fahrzeugen. Dies sind beispielsweise Motormodelle, Getriebemodelle oder Reifenmodelle. Er kennt die vorhandenen Simulationsmodelle seiner zugeordneten Teilstruktur und entscheidet auf Basis der Anforderung, ob vorhandene Modelle angepasst werden oder eine Neuerstellung erfolgt. In beiden Fällen erfolgt die Verfeinerung der Anforderungsdefinition. Eine Teilstruktur kann ein einzelnes Modell oder eine Gruppe von Modellen sein. Ein Modellverantwortlicher könnte beispielsweise die Modellgruppe Antriebsstrang verantworten, die unter anderem die vormals genannten Motor- und Getriebemodelle enthält. In diesem Fall weißt der Modellverantwortlicher der Modellgruppe die Teilaufgaben wiederum den jeweiligen Modellverantwortlichen der Einzelmodelle seiner Modellgruppe zu. Ist keine Untermodellerstellung notwendig, weißt der Modellverantwortliche die Modellierungsaufgabe einem Modellersteller zu. Dieser setzt daraufhin die Anforderungen in das jeweilige Modell um. Alle Modelle des Anwendungsfalls dieser Arbeit liegen dabei in Subversion [PCSF08] versioniert vor. Die Verfolgung der Abarbeitung erfolgt während der Modellerstellung wiederum in JIRA. Eine Entwicklungsaufgabe wird stets durch den Verweis auf ein in Subversion versioniertes Modell in einer bestimmten Version abgeschlossen.

Gemäß Entwicklungsprozess folgt der Modellerstellung zunächst eine Reviewphase. Aufgrund der hohen Anzahl an Einzelmodellen, die im späteren Verlauf zu einem Gesamtmodell integriert werden, ist es notwendig, Vorgaben hinsichtlich der Modellierung in Form

von Modellierungsrichtlinien zu definieren. Diese Vorgaben beschreiben beispielsweise Anordnung und Aufbau von Schnittstellen, Menge und Format der Dokumentation aber auch die zulässigen und nicht zulässigen Modellierungselemente. Genügt das Modell den Anforderungen des Modellreview, folgen die Schritte der Verifikation und die Validierung. Die Verifikation wird in der Regel vom den Modellentwickler durchgeführt. Basierend auf der ursprünglichen Modellanforderung werden Testfälle spezifiziert, die in Form von Stimuli als Eingabe- und Referenzwerten als Ausgabedaten beschrieben werden. Entsprechen die Ausgabedaten bei Eingabe der Stimuli für alle Testfälle den Referenzwerten erfolgt die Freigabe des Modells als verifiziert und entspricht damit den Anforderungen. Modelle im Sinne des beschriebenen Prozesses bilden stets reale Komponenten eines Fahrzeugs nach. Die Validierung eines Modells stellt einen Vergleich zwischen Testfällen einer Messung an realen Komponenten und der Berechnung des Modells dar. Entsprechen die Ausgabedaten der Berechnung denen der Messung an realen Komponenten bei gleichen Stimuli, so entspricht das modellierte Modellverhalten, dem der realen Komponente und kann diese damit in der Simulation gleichwertig ersetzen. Die beschriebenen Freigabestufen Review, Verifikation und Validierung sind als aufeinander aufbauende Qualitätsstufen für die Modellgüte zu verstehen und werden im Prozess auch als solche genutzt. Die zu erreichende Qualitätsstufe wird zusammen mit der Modellanforderung beschrieben. Der Entwicklungsprozess endet mit der Freigabe des entwickelten Simulationsmodells in der angestrebten Qualitätsstufe.

7.2 Prozessintegrierte Datenerfassung und -analyse

Die formale Abarbeitung des in Abschnitt 7.1 beschriebenen Prozesses wird durch den Einsatz von Entwicklungsprozesswerkzeugen unterstützt, die in der Regel aus der klassischen Softwareentwicklung stammen. Diese werden, ergänzt durch prozessübergreifende Softwaresysteme, im Folgenden hinsichtlich ihres Informationsumfanges und Informationsgehaltes für den Anwendungsfall der vorliegenden Arbeit beschrieben.

Das Umfeld des Anwendungsfalls basiert auf dem Datenbestand der zentralen Hardware-in-the-Loop-Modellentwicklung der AUDI AG. Im Prozess arbeiten rund 150 direkt beteiligte Personen (Ersteller und Nutzer), sowie rund 100 weitere im indirekten Bereich (wiederverwendende Nutzer). Die Koordination der Entwicklungstätigkeiten dieser Personen erfolgt über die Systeme Subversion und Atlassian JIRA. Als indirekt angebundene Systeme wurde der Microsoft Exchange Server für den Zugriff auf Kontaktdaten, sowie das System MBT gewählt.

Subversion Subversion (SVN) dient im Prozess als zentrale Datenhaltung, sowie zur Versionierung und Variantierung der Simulationsmodelle. Die Ablagestrukturen sind bis auf Entwicklungsartefaktebene hinsichtlich ihrer hierarchischen Struktur sowie der Benennungsnomenklatur definiert. Somit ergibt sich eine semistrukturierter Datenbestand als Datenquelle für das modulare Informationsmanagementsystem. Der Aufbau lässt erste Schlüsse auf die hinterlegten Entwicklungsartefakte zu. Beispielsweise lässt sich aus der Struktur ableiten, dass ein Bibliothekskonzept umsetzt ist, in dem Modelle gleichen Typs in einem Verzeichnis, in der Regel sogar innerhalb einer Simulink-Datei, abgelegt werden. Für die Nutzung innerhalb der Fahrzeugprojekte werden sogenannte SVN-Externals genutzt. Dies sind

Verlinkungen innerhalb des Verzeichnisbaumes, die eine Verbindung zwischen Modellbibliothek und Modellverwendung herstellen und somit eine Variantenbildung ermöglichen. Darüber hinaus ist erkennbar, dass Modelle (Verzeichnis „libs") und Parametersätze (Verzeichnis „ini") in unterschiedlichen Verzeichnissen der gleichen Hierarchieebene hinterlegt werden. Auf gleicher Hierarchieebene werden außerdem im Verzeichnis „doku" Dokumentationen abgelegt. Diese bestehen in der Regel aus beschreibenden Daten zu den Modellen und Parametern in Form von Microsoft Office Dokumenten, wie Word- und Excel-Dateien.

Neben der reinen Datenablage hält Subversion zu jedem verwalteten Datenelement Meta-Daten, wie Versionen, Änderungshistorien oder Autoreninformationen. Insbesondere die Änderungskommentare sind für die Informationsextraktion von Bedeutung. Einerseits lassen sich hieraus Informationen zu Modellnutzungen und Entwicklungsständen extrahieren. Andererseits findet innerhalb der Kommentare eine Verknüpfung zur relevanten JIRA-Tickets in Form von sogenannten Ticket-IDs statt, die für die automatische Analyse eine direkte Verknüpfung zwischen Entwicklungsartefakt und -prozess ermöglichen.

Durch die Anbindung an Subversion können mindestens folgende Informationsquellen erreicht werden:

- Entwicklungsartefakt „Simulationsmodell": Schnittstellen, Modellqualität, Dokumentation im Modell
- Entwicklungsartefakt „Parametersatz": Eigenschaften und Modellierungsformen von Modellen
- Dokumentationsinformationen aus Word- und Excel-Dateien
- SVN-Struktur: interner Aufbau der Modelle als Einzelmodell oder Modellbibliothek, Nutzung von Modellen in Gesamtfahrzeugmodellen
- SVN-Metadaten: Autoren- Änderungs-, Versionsinformationen, JIRA-Verknüpfung

Atlassian JIRA Das System Atlassian JIRA dient im Prozess als Anforderungsmanagementsystem, zentrale Aufgabensteuerung und Lebenszyklusverwaltung für Entwicklungsartefakte. Im System wurde der in Abschnitt 7.1 beschrieben Prozess in Form von Ticket-Typen für Anforderungen und Aufgaben sowie als Zustandsgraph für die Ticketzustände hinterlegt. Der hinterlegte Zustandsgraph entspricht dabei dem in Abschnitt 5.2.6 vorgestellten. Entlang des V-Modells werden die Modellanforderung zunächst in Anforderungstickets überführt und anschließend in Entwicklungsaufgabe heruntergebrochen. Bekommt ein Entwickler eine Aufgabe zugewiesen, erstellt er das entsprechende Entwicklungsartefakt, pflegt es in Subversion ein und stellt eine Verknüpfung zum zugehörigen Aufgabenticket her. Über die Verknüpfung des Aufgabentickets zum Anforderungsticket ergibt sich eine Rückverfolgbarkeit zu den Anforderungen.

Per Definition im Entwicklungsprozess ist festgelegt, dass beim Schließen eines Tickets, gemeint ist das Erreichen eines finalen Zustandes im Zustandsgraphen, alle im Ticket definierten Aufgaben oder Anforderungen erfüllt sind. Auf das verknüpfte Entwicklungsartefakt angewendet bedeutet dies, dass alle geforderten Eigenschaften als erfüllt angenommen und somit für dieses Entwicklungsartefakt dokumentiert werden können. Tickets besteht neben in Prosatext formulierten Inhalten aus einer Menge festgelegter Attribute, die zusätzliche Informationen zum Ticket enthalten. Dazu gehören Lebenzykluszustände, Ticketauto-

ren, -bearbeiter, Versionsinformationen und auch Variantenverknüpfungen beispielsweise zu Fahrzeugprojekten.

Durch die Anbindung an Atlassian JIRA können mindestens folgende Informationsquellen erreicht werden:

- Zustände von Entwicklungsartefakten im Entwicklungsprozess
- Zuständigkeiten von Personen in den Phasen des V-Modells
- Eigenschaften und Anforderungen, die ein Entwicklungsartefakt erfüllt
- Kontextinformationen zu Verwendung und Zugehörigkeit zu Projekten

Microsoft Exchange Server Das System Microsoft Exchange Server wird unter anderem zur zentralen Verwaltung von Kontaktinformationen von Personen vom Unterstützungsprozess der Unternehmens-IT zur Verfügung gestellt. Es wird im Rahmen dieser Arbeit als indirektes System betrachtet, da es selbst keine Informationen zu Entwicklungsartefakten enthält. Für den betrachteten Anwendungsfall der Arbeit sind die Kontaktinformationen insbesondere für zwei Szenarien interessant. Zunächst stellen Ansprechpartner und Verantwortlichkeiten einen wichtigen Anteil der Dokumentation von Entwicklungsartefakten dar, da diese für Nutzer individuelle Fragestellung über die reine Dokumentation hinaus beantworten können. Darüber hinaus liefern diese Kontaktinformationen Daten zur Organisationseinheiten der beteiligten Rollen. Dies können Modellersteller, wie auch Nutzer sein. Befindet sich beispielsweise ein Modellersteller in einer Organisationseinheit für Hardware-in-the-Loop-Simulation, so kann davon ausgegangen werden, dass die von ihm erstellten Simulationsmodelle für diese Simulationsform geeignet sind. Bei einem Modellnutzer aus einer Hardware-in-the-Loop-Organisationseinheit kann angenommen werden, dass er wiederum Modell für diese Simulationsform sucht. Relevant sind diese Informationen insbesondere für das in Abschnitt 5.7.2 beschriebene Referenzmodell „Umfeld".

In den Informationssystemen Subversion und Atlassian JIRA sind Kontaktinformationen lediglich in Form von Nutzernamen beziehungsweise Klartextnamen vorhanden. Diese können jedoch genutzt werden, um den entsprechenden Eintrag im Microsoft Exchange Server zu identifizieren und die Kontaktinformationen zu vervollständigen.

Durch die Anbindung an den Microsoft Exchange Server können mindestens folgende Informationsquellen erreicht werden:

- Kontaktdaten zu Personen, wie Name, E-Mail, Telefon
- Zuordnung zu Organisationseinheiten

MBT Baubarkeitskatalog Das System MBT, hierbei steht MBT für „Modell - Bauteil - Technik", wird vom Unterstützungsprozess der Konzern-IT als zentrales Informationssystem für Baubarkeit zur Verfügung gestellt. Es enthält eine Repräsentation aller Komponenten eines jeden Fahrzeuges des Volkswagenkonzerns. Für Fahrzeugprojekte werden auf diesem Wege beispielsweise die unterschiedlichen Derivate gepflegt. Für alle Komponenten werden darüber hinaus Einsatz- und Entfallsdaten gepflegt. Dadurch kann beispielsweise für ein Fahrzeug geplant werden, dass ein Motor bis zu einem bestimmten Datum in Fahrzeugen verbaut wird und anschließend von einem anderen Motor abgelöst wird. Eine weitere, für

den Anwendungsfall der Arbeit wichtige Information des Systems ist die Abbildung der Gültigkeit von Komponentenkombinationen. Aus dem System MBT lassen sich baubare Kombinationen, wie Motor-Getriebe-Kombinationen, auslesen. Für die Dokumentation und insbesondere die Empfehlung von Simulationsmodellen in der Suche bedeutet dies, dass bei einer Verknüpfung zwischen Motoren und Getrieben des MBT und den Simulationsmodellen des Entwicklungsprozessen bei der Vorauswahl eines bestimmten Motormodells gültige Getriebemodelle empfohlen werden können.

Durch die Anbindung an MBT können mindestens folgende Informationsquellen erreicht werden:

- Fahrzeugprojektbeziehungen und -bezeichnungen
- Verknüpfung von Modell - Echtteil - Beziehungen und Modellvarianten
- Aussagen zu Gültigkeiten von Modellkonfigurationen
- Extraktion von Echtteileigenschaften, welche durch das Modell zu repräsentieren sind

7.3 Definition der Anwendungsszenarien

Für den Nachweis der Übertragbarkeit der im Rahmen der Arbeit entwickelten Konzepte für modellbasierte Entwicklungsprozesse auf den in Abschnitt 7.1 beschriebenen konkreten Anwendungsfall der automobilen Funktionsabsicherung angewendet werden. Dabei wird der entstandenen Prozess der Informationmodellerstellung, Informationsextraktion und Wiederverwendung für ein Simulationsmodell im Detail durchlaufen und das Ergebnis jedes Prozesschrittes analysiert.

Als Simulationsmodell wurde ein Motormodell des Anwendungsfalls ausgewählt, dass sich innerhalb einer Modellbibliothek befindet. Dabei handelt es sich konkret um ein Simulinkmodell für einen 2,0L Verbrennungsmotor, das für die Simulation am Hardware-in-the-Loop-Prüfstand für ein aktuelles Fahrzeugprojekt entwickelt wurde. Für einen Anwender, der dieses Modell in seiner Simulation nutzen möchte, aber die im Entwicklungsprozess genutzten Werkzeug nicht kennt, stellt sich dieses als Dateipfad wie folgt dar: „//10_Source/20_Bibliotheken/01_Motor/lib/Motor_Master_Lib.mdl". Alle weiteren Informationen benötigen eine Interaktion mit direkt im Prozess beteiligten Kollegen, um zunächst den Inhalt des Modells zu verstehen und darauf aufbauend die Nutzbarkeit bewerten zu können. Im Rahmen der Validierung wird die Generierung des Modellkontexts im Sinne der vorliegenden Arbeit für dieses konkrete Simulationsmodell in fünf Anwendungsszenarien demonstriert. Mit der erfolgreichen Erstellung des Modellkontextes erhält der beschriebene Anwender alle für ihn relevanten Informationen, um eine Nutzenbewertung für seinen Anwendungsfall durchführen zu können.

Die Anwendungsszenarien erstrecken sich dabei über alle Phasen der Informationsverarbeitung des modularen Informationsmanagementsystems - beginnend bei der Erstellung eines Informationsmodells, über die Datenakquise bis hin zur Wiederverwendung der automatisch generierten Dokumentation. Für jedes Anwendungsszenario werden Voraussetzungen für die Durchführung, sowie das erwartete Ergebnis als Messkriterium der erfolgreichen Anwendung angegeben.

Anwendungsszenario 1	Modellbasierte Erstellung eines Informationsmodells
Voraussetzung: Es existiert ein formaler Entwicklungsprozess sowie eine Definition notwendiger Entwicklungsartefakte und deren zugehöriger Mindestdokumentation	
Erwartetes Ergebnis: Es existiert ein auf Basis von EMF (vgl. Abschnitt 4.3.1) modellbasiert erstelltes, generierbares Informationsmodell, das den Modellkontext eines Entwicklungsartefaktes im formalen Entwicklungsprozess repräsentiert. Die Spezifikation entspricht den Modellierungsrichtlinien aus Abschnitt 5.4.3 und erlaubt damit die Generierung einer dynamischen API.	

Tabelle 7.1: Anwendungsszenario - Modellbasierte Erstellung eines Informationsmodells

Anwendungsszenario 2	Erfassung einer prozessintegrierten Basisdokumentation
Voraussetzung: Es existieren quellsystemspezifische Adapter für jedes berücksichtigte Quellsystem des Anwendungsfalls. Darüber hinaus existiert mindestens ein Adapter für jede Informationsklasse des Informationsmodells.	
Erwartetes Ergebnis: Nach Durchführung der Datenakquisephase existiert für jedes Entwicklungsartefakt ein Modellkontext gemäß des definierten Informationsmodells. Der Nachweis wird durch exemplarische Überprüfung erbracht.	

Tabelle 7.2: Anwendungsszenario - Erfassung einer prozessintegrierten Dokumentation

Anwendungsszenario 3	Qualitätsbewertung von Simulationsmodellen
Voraussetzung: Das modulare Informationsmanagementsystem muss Zugriff auf die Simulationsmodelle des Anwendungsfalles haben.	
Erwartetes Ergebnis: Die Qualitätsbewertung wird für mindestens drei Simulationsmodell exemplarisch durchgeführt. Als Vergleichsreferenz wir der Durchschnitt der Messwerte über alle zugreifbaren Simulationsmodelle genutzt. Aufgrund der Menge an Einzelmetriken wird der Nachweis anhand des Vergleichs exemplarischer Messwerte erbracht.	

Tabelle 7.3: Anwendungsszenario - Qualitätsbewertung von Simulationsmodellen

Anwendungsszenario 4	Erfassung einer prozessübergreifenden Dokumentation
Voraussetzung: Es existiert eine inhaltliche Verknüpfung zwischen den Modellattributen eines Simulationsmodells und der Echtteilrepräsentation im System MBT	
Erwartetes Ergebnis: Die Dokumentation und damit die verfügbaren Informationen der Simulationsmodelle werden durch Informationen der von ihnen repräsentierten Echtteile ergänzt. Der Nachweis erfolgt am Beispiel eines Motormodells, dessen Informationen durch fahrzeugprojektspezifische Leistungsdaten vervollständigt werden.	

Tabelle 7.4: Anwendungsszenario - Erfassung einer prozessübergreifenden Dokumentation

Anwendungsszenario 5	Generierung eines Suchindex für Elastic Search
Voraussetzung: Es existiert für jedes zu indexierende Simulationsmodell ein Modellkontext im Sinne der vorliegenden Arbeit.	
Erwartetes Ergebnis: Der Suchindex liegt vor und kann für die zielgerichtete Suche nach Simulationsmodellen genutzt werden.	

Tabelle 7.5: Anwendungsszenario - Durchführung kriterienbasierter Suchen

7.4 Durchführung und Auswertung der Validierung

Die Validierung der im Rahmen der vorliegenden Arbeit entwickelten Ansätze wird durch die Anwendung auf den in Abschnitt 7.1 beschriebenen Modellentwicklungsprozess durchgeführt. Zum Nachweis werden die in Abschnitt 7.3 vorgestellten Anwendungsszenarien anhand eines Demonstationsaufbaus im produktiven Entwicklungsprozess am Beispiel eines realen Fahrzeugprojektes angewendet und anschließend bewertet.

Abbildung 7.2 zeigt einen Überblick der Informationsverarbeitung des modularen Informationsmanagementsystem unter Nutzung der in Kapitel 6 beschriebenen prototypischen Implementierung und den in Abschnitt 7.2 vorgestellten Quellsystemen des Modellentwicklungsprozesses.

Ausgehend von einem in Anwendungsszenario 1 (vgl. Tabelle 7.1) modellbasiert erstellten Informationsmodells erfolgt zunächst die Datenakquise in den Quellsystemen Subversion, Atlassian JIRA und Microsoft Exchange. Zur Anwendung kommt hier die dynamische API (vgl. Abschnitt 5.5.1) zusammen mit den in der prototypischen Entwicklung umgesetzten Adaptern, um die relevanten Daten prozessintegriert extrahieren zu können. Eine Anpassung der Quellsysteme ist dafür nicht notwendig. Die dynamische API übergibt die Informationen anschließend an den Modellkontextgenerator (vgl. Abschnitt 5.8), der auf Basis des modellbasierten Informationsmodells konkrete Instanzen für jedes Simulationsmodell des Quellsystems Subversion erstellt. Diese Phase der Informationsverarbeitung wird in Anwendungsszenario 2 (vgl. Tabelle 7.2) validiert.

Im Anschluss erfolgt die Qualitätsanalyse für jedes im Informationssystem dokumentierte Simulationsmodell. Die Messung erfolgt durch die Anwendung der Metriken des in Abschnitt 4.4.2 vorgestellten Qualitätsmodells nach [Sch12]. Die Ergebnisse werden in die Modellkontexte der Simulationsmodell eingepflegt und dienen der Bewertung der Simulationsmodelle im Rahmen einer späteren kriterienbasierten Modellbewertung nach dem Basisreferenzmodell „Qualität" (vgl. Abschnitt 5.7.2). Diese Phase der Informationsverarbeitung wird in Anwendungsszenario 3 (vgl. Tabelle 7.3) validiert.

Im nächsten Schritt erfolgt die Transformation der Modellkontexte nach dem in Abschnitt 5.8 beschriebenen Verfahren in eine graphbasierte Repräsentation. Diese wird in der Graphdatenbank Neo4j persistiert. Anschließend werden unter Nutzung der Modellkontextdaten und der statischen API (vgl. Abschnitt 5.5.1) indirekt mit den Simulationsmodellen verknüpfte Daten aus geeigneten Quellsystemen extrahiert. Im konkreten Anwendungsfall kommt hier das System MBT zum Einsatz, um die Modellkontexte von Motormodellen mit ihren repräsentierten, realen Fahrzeugmotordaten zu verknüpfen. Anwendungsszenario 4 (vgl. Tabelle 7.3) validiert diese Datenakquise von indirekt verknüpften Informationen. Abbildung 7.2 zeigt beispielhaft die mittels der statischen API generierten Informationen. Dies

können Relationen zwischen Modellkontexten oder an Modellkontexte angehängte, zusätzliche Informationen sein. Wie in Abschnitt 5.5.1 definiert erfolgt die Verknüpfung stets auf Wurzelknotenebene.

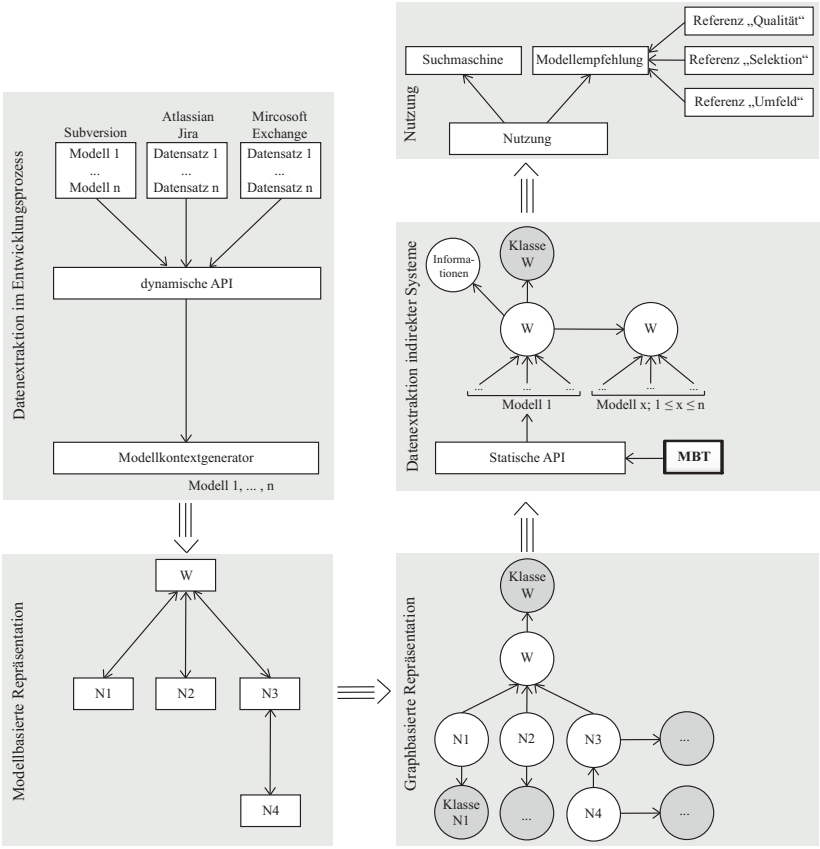


Abbildung 7.2: Überblick zur Durchführung der Validierung am Anwendungsfall

Abschließend erfolgt die Nutzung der resultierenden Wissensbasis der Simulationsmodelle des Anwendungsfalls für Generierung eines Suchindex für Elastic Search. Dieser Schritt dient der Validierung der Wiederverwendbarkeit der gesammelten Informationen zur Unterstützung des modellbasierten Entwicklungsprozesses. Anwendungsszenario 5 (vgl. Tabelle 7.5) validiert die Durchführung dieser Suchindexgenerierung.

7.4.1 Beschreibung der Datenbasis

Als Datenbasis zur Validierung der Konzepte der vorliegenden Arbeit dient ein aktuelles Fahrzeugprojekt der AUDI AG, das in der in Abschnitt 7.1 beschriebenen Toolkette entwickelt und genutzt wird. Konkret handelt es sich um ein Gesamtfahrzeugmodell (vgl. Abschnitt 7.1) für die Hardware-in-the-Loop-Simulation, wie es am Prüfstand Verwendung findet.

Das Gesamtmodell besteht aus 305 Simulinkmodellen, von denen 132 Simulinkmodelle für die Modellierung konkreter Fahrzeugeigenschaften oder -funktionen genutzt werden. Die übrigen Modelle bilden den architektonischen Rahmen für das Gesamtfahrzeugmodell, d.h. sie dienen der Verschaltung der genannten Simulinkmodelle zu einem simulationsfähigen Modell. Darüber hinaus enthält die Datenbasis Parametersätze, Dokumentationsdateien sowie Matlab-Skripte, die ebenfalls zur Architektur des Gesamtfahrzeugmodells gehören.

Die Datenbasis liegt versioniert im Subversion Repository und besteht in der genutzten Revision als ausgecheckte Kopie aus 8078 Dateien mit einer Gesamtgröße von 3,46GB Daten auf der Festplatte. Die Daten sind semistrukturiert (vgl. Abschnitt 4.5) im Dateisystem geordnet.

Die beschriebenen 132 Simulinkmodelle mit konkreter Repräsentation des Fahrzeugs sollen im Rahmen der folgenden Untersuchungen näher betrachtet werden, da sie den das eigentliche Entwicklungswissen abbilden und ihre Dokumentation und Wiederverwendung von Bedeutung ist.

Als Grundlage der Erfassung einer minimalen Informationsbasis dient die bereits in Abschnitt 5.1 erwähnte Erhebung der aus Sicht der Entwickler und Anwender wichtigsten Informationen für die Dokumentation von Simulationsmodellen. Dazu wurden im Rahmen der Arbeit Abstimmungen in 8 Fachabteilungen in den Bereichen Elektronik, Antrieb, Fahrwerk sowie Karosserie durchgeführt. Die Fachabteilungen decken dabei Model-, Software- und Hardware-in-the-Loop simulierende Fachabteilungen berücksichtigt. Tabelle 7.6 listet die Ergebnisse der Abstimmungen und die im Rahmen der Arbeit untersuchten Quellsysteme, die diese Informationen enthalten. Nach Abschluss der Untersuchungen wurden die gelisteten Informationen als Standard für eine Mindestdokumentation bestätigt und werden seit dem bereits in einigen Fachbereichen für jedes Simulationsmodell erfasst. In der Regel werden die Informationen aktuell händisch in Microsoft Word-Dokumenten erfasst, was sie für eine Suche nicht zugreifbar macht.

7.4.2 Durchführung und Auswertung der Anwendungsszenarien

Der vorliegende Abschnitt beschreibt die Durchführung und Auswertung der in Abschnitt 7.3 vorgestellten Anwendungsszenarien.

Durchführung und Auswertung des Anwendungsszenarios 1 Der erste Anwendungsfall (vgl. Tabelle 7.1) dient der Validierung der modellbasierten Erstellung eines Informationsmodells. Die Voraussetzung eines formalen Entwicklungsprozesses ist gegeben. Dieser wurde bereits in Abschnitt 7.1 im Detail vorgestellt. Darüber hinaus existiert eine definierte Mindestdokumentation, die in Tabelle 7.6 dargestellt wurde. Die Mindestdokumentation

entspricht der in Abschnitt 5.1 definierten minimalen Informationsbasis. Diese wird für die Durchführung des Anwendungsfalles mittels EMF und den in Abschnitt 5.4.3 definierten Modellierungsrichtlinien in eine modellbasierte Repräsentation überführt. Abbildung 7.3 zeigt das daraus resultierende Gesamtmodell im Überblick. Die Modellierung wurde spaltenweise entsprechend der Kategorien Meta-Daten, Architektur, Projekt und Qualität umgesetzt. Für die Verortung des Simulationsmodells wurde im Meta-Daten-Zweig ein modellspezifischer Zweig für die Modelldatei sowie zugeordnete Parametersätze modelliert.

Bezeichnung	**Wert**	**Quellsystem**
Meta-Daten		
Modellname	String	SVN
Beschreibung	String	JIRA
Modelltyp	String	JIRA
Dateipfad	URI	SVN
Ersteller	Name	SVN, SVN
Erstellungsdatum	Datum	SVN
Fachabteilung	String	Exchange
Version	Revisionsnummer	SVN
Schnittstellen		
Signalname	String	Simulationsmodell
Signaltyp	Datentyp	Simulationsmodell
Defaultwert	Datentyp	Simulationsmodell
Fahrzeugprojekt		
Bezeichnung	String	JIRA, MBT
Projektidentifikation	String	JIRA, MBT
Parametrierung		
Parametername	String	SVN
Parametersatz	URI	SVN
Bedeutung	String	JIRA
Qualität		
Reifegrad	String	JIRA, Simulationsmodell

Tabelle 7.6: Überblick minimaler Anforderungen an eine Informationsbasis

In jedem Zweig wurden die zu dokumentierenden Entwicklungsartefakte hierarchisch hinsichtlich ihrer enthaltenen Artefakte in einzelnen Informationsklassen (vgl. Abschnitt 5.4.3) modelliert. Einem Modell beispielsweise sind eine Modelldatei sowie potenziell mehrere Parametersätze zugeordnet. Die beschreibenden Informationen einer jeden Informationsklasse sind als Attribute der jeweiligen Klasse modelliert.

In der modellbasierten Repräsentation wurden alle dokumentationsrelevanten Informationen nach Tabelle 7.6 modelliert. Anschließend wurde unter Nutzung von Codegenerierung das Informationsmodell der dynamische API automatisch generiert. Abbildung 7.4 zeigt die generierten Informationsklassen, die für die dynamische API genutzt werden. Mittels Reflection werden diese Informationsklassen zur Laufzeit geladen und erlauben dadurch die dynamische Anpassung der API mit der Veränderung der modellbasierten Repräsentation (vgl. Abschnitt 5.5.3).

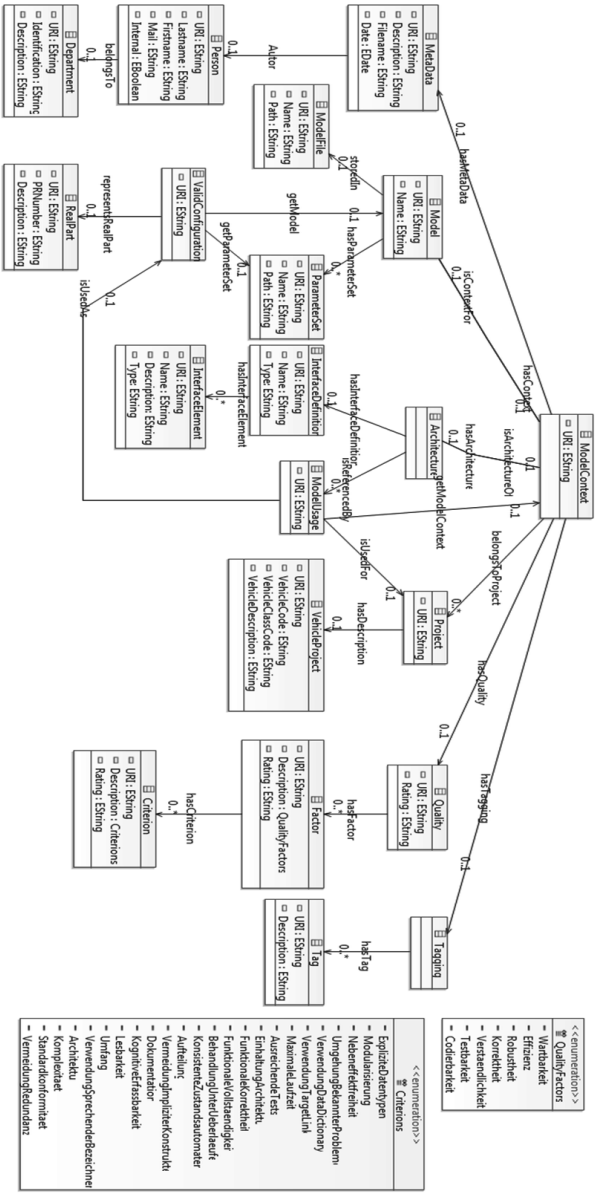


Abbildung 7.3: EMF-Modell des entwickelten Modellkontexts

Im Anwendungsfall 1 wurde die für die modellbasierte Entwicklung in der automobilen Funktionsabsicherung minimale Informationsbasis in eine modellbasierte Repräsentation, den Modellkontext, nach dem Konzept der vorliegenden Arbeit überführt. Damit hat der Anwendungsfall die Nutzbarkeit der Repräsentationsform gezeigt. Darüber hinaus wurde die Codegenerierbarkeit auf Grundlage der definierten Modellierungsrichtlinien nachgewiesen.

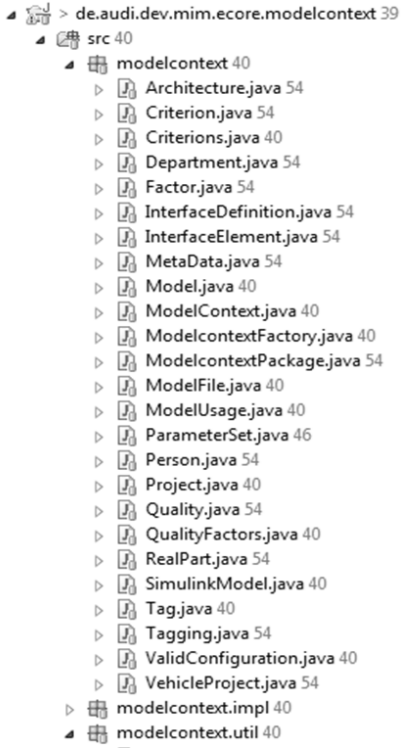


Abbildung 7.4: Generierte Informationsklassen der dynamischen API

Durchführung und Auswertung des Anwendungsszenarios 2 Der zweite Anwendungsfall (vgl. Tabelle 7.2) soll die Datenerfassung einer prozessintegrierten Dokumentation auf Basis der Konzepte der vorliegenden Arbeit nachweisen. Wie in Abschnitt 6.1 beschrieben wurden im Rahmen der Arbeit Adapter für die Datenextraktion aus den Datenhaltungssystemen des für die Validierung genutzten Entwicklungsprozesses entwickelt.

Im Folgenden wird der Prozess der Datenakquise für ein konkretes Beispielmodell nachvollzogen. Dies ist für die Validierung ausreichend, da in gleicher Weise der beschriebene Prozess für alle 132 Simulationsmodelle der Datenbasis wiederholt wird.

Als Beispiel dient die Dokumentation eines Motormodells. Im genutzten Fachprozess werden Simulationsmodelle in Form von Modellbibliotheken abgelegt. Modellbibliotheken enthalten in einer Modelldatei auf oberster Modellebene nur sogenannte Subsysteme, ohne dass diese untereinander verknüpft sind. Die Verwendung der Modelle im Gesamtfahrzeugmodell erfolgt dann durch Referenzierung auf die Modelldatei und das Subsystem der Modellbibliothek.

Prozessual beginnt die Datenakquise mit der Entdeckung der Simulationsmodelldatei im Subversion Repository. In der ersten Phase erfolgt die Analyse der Modelldatei innerhalb des Subversion-Adapters und die Erkennung, ob es sich um ein Einzelmodell oder eine Modellbibliothek handelt. Wie bereits beschrieben wird anhand der obersten Modellebene hier die Unterscheidung durchgeführt. Darüber hinaus gibt es Schlüsselwörter innerhalb des Modellpfades im Repository, dass Rückschlüsse auf ein Einzelmodell oder eine Bibliothek zulässt. So wären beispielsweise Dateipfade mit den Zeichenketten „Lib" oder „Bibliothek" Indizien für eine Einsortierung. Im konkreten Entwicklungsprozess der Validierung sind diese Bezeichnungen aufgrund der bereits stark strukturieren Arbeitsweise im Pfad verpflichtend und können als Referenz herangezogen werden.

Nach der Entdeckungsphase wird zunächst eine Instanz des Modellkontextes für diese Datei über die dynamische API angefordert. Grundlage dafür ist der Dateipfad im Repository. Existiert bereits ein Modellkontext für dieser Datei, so wird eine mit den bereits bekannten Daten vorbefüllte Instanz des Modellkontextes zurückgegeben. Ist noch kein Modellkontext vorhanden, so wird eine neue Instanz erzeugt und zurückgegeben.

Ausgehend von der Annahme, dass es ich um einen neuen Modellkontext handelt werden in diesen vom Subversion-Adapter zunächst die dort vorhandenen Meta-Daten hinterlegt. Dazu gehören mindestens der Modellpfad, die Modellrevision, die Nutzerinformation der letzten Revision und das Änderungsdatum. Darüber hinaus erfolgt die Suche nach relevanten Dokumenten für die Analyse. Dazu zählen insbesondere Office-Dokumente, wie Microsoft Word oder Excel-Dateien, aber auch Parametersätze. Diese werden in der Regel in relativen Pfaden innerhalb der hierarchischen Struktur (vgl. Abbildung 7.5) abgelegt und können somit vom Adapter zielgerichtet zugeordnet werden. Für Parametersätze werden entsprechend des Modellkontextes Name und Dateipfad hinterlegt. Die Office-Dokumente werden später für weitere Informationsextraktionen genutzt. Als Modellname wird für Einzelmodelle der Dateiname ohne Dateierweiterung und für Modellbibliotheken die Bezeichnung des Subsystems auf oberster Ebene herangezogen.

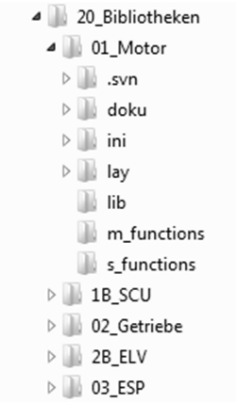


Abbildung 7.5: Hierarchische Ablagestrukturen der Datenbasis

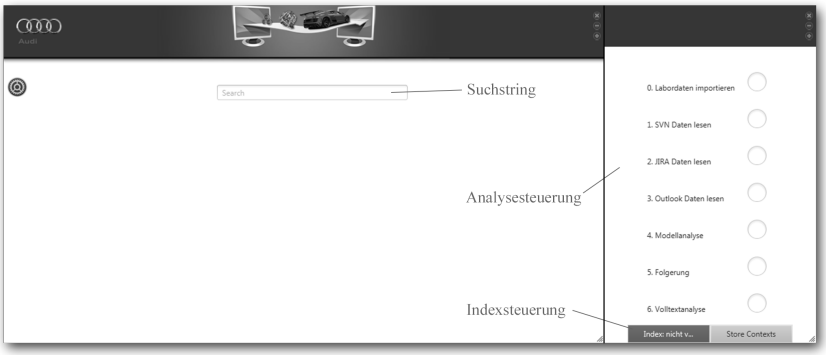


Abbildung 7.6: Überblick des entwickelten Demonstrators

Abbildung 7.6 zeigt den im Rahmen der Arbeit im entwickelten Demonstrator. Die Gesamtanalyse der entwickelten Dokumentationsmethodik kann dabei entweder vollständig automatisiert oder manuell (Analysesteuerung) in ihren einzelnen Verarbeitungsschritten durchlaufen werden. Eine Indexssteuerung aktualisiert auch für unvollständige Modellkontexte den umgesetzten Suchindex, der mithilfe eines Suchstrings die Ergebnisse der Analyse anzeigt. Um die Analyseschritte im einzelnen Nachvollziehen zu können, werden im Rahmen dieses Abschnitts die einzelnen Phasen Schritt für Schritt durchlaufen und beispielhaft die Auswirkungen auf den Informationsgehalt der Wissensbasis betrachtet. Dazu wird in den Abbildungen jeweils das Suchergebnis, ein Ausschnitt des Modellkontextes, für das oben genannte und besonders betrachtete Simulationsmodell im linken Teil dargestellt.

Abbildung 7.7 zeigt das Ergebnis nach Abschluss des ersten Schritts, der SVN-Analyse (vgl. Abschnitt 7.2). Initial wurde eine Bibliothek erkannt, ein Modellkontext für die oberste Ebene angelegt und die Modelldatei in den Kontext eingetragen. Die konkreten Bezeichnungen von Personen und Projekten wurden dabei in den Abbildungen durch Bezeichner im Format: „<Bezeichner>" ersetzt, da es sich hierbei in der Datenbasis um tatsächliche Mitarbeiter und Projekte der AUDI AG handelte. Auf oberster Ebene enthält die Modelldatei also ein Subsystem mit der Bezeichnung „Projektspezifische Blöcke <Fahrzeugprojekt>", welches das Motormodell dieses Fahrzeugprojektes repräsentiert. Die Qualifikation des Modells als Motormodell lässt sich hierbei anhand des Dateipfades mit entsprechenden Bezeichnern realisieren.

Neben den hier sichtbaren Informationen werden im Rahmen der Analyse des Subversion Repositories weitere Daten extrahiert. Dazu zählen unter anderem die Nutzernamen der letzten Bearbeiter dieses Modells, sowie die Kommentare aller Versionen, die für dieses Modell existieren. Diese dienen als Ausgangsbasis für weitere Analysen. Die Nutzernamen werden dem Modellkontext als erste Elemente der Klasse „Personen" hinzugefügt und später um weitere Informationen ergänzt. Die Ergebnisse der Kommentaranalyse hängen stark von deren Strukturierung ab. Eine Volltextanalyse kann hierbei bereits Rückschlüsse auf Projekte, Dateizusammenhänge und Modellinhalte ziehen. Diese sind in der Regel abhängig von Dokumentationsumfang und Standardisierung von Begriffen, beispielsweise durch Thesauri (vgl. Abschnitt 4.1.4). Im konkreten Anwendungsfall der Arbeit liefern die Kommentare beispielsweise Verweise auf relevante Tickets im System Atlassian JIRA (vgl. Abschnitt 7.2), die nach einer definierten Formatierung im Kommentar hinterlegt werden. Abbildung 7.8 stellt den Modellkontext des im Rahmen der Arbeit betrachteten Beispielmodells nach Abschluss des jeweiligen Analyseschritts dar. Grau markiert sind die bekannten Attribute. Für grau markierte Informationsklassen sind alle Attribute bekannt.

Nach Abschluss der SVN-Analyse liegen unter anderem die für dieses Simulationsmodell relevanten Ticket-Identifikatoren (im Folgenden als Ticket ID bezeichnet), im System Atlassian JIRA vor. Es folgt die JIRA-Analyse, die unter Nutzung des JIRA-Adapters und der Ticket IDs die dort abgelegten Informationen extrahiert. Ein Ticket in Atlassian JIRA bezeichnet eine Informationseinheit innerhalb des Softwaresystem, das identifiziert durch eine systemweit eindeutige Ticket ID Informationen eines konkreten Sachverhaltes dokumentiert. Meist sind unter diesem Sachverhalt Aufgaben oder Anforderungen an jemanden oder etwas zu verstehen. Ein Ticket ist in der Regel gekennzeichnet durch eine Ticket ID, eine Menge verpflichtender sowie optionaler Attribute, mindestens einen beschreibenden Text, sowie Verknüpfungen zu weiteren, mit diesem Ticket in logischem Zusammenhang stehenden Tickets. Im konkreten Anwendungsfall werden für jedes durch seine Ticket ID zum Simulationsmodell verknüpfte Ticket folgende Informationen extrahiert:

- Fahrzeugprojekt: hinterlegt als Attribut
- Bearbeiter: hinterlegt als Attribut
- Ticketstatus: hinterlegt als Attribut
- Verknüpfte Ticket IDs: hinterlegt als Attribut
- Beschreibungstext: hinterlegt als Volltext
- Dokumente: hinterlegt als Dateianhänge

Abbildung 7.7: Demonstratorsuchergebnis nach SVN-Analyse

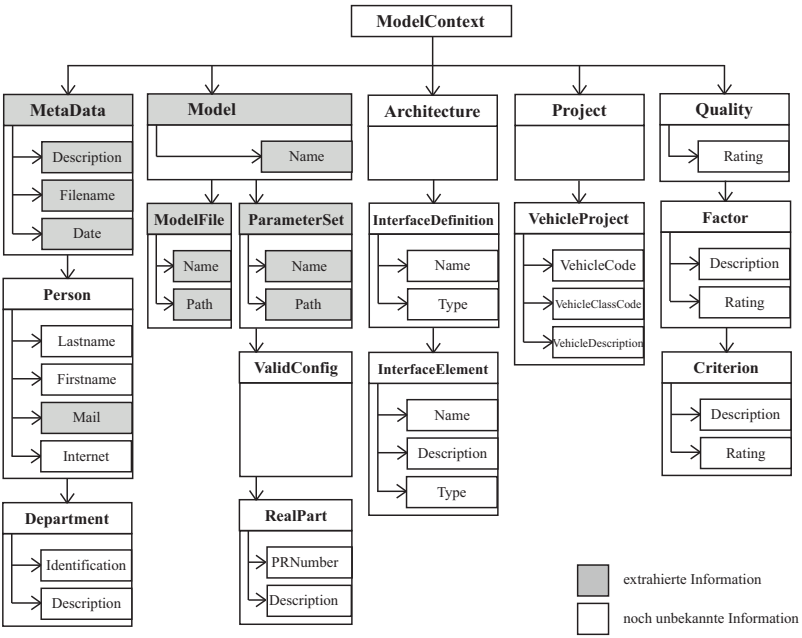


Abbildung 7.8: Modellkontext nach SVN-Analyse

Die Information der Fahrzeugprojekte erlaubt Rückschlüsse auf die Nutzung eines Simulationsmodells im Kontext dieser Projekte und wird direkt in den Modellkontext übernommen. Ein Bearbeiter ist eine für dieses Ticket verantwortliche Person, die in der Regel die Aufgabe der Bearbeitung des Tickets und damit auch des Simulationsmodells erhält. Mit dieser Information kann der mit Hilfe der SVN-Informationen bestimmte Ansprechpartner zum Simulationsmodell validiert werden. Der Ticketstatus erlaubt Aussagen über den Bearbeitungsstatus. Dieser erlaubt Rückschlüsse darauf, ob ein Modell fertig entwickelt ist und genutzt werden kann, oder ob es sich beispielsweise noch in Entwicklung befindet. Diese Information wird als Meta-Datum im Modellkontext hinterlegt. Die verknüpften Ticket IDs erlauben es, ein weitere für dieses Simulationsmodell relevante Tickets zu identifizieren. Beschreibungstexte sowie Dokumente werden, wie auch schon die SVN-Kommentare, auf Schlüsselworte durchsucht, die weitere Informationen zum Simulationsmodell enthalten. Im konkreten Anwendungsfall enthalten diese Texte beispielsweise für HIL-Prüfstände die konkrete Bezeichnung des Prüfstandes, an dem dieses zum Einsatz kommt. Abbildung 7.9 zeigt den gewählten Auszug des Modellkontextes am Beispiel des Demonstrators nach der JIRA-Analyse und Abbildung 7.10 den daraus resultierenden Befüllungsgrad des Modellkontextes.

Im dritten Schritt erfolgt die Analyse des Microsoft Exchange Servers. Die im Rahmen der SVN- und JIRA-Analyse gesammelten Daten zu Personen bestehen aus Nutzernamen für das jeweilige Softwaretool, die die jeweilige Person im System eindeutig identifizieren. Im Fall von SVN ist dies beispielsweise die E-Mail-Adresse. Im zentralen Microsoft Exchange Server werden alle Mitarbeiter gepflegt. Der Exchange-Adapter des Demonstrators erlaubt den Zugriff auf die Adressbücher des Exchange Servers. Anhand der bereits extrahierten E-Mail-Adressen lassen sich die Einträge der für ein Modell relevanten Personen eindeutig identifizieren. Aus den Einträgen werden folgende Informationen extrahiert:

- Name und Vorname
- Fachabteilung - sowohl das Abteilungskürzel, als auch die Bezeichnung
- Firmenzugehörigkeit - insbesondere bei externen Mitarbeitern relevant
- Standort des Arbeitsplatzes
- Telefonnummer

Mit diesen Informationen lassen sich die personenbezogenen Daten des Modellkontextes vollständig ausfüllen. Abbildung 7.11 zeigt den gewählten Auszug des Modellkontextes nach der Exchange-Analyse und Abbildung 7.12 wieder den korrespondierenden Befüllungsgrad des Modellkontextes.

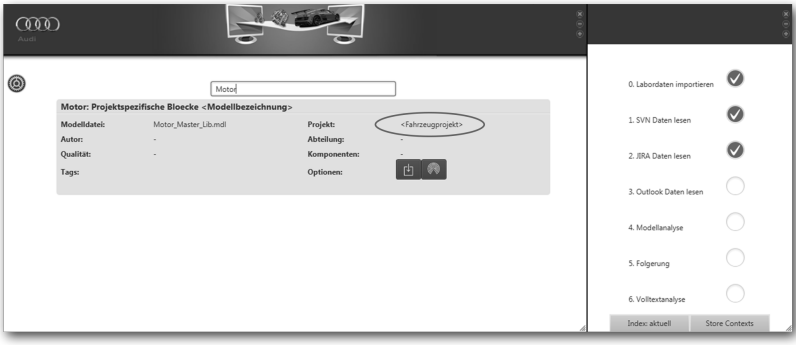


Abbildung 7.9: Demonstratorsuchergebnis nach JIRA-Analyse

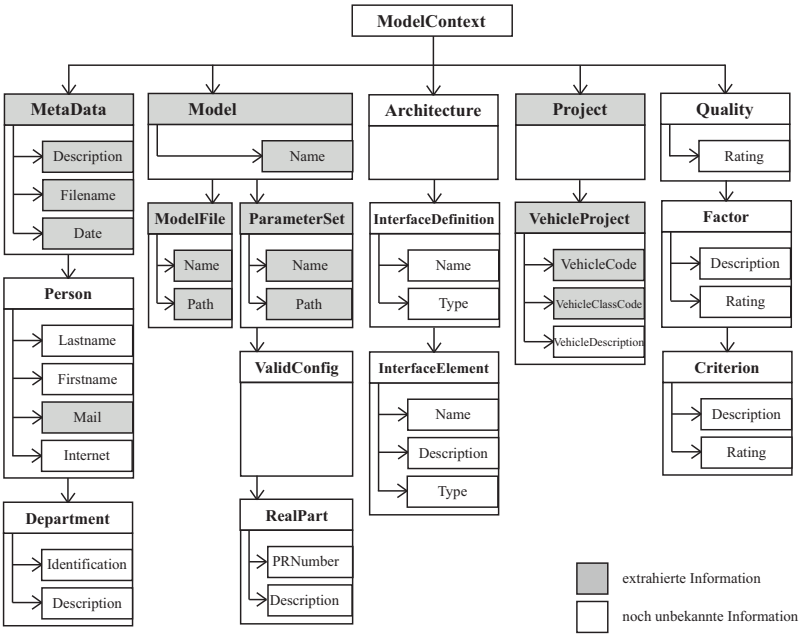


Abbildung 7.10: Modellkontext nach JIRA-Analyse

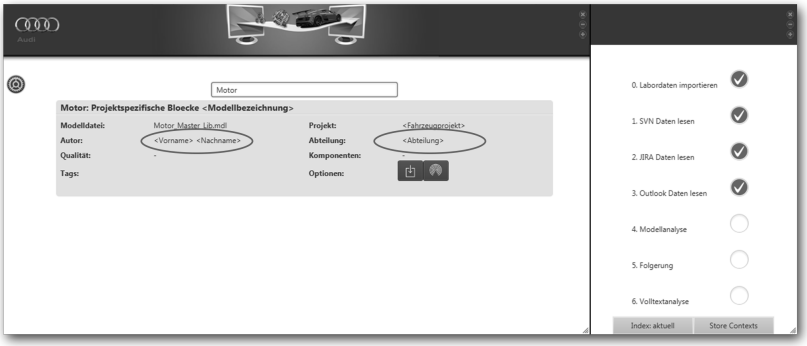


Abbildung 7.11: Demonstratorsuchergebnis nach Exchange-Analyse

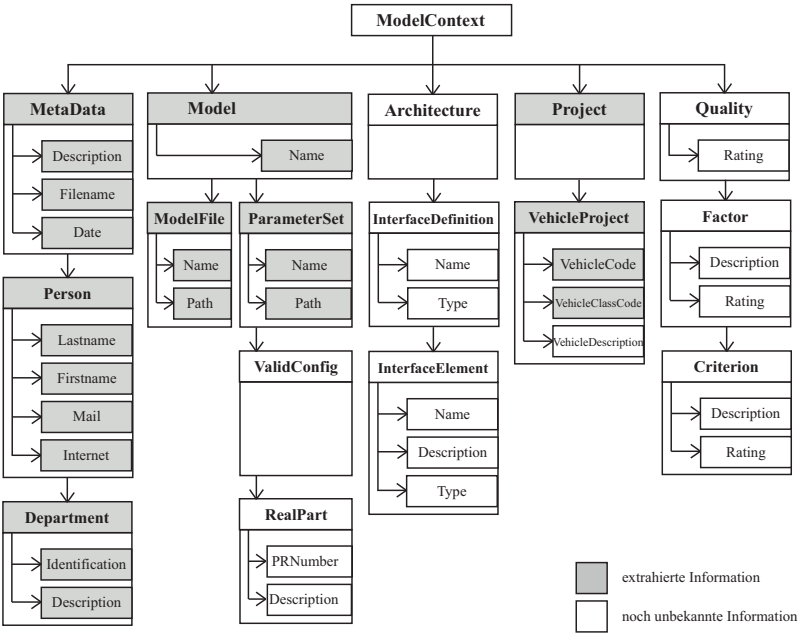


Abbildung 7.12: Modellkontext nach Exchange-Analyse

Durchführung und Auswertung des Anwendungsszenarios 3 Der dritte Anwendungsfall (vgl. Tabelle 7.3) überprüft die Informationsextraktion aus dem Simulationsmodell selbst. Voraussetzung für die Durchführung der Analyse ist die lokale Verfügbarkeit der Modelldatei auf dem Rechner des Demonstrators. Im konkreten Anwendungsfall wurde dafür die aktuelle Revision, die so genannte Head-Revision, des genutzten Fahrzeugprojektes auf den Testrechner ausgecheckt.

Der Zugriff auf die Modelldatei erfolgt unter Nutzung der in Abschnitt 6.1.1 beschriebenen Bibliothek. In einem ersten Schritt werden die Modellschnittstellen auf oberster Ebene extrahiert. Auf Basis der Modellierungsrichtlinien des Entwicklungsprojektes bestehen diese stets aus so genannten „Bus-Selector"-Blöcken am Eingang und „Bus-Creator"-Blöcken am Ausgang. Ein Bus ist hierbei als hierarchisch strukturierte Zusammenfassung von Signalen zu verstehen, die am Eingang in einzelne Signale aufgeteilt und am Ausgang wieder zu einem Bus zusammengeführt werden. Die Signale im Eingangs- bzw. Ausgangs-Bus repräsentieren die Schnittstellen des Modells. Bei der Analyse lassen sich die Signalbezeichnung, der Datentyp, sowie die Zugehörigkeit zum Bus extrahieren und im Modellkontext als Schnittstelleninformation einpflegen.

Im Anschluss an die Schnittstellenanalyse erfolgt die Qualitätsmessung nach dem in Abschnitt 4.4.2 erläuterten Qualitätsmodell. Dafür wurden im Rahmen des Demonstrators insgesamt 49 Metriken implementiert. Eine Metrik misst dabei stets einen Kennwert des Modells, indem beispielsweise das Vorhandensein bestimmter Blöcke, die Anordnung der Blöcke oder die Modellierungstiefe gemessen werden. Diese Einzelmessungen werden anschließend zu Faktoren zusammengefasst, wobei jede Metrik einen Qualitätsaspekt zum jeweiligen Faktor beiträgt.

Wie in Abschnitt 5.7 erläutert ist die Qualität eines Modells stets ein relativer Wert bezogen auf eine Referenzgröße. Exemplarisch soll im Rahmen der Validierung das gewählte Beispielmodell eines Motors im Vergleich zu anderen Motormodellen betrachtet werden. Die Modellbibliothek des gewählten Beispielmodells enthält neben dem des gewählten Fahrzeugprojektes noch drei weitere Motormodelle, die sich in Teilen aus gemeinsamen Blöcken zusammensetzen und darüber hinaus individuelle Anpassungen enthalten. Für alle vier Simulationsmodelle wurden die implementierten Metriken gemessen. Die Abbildungen 7.13 und 7.14 zeigen die Messwerte aller Modelle im Überblick. Da die konkreten Messwerte der Metriken sich numerisch stark unterscheiden, wurden die Messergebnisse auf zwei Diagramme aufgeteilt und anhand des gewählten Referenzprojekte aufsteigend sortiert. Innerhalb der Diagramme sind jeweils vier Messkurven dargestellt. Das Referenzprojekt stellt das für diese Validierung gewählte Motormodell dar. Da es sich dabei um den Teil eines Gesamtfahrzeugmodells handelt, wurde darüber hinaus ein Einzel-HIL-Modell für die Analyse ausgewählt. Ein Einzel-HIL prüft eine einzelne Komponente, wie beispielsweise ein Motorsteuergerät, und benötigt in der Regel einfacher implementierte Modelle, da nicht alle Schnittstellen sondern lediglich die zur betrachteten Komponente umgesetzt sein müssen. Als weitere Vergleichsmodelle wurde das Motormodell eines anderen Fahrzeugprojektes als Vergleichsprojekt gewählt. Außerdem wurde ein Plattformmodell ausgewählt. Plattformen sind dadurch gekennzeichnet, dass sie in der Regel für mehrere Fahrzeugprojekte als Grundlage dienen. Sie stellen dabei ein überspezifiziertes Modell dar, dass für den jeweiligen Anwendungsfall eingeschränkt wird. Es ist zu erkennen, dass die Werte jeder einzelnen Messung vergleichbar sind.

In den Diagrammen wurden Beispiele für die Vergleichbarkeit der Metriken markiert. In

Abbildung 7.13 zeigt Markierung 1 den Messwert „MagicConstantCount". Unter magischen Konstanten werden Blocks verstanden, die keinen Namen besitzen, aber im Modell einen konstanten Eingangswert für die Berechnung implementieren. Diese magischen Konstanten sind in der Regel unerwünscht, da sie das Simulationsmodell unleserlich und für den Nutzer unverständlich machen. Es ist nicht auf den ersten Blick erkennbar, welche Bedeutung der Wert für die Berechnung hat. Der Vergleich zeigt, dass das Referenzprojekt hier beispielsweise schlechter ist, als das Vergleichsprojekt, aber besser als das Plattformmodell. Markierung 2 der Abbildung 7.13 zeigt den Messwert „BusCreatorBlockCount". Dieser Messwert gibt die Anzahl der Bus-Creator-Blocks an. Diese dienen, wie bereits erwähnt, in der Regel der Schnittstellendefinition. Der Messwert lässt daher Rückschlüsse auf die Schnittstellenkomplexität zu. Im Vergleich des Referenzprojektes mit dem Einzel-HIL-Modell zeigt sich der bereits beschriebene Sachverhalt, dass die Schnittstellen für Einzel-HIL-Modelle meist einfacher sind, als die von Gesamtfahrzeugmodellen. Markierung 3 in Abbildung 7.13 zeigt den Messwert für die „GotoBlockCount"-Metrik. Mit Goto-Blöcken werden kannten im Modell eliminiert, die eine Darstellung übersichtlicher erscheinen lassen. Dadurch geht jedoch die Nachverfolgbarkeit von Signalpfaden im Modell verloren. Was für den Modellersteller übersichtlicher ist, führt für einen anderen Nutzer des Modells dazu, dass er dieses nicht mehr nachvollziehen kann. Auch hier zeigt der Vergleich, dass das Referenzprojekt zum Vergleichsprojekt eher schlecht abschneidet.

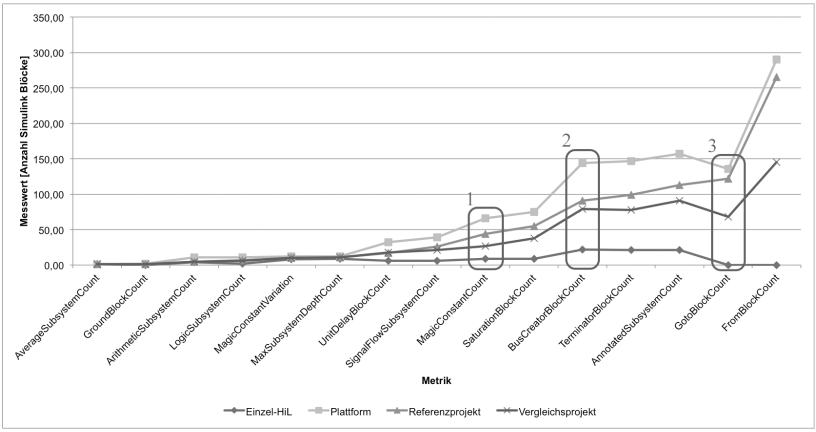


Abbildung 7.13: Qualitätsmessung des Referenzprojektes (kleine Messwerte)

In Abbildung 7.14 zeigt Markierung 4 den Messwert der „LineAnnotationCount"-Metrik. Der konkrete Messwert muss stets in Relation zum in Markierung 6 gezeigten Messwert „LineCount" betrachtet werden. Während „LineCount" die Gesamtanzahl der Signalpfade zählt, zählt „LineAnnotationCount" die Anzahl der Signalpfade mit einer Bezeichnung. In einem idealen Modell sind beide Messwerte gleich, da in diesem Fall jeder Signalpfad dokumentiert ist. Die beiden Messwerte erlauben damit eine relative Aussage zur Dokumentationsqualität. Eine der wichtigsten Metriken stellt der in Abbildung 7.14 mit Markierung

5 hervorgehobene Messwert „TotalBlockCount" dar. Er dient in der Qualitätsbestimmung als Normierungfaktor über einen Großteil der Metriken. Der Messwert zeigt die Anzahl der zum Modell gehörigen Simulink-Blöcke. Betrachtet man beispielsweise den Messwert der magischen Konstanten, so kann ein Modell mit 10000 Blöcken die doppelte Anzahl an magischen Konstanten enthalten, wie ein Modell mit 5000 Blöcken, um die gleiche Qualität zu erreichen.

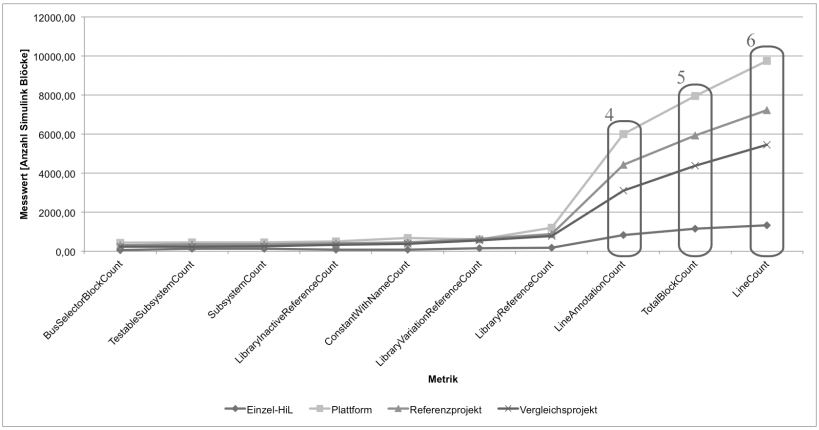


Abbildung 7.14: Qualitätsmessung des Referenzprojektes (große Messwerte)

Die einzelnen Messwerte der Metriken werden anschließend nach dem in Abschnitt 4.4.2 beschrieben Qualitätsmodell nach [Sch12] zu den einzelnen Faktoren und letztendlich zum Qualitätswert aggregiert und in den Modellkontext eingepflegt. Abbildung 7.15 zeigt den finalen Qualitätswert von 18% zum Referenzmodell (vgl. Abschnitt 5.7.2). Dieser relativ niedrige Wert entspricht den Erwartungen, die bereits aus der lokalen Betrachtung einzelner Messwerte im Rahmen dieses Abschnitts hervorgehen. Nach Abschluss der Qualitätsanalyse ist der minimale Modellkontext vollständig erzeugt. Für das gewählte Simulationsmodell konnte dieser vollständig aus bestehenden Datenquell extrahiert werden. Abbildung 7.16 zeigt den Befüllungsgrad des Modellkontextes.

Abbildung 7.15: Demonstratorsuchergebnis nach Qualitätsanalyse

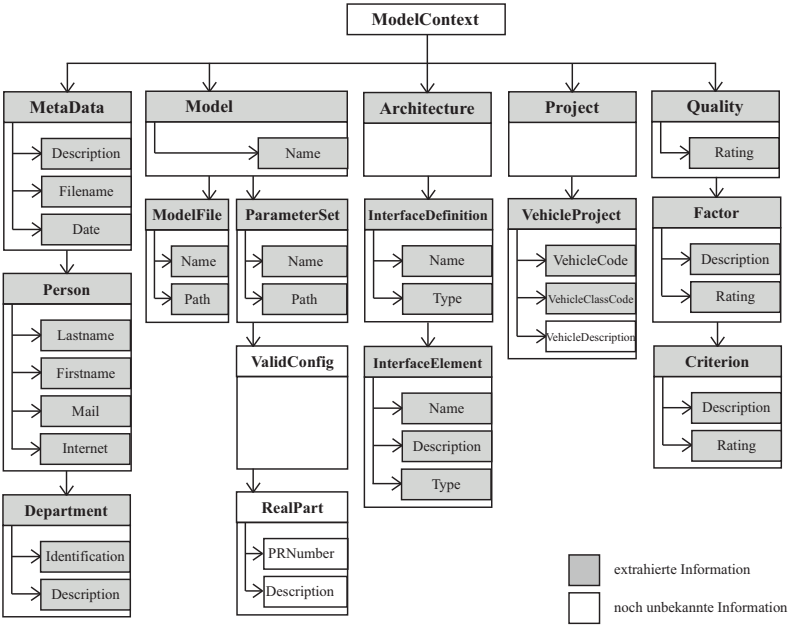


Abbildung 7.16: Modellkontext nach Qualitätsanalyse

Durchführung und Auswertung des Anwendungsszenarios 4 Im vierten Anwendungsfall (vgl. Tabelle 7.4) wird die Integration von Datenquellen außerhalb des Entwicklungsprozesses gezeigt. Im konkreten Anwendungsfall wird das System MBT (vgl. Abschnitt 7.2) angebunden und versucht eine Verknüpfung mit der realen Fahrzeugkomponente herzustellen, die vom Modell repräsentiert wird. Dabei handelt es sich im einen Motor, der in diesem Fahrzeugprojekt zum Einsatz kommen soll. Die Verknüpfung wird durch eine Volltextsuche unter Verwendung einer Mustererkennung auf den im Rahmen der vorherigen Analyseschritte extrahierten Texten durchgeführt. Diese Muster müssen abhängig vom Anwendungsgebiet definiert werden. Für die AUDI AG existieren dabei eine Reihe von Konventionen, die die Bezeichnung von Fahrzeugprojekten, Bauteilen und deren Zusammengehörigkeiten regeln. Im Falle des MBT kann anhand des Fahrzeugprojektes, sowie der Einschränkung auf die Komponente Motor, der Suchraum auf die dort registrierten Motoren eingeschränkt werden. Anschließend können diese Motoren mit geeigneten Suchmustern identifiziert werden. Relevante Begriffen wären dafür beispielsweise „TDI" oder „TFSI" und Hubraumbezeichnungen, wie „1,6L" oder „2,0L". Darüber hinaus existieren für jeden Motor eindeutige Identifikationsnummern. Mit den dadurch gesammelten Informationen lassen sich die oben genannten, zuvor extrahierten Texte des Simulationsmodells zielgerichtet nach Zusammenhängen durchsuchen. Abbildung 7.17 zeigt das Ergebnis für das Beispielmodell, dass als 2,0L Motor identifiziert werden konnte. Damit ergibt sich für den Nutzer der Mehrwert einer Verknüpfung des Simulationsmodells mit dem von ihm repräsentierten, realen Objekt durch Nutzung von Informationsquellen, die ihm zuvor innerhalb seines Entwicklungsprozesses nicht zur Verfügung standen. Abbildung 7.18 stellt den nach Abschluss der MBT-Analyse vollständig befüllten Modellkontext dar. Es ist zu erwähnen, dass die Informationsklassen für die Echtteile (vgl. Abbildung 7.18 unter „RealPart") im Rahmen der Umsetzung des Demonstrators im Modellkontext modelliert wurden, jedoch unter Nutzung der statische API (vgl. Abschnitt 5.5.3) befüllt werden.

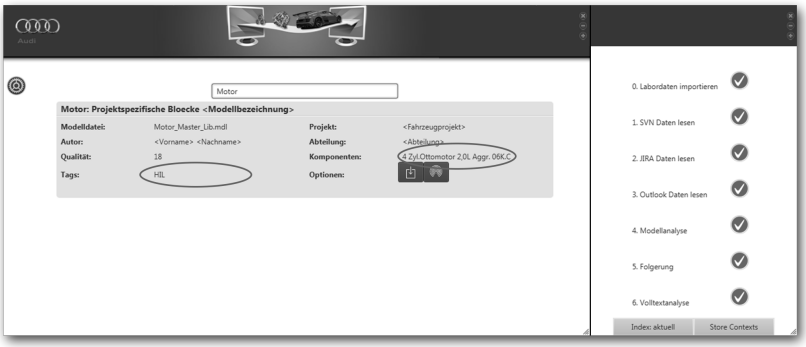


Abbildung 7.17: Demonstratorsuchergebnis nach MBT-Analyse

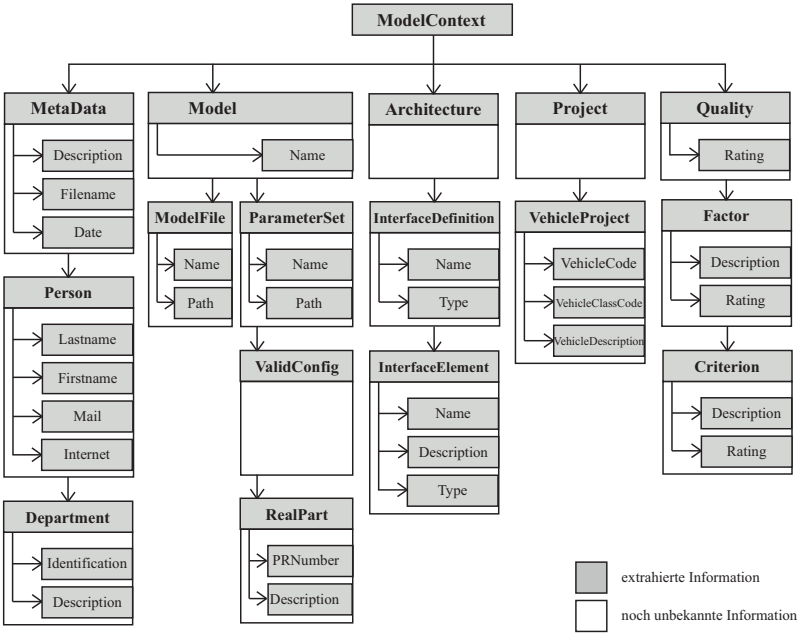


Abbildung 7.18: Modellkontext nach MBT-Analyse

Durchführung und Auswertung des Anwendungsszenarios 5 Im fünften Anwendungsfall (vgl. Tabelle 7.5) wird die Überführung der im Rahmen der Analyse generierten Modellkontexte in einen Suchindex für Elastic Search gezeigt. Dadurch wird die Wiederverwendbarkeit der Datenbasis nachgewiesen. Wie in Abschnitt 4.5.3 erläutert erfolgt die Konfiguration eines Suchindex durch JSON-Konstrukte, eine hierarchische Strukturierung der Informationen. Der Algorithmus 7.1 zeigt das konkrete JSON-Konstrukt für das im Rahmen der Evaluierung betrachtete Motormodell. Das JSON-Konstrukt wird über eine Konfigurationsschnittstelle an den Elastic Search Server übergeben. Anschließend muss der Suchindex neu generiert werden. Dies erfolgt im Demonstrator durch die in Abbildung 7.6 dargestellte Indexsteuerung. Die Funktionsfähigkeit der Suche wurde bereits in den Demonstratorabbildungen 7.7, 7.9, 7.11 sowie 7.17 gezeigt.

Algorithmus 7.1 JSON-Konstrukt zur Indexgenerierung

```
{
"RealPart" : "4Zyl.Ottomotor 2,0L Aggr.",
"ProjectClass" : "<Klasse>",
"ProjectAbbreviation" : "<Abkürzung>",
"ModelName" : "Projektspezifische Bloecke <Modellbezeichnung>",
"Filename" : "Motor_Master_Lib.mdl",
"URI" : "http://www.audi.de/xil-context/c4a3fe2b-6e8d2f895091",
"PRN" : "<PR-Nummer>",
"Date" : "Thu Aug 20 15:55:58 CEST 2015",
"Author_Firstname" : "<Vorname>",
"Author_Name" : "<Nachname>",
"Author_Mail" : "<E-Mail>",
"QualityRating" : "18",
"ProjectDescription" : "<Fahrzeugprojekt>"

[...]

}
```

Laufzeit Bewertung für den Anwendungsfall Nachdem die Anwendbarkeit der Konzepte der vorliegenden Arbeit für ein konkretes Simulationsmodell mit der Durchführung der Anwendungsszenarien 1 bis 5 demonstriert wurde, folgt an dieser Stelle eine Bewertung der Laufzeit des Demonstrators für den gewählten Anwendungsfall. Dafür wurden die Analyseschritte für unterschiedliche Anzahlen von Simulationsmodellen in einer konstanten Hardwareumgebung durchgeführt und jeweils die Ausführungszeit aufgenommen. Die Umgebung der Analyse stellte sich dabei wie folgt dar:

- Ausführungsrechner: HP Elitebook 2570p
- Prozessor: Intel Core i5-3320M 2,60Ghz
- Hauptspeicher: 16 GB
- Betriebssystem: Windows 7 Enterprise 64-Bit

- Speicher für lokale Daten: Transcend USB 3.0 Solid State Disk (SSD) angebunden über einen USB 3.0 Port
- Ausführungsumgebung: JDK 8
- Netzwerkanbindung: Wireless Lan 802.11g
- Anbindung an SVN: SVNKit 1.7.8
- Anbindung an JIRA: JIRA 6 Rest API

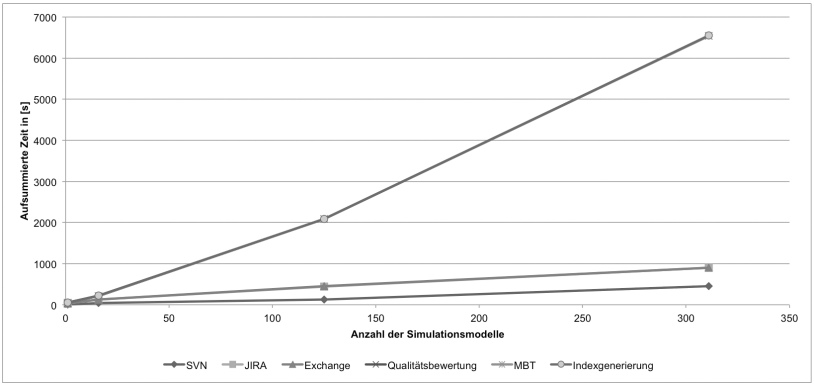


Abbildung 7.19: Zeitmessung für unterschiedliche Datenbasisgrößen

In Abbildung 7.19 ist das Messergebnis als zeitliche Aufsummierung der einzelnen Analyseschritte für die Messgrößen 1, 16, 132 und 305 Simualtionsmodelle dargestellt. Zur Übersicht wurde der letzte Analyseschritt, die Indexgenerierung für Elastic Search rot markiert. Dieser repräsentiert die Gesamtlaufzeit für die jeweilige Messung. Es zeigt sich eine lineare Laufzeitkomplexität $O(n)$ in Abhängigkeit von der Anzahl n der analysierten Simulationsmodelle. Dieses lineare Verhalten zeigt sich in allen Analyseschritten, auch wenn diese sich hinsichtlich ihrer individuellen Laufzeit stark unterscheiden. Auffällig sind die Laufzeiten der SVN-, JIRA- und Modellanalye. Bei SVN und JIRA lässt sich diese Laufzeit auf die zugrundeliegende Netzwerkkommunikation zurückführen, da jede Anfrage an die jeweiligen Server zunächst autorisiert werden muss. Im Falle von JIRA kommt hier eine zustandslose Restful API zum Einsatz, bei der jeder einzelne Zugriff einen individuellen Verbindungsaufbau benötigt. Für die Modellanalyse lässt sich die Laufzeit auf den relativ langsamen Einzeldateizugriff im Dateisystem zurückführen. Durch den Wechsel der genutzten Hardware von einer HDD Festplatte auf eine SSD Festplatte konnte im Rahmen der Arbeit bereits ein Geschwindigkeitsgewinn im Bereich von zwei Zehnerpotenzen erreicht werden. Dennoch bleibt der Zugriff auf die Modelldateien auch bei dieser Hardwarekonfiguration der Engpass der Analyse. Darüber hinaus zeigte die Analyse, dass auch die Anzahl der Blöcke in den analysierten Simulinkmodellen einen Einfluss auf die Laufzeit hat. Die Analysezeit für ein einzelnes Modell schwankt dabei in Abhängigkeit von der Blockanzahl, während, wie die Messergebnisse zeigen, im Mittel die Laufzeit eher von der Anzahl der Modelle abhängig ist.

Mit der linearen Skalierung des entwickelten Ansatzes lässt sich dieser auf Modellanzahlen von mehreren hundert Simulationsmodellen, wie sie heute im Entwicklungsprozess vorkommen, ohne Einschränkungen anwenden. Optimierungspotenzial für die Beschleunigung der Analyse liegt insbesondere in der Anbindung von Prozessdatenablagen, wie SVN und JIRA, sowie in der Anbindung lokaler Datenablagen mit möglich geringer Zugriffzeit auf die Simulationsmodelle.

Laufzeitabschätzung zur Nutzung der resultierenden Wissensbasis Neben der Generierung stellt auch die Laufzeit des Informationszugriffs auf die resultierende, graphbasiert Wissensbasis eine wichtige Kenngröße dar. Ausgehend von dem in Abschnitt 5.4.3 beschriebenen Verfahren der Generierung wird im Folgenden eine Laufzeitanalyse für eine wachsende Wissensbasis durchgeführt. Abbildung 7.20 zeigt zur Verdeutlichung der Zusammenhänge beispielhaft zwei Ausschnitte von Modellkontexten. Die Informationsklassen Modellkontext, Projekt und Verwendung sind mit ihren jeweiligen Instanzen direkt verknüpft. Darüber hinaus stellt die Instanz der Modellkontext-Informationsklasse nach Abschnitt 5.4.3 stets den Wurzelknoten eines konkreten Modellkontextes dar.

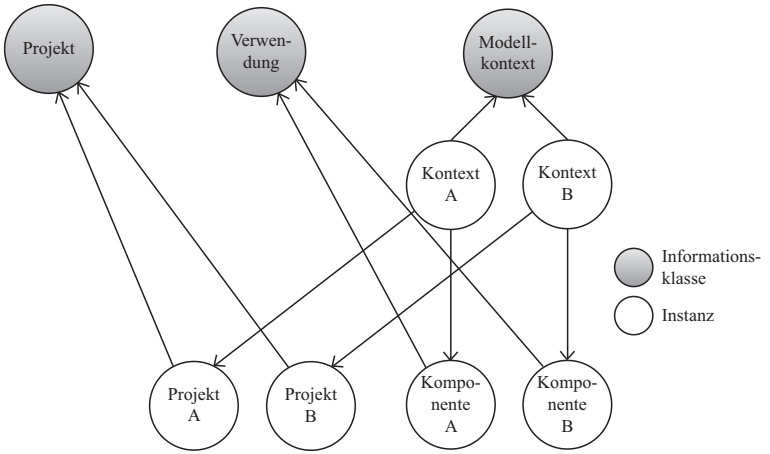


Abbildung 7.20: Beispiel für Zugriff auf die graphbasierte Repräsentation

Der Zugriff und die Auswertung von Graphbeziehungen hängt stets von der technischen Umsetzung ab. Im konkreten Anwendungsfall wurde Neo4j als Graphdatenbank mit der Abfragsprache Cypher gewählt. Da die interne Abarbeitung von Cypher-Abfragen nicht im Fokus der vorliegenden Arbeit steht, wird hier insbesondere die Worst-Case-Laufzeit für den Zugriff auf ein Datum betrachtet.

Abfragen im resultierenden Graphen suchen stets Beziehungen, die von einer bekannten Instanz einer Informationklasse, in der Regel einer Modellkontext-Instanz, ausgehen. Sei k die Anzahl aller Informationsklassen im Informationsmodell und m die Anzahl der Modellkontext-Instanzen im Graphen. Wie im Beispielgraph der Abbildung 7.20 deutlich wird, können

diese Modellkontext-Instanzen je nach Modellierung direkt mit allen Instanzen der sie beschreibenden Informationsklassen oder über hierarchische Beziehungen der jeweiligen Informationklassen mit maximal allen anderen (k-1)-Informationsklassen verknüpft sein. Somit müssen für eine Suche über alle m Modellkontext-Instanzen die verknüpften (k-1) Informationsklassen durchlaufen werden. Es ergibt sich eine Laufzeit von $O((k-1)*m)$, wobei k durch die modellbasierte Repräsentation des Informationsmodell vorgegeben und bezüglich der Laufzeit für einen definierten Anwendungsfall als konstant zu betrachten ist (vgl. Abschnitt 5.4.3). Dadurch ergibt sich eine Laufzeitkomplexität von $O(m)$ für den Informationszugriff.

7.5 Zusammenfassung

Im Kapitel 7 wurden die im Rahmen der vorliegenden Arbeit entwickelten Konzepte am konkreten Anwendungsfall, einem Entwicklungsprozess der automobilen Funktionsabsicherung und einem konkreten, aktuellen Fahrzeugprojekt, validiert. Dafür wurde zunächst in Abschnitt 7.1 der Entwicklungsprozess beschrieben. In Abschnitt 7.2 wurden darauf aufbauend die eingesetzten Softwaresyteme hinsichtlich ihrer Rolle im Prozess sowie ihres Informationsgehaltes analysiert.

In Abschnitt 7.3 wurden die zur Validierung genutzten Anwendungsszenarien 1-5 vorgestellt, die sich an der Verarbeitungsreihenfolge des Konzeptes für die prozessintegrierte Dokumentation von Simulationsmodellen der vorliegenden Arbeit orientieren. Diese Anwendungsszenarien wurden in Abschnitt 7.4 in ihrer Durchführung beschrieben und hinsichtlich ihrer Ergebnisse bewertet.

Bei der Auswertung der Anwendungsfälle konnte die Gültigkeit der Konzepte für ein ausgewähltes Simulationsmodell vollständig nachvollzogen und ein lückenloser Modellkontext automatisiert erzeugt werden. Die Ausführung der Analyse wurde für alle Simulationsmodelle des gewählten Fahrzeugprojektes. Anhand dieser Stichprobe von 132 Simulationsmodellen zeigte sich, dass die Vollständigkeit des Modellkontextes stark von der Einhaltung der Prozessvorgaben zur Nutzung der Softwaresysteme abhängig ist. Die betrachteten Systeme SVN und Microsoft Exchange lieferten eine vollständige Datenbasis für jedes der 132 Simulationsmodelle und alle nach Abschnitt 7.2 erwarteten Daten. Insbesondere das System JIRA lieferte lediglich unvollständige Daten, durch fehlende Einträge von Attributen, wie beispielsweise der Zugehörigkeit zu einem Fahrzeugprojekt.

Für die Laufzeitkomplexität konnte eine lineare Entwicklung in Abhängigkeit von der Anzahl der analysierten Simulationsmodelle gezeigt werden. Damit lässt sich der Ansatz auf heute gängige Entwicklungsprozesse ohne Einschränkungen anwenden.

Somit lässt sich Zusammenfassen, das die Konzepte der vorliegenden Arbeit das Ziel einer automatisierten, prozessintegrierten Dokumentation erreichen. Mit den dargestellten Methoden der Informationsextraktion lassen sich dokumentationsrelevante Inhalte aus nahezu allen, im Prozess verfügbaren Datenquellen extrahieren. Die Qualität der Dokumentationsergebnisse hängt dabei von der Einhaltung bzw. der Verlässlichkeit von Prozessvorgaben ab.

8 Zusammenfassung und Ausblick

Die Schaffung, Wahrung und Nutzung von Wissen stellt eine wichtige Säule für die Konkurrenzfähigkeit von Unternehmen am Markt dar. Noch vor wenigen Jahren waren die individuellen Leistungen eines einzelnen Mitarbeiters oder einer kleinen Gruppe für eine Entwicklungstätigkeit auf höchstem Niveau ausreichend. Heute erhöht sich die Komplexität von Entwicklungsprozessen, wirtschaftlichen Herausforderungen und technologischen Möglichkeiten so schnell, dass Unternehmen durch die Anpassung ihrer Entwicklungsstrukturen aktiv gegensteuern müssen, um diese in Zukunft beherrschen zu können. In diesem Zusammenhang werden neue Lösungen benötigt, die eine optimale Zusammenarbeit früherer, individuenzentrierter Organisationsstrukturen unterstützen.

Vor diesem Hintergrund steht die moderne Funktionsentwicklung der Automobilindustrie vor der Herausforderung immer neue, hochgradig vernetzte Fahrzeugfunktionen zu entwickeln und in immer kürzerer Zeit und immer geringeren Kosten in den Markt zu bringen. Um dies in den Entwicklungsprozessen der Automobilhersteller abbilden zu können, kommt der virtuellen Entwicklung und Absicherung dieser Fahrzeugfunktionen in wachsendem Maße Bedeutung zu. Ehemals von kleinen Gruppen getriebene Entwicklungsprozesse und die Absicherung von Funktionen im realen Fahrversuch reichen für die Komplexität heutiger Funktionen nicht mehr aus.

Diese Entwicklungen führen zu Anpassungen der Funktionsentwicklungsprozesse weg von einer organisatorisch getrennten hin zu einer vernetzten Entwicklung, in der Wissenstransfer und Kommunikation über den Erfolg oder Misserfolg entscheiden. Für die automobile Funktionsentwicklung hat sich hier ein durchgängig modellbasierter Entwicklungsprozess etabliert. Dieser beginnt mit der modellbasierten Softwareentwicklung und reicht über virtuelle Absicherungsmethoden von Model- über Software- bis Hardware-in-the-Loop-Simulation. Die beschriebene Transformation der Entwicklungsprozesse hin zur vernetzten Entwicklung stellt aktuell hohe Anforderungen an die Funktionsentwicklungsabteilungen der Automobilindustrie. Es fehlt an Methoden und Werkzeugen zur verteilten Entwicklung und zum Wissenstransfer.

An diesem Punkt greift die vorliegende Arbeit die Herausforderungen der vernetzten Entwicklung in modellbasierten Entwicklungsprozessen auf. Grundlage der Untersuchungen stellt dabei der Stand der Technik in den Bereichen modellbasierte Entwicklung (vgl. Abschnitt 4.3) sowie Informations- und Wissensmanagement (vgl. Abschnitt 4.1) dar. In einem ersten Arbeitsschwerpunkt (vgl. Abschnitt 1.2) wird eine minimale Informationsbasis für modellbasierte Entwicklungsprozesse hergeleitet und in Form des sogenannte Modellkontexts definiert (vgl. Abschnitt 5.1). Darauf aufbauend und auf Basis der These, dass alle benötigten Informationen bereits heute in den Entwicklungsprozessen verfügbar sind, wird die Architektur eines Informationssystems zur prozessintegrierten Dokumentation der Modellkontexte entwickelt (vgl. Abschnitte 5.3, 5.4, 5.5). Diese Architektur bildet den Lösungsansatz für den zweiten Arbeitsschwerpunkt der vorliegenden Arbeit, die automatische Erfassung von Modellkontexten (vgl. Abschnitt 1.2). Den dritten Arbeitsschwerpunkt der Ar-

beit stellt die Untersuchung von Methoden der zielgerichteten Nutzung der erzeugten Modellkontexte zur Unterstützung der kollaborativen Arbeit in vernetzten, modellbasierten Entwicklungsprozessen (vgl. Abschnitte 5.6, 5.7) dar.

Die Ergebnisse der Arbeit fasst das modulare Informationsmanagementsystem zu einem Wissensmanagementframework für modellbasierte Entwicklungsprozesse zusammen. Wo aktuelle Wissensmanagementlösungen für die Erstellung einer Wissensbasis in der Regel besonders geschultes Personal zur Erstellung und Wartung von Ontologien benötigen, bildet den Ausgangspunkt des modularen Informationsmanagementsystems eine modellbasierte Wissensrepräsentation. Diese wird mithilfe einer an UML angelehnten Notation erstellt und kann damit durch die Nutzer in einer ihnen vertrauten Sprache im modellbasierten Entwicklungsprozess selbständig erstellt, gepflegt und angepasst werden.

Die prozessintegrierte Dokumentation der für den Modellkontext relevanten Informationen wird durch eine Adapterschicht realisiert, deren Adapterendpunkte dynamisch durch Codegenerierung aus der modellbasierten Wissensrepräsentation erzeugt werden (vgl. Abschnitt 5.5). Anschließend kann der Modellkontext über diese dynamische API mithilfe quellsystemspezifischer Adapter befüllt werden. Durch dieses im Rahmen der Arbeit entwickelte Konzept lässt sich das Wissensmanagementframework modular erweitern. Ist eine Erweiterung der modellbasierten Wissensrepräsentation notwendig, so kann diese direkt im System durch die Anwender realisiert werden. Um die neuen Informationen in den dann erweiterten Modellkontext einzupflegen, muss das System um einen zur Erweiterung passenden Adapter ergänzt werden.

Ein besonderer Fokus des Wissensmanagementframeworks liegt auf der Erweiterbarkeit der Wissensbasis auch über den angebundenen modellbasierten Entwicklungsprozess hinaus. Aus diesem Grund wird die hinsichtlich der dynamischen Erweiterung recht unflexible modellbasierte Repräsentation nach der Datenakquise aus dem Entwicklungsprozess in eine Graphrepräsentation überführt (vgl. Abschnitt 5.8). Dadurch ergeben sich einerseits Möglichkeiten der Anbindung indirekter Informationsquellen, die über das reine Prozesswissen hinaus gehen (vgl. Abschnitt 5.4.3). Andererseits erlaubt die Graphrepräsentation eine hohe Flexibilität bei der Anbindung von nutzenden Systemen an die Wissensbasis. Als Nutzungsszenarien werden im Rahmen der vorliegenden Arbeit zwei Anwendungsfälle untersucht. Einerseits wird die Überführung der Wissensbasis in den Suchindex einer Suchmaschine durchgeführt und bewertet (vgl. Abschnitt 5.6). Andererseits werden Methoden der kriterienbasierten Modellbewertung erarbeitet und drei Referenzmodelle der Bewertung erarbeitet (vgl. Abschnitt 5.7).

Der Nachweis der Anwendbarkeit des entwickelten Wissensmanagementframeworks wird anschließend durch die prototypische Implementierung und Anwendung auf einen modellbasierten Entwicklungsprozess der automobilen Funktionsentwicklung erbracht (vgl. Kapitel 7). Damit konnten die drei Arbeitsschwerpunkte und die damit verbundenen Ziele erreicht werden. Darüber hinaus konnte eine lineare Laufzeitkomplexität in Abhängigkeit von der Anzahl der Simulationsmodelle gezeigt und dadurch eine Übertragbarkeit auf heute gängige Entwicklungsprozesse nachgewiesen werden.

Im nächsten Schritt steht die Implementierung des entwickelten modularen Informationsmanagementsystems im Rahmen des Projektes XIL-Datenmanagement an. Dadurch werden die Ergebnisse in die produktiven Entwicklungsprozesse der automobilen Funktionsabsicherung der AUDI AG integriert und die Integration in weitere modellbasierte Entwicklungsprozesse durch neue Adapter gefördert.

Der im Rahmen der vorliegenden Arbeit entwickelte, neue Ansatz eines durchgängigen, modularen Informationsmanagementsystem für modellbasierte Entwicklungsprozesse erfüllt die in Abschnitt 1.2 gesetzten Ziele der Arbeit. Mit der modellbasierten Repräsentation eines Informationsmodells wird es dadurch möglich, die Nutzer des Systems in die Konfiguration einzubeziehen und den Dokumentationsprozess für sie transparent zu gestalten. Damit löst der Ansatz unter anderem die Herausforderung früherer Ansätze (vgl. Abschnitte 4.1.1, 4.5.2), bei denen zunächst ein Wissenstransfer vom Nutzer zu gesondert ausgebildeten Experten notwendig war, um die Informationsmodelle in Form von Ontologien zu erstellen.

Darüber hinaus erlaubt der Ansatz durch die direkte Integration der Datenakquise in den Entwicklungsprozess, die Anwendung auf bestehende Prozesse, ohne dafür verändernd in den Prozess oder den Dokumentationsaufwand eingreifen zu müssen. Damit ist er insbesondere für die Nutzung in bestehenden Prozessen geeignet. Durch eine zielgerichtete Anreicherung mit Informationen, die heute nicht direkt im Entwicklungsprozess genutzt werden, lassen sich durch die weiteren Informationsverarbeitungsschritte des Ansatzes (vgl. Abschnitt 5.6) neue Anwendungen, wie die zielgerichtete Wiederverwendung der Entwicklungsartefakte durch Such- und Empfehlungssysteme, realisieren. Dies unterstützt die hochgradig vernetzten Entwicklungsprozesse und sichert einen zielgerichteten Informationsfluss. Aufgrund des modularen Ansatzes lässt sich dieser nicht nur in der Automobilindustrie anwenden, sondern auch auf modellbasierten Entwicklungsprozesse anderer Industriezweige übertragen.

Bezüglich der wissenschaftliche Weiterentwicklung des Wissensmanagementframeworks konnten weitere interessante Forschungsfelder identifiziert werden, die jedoch über den Rahmen der vorliegenden Arbeit hinausgehen. Zum einen kann die Qualität der Ergebnisse der Datenakquise, insbesondere in unstrukturierten Daten, durch Natural Language Processing (vgl. Abschnitt 5.5.2) mithilfe eines anwendungsspezifisches Sprachmodells deutlich verbessert werden. Der Fokus in diesem Forschungsfeld liegt heute besonders im medizinischen Umfeld und bietet hohes Potenzial für weitere Anwendungsgebiete. Darüber hinaus stellt die Nutzung graphbasierter Indizes einen Ansatz zur Verbesserung der Suchergebnisse und damit der Wiederverwendung der Entwicklungsartefakte dar. Ein möglicher Untersuchungsschwerpunkt wäre die These, dass graphbasierte Indizes die semantischen Beziehungen innerhalb des Wissensgraphen besser als beispielsweise hierarchisch strukturierte Indizes berücksichtigen und die Ergebnisgüte dadurch erhöht wird.

Abschließend stellt die tiefere Integration des Wissensmanagementframeworks in die vernetzten modellbasierten Entwicklungsprozesse ein weiteres Forschungsfeld dar. Das im Rahmen der prozessintegrierten Dokumentation generierte Wissen erlaubt es Aussagen über die Entwicklungsartefakte von der Ebene der Schnittstellen bis zur Ebene der projektabhängigen Gültigkeit der Kombination zweier oder mehr Entwicklungsartefakte zu treffen. Ein Anwendungsfall hierfür wäre beispielsweise die Nutzung dieses Wissens für die Unterstützung eines automatisierten Modellaufbaus, in dem auf Basis definierter Anforderungen eine gültige, simulationsfähige Menge an Entwicklungsartefakten gesucht und einer Simulationsumgebung übergeben wird.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[ABH+98] ABECKER, Andreas; BERNARDI, Ansgar; HINKELMANN, Knut; KÜHN, Otto; SINTEK, Michael: *Toward a technology for organizational memories*. In: *IEEE Intelligent Systems*, Bd. 13: S. 40–48, 1998.

[AG12] AG, IntraFind: *iFinder Enterprise Search*, 2012, http://www.intrafind.de/de/produkte/ifinder.
URL `http://www.intrafind.de/de/produkte/ifinder`

[Ang14] ANGERMANN, Anne: *Matlab, Simulink, Stateflow: Grundlagen, Toolboxen, Beispiele*, Oldenbourg, München, 8. aufl Aufl., 2014, ISBN 978-3-4867-7845-8.

[Atl14] ATLASSIAN: *Atlassian JIRA*, 2014, https://www.atlassian.com/de/software/jira.
URL `https://www.atlassian.com/de/software/jira`

[BBL76] BOEHM, B. W.; BROWN, J. R.; LIPOW, M.: *Quantitative Evaluation of Software Quality*. In: *Proceedings of the 2Nd International Conference on Software Engineering*, ICSE '76, S. 592–605, IEEE Computer Society Press, Los Alamitos, CA, USA, 1976.
URL `http://dl.acm.org/citation.cfm?id=800253.807736`

[BE08] BALZERT, Helmut; EBERT, Christof: *Lehrbuch der Sofwaretechnik*, Lehrbücher der Informatik, Spektrum Akademischer Verlag, Heidelberg, 2. aufl Aufl., 2001-2008, ISBN 978-3-8274-1161-7.

[Bec04] BECKETT, Dave: *RDF/XML Syntax Specification (Revised)*, W3C recommendation, W3C, Febr. 2004, http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/.

[BK14] BARMPIS, Konstantinos; KOLOVOS, Dimitrios: *Evaluation of Contemporary Graph Databases for Efficient Persistence of Large-Scale Models*. In: *Journal of Object Technology*, Bd. 13(3): S. 3:1–26, 2014.

[BL89] BERNERS-LEE, Tim: *Information Management: A Proposal*. In: *CERN Proposal*, 1989, http://www.w3.org/History/1989/proposal.html.
URL `http://www.w3.org/History/1989/proposal.html`

[BLHL01] BERNERS-LEE, Tim; HENDLER, James; LASSILA, Ora: *The Semantic Web*. In: *Scientific American*, Bd. 284(5): S. 34–43, Mai 2001.
URL `http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21`

[BO12] BLOCHWITZ, T.; OTTER, M.: *Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models*, . In: *Proceedings of the 9th International Modelica Conference*, S. 173–184, 2012.

[Bre09] BRECHTER, Daniel: *MeMo: Eine offene Integrationsplattform zur modellbasierten Entwicklung von Fahrzeugsystemen*, Bd. 3 von *AutoUni-Schriftenreihe*, Logos-Verl., Berlin, 2009, ISBN 978-3832521615.

[BS14] BRÜCKNER, Constantin; SWYNNERTON, Bettina: *Busbasiertes Architekturkonzept für Hardware-in-the-Loop-Prüfstände*. In: *ATZ Elektronik*, Bd. 9(3): S. 52–56, 2014.

[BSMG12] BAUER, Stefan; STÜBER, Nico; MELLER, Frank; GRUBER, Karl: *Entwicklung einer Co-Simulationsprozesskette und deren Integration in den Prozess*. In: *SIMVEC - Berechnung, Simulation und Erprobung im Fahrzeugbau 2012, 20./21. November 2012, Baden-Baden*, VDI-Berichte 2169, S. 609–616, VDI Verlag GmbH, Düsseldorf, 2012, ISBN 978-3-18-092169-3.

[Bul02] BULLINGER, H.-J: *Technologiemanagement: Forschen und Arbeiten in einer vernetzten Welt*, Springer, Berlin and New York, 2002, ISBN 978-3-540-41891-7.

[CDF+05] CONRAD, M.; DÖRR, H.; FEY, I.; POHLHEIM, H.; STÜRMER, I.: *Guidelines und Reviews in der Modell-basierten Entwicklung von Steuergeräte-Software*. In: *2. Tagung Simulation und Test in der Funktions- und Softwareentwicklung für die Automobilelektronik*, Expert-Verlag, Berlin, 2005.

[CDSS02] CONRAD, Mirko; DÖRR, Heiko; STÜRMER, Ingo; SCHÜRR, Andy: *Graph Transformations for Model-based Testing*. In: *GI-LECTURE NOTES IN INFORMATICS, P-12*, S. 39–50, 2002.

[CFDY99] CONRAD, M; FEY, I; DÖRR, H; YAP: *Modelbased Generation and Structured Representation of Test Scenarios*. In: *Proceedings of the Workshop on Software-Embedded Systems Testing*, Maryland, USA, 1999.

[CFS05] CONRAD, Mirko; FEY, Ines; SADEGHIPOUR, Sadegh: *Systematic Model-Based Testing of Embedded Automotive Software*. In: *Electron. Notes Theor. Comput. Sci.*, Bd. 111: S. 13–26, Jan. 2005, ISSN 1571-0661.
URL `http://dx.doi.org/10.1016/j.entcs.2004.12.005`

[CM78] CAVANO, Joseph P.; MCCALL, James A.: *A Framework for the Measurement of Software Quality*. In: *Proceedings of the Software Quality Assurance Workshop on Functional and Performance Issues*, S. 133–139, ACM, New York, NY, USA, 1978.
URL `http://doi.acm.org/10.1145/800283.811113`

[Con14] CONSORTIUM, World Wide Web: *World Wide Web Consortium (W3C)*, 2014, http://www.w3.org.
URL `http://www.w3.org`

[Dar14] DARMSTADT, TU: *Darmstadt Knowledge Processing Repository (DKPro)*, 2014, https://www.ukp.tu-darmstadt.de/research/current-projects/dkpro/.
URL `https://www.ukp.tu-darmstadt.de/research/current-projects/dkpro/`

[Dei14] DEISSENBÖCK, Florian: *Conqat - Simulink Library*, 2014, https://www.cqse.eu/en/products/simulink-library-for-java/overview/.
URL `https://www.cqse.eu/en/products/simulink-library-for-java/overview/`

[Den12] DENGEL, Andreas: *Semantische Technologien: Grundlagen - Konzepte - Anwendungen*, SpringerLink : Bücher, Springer Berlin Heidelberg, Heidelberg, 2012, ISBN 978-3-8274-2664-2.

[dG14a] dSPACE GmbH: *dSPACE GmbH*, 2014, http://www.dspace.com/. URL http://www.dspace.com/

[dG14b] dSPACE GmbH: *Variant-Based Workflow Management*, 2014. URL http://www.dspace.com/shared/data/pdf/2014/Variant-Based_Workflow_Management.pdf

[DHM12] Deicke, Markus; Hardt, Wolfram; Martinus, Marcus: *VAP 2.0 - Die nächste Generation der virtuellen Absicherungsplattform*, . In: *4. AutoTest - Fachkonferenz zum Thema Test von Hard- und Software in der Automobilentwicklung*, S. 133–140, Forschungsinstitut für Kraftfahrwesen und Fahrzeugmotoren, Stuttgart, 2012.

[Die06] Diestel, Reinhard: *Graphentheorie*, Springer-Lehrbuch, Springer, Berlin [u.a.], 3., neu bearb. und erw. aufl Aufl., 2006, ISBN 3-540-21391-0.

[EB07] Eisemann, Ulrich; Beine, Michael: *Werkzeugeinsatz in der Entwicklung von Automotive-Software*, . In: *Automobil-Elektronik*, Bd. 1-2007, S. 48–49, 2007.

[EI15] EMF-IncQuery: *EMF-IncQuery*, 2015, https://www.eclipse.org/incquery/. URL https://www.eclipse.org/incquery/

[Eig08] Eigner, M.: *Product lifecycle management*, Springer, London, 2008, ISBN 978-3-540-44373-5.

[EMH14] Englisch, Norbert; Masrur, Alejandro; Hardt, Wolfram: *Testmethode für den applikationsspezifischen Test von AUTOSAR Steuergeräten*, . In: *5. AutoTest - Fachkonferenz zum Thema Test von Hard- und Software in der Automobilentwicklung*, Forschungsinstitut für Kraftfahrwesen und Fahrzeugmotoren, Stuttgart, 2014.

[Fac14] Facility, MetaObject: *MetaObject Facility (MOF)*, 2014, http://www.omg.org/mof/. URL http://www.omg.org/mof/

[FINte] FIND: *Forschungsprojekt FIND - Flexible Daten- und Informationsvernetzung*, 2012 - heute, http://www.v2c2.at/research/information-process-management/modular-information-management/. URL http://www.v2c2.at/research/information-process-management/modular-information-management/

[FS07] Fey, Ines; Stürmer, Ingo: *Quality Assurance Methods for Model-based Development: A Survey and Assessment*, . In: *SAE 2007 Transactions Journal of Passenger Cars: Mechanical Systems*, Bd. 116-6, S. 374–380, 2007.

[Gam95] Gamma, Erich: *Design patterns: Elements of reusable object-oriented software*, Addison-Wesley professional computing series, Addison-Wesley, Reading and Mass, 1995, ISBN 978-0201633610.

[Goo14] Google: *Google Search Appliance*, 2014, https://www.google.com/work/search/products/gsa.html. URL https://www.google.com/work/search/products/gsa.html

[GT12] Gruber, Thomas; Thiel, Sebastian: *Herausforderung XIL-Datenmanagement*. In: *NAFEMS Deutschsprachige Konferenz '12*, NAFEMS GmbH, Bamberg, 8.-9. Mai 2012, ISBN 978-1-87437-667-5.

[GTH13] GRUBER, Thomas; THIEL, Sebastian; HARDT, Wolfram: *Managing Knowledge Across Disciplines and Departments for Automotive X-in-the-Loop-Simulation*. In: *Proceedings of the 13th International Conference on Knowledge Management and Knowledge Technologies*, i-Know '13, S. 33:1–33:5, ACM, New York, NY, USA, 2013, ISBN 978-1-4503-2300-0.
URL `http://doi.acm.org/10.1145/2494188.2499148`

[Gur13] GUREVYCH, Iryna: *Language processing and knowledge in the web: 25th international conference, GSCL 2013, Darmstadt, Germany, September 25 - 27, 2013 ; proceedings*, Bd. 8105 von *Lecture notes in computer science*, Springer, Berlin [u.a.], 2013, ISBN 978-3-642-40721-5.

[Hak13] HAKEN, Karl-Ludwig: *Grundlagen der Kraftfahrzeugtechnik*, Fahrzeugtechnik, Hanser, München, 3. auflage Aufl., 2013, ISBN 978-3-4464-3527-8.

[Har01] HARTMANN, Nico: *Automation des Tests eingebetteter Systeme am Beispiel der Kraftfahrzeugelektronik*, Dissertation, Universität Fredericiana Karlsruhe, Karlsruhe, 2001.

[Har02] HARDT, W. et. al: *IPCHL - A Description Language for Semantic IP Characterization*, . In: *Best of FDL'02, System Specification & Design Languages*, 2002.

[Har10] HARDT, Wolfram: *Vorlesungsskript: Hardware Software Codesign*, Technische Universität Chemnitz, 2010.

[Hed12] HEDTSTÜCK, Ulrich: *Einführung in die theoretische Informatik: Formale Sprachen und Automatentheorie*, Oldenbourg, München, 5., überarb. aufl Aufl., 2012, ISBN 978-3486714043.

[HKRS08] HITZLER, Pascal; KRÖTZSCH, Markus; RUDOLPH, Sebastian; SURE, York: *Semantic Web: Grundlagen*, EXamen.press, Springer-Verlag Berlin Heidelberg, Berlin and Heidelberg, 2008, ISBN 978-3-540-33994-6.

[HM10] HAWKING, Stephen W.; MLODINOW, Leonard: *Der große Entwurf: Eine neue Erklärung des Universums*, Rowohlt, Reinbek bei Hamburg, 1. aufl Aufl., 2010, ISBN 978-3-49802-991-3.

[Hun14] HUNGER, Michael: *Neo4j 2.0: Eine Graphdatenbank für alle*, Schnell + kompakt, Entwickler.press, Frankfurt and M, 2014, ISBN 978-3-86802-128-8.

[IBM14] IBM: *IBM Rational Doors*, 2014.
URL `http://www-03.ibm.com/software/products/de/ratidoor`

[Ind14] INDEN, Michael: *Java 8 - Die Neuerungen: Lambdas, Streams, Date And Time API und JavaFX 8 im Überblick*, dpunkt, Heidelberg and Neckar, 1. aufl Aufl., 2014, ISBN 978-3864902017.

[Int02] INTERNATIONALE ORGANISATION FÜR NORMUNG: *ISO-646: Information technology - ISO 7-bit coded character set for information interchange*, 2002.

[Int09] INTERNATIONALE ORGANISATION FÜR NORMUNG: *ISO-9000:2005: Quality management systems - Fundamentals and vocabulary*, 2009.

[Int11a] INTERNATIONALE ORGANISATION FÜR NORMUNG: *ISO-25964: Thesauri and interoperability with other vocabularies*, 2011.
URL `http://www.niso.org/schemas/iso25964/`

[Int11b] INTERNATIONALE ORGANISATION FÜR NORMUNG: *ISO-26262: Road vehicles - Functional safety*, 2011.

[Int12] INTERNATIONALE ORGANISATION FÜR NORMUNG: *ISO-19505: Object Management Group Unified Modeling Language (OMG UML)*, 2012.

[Int14] INTERNATIONALE ORGANISATION FÜR NORMUNG: *ISO-25000:2014: Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE)*, 2014.

[KB13] KIFFE, Gerhard; BOCK, Thomas: *Standardisierte Entwicklungsumgebung für die Softwareeigenentwicklung bei Audi*, dSpace Anwenderkonferenz 2013 - Vortrag, 2013.

[KF09] KINDEL, Olaf; FRIEDRICH, Mario: *Softwareentwicklung mit AUTOSAR: Grundlagen, Engineering, Management in der Praxis*, Dpunkt-Verl., Heidelberg, 1. aufl Aufl., 2009, ISBN 978-3898645638.

[KH06] KOCHEM, M.; HOLZMANN, H.: *Einsatz der Road-Lab-Math Strategie bei der simulationsbasierten Entwicklung von Fahrdynamikregelsystemen*, . In: *Steuerung und Regelung von Fahrzeugen und Motoren - AUTOREG 2006*, Bd. 1931 von *VDI-Berichte*, S. 695–704, VDI-Verl., Düsseldorf, 2006, ISBN 3-18-091931-0.

[Köc08] KÖCHEL, Prof. Dr. Peter: *Vorlesungsskript:Algorithmen und Programmierung*, Technische Universität Chemnitz, 2008.
URL `https://www.tu-chemnitz.de/informatik/ModSim/skript_aup`

[KR08] KUROSE, James F.; ROSS, Keith W.: *Computer networking: A top-down approach*, Pearson/Addison Wesley, Boston, 4th ed Aufl., 2008, ISBN 978-0-321-49770-3.

[Kra07] KRATZKE, Nane: *Modellbasierte Analyse interorganisationaler Wissensflüsse*, Gito-Verl., Berlin, 2007, ISBN 978-3-9367-7192-3.

[Kre01] KREINES, David C.: *Oracle SQL: Die umfassende Referenz ; [für Oracle-DBAs und Entwickler]*, O'Reilly, Beijing and Cambridge and Farnham and Köln and Paris and Sebastopol and Taipei and Tokyo, dt. ausg., 1. aufl Aufl., 2001, ISBN 3-89721-183-1.

[LB05] LAMBERG, Klaus; BEINE, Michael: *Testmethoden und –tools in der modellbasierten Funktionsentwicklung*, . In: DSPACE GMBH (Hrsg.): *ASIM: Fachtagung Simulations- und Testmethoden für Software in Fahrzeugsystemen*, Bd. 2005, 2005.

[LH02] LEEN, G.; HEFFERNAN, D.: *Expanding automotive electronic systems*, . In: *IEEE Computer*, Bd. 35, S. 88–93, IEEE, 2002.

[Ltd14] LTD., ARM: *ARM Ltd.*, 2014, architektur: http://www.arm.com/products/processors/instruction-set-architectures/.
URL `http://www.arm.com/`

[McC76] MCCABE, T.J.: *A Complexity Measure*. In: *Software Engineering, IEEE Transactions on*, Bd. SE-2(4): S. 308–320, Dec 1976, ISSN 0098-5589.

[NIS05] NISO: *Guidelines for the construction, format, and management of monolingual controlled vocabulary*, National information standards series, NISO Press, Bethesda and Md, 2005, ISBN 978-1-880124-65-9.

[Nor02] NORTH, Klaus: *Wissensorientierte Unternehmensführung: Wertschöpfung durch Wissen*, Gabler, Wiesbaden, 3., aktualisierte und erw. aufl Aufl., 2002, ISBN 978-3-4093-3029-9.

[Obj14] OBJECT MANAGEMENT GROUP: *OMG Systems Modeling Language*, 2014, http://www.omgsysml.org.
URL `http://www.omgsysml.org`

[PCSF08] PILATO, C. Michael; COLLINS-SUSSMAN, Ben; FITZPATRICK, Brian W.: *Version control with Subversion*, O'Reilly Media, Sebastopol and CA, 2nd ed Aufl., ©2008, ISBN 978-0-5965-1033-6.

[pmb13] *A guide to the Project Management Body of Knowledge (PMBOK guide), fifth edition*, Project Management Institute, Newtown Square and Pa, 5th ed Aufl., 2013, ISBN 978-1935589679.

[POI14] POI, Apache: *Apache POI Projekt*, 2014, https://poi.apache.org.
URL `https://poi.apache.org`

[PSMP12] PATEL-SCHNEIDER, Peter; MOTIK, Boris; PARSIA, Bijan: *OWL 2 Web Ontology Language XML Serialization (Second Edition)*, W3C recommendation, W3C, Dez. 2012, http://www.w3.org/TR/2012/REC-owl2-xml-serialization-20121211/.

[PTC14] PTC: *PTC Integrity*, 2014.
URL `http://de.ptc.com/product/integrity`

[Rau01] RAU, A.: *Einsatz von Metriken bei der modellbasierten Software-Entwicklung*, . In: *Simulaitonstechnik - 15. Symposium in Paderborn*, 2001.

[Rep15] REPOSITORY, CDO Model: *CDO Model Repository*, 2015, http://wiki.eclipse.org/CDO.
URL `http://wiki.eclipse.org/CDO`

[Ric09] RICHTER, Harald: *Elektronik und Datenkommunikation im Automobil*, lfl Technical Report lfl-09-05, TU Clausthal, Mai 2009.

[RTK] RUTHERGLEN, Jason.; TABORA, Ryan.; KRUPANSKY, Jack.: *Lucene and Solr: The definitive guide : the comprehensive guide to Lucene and Solr for realtime big data*, ISBN 978-1449359959.

[Sch98] SCHEER, August-Wilhelm: *ARIS - Vom Geschäftsprozeß zum Anwendungssystem*, Springer Berlin Heidelberg, Berlin and Heidelberg, dritte, völlig neubearbeitete und erweiterte auflage Aufl., 1998, ISBN 978-3-642-97820-3.

[Sch12] SCHEIBLE, Jan: *Automatisierte Qualitätsbewertung am Beispiel von MATLAB Simulink-Modelln in der Automobil-Domäne*, Dissertation, Eberhard Karls Universität Tübingen, Tübingen, 2012.
URL `https://publikationen.uni-tuebingen.de/xmlui/handle/10900/49708`

[Sch14] SCHUBA, Daniel: *Javadoc*, 2014, http://www.java-doc.de/.
URL `http://www.java-doc.de/`

[SDP08] STÜRMER, I.; DZIOBECK, C.; POHLHEIM, H.: *Modeling Guidelines and Model Analysis Tools in Embedded Automotive Software Development*, . In: *Proceedings of Dagstuhl Seminar Modellbasierte Entwicklung eingebetteter Systeme (MBEES 2008)*, S. 28–39, Schloss Dagstuhl, 2008.

[Sea14] SEARCH, Elastic: *Elastic Search Projekt*, 2014, https://www.elastic.co. URL `https://www.elastic.co`

[Sed02] SEDGEWICK, Robert: *Algorithmen*, Informatik, Addison-Wesley, München and Boston [u.a.], 2. aufl Aufl., 2002, ISBN 978-3-8273-7032-7.

[SL07] SPILLNER, Andreas; LINZ, Tilo: *Basiswissen Softwaretest: Aus- und Weiterbildung zum Certified Tester ; Foundation Level nach ISTQB-Standard*, Dpunkt-Verl., Heidelberg, 3., überarb. und aktualisierte aufl., korrigierter nachdr Aufl., 2007, ISBN 3-89864-358-1.

[Sol14] SOLR, Apache: *Apache Solr Projekt*, 2014, http://lucene.apache.org/solr/. URL `http://lucene.apache.org/solr/`

[SPR10] STÜRMER, I.; POHLHEIM, H.; ROGIER, T.: *Berechnung und Visualisierung der Modellkomplexität bei der modellbasierten Entwicklung sicherheits-relevanter Software*, . In: *Automotive - Safety & Security*, S. 69–82, Shaker Verlag, 2010.

[SS09] STAAB, Steffen; STUDER, Rudi: *Handbook on ontologies*, International handbooks on information systems, Springer, Berlin, 2nd ed Aufl., 2009, ISBN 978-3-540-92673-3.

[Sta73] STACHOWIAK, Herbert: *Allgemeine Modelltheorie*, Springer-Verlag, Wien and New York, ©1973, ISBN 978-3211811061.

[Sta07] STAHL, Thomas: *Modellgetriebene Softwareentwicklung: Techniken, Engineering, Management*, Dpunkt-Verl., Heidelberg, 2., aktualisierte und erw. aufl Aufl., 2007, ISBN 978-3-898-6444-88.

[Ste09] STEINBERG, Dave: *EMF: Eclipse Modeling Framework*, The eclipse series, Addison-Wesley, Upper Saddle River and NJ, 2nd ed., rev. and updated Aufl., ©2009, ISBN 978-0321331885.

[Str12] STRASSER, Benedikt: *Vernetzung von Test- und Simulationsmethoden für die Entwicklung von Fahrerassistenzsystemen*, Bd. 69 von *Audi-Dissertationsreihe*, Cuvillier, Göttingen, 1. aufl Aufl., 2012, ISBN 978-3-95404-263-0.

[SVN14] SVNKIT: *SVNKit Projekt*, 2014, http://svnkit.com. URL `http://svnkit.com`

[SZ13] SCHÄUFFELE, Jörg; ZURAWKA, Thomas: *Automotive Software Engineering: Grundlagen, Prozesse, Methoden und Werkzeuge effizient einsetzen*, ATZ-MTZ-Fachbuch, Springer Vieweg, Wiesbaden, 5., überarb. und aktualisierte aufl Aufl., 2013, ISBN 978-3-8348-2469-1.

[Teu11] TEUFEL, Marc: *Eclipse 4*, entwickler press, Frankfurt, 2011, ISBN 978-3868020632.

[Thi12] THIEL, Sebastian: *Petri-Netz-basierte Verifikation von funktionalen Testfällen*, Bd. Bd. 57 von *Audi-Dissertationsreihe*, Cuvillier, Göttingen, 1. aufl Aufl., 2012, ISBN 978-3-95404-056-8.

[VLKH04] VISARIUS, Markus; LESSMANN, Johannes; KELSO, Frank; HARDT, Wolfram: *Generic integration infrastructure for IP-based design processes and tools with a unified XML format*. In: *Integration*, Bd. 37(4): S. 289–321, 2004. URL `http://dx.doi.org/10.1016/j.vlsi.2004.01.001`

[vNC14] VON NEUMANN-COSEL, Kilian: *Virtual Test Drive - Simulation umfeldbasierter Fahrzeugfunktionen*, Dissertation, TU München, 2014.

[Voe03] VOERG, Andreas: *TOOLIP - Tools and Methods for IP*, 2003, http://toolip.fzi.de. URL `http://toolip.fzi.de`

[Voi08] VOIGTLÄNDER, Pierre: *ADTF: Framework for driver assistance and safety systems*, . In: *FISITA 2008: World Automotive Congress - Proceedings-Book of Abstracts*, Bd. 7, S. 563–567, 2008.

[WB05] WÖRN, Heinz; BRINKSCHULTE, Uwe: *Echtzeitsysteme: Grundlagen, Funktionsweisen, Anwendungen ; mit 32 Tabellen*, EXamen.press, Springer, Berlin [u.a.], 2005, ISBN 3-540-20588-8.

Eingebettete, Selbstorganisierende Systeme

im Universitätsverlag Chemnitz

(8) Meisel, André (2010)
Design Flow für IP basierte, dynamisch rekonfigurierbare, eingebettete Systeme
ISBN 978-3-941003-15-6
Volltext: http://nbn-resolving.de/urn:nbn:de:bsz:ch1-201000890

(9) Vodel, Matthias (2010)
Funkstandardübergreifende Kommunikation in Mobilen Ad Hoc Netzwerken
ISBN 978-3-941003-18-7
Volltext: http://nbn-resolving.de/urn:nbn:de:bsz:ch1-201001164

(10) Reichelt, Toni (2012)
A Model Driven Approach for Service Based System Design Using Interaction Templates
ISBN 978-3-941003-60-6
Volltext: http://nbn-resolving.de/urn:nbn:de:bsz:ch1-qucosa-85986

(11) Caspar, Mirko (2013)
Lastgetriebene Validierung Dienstbereitstellender Systeme
ISBN 978-3-941003-84-2
Volltext: http://nbn-resolving.de/urn:nbn:de:bsz:ch1-qucosa-110257

(12) Heller, Ariane (2013)
Systemeigenschaft Robustheit : ein Ansatz zur Bewertung und Maximierung von Robustheit eingebetteter Systeme
ISBN 978-3-941003-87-3
Volltext: http://nbn-resolving.de/urn:nbn:de:bsz:ch1-qucosa-113369

(13) Vodel, Matthias (2014)
Energieeffiziente Kommunikation in verteilten, eingebetteten Systemen
ISBN 978-3-944640-05-1
Volltext: http://nbn-resolving.de/urn:nbn:de:bsz:ch1-qucosa-131891

(14) Tudevdagva, Uranchimeg (2014)
Structure Oriented Evaluation Model for E-Learning
ISBN 978-3-944640-20-4
Volltext: http://nbn-resolving.de/urn:nbn:de:bsz:ch1-qucosa-146901

(15) Gruber, Thomas (2016)
Prozessintegrierte Dokumentation und optimierte Wiederverwendung von Simulationsmodellen der automobilen Funktionsabsicherung
ISBN 978-3-944640-89-1
Volltext: http://nbn-resolving.de/urn:nbn:de:bsz:ch1-qucosa-202845