Forschungsberichte

IPEK
Institut für Produktentwicklung
am KIT ▪ o. Prof. Dr. -Ing. Dr. h.c. A. Albers

Johannes Christian Zingel

Basisdefinition einer gemeinsamen Sprache der Produktentwicklung im Kontext der Modellbildung technischer Systeme und einer Modellierungstechnik für Zielsystem und Objektsystem technischer Systeme in SysML auf Grundlage des ZHO-Prinzips

Basis definition of a common language of product engineering in the context of modeling of technical systems and a modeling technique for the systems of objectives and objects of technical systems on the basis of the ZHO-principle

Band 70

Systeme ▪ Methoden ▪ Prozesse

Hrsg.: Prof. Dr.-Ing. Dr. h.c. A. Albers

Forschungsberichte

IPEK
Institut für Produktentwicklung
am KIT ▪ o. Prof. Dr.-Ing. Dr. h.c. A. Albers

Johannes Christian Zingel

Basisdefinition einer gemeinsamen Sprache der Produktentwicklung im Kontext der Modellbildung technischer Systeme und einer Modellierungstechnik für Zielsystem und Objektsystem technischer Systeme in SysML auf Grundlage des ZHO-Prinzips

Basis definition of a common language of product engineering in the context of modeling of technical systems and a modeling technique for the systems of objectives and objects of technical systems on the basis of the ZHO-principle

Band 70

Herausgeber: o. Prof. Dr.-Ing. Dr. h.c. A. Albers

Druck: Stolzenberger Druck und Werbung GmbH & Co. KG, Leimen
06224-7697915

ISSN 1615-8113

Basisdefinition einer gemeinsamen Sprache der Produktentwicklung im Kontext der Modellbildung technischer Systeme und einer Modellierungstechnik für Zielsystem und Objektsystem technischer Systeme in SysML auf Grundlage des ZHO-Prinzips

Zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften
der Fakultät für Maschinenbau
Karlsruher Institut für Technologie (KIT)

genehmigte
Dissertation

von

Dipl.-Ing. Johannes Christian Zingel
aus Dernbach (WW)

Tag der mündlichen Prüfung: 18. Oktober 2013
Hauptreferent: o. Prof. Dr.-Ing. Dr. h.c. A. Albers
Korreferent: Prof. Dr.-Ing. Udo Lindemann

Vorwort des Herausgebers **(Stand: März 2013)**

Wissen ist einer der entscheidenden Faktoren in den Volkswirtschaften unserer Zeit. Der Unternehmenserfolg wird in der Zukunft mehr denn je davon abhängen, wie schnell ein Unternehmen neues Wissen aufnehmen, zugänglich machen und verwerten kann. Die Aufgabe eines Universitätsinstitutes ist es, hier einen wesentlichen Beitrag zu leisten. In den Forschungsarbeiten wird ständig Wissen generiert. Dieses kann aber nur wirksam und für die Gemeinschaft nutzbar werden, wenn es in geeigneter Form kommuniziert wird. Diese Schriftenreihe dient als eine Plattform zum Transfer und macht damit das Wissenspotenzial aus aktuellen Forschungsarbeiten am IPEK - Institut für Produktentwicklung Karlsruhe[1] am Karlsruher Institut für Technologie (KIT) verfügbar.

Die Forschungsfelder des Institutes sind die methodische Entwicklung und das Entwicklungsmanagement, die rechnergestützte Optimierung von Strukturen und Systemen, die Antriebstechnik mit einem Schwerpunkt auf den Gebieten Antriebsstrang-Engineering und Tribologie und Monitoring von Lager- und Funktionsreibsystemen, die NVH mit dem Fokus auf Schwingungen und Akustik an Komponenten und am Gesamtfahrzeug, die Mikrosystemtechnik mit dem Fokus auf die zugehörigen Entwicklungsprozesse sowie die Mechatronik. Die Forschungsberichte werden aus allen diesen Gebieten Beiträge zur wissenschaftlichen Fortentwicklung des Wissens und der zugehörigen Anwendung – sowohl den auf diesen Gebieten tätigen Forschern als auch ganz besonders der anwendenden Industrie – zur Verfügung stellen. Ziel ist es, qualifizierte Beiträge zum Produktentwicklungsprozess zu leisten.

Albert Albers

[1] Eh.: Institut für Maschinenkonstruktionslehre und Kraftfahrzeugbau, Universität Karlsruhe (TH)

Vorwort zu Band 70

Der moderne Maschinen- und Fahrzeugbau ist gekennzeichnet von einer immensen Komplexität und Kompliziertheit der entwickelten technischen Systeme. Insbesondere durch die fortschreitende Mechatronisierung sowie die Miniaturisierung in großen Teilbereichen werden technische Systeme entwickelt, die das Mitwirken und Zusammenwirken einer Vielzahl von Disziplinen, Experten und Entwicklungsmethoden und Werkzeugen erfordern. Um diesen Herausforderungen zu begegnen, werden von unterschiedlichen Forschungsgruppen neue Ansätze zur Unterstützung dieses Entwicklungsprozesses angegangen. So beschreibt Lindemann mit dem Münchner Produktkonkretisierungsmodell (MKM) neuartige Vorgehensweisen zur vernetzten Entwicklung technischer Lösungen. Gausemeier und seine Gruppe betrachten insbesondere den Aspekt der Mechatronik und führen hierzu ebenfalls konzeptionell neue Ansätze zu deren Beschreibung und Synthese ein. Die Gruppe um Albers arbeitet ebenfalls seit mehr als 15 Jahren auf dem Gebiet neuartiger Konzepte für Entwicklungsprozesse. Basis dieser Arbeiten sind das von Ropohl definierte ZHO-Modell der Systemtechnik und die Grundlagen der Systemtheorie. Auch hier ist es Ziel, durch die Verknüpfung der unterschiedlichen disziplinären Aspekte, ihrer Vorgehensweisen sowie Werkzeuge und Tools, aber auch deren Denkweise eine strukturierte Vorgehensunterstützung bei der Synthese und Analyse komplexer technischer Systemlösungen auf einer Kommunikationsebene mit Meta-Modellen zu erarbeiten. Von entscheidender Bedeutung in diesem Zusammenhang ist eine durchgängige und fachdisziplinenübergreifende Modellierungssprache, auf deren Basis dann auch entsprechende Regeln und Vorgehensweisen definiert werden können. An diesem Konzept arbeitet die Gruppe seit einigen Jahren und setzt hier u. a. die Ansätze des Model-Based-Systems-Engineering (MBSE) ein. An dieser Stelle setzt die wissenschaftliche Arbeit von Herrn Dr.-Ing. Christian Zingel an. Er hat sich zum Ziel gesetzt, die modellbasierte Systementwicklung in der Produktentstehung durch die Definition einer gemeinsamen Sprache für den Kontext der Modellbildung technischer Systeme sowie ihrer Formalisierung durch Modellbildung, Modellierungsregeln und ein Vorgehensmodell für die Anwendung der entsprechenden Modellierungstechnik zu unterstützen. Die Zielsetzung der Arbeit von Herrn Zingel wird auf der Basis von grundlegenden Forschungshypothesen formuliert. Diese fünf grundlegenden Forschungshypothesen fordern eine gemeinsame kompakte Sprachbasis für den Kontext der Modellbildung; postulieren, dass über eine anwenderfreundliche multidisziplinäre Modellbildung eine nachhaltige, fachdisziplinübergreifende Kommunikation möglich wird; fordern, dass ein entsprechendes Modell eindeutig, aber auch hochflexibel sein muss; postulieren,

dass es nicht Ziel sein soll, mit der fachdisziplinübergreifenden Modellierungssprache bestehende Modelle zu ersetzen, sondern diese miteinander zu vernetzen und zu verknüpfen und postulieren abschließend aus Sicht des Anwenders, dass die entsprechende Modellierungssprache Informationen und deren Zusammenhänge anwendergerecht darstellen können muss. Auf der Basis einer umfangreichen Analyse des Standes der Forschung gelingt es Herrn Dr.-Ing. Christian Zingel in seiner wissenschaftlichen Arbeit auf der Basis klar definierter Zielsysteme für die Modellierungssprache und die Modellierungstechnik diese systematisch mit wissenschaftlichen Methoden zu erarbeiten und formal zu beschreiben. Die Arbeit leistet einen wichtigen Beitrag zur Karlsruher Schule für Produktentwicklung und gibt wesentliche Impulse für die Forschung auf dem Gebiet der Produktentstehungsprozesse.

Oktober, 2013 Albert Albers

Kurzfassung

Entstehungsprozesse technischer Systeme sind komplexe Handlungssysteme zahlreicher handelnder Menschen verschiedener Fachexpertisen unter Verwendung unterschiedlichster Ressourcen zur Bildung, Speicherung und Kommunikation von Wissen in Form von Zielen und Objekten. Die vorliegende Arbeit begegnet den Herausforderungen einer erfolgreichen Kommunikation in der fachdisziplinübergreifenden Zusammenarbeit durch eine Basisdefinition einer gemeinsamen Sprache der Produktentwicklung für den Kontext der Modellbildung technischer Systeme. Darauf basierend wird eine Modellierungstechnik als zentrales Werkzeug zur Handhabung der umfangreichen und heterogenen Daten und Informationen vorgestellt. Der Aufbau der Arbeit gliedert sich wie nachfolgend beschrieben:

Aus der Diskussion der Stärken und Schwächen existierender multidisziplinärer Modellierungssprachen und Methoden werden die Motivation und die Zielsetzung für die angestrebte Modellierungstechnik abgeleitet. Diese besteht aus einer Sprache zur Modellierung der fachdisziplinübergreifend relevanten Ziel- und Objektsystemelemente sowie einer Methode zu deren zielführenden Anwendung. Anschließend wird das Forschungsdesign der Entwicklung der angestrebten Ziele vorgestellt, gefolgt von den Ergebnissen der Forschungsarbeit. Zunächst wird die Basisdefinition einer gemeinsamen Sprachbasis der Produktentwicklung im Kontext der Modellbildung technischer Systeme vorgestellt, worin die Syntax zentraler Begriffe und deren semantische Zusammenhänge definiert werden. Darauf basierend wird eine formale Sprache auf Basis der Systems Modeling Language (SysML) zur durchgängigen, ganzheitlichen und anwendergerechten Modellierung der fachdisziplinübergreifend relevanten Elemente des Zielsystems und des Objektsystems technischer Systeme spezifiziert. Sie greift die Konzepte des Contact & Channel – Ansatzes zur integrativen Betrachtung des Zusammenhangs von Funktion und ausführender Gestalt in der Analyse und Synthese multidisziplinärer technischer Systeme auf. Durch diese sollen ein maximaler Lösungsraum für innovative Lösungen gewahrt und bauteilbehaftetes Denken überwunden werden. Die Aspekte der Modellierungstechnik werden an Beispielen erläutert und in ein flexibles Vorgehensmodell eingeordnet. Darüber hinaus wird ein Konzept zur Integration der Sprache in eine Produktentwicklungsumgebung vorgestellt. Anschließend wird die Modellierungstechnik an mehreren Anwendungsbeispielen validiert und diskutiert. Aus der Diskussion ihrer Potentiale und resultierenden Herausforderungen werden Vorschläge für mögliche, weiterführende wissenschaftliche Arbeiten abgeleitet.

abstract

Development processes of technical systems are complex operation systems of numerous actors with different expertise using highly diverse resources in order to generate, store and communicate knowledge in terms of objectives and objects. The research work at hand faces the challenges of successful communication in discipline-crossing collaboration through a basic definition of a common language for product development in the context of modeling technical systems. Furthermore, a modeling technique as core tool for handling the comprehensive and heterogeneous data and information is presented. The research work is structured as following:

Deriving from the discussion of strengths and weaknesses of existing multidisciplinary modeling languages and methods, the motivation and targets regarding the intended modeling technique are defined. The technique consists of a language for modeling discipline-crossing relevant elements of the Systems of Objectives and the System of Objects as well as a method for its purposeful and expedient application. The definition of the System of Objectives of the modeling technique is followed by an overview of its development operations and its resulting objects.

The basic definition of a common language for product development in the context of modeling technical systems defines the syntax of basic terms and their semantic context. Based on those definitions, a formal language based on the Systems Modeling Language (SysML) for continuous, comprehensive and user-friendly modeling of cross disciplinary relevant elements of the System of Objectives and the System of Objects of technical systems is specified. It seizes on the concepts of the Contact & Channel – Approach for integrated analysis and synthesis of function and form of multi-disciplinary systems. Consequently, a maximal solution space shall be kept and component-afflicted thinking shall be overcome. The different aspects of the modeling technique are elucidated using examples from conducted research projects and are arranged in a flexible procedure model. Furthermore, a concept for integrating the language into a product development environment is presented.

Afterwards, the modeling technique is validated and limitations are discussed using several application examples. Finally, proposals for further scientific research efforts are derived from the discussion of potentials and emerging challenges.

Danksagung

Diese Arbeit entstand im Rahmen meiner Tätigkeit als akademischer Mitarbeiter zunächst in der Forschungsgruppe NVH/Driveability und später Systemische Mobilität am IPEK – Institut für Produktentwicklung am Karlsruher Institut für Technologie (KIT).

Mein größter Dank gebührt meinem Doktorvater, Herrn o. Prof. Dr.-Ing. Dr. h.c. Albert Albers, für die Betreuung meiner wissenschaftlichen Arbeit, das entgegengebrachte Vertrauen und die Ermöglichung der tollen Rahmenbedingungen für meine Forschungsarbeiten. Insbesondere werde ich die fruchtbaren und spannenden Diskussionen sowie seine konstruktiven Denkanstöße in Erinnerung behalten.

Für die Übernahme des Koreferates bedanke ich mich ganz herzlich bei Herrn Prof. Dr.-Ing. Udo Lindemann vom Lehrstuhl für Produktentwicklung der TU München. Ich hatte das Glück, unter anderem in einem Forschungsprojekt mit seinem kompetenten Team zusammenarbeiten zu dürfen, woraus nette Kontakte und tolle Erfahrungen entsprangen.

Weiterhin möchte ich insbesondere den Kollegen der Entwicklungsmethodik und Management für die tolle Zusammenarbeit und den sehr intensiven wissenschaftlichen Austausch danken, die meine Arbeit maßgebend mit geprägt haben. Dies gilt ferner auch für alle Kolleginnen und Kollegen am Institut. Nicht nur die wissenschaftlichen Diskussionen und fruchtbaren Impulse, sondern auch das sehr angenehme und offene Arbeitsklima haben mich stets begeistert und angespornt.

Auch über die Grenzen des IPEK hinaus habe ich zahlreiche interessante Kontakte knüpfen können. Vor allem die tolle Zusammenarbeit mit den Kollegen vom Virtuellen Fahrzeug (ViF) sowie der AVL (beide in Graz) hat viele Ergebnisse dieser Arbeit überhaupt erst ermöglicht.

Zudem danke ich meinen Eltern Margot und Hans-Walter Zingel, die mich stets bei meiner Arbeit motiviert und unterstützt haben und mir in allen Lebenslagen den Rücken freigehalten haben.

Schließlich möchte ich mich ganz besonders bei meiner Partnerin Anna-Lena Möller für ihr Verständnis, ihre Geduld und die vielfältige Unterstützung in den letzten Jahren bedanken.

Karlsruhe, im Juli 2013

Johannes Christian Zingel

“So that we may say the door is now opened, for the first time, to a new method fraught with numerous and wonderful results, which in future years will command the attention of other minds”.

Galileo Galilei, im 16. Jhdt.

Inhalt

Abkürzungsverzeichnis

Abkürzung	Bedeutung	Erläuterung
AD	Axiomatic Design	Kapitel 2.4.5
AST	Abstract Syntax Tree	Kapitel 2.8.4
AUTOSAR	AUTomotive Open System ARchitecture	Kapitel 2.6.8
BPMN	Business Process Modeling Notation	Kapitel 2.6.4
C&C²-A	Contact & Channel – Ansatz	Kapitel 2.4.9
C&C²-M	Contact & Channel – Model	Kapitel 2.4.9
CONSENS	Conceptual Design Specification Technique for the Engineering of Complex Systems	Kapitel 2.6.5
CPM	Characteristics Properties Modelling	Kapitel 2.4.7
DfX	Design for X	Kapitel 2.4.4
DSL	Domain-Specific Language	Kapitel 2.6.3
EAST-ADL	Electronics Architecture and Software Technology – Architecture Description Language	Kapitel 2.6.7
F-Logic	Frame-Logic	Kapitel 2.7.3
FAS	Funktionale Architekturen für Systeme - Methode	Kapitel 2.6.2.6, 2.10.2
GDT	General Design Theory	Kapitel 2.4.2
GRM	Generic Reference Model	Kapitel 2.4.1
MARTE	Modeling and Analysis of Real Time and Embedded systems	Kapitel 2.6.6
MDA	Model-Driven Architecture	Kapitel 2.8.1
MKM	Münchner Produktkonkretisierungsmodell	Kapitel 2.4.8
MOF	Meta Object Facility	Kapitel 2.8.2
OCL	Object Constraint Language	Kapitel 2.8.3
OPM	Object Process Methodology	Kapitel 2.10.3
OWL	Web Ontology Language	Kapitel 2.7.2
QVT	Query View Transformation	Kapitel 2.8.5
RDF	Resource Description Framework	Kapitel 2.7.1

STEP	Standard for the Exchange of Product Model Data	Kapitel 2.8.8
SysML	Systems Modeling Language	Kapitel 2.6.2
UDT	Universal Design Theory	Kapitel 2.4.3
UML	Unified Modeling Language	Kapitel 2.6.1
XMI	XML Metadata Interchange	Kapitel 2.8.7
XML	Extensible Markup Language	Kapitel 2.8.6

Begriffsglossar

Kontext: Die hier aufgeführten Begriffsdefinitionen sind auf den Kontext der Modellbildung technischer Systeme beschränkt, da eine generalisierte Definition, die sämtlichen Kontexten entspräche, nicht zielführend umsetzbar ist.

Hinweis: Aufgrund der Mehrfachbelegung einiger Begriffe werden Elemente der SysML in ihrer englischen Originalbezeichnung aufgeführt und durch *Kursivdruck* hervorgehoben.

Aktivität *Activity*	1. Element des Handlungssystems im Integrierten Produktentstehungsmodell (iPeM). Unterscheidung in Makroaktivitäten der Produktentwicklung und Mikroaktivitäten der Problemlösung. 2. Modellelement der SysML zur Modellierung von logischen Abläufen und der Verarbeitung von Objektflüssen in Aktivitätsdiagrammen.
Anwendungsfall (*Use Case*)	Ein Anwendungsfall beschreibt zeitlich zusammenhängende und zielgerichtete logische Abfolge von Funktionen eines technischen Systems.[2]
Architektur	Beschreibung von Funktionen und Strukturen eines Systems sowie deren Wechselwirkungen (in Anlehnung an PAHL ET AL.[3])
Aspekt	Gruppierung von Informationen mit einem engen semantischen Zusammenhang (z.B. funktionale Struktur, Anforderungen, Zustände etc.).
Attribut	Attribut ist ein Sammelbegriff für Kennzeichen, Merkmale, Eigenschaften von Individuen bzw. Systemen, Relationen zwischen Individuen/Systemen, Eigenschaften von Eigenschaften, Eigenschaften von Relationen usw.[4]
Baustruktur	→ Physische Struktur
Bauteil	Ein Bauteil ist ein rangniedrigstes Sachsystem, das elementar im Sinne der technischen Betrachtung ist, also

[2] Vgl. Weilkiens (2008) sowie Kapitel 5.2.7
[3] Pahl et al. (2007), S. 697
[4] Vgl. Stachowiak (1973) und Ropohl (1975)

	nicht weiter sinnvoll zerlegt werden kann, da man es sonst mit mikrophysikalischen oder chemischen Sachverhalten zu tun hat.[5]
Begrenzungsfläche (BF)	Begrenzungsflächen sind feste Oberflächen von Körpern oder generalisierte Grenzflächen von Flüssigkeiten, Gasen oder Feldern, die nie Wirkflächen sind.[6]
Bewertung	Eine Bewertung charakterisiert eine oder die Summe mehrerer interpretierter Wirkungen in Form einer Eigenschaft.
Block	Element der SysML, das in Strukturdiagrammen zur Beschreibung einer Klasse von Systembausteinen mit Parametern und Schnittstellen eingesetzt wird.[7]
Block Property (Part)	Element der SysML, das einen Block in einer Rolle mit Schnittstellen zu Nachbarsystemen repräsentiert.
Block Property (Value)	Element der SysML, das einen Parameter mit einem Werte-/Datentyp/Einheit repräsentiert.
Connector (C)	Connectoren integrieren die wirkungsrelevanten Merkmale, Eigenschaften und Wirkflächen durch Modelle der mit dem System interagierenden Systemumgebung. Sie liegen im betrachteten System, jedoch nicht im Gestaltungsraum.[6]
Connector	Relation der SysML zur Verknüpfung von →Ports (Schnittstellen) in Blockdiagrammen.
Diskrete Funktion	Die Diskrete Funktion ist eine Ist-Funktion, die in der Transition zwischen zwei Systemzuständen ausgeführt wird. Sie wird durch eine charakteristische Änderung von Inputs ausgelöst und endet mit Abschluss der Transition der betreffenden Zustandsattribute (Merkmale, Eigenschaften, WFP).[8]
Effekt	Ein Effekt beschreibt eine naturwissenschaftliche Gesetzmäßigkeit zur Bestimmung des Verhältnisses

[5] Vgl. Ropohl (1975), S. 35
[6] Vgl. Kapitel 2.4.9
[7] Vgl. Kapitel 2.5

	zwischen Inputs und Outputs in WFP und LSS unter Einbezug ihrer relevanten Merkmale und Eigenschaften.[8]
Eigenschaft	Eine Eigenschaft ist ein nicht unmittelbar vom Entwickler beeinflussbares Attribut eines Technischen Systems.[8]
Entität	Beschreibt ein eindeutig zu bestimmendes Objekt innerhalb eines Modells, das über einen eindeutigen Bezeichner verfügt sowie Eigenschaften und Relationen erhalten kann.
Funktion	Eine Beziehung zwischen zwei oder mehr Attributen durch die Zuordnung von Werten eines Attributs zu bestimmten Werten des/der anderen Attributs/e (kann, muss aber nicht mathematisch beschreibbar sein).[9]
Gestalt	Die Gestalt ist die quantitative Beschreibung der Summe aller Merkmale und Struktureigenschaften der physischen Struktur eines technischen Systems.[8]
Ist-Funktion	Eine Ist-Funktion als Teil des Objektsystems beschreibt das in ihrer Wirkstruktur stattfindende Transformationsprinzip von Inputs in Outputs durch zugrundeliegende Effekte sowie funktionsrelevante Merkmale und Eigenschaften.[10]
Klasse (*Class*)	Element der UML, das eine Klassifizierung von Objekten mit gleichartigen Eigenschaften erlaubt.[11]
Komponente	Physischer Systembestandteil, kann sowohl aus Hard- als auch Software oder einer Kombination beider bestehen. Komponenten sind Bestandteil der physischen Systemstruktur.
Konstrukt	Überbegriff für ein Element einer Modellierungssprache, das entweder eine Entität, eine Relation oder ein Diagramm sein kann.
Kontinuierliche Funktion	Die Kontinuierliche Funktion ist eine Ist-Funktion, die innerhalb eines Systemzustandes ausgeführt wird. Sie dient der Zustandserhaltung und muss durch bestimmte

[8] Vgl. Kapitel 5.2.7
[9] Nach Ropohl (1975), S. 26/27
[10] vgl. Kapitel 2.1
[11] vgl. Kapitel 2.5

	Ausprägungen von Inputs beendet werden.[8]
Merkmal	Ein Merkmal ist ein Attribut eines Strukturelements eines technischen Systems (z.B. Datenformat, Interfaceart, Form, Lage, Stoff) und wird durch den Entwickler festgelegt.[8]
Metamodell	Modell zur Beschreibung der abstrakten Syntax und der statischen Semantik einer Modellierungssprache.
Modell	Ein Modell ist eine abstrahierte Beschreibung der Realität.[12]
Modellartefakt	Bezeichnet Modellelemente, die in Diagrammen zur Modellierung verwendet werden können, die entweder Entitäten oder Relationen darstellen.
Leitstützstruktur (LSS)	Leitstützstrukturen (LSS) leiten während der Funktionserfüllung zwischen genau zwei Wirkflächenpaaren Energie, Stoff und/oder Information. Eine Leitstützstruktur kann sich dabei abhängig vom Detaillierungsgrad der Modellbildung über Systeme oder Subsysteme hinweg erstrecken. Leitstützstrukturen existieren gemeinsam mit den zugehörigen Wirkflächenpaaren ausschließlich im Zeitraum der Funktionserfüllung. Leitstützstrukturen können nur in Volumina von Festkörpern, Flüssigkeiten, Gasen oder felddurchsetzten Räumen oder deren Kombination realisiert werden.[6]
Parameter	Der Begriff Parameter dient in dieser Arbeit einerseits als Oberbegriff für Merkmale und Eigenschaften technischer Systeme und deren Subsysteme und ist andererseits eine Bezeichnung für das SysML-Element *Block Property (Value)*, das Eigenschaften und Merkmale in SysML-Modellen repräsentiert.
Physische Struktur	Die Physische Struktur (Baustruktur) enthält die Charakteristika der Wirkstruktur und konkretisiert die LSS und WFP durch die Ausarbeitung zu resultierenden physischen Komponenten und Schnittstellen. Diese umfassen darüber hinaus Reststrukturen und nicht funktionsrelevante Merkmale und Eigenschaften.[8]

[12] Nach Alt (2012)

Profil (*Profile*)	Eine Menge von Stereotypen und Tag Definitions zur Erweiterung der UML/SysML um zusätzliche Entitäten und Relationen.
Relation	Eine Relation ist ein Zusammenhang zwischen einem Attribut eines (Sub)Systems und einem Attribut eines anderen (Sub)Systems.[13]
Reststruktur (RS)	Reststrukturen sind Volumina von Körpern, Flüssigkeiten, Gasen oder felderfüllte Räume, die nie Tragstruktur werden.[6]
Schnittstelle *Port*	Bezeichnet einen Interaktionsbereich zwischen zwei Systemen. Die SysML V 1.2 spezifiziert *Standard Ports* (Datenflussports der UML) und *Flow Ports* (Objektflussports der SysML). Ab V1.3 wurden diese als *Port* zusammengefasst[14]. In dieser Arbeit werden Ports zur Modellierung von Wirkflächen eingesetzt.
Sicht	Eine Sicht ist eine Projektion eines Modells, die es von einer bestimmten Perspektive oder einem Standpunkt aus zeigt und Dinge weglässt, die für diese Perspektive nicht relevant sind.[15]
Stereotyp (*Stereotype*)	Element des Erweiterungsmechanismus der MOF, ergänzt ein bestehendes Modellelement um weitere Eigenschaften und Semantik. Das neu definierte Modellelement kann neben dem Namen auch eine neue grafische Darstellung erhalten.[16]
Struktur	Eine Struktur ist die Menge der Relationen eines Systems.[17]
Struktureigenschaft	Struktureigenschaften charakterisieren gemessene oder berechnete statische Strukturkennwerte (z.B. Gewicht, Kosten, Montagegerechtheit, Ästhetik).[8]

[13] Nach Ropohl (1975), S. 28
[14] Vgl. OMG SysML (2012)
[15] Nach Alt (2012)
[16] Vgl. Weilkiens (2008)
[17] Nach Ropohl (1975)

System	Ein System ist das Modell einer Ganzheit, die Beziehungen zwischen Attributen (Inputs, Outputs, Zustände etc.) aufweist, die aus miteinander verknüpften Teilen bzw. Subsystemen besteht, und die von ihrer Umgebung bzw. von einem Supersystem abgegrenzt wird.[10]
Tag definition	Element des Erweiterungsmechanismus der MOF[18] zur Definition neuer Relationen (Reference Tag) oder Attribute (Non-Reference Tag) als Teil eines Profils.
Technische Funktion	Eine Technische Funktion beschreibt den Zusammenhang zwischen Inputs (Stoff, Energie, Information) - den Ursachen – und Outputs – den Wirkungen, der auch gleichzeitig bidirektional in Form von Wechselwirkungen bestehen kann.[10]
Technisches System	Ein Technisches System (Sachsystem) ist ein künstlich entstandenes, zweckorientiertes, offenes und dynamisches System.[10]
Testfall	Ein Testfall beschreibt eine konkrete Ausprägung eines Anwendungsfalls. Er beschreibt eine Sequenz von Beaufschlagungen konkreter Inputverläufe über der Zeit, wodurch die Ausführung von Ist-Funktionen verursacht wird. Dadurch bilden sich Wirknetze aus. Testfälle validieren den Zweck eines technischen Systems durch Messung und Interpretation der Outputs von Ist-Funktionen bzw. verifizieren Merkmale seiner Struktur.[8]
Tragstruktur (TS)	Tragstruktur (TS) ist die Menge aller möglichen Leitstützstrukturen.[6]
Transition	Eine Transition beschreibt einen Zustandswechsel und verändert die Anordnung von WFP und/oder die Eigenschaften von LSS in der Struktur eines technischen Systems oder Teilsystems.[8]
Ursache	Eine Ursache löst Funktionen in Form charakteristischer Inputs aus.[8]
Verhalten	Das Verhalten ist die Reaktion eines Technischen Systems

[18] Vgl. Kapitel 2.8.2

	auf konkrete Inputs, die in Wirknetzen gewandelt werden. Die resultierenden Outputs sind an den Systemgrenzen als Wirkungen messbar und ermöglichen eine Interpretation (Validierung) des Systemverhaltens.[8]
Verhaltenseigenschaft	Verhaltenseigenschaften charakterisieren direkt gemessene Outputs eines Wirknetzes (z.B. NVH-Größen, Leistung) oder durch Menschen hinsichtlich der Zielerreichung interpretierte Outputs (z.B. Umwelteigenschaften, Sicherheit) eines technischen Systems.[8]
Wechselwirkung	Eine Wechselwirkung ist eine bidirektionale Interaktion zweier Funktionen über ein WFP, die gleichzeitig Ursache (Input) und Wirkung (Output) beider beteiligter Funktionen darstellt.[8]
Wirkfläche (WF)	Wirkflächen sind feste Oberflächen von Körpern oder generalisierte Grenzflächen von Flüssigkeiten, Gasen oder Feldern, die dauernd oder zeitweise im Kontakt zu einer weiteren Wirkfläche stehen und am Energie-, Stoff- und/oder Informationsaustausch des technischen Systems beteiligt sind.[6]
Wirkflächenpaar (WFP)	Wirkflächenpaare werden gebildet aus zwei beliebig geformten Wirkflächen, die in Wirkkontakt stehen, in dem Energie, Stoff und/oder Information übertragen werden.[6]
Wirkkontakt (WK)	Der Wirkkontakt ist der Teil eines Wirkflächenpaares, in dem aktuell die Wechselwirkungen stattfinden.[6]
Wirknetz	Ein Wirknetz beschreibt die Propagation (Wandlung) der beaufschlagten Inputs durch eine Wirkstruktur im Moment der Ausführung technischer Funktionen unter definierten, konkreten Konditionen (Inputverläufe über der Zeit sowie Zustandsgrößen der beteiligten LSSen) bis hin zu den resultierenden Outputs.[8]
Wirkstruktur	Die Wirkstruktur beschreibt das Gefüge (Aufbau und Anordnung) der LSS, WFP und C eines technischen Systems sowie deren funktionsrelevanten Merkmale und Eigenschaften. Sie ist die Summe aller möglichen Wirknetze einer Funktion.[8]
Wirkung	Eine Wirkung beschreibt die aus einer Funktion

	resultierenden Outputs, die innerhalb eines technischen Systems gleichzeitig Ursachen (Inputs) mit Auslösung weiterer Funktionen sein können. An der Systemgrenze sind Wirkungen messbar.[8]
Ziel-Funktion	Eine Ziel-Funktion als Teil des Zielsystems beschreibt einen Zweck bzw. eine Aufgabe des zu entwickelnden technischen Systems aus Sicht des Anwenders (als Teil der Systemumgebung).[10]
Zustand	Ein Zustand repräsentiert eine konkrete Ausprägung der Struktur eines technischen Systems oder eines Subsystems mit einer bestimmten Menge von Werte- oder Wertebereichskombinationen ihrer Attribute.[8]
Zweck	Ein Zweck eines technischen Systems ist eine von einer menschlichen Instanz erwartete Funktion in Interaktion mit seiner Umgebung mit Hilfe bestimmter Inputs und unter bestimmten Randbedingungen.[10]

1 Einleitung

Die heutige Gesellschaft ist auf vielfältige Weise von technischen Systemen geprägt. Seit Beginn der Industrialisierung vor gut zweihundert Jahren und insbesondere dem Wandel zur Informationsgesellschaft in den letzten Dekaden wird jedes Individuum mit einer rasant zunehmenden Komplexität seiner Umwelt konfrontiert. Dabei schreitet die technische Entwicklung derart schnell voran, dass der Mensch kaum in der Lage ist, das selbst geschaffene „Technotop“ zu verstehen oder gar zu bewältigen[19]. Die immense Herausforderung im Umgang mit Komplexität äußert sich insbesondere durch den Innovationsdruck von Herstellern technischer Produkte. Der sich globalisierende Wettbewerb zwingt Unternehmen einerseits zu kürzeren Entwicklungszeiten, andererseits hemmt die Zunahme der Systemkomplexität durch neue Technologien und den Einbezug neuer Branchen in die Entwicklung neuer Innovationen den Fortschritt. Eine Studie des Fraunhofer Instituts für System- und Innovationsforschung (ISI) aus 2005 belegt, dass sich die Altersstruktur von Produkten der deutschen Metall- und Elektroindustrie, gemessen am Umsatzanteil in den letzten Jahren nicht verkürzt hat[20]. Eine weitere Studie des Instituts aus 2009 konnte darüber hinaus belegen, dass weniger FuE-intensive Unternehmen sich insbesondere durch prozessbezogene Leistungsindikatoren wie Qualität, Effizienz und Flexibilität weiterhin erfolgreich am Markt behaupten können. Sie erwirtschaften laut der Studie aktuell rund 41 Prozent der industriellen Wertschöpfung und beschäftigen etwa die Hälfte aller Erwerbstätigen in der deutschen Industrie[21].

Diese Fakten erlauben den Rückschluss, dass viele Unternehmen heute noch nicht in der Lage sind, mit der steigenden Komplexität durch neue Technologien umzugehen und sich die damit einhergehenden, vielfältigeren Innovationspotentiale zunutze zu machen. Beispielsweise bewegt sich besonders die Automobilindustrie in einem Spannungsfeld aus Innovation, Kosten und Zeit[22]. Hier führt der durch zunehmende Globalisierung massiv verschärfte Wettbewerb immer häufiger zu Qualitätsmängeln, wie der Jahresbericht 2010 des Kraftfahrt-Bundesamtes[23] eindrucksvoll in der deutlich steigenden Anzahl an Rückrufaktionen[24] belegt.

[19] Ropohl (2009)
[20] Kinkel (2005)
[21] Som et al. (2011)
[22] Fachbach et al. (2012)
[23] Immen (2010)
[24] Beispiel von Toyota (weltgrößter Automobilhersteller): FTD (2012)

Insbesondere die Elektronifizierung des Automobils durch zahlreiche neue Assistenzsysteme entwickelt sich einerseits zu einem immer bedeutenderen Innovationstreiber, stellt die Automobilhersteller aber andererseits vor immer größere Herausforderungen bei ihrer erfolgreichen Integration in das Produkt[25]. Immer häufiger werden aufgrund des Wettbewerbsdrucks unausgereifte Produkte mit teilweise gravierenden Folgen für die Sicherheit und die Selbstverantwortung des Menschen auf den Markt gebracht[26].

Die vorliegende Arbeit fokussiert sich daher auf die Herausforderungen an Manager und Produktentwickler durch Wissenslücken, die aus der steigenden Komplexität durch die Integration neuer Disziplinen in die Entwicklung technischer Systeme entstehen. Die früher rein auf Mechanik fokussierte Produktentwicklung wird immer mehr zu einem mechatronischen Systementwicklungsprozess und erfordert folglich auch neue Herangehensweisen bei der Handhabung der einhergehend steigenden Systemkomplexität. Dieser auf der Systemtheorie basierende Forschungs- und Entwicklungstrend wird als „Systems Engineering (SE)“ bezeichnet. Diese Disziplin hat ihre Anfänge bereits Anfang des 20. Jahrhunderts und gewann ab den 50er Jahren durch das Wettrüsten der Weltmächte und ihre führenden Technologietreiber wie dem Department of Defense der USA und der NASA mit ihrem Raumfahrtprogramm APOLLO durch die steigende Systemkomplexität stark an Bedeutung[27]. In den letzten Jahren ist aus dem „traditionellen“ SE aufgrund der voranschreitenden Computertechnologie ein weiteres Forschungsfeld entstanden: das „Model-Based Systems Engineering (MBSE)“. MBSE verfolgt das Ziel, Produktentwicklung durch einen Wechsel von der heterogenen, dokumentenzentrierten Entwicklung hin zu einer konsistenten und vernetzten, modellbasierten Entwicklungsmethodik effizienter zu gestalten. Ein Mehrwert durch den durchgängigen Einsatz von Modellen in der Produktentwicklung konnte bereits nachgewiesen werden[28]. Im MBSE übernimmt ein interdisziplinäres Systemmodell eine zentrale Rolle bei der Speicherung, Vernetzung und Repräsentation der im Produktentstehungsprozess entstehenden Daten und Informationen. Der weltweite Dachverband des Systems Engineering INCOSE[29] hat in Kooperation mit der OMG[30] die Systems Modeling Language (SysML) entwickelt und 2007 erstmals standardisiert. Seitdem setzt sie sich als Werkzeug zur disziplinübergreifenden

[25] Ackermann (2007)
[26] Weyer (2006)
[27] Hitchins (2007)
[28] Z. B. durch Broy et al. (2011)
[29] International Council on Systems Engineering
[30] Object Management Group

Systemmodellierung zunehmend durch. Im Sinne der Interoperabilität[31] und der Flexibilität wird diese durch anwenderspezifische Profile[32] erweiterbare, grafische Modelliersprache auch in der vorliegenden Arbeit eingesetzt[33].

Obwohl MBSE über ein großes Potential zur Steigerung der Entwicklungseffizienz verfügt, hat es sich in der industriellen Produktentwicklungspraxis trotz diverser Bemühungen durch Pilotprojekte[34] noch nicht verbreitet durchsetzen können[35]. Dieser Sachverhalt beruht insbesondere auf der Herausforderung für Entwickler, den Transfer weg von einer geometrie- und bauteilbehafteten hin zu einer abstrahierten, funktionszentrierten Denkweise zu bewältigen[36], sowie dem Bedarf nach einer bis heute nicht hinreichend vorhandenen durchgängigen Methoden- und Werkzeugunterstützung[37]. CLOUTIER UND BONE befragten 2010 in einer von der OMG beauftragten Studie SysML-Anwender nach dem Nutzen im Verhältnis zum Aufwand sowie nach Verbesserungspotentialen der Modellierungssprache[38]. Sie identifizierten zwar ein hohes Potential der SysML, einen Beitrag zur Steigerung der Effizienz in der Produktentwicklung leisten zu können, jedoch auch zahlreiche offene Fragestellungen und Weiterentwicklungspotentiale.

Die breite Mehrheit der zahlreichen Publikationen zum Thema Model-Based Systems Engineering befasst sich mit softwarelastigen Systemen und eingebetteten Systemen (Elektrik-/Elektronik). Dies begründet sich auf den Ursprüngen des MBSE in der objektorientierten Softwareentwicklung mit der Unified Modeling Language[39]. Auf deren Basis sind bereits zahlreiche Modellierungssprachen in industriegetriebenen Projektkonsortien entstanden, die verbreitet in der industriellen Produktentwicklung Anwendung finden. Beispiele hierfür sind AUTOSAR, EAST-ADL oder MARTE[40]. Auch die SysML, die zur ganzheitlichen Modellierung multidisziplinäre Systeme entwickelt wurde, basiert auf der UML. Da sie seit ihrer Erstvorstellung in 2007 noch immer keine verbreitete Anwendung findet, liegt der Hypothese nahe, dass

---

[31] Valilei und Houshmand (2009)

[32] Alt (2012), Shah et al. (2009)

[33] Die Begründung für die Wahl der SysML als Werkzeug wird in Kapitel 4.1.1 formuliert

[34] beispielweise Alt (2009), Andersson et al. (2009), Piques und Andrianarison (2012), Becker et al. (2013)

[35] Birkhofer (2011), Kasser (2010), Tazir (2011), Tazir (2012)

[36] Insbesondere Konstrukteure verwenden primär geometrisch behaftete Produktrepräsentationen, auch bereits sehr früh in der Ideenfindung und Prinzipmodellierung, vgl. Hacker et al. (2002), Eckert et al. (2010), Hannah et al. (2012)

[37] Vgl. Broy et al. (2010)

[38] Cloutier und Bone (2010)

[39] OMG UML (2011a), OMG UML (2011b)

[40] AUTOSAR (2011), ATESST2 Consortium (2010a), ATESST2 Consortium (2010b), OMG MARTE (2011)

Entwickler mit weniger IT-affiner Ausbildung größere Schwierigkeiten bei der Anwendung objektorientierter Modellierungssprachen haben.

Die vorliegende Arbeit adressiert Herausforderungen, die sich aus der Anwendung der SysML für die Modellierung multidisziplinärer Systeme ergeben. Der Betrachtungsschwerpunkt der Anwendung der Sprache liegt hierbei auf klassischen Maschinenbaudisziplinen wie der Konstruktion sowie ferner Personen mit betriebswirtschaftlicher Ausbildung, welche oft mit Management-Verantwortlichkeiten in der industriellen Produktentwicklung betraut sind. Insbesondere werden die Anwendbarkeit und die Akzeptanz der SysML evaluiert und daraus Anforderungen an eine anwendergerechtere Modellierungstechnik abgeleitet. Darauf basierend werden das Zielsystem und das Handlungssystem zur Entwicklung einer neuen Technik zur modellbasierten Systementwicklung, bestehend aus den folgenden zentralen Elementen, in den Kapiteln 4.1 und 4.2 vorgestellt:

- Basisdefinition zentraler Begriffe und deren semantischer Zusammenhänge zur fachdisziplinübergreifenden Beschreibung von Ziel- und Objektsystem technischer Systeme.
- Bereitstellung einer Beschreibungsmethodik der initialen Spezifikation des Zielsystems und seiner dynamischen Entwicklung im Verlauf des Handlungssystems der Produktentwicklung sowie seiner Wechselwirkungen mit dem entstehenden Objektsystem.
- Applikation der Konzepte des Contact & Channel – Ansatzes (C&C²-A) nach ALBERS[41] in der SysML als Werkzeug zur funktionsbasierten Beschreibung der Ergebnisse aus der Analyse und der Synthese technischer Systeme[42].
- Definition eines Anwendungsframeworks für den flexiblen Einsatz der Modellierungssprache, angelehnt an Konzepte wie bspw. den fraktalen Charakter des integrierten Produktentstehungsmodells (iPeM) nach ALBERS[43].
- Vorstellung eines Konzepts zur Integration der Modellierungssprache in eine durchgängig IT-gestützte Entwicklungsumgebung.

In Kapitel 4.3 wird ein Überblick über die resultierenden Elemente des Objektsystems gegeben, die in den nachfolgenden Kapiteln ausführlich vorgestellt werden. Zunächst wird in Kapitel 5 die Basisdefinition einer gemeinsamen Sprache der

[41] Albers et al. (2008a), Albers et al. (2008b), Albers et al. (2009a), Alink (2010)

[42] Dies umfasst insbesondere die funktionsbasierte Modellierung mit integrierter Betrachtung funktionsrelevanter physischer Merkmale sowie die Modellierung von physischen Strukturen eines Systems unter Berücksichtigung ihrer Dynamik

[43] Albers (2010), Albers und Braun (2011a), Albers und Braun (2011b)

Produktentwicklung formuliert, die gleichzeitig die syntaktische und semantische Grundlage der Modellierungstechnik darstellt. In Kapitel 6 wird das Anwendungsframework der Modellierungstechnik vorgestellt, gefolgt von der detaillierten Einführung der Aspekte der Ziel- und Objektsystemmodellierung an Beispielen aus den Evaluationsprojekten im Bereich der Hybridantriebsstrangentwicklung. Darüber hinaus werden Möglichkeiten und Ansätze aufgezeigt, die vorgestellte Methode ganzheitlich und nachhaltig in industrielle Entwicklungsumgebungen einzubetten. Hierzu wird neben den Ergebnissen durchgeführter Vorarbeiten zu Modellkopplungen ein Konzept für die Integration der Modellierungstechnik in einen übergeordneten, funktionsbasierten Produktportfolio-Lenkungsprozess vorgestellt. Durch die Kopplung der SysML mit Ontologiesprachen soll die semantische Kopplung von Modellelementen mit dem Ziel ermöglicht werden, verschiedene Produktmodelle untereinander sowie mit anderen Modellen zu vernetzen und management-relevante Informationen zu extrahieren. Damit wird beispielsweise die Bildung neuer, auf das Management zugeschnittener Sichten zur Unterstützung strategischer Entscheidungen ermöglicht, wie sie in Produktportfolio-Lenkungsprozessen erforderlich sind. Anschließend werden diese Ergebnisse in ein Konzept eines Rahmenwerks durchgängiger IT-basierter Modellkopplung integriert, das gleichzeitig die langfristige Vision einer modellbasierten Entwicklungsumgebung darstellt.

Die Modellierungstechnik wurde in verschiedenen Anwendungsszenarien validiert, die in Kapitel 7 vorgestellt und diskutiert werden. Dies umfasst insbesondere ihre Validierung in der Modellierung von Gesamtfahrzeug-Entwicklungsumgebungen und des Mensch-Maschine-Interfaces eines Rollenprüfstands-Automatisierungssystems, die im Rahmen eines Forschungsprojektes und mehrerer Industriepilotprojekte durchgeführt wurde. Ferner trug ein weiteres Anwendungsszenario im Bereich der funktionalen Modellierung von Laserschneidmaschinen zur Validierung bei.

Die wesentlichen Fortschritte und Erkenntnisse aus der vorliegenden Arbeit werden in Kapitel 8 zusammengefasst und darauf basierend mögliche wissenschaftliche Anknüpfungspunkte in einem Ausblick auf zukünftige Forschungsthemen erörtert.

2 Grundlagen und Stand der Forschung

Moderne technische Produkte charakterisiert durch den rasanten technischen Fortschritt eine hohe Multidisziplinarität, deren Handhabung nur ein systemischer Ansatz gerecht werden kann. Dessen Ziel ist der Aufbau eines ganzheitlichen Systemverständnisses unter Einbezug sämtlicher Wechselwirkungen und Randbedingungen, die sich aus der Interaktion des technischen Systems mit dem Menschen und technischen Nachbarsystemen sowie aus den Interaktionen der Systemkomponenten untereinander ergeben. Zu diesem Zweck wurden in der jüngeren Vergangenheit zahlreiche theoretische Vorgehensmodelle und Modellierungssprachen mit unterschiedlichen Zielsetzungen entwickelt. Dieses Kapitel stellt zunächst die Grundlagen der System- und Modelltheorie vor und definiert zentrale Grundbegriffe. Auf Basis dieses Verständnisses werden die bedeutendsten wissenschaftlichen Ansätze und Vorgehensmodelle sowie Modellierungssprachen vorgestellt und deren Potentiale und Herausforderungen diskutiert.

2.1 Systemtheorie der Technik und Funktionsbegriff

Erste Erwähnung systemischen Denkens findet sich bereits in der griechischen Philosophie, wo bereits zwischen einer reinen Zusammenfassung einer Vielfalt an Elementen zu einer Menge („pan“) und deren Anordnung in einer Ganzheit („holon“) unterschieden wurde. Dies entspricht in Grundzügen bereits dem strukturalen Systemkonzept. Bedeutende Erkenntnisse gelangen dem Biologen BERTALANFFY Mitte des 20. Jahrhunderts in seiner *Allgemeinen Systemlehre*: „Die Eigenschaften und Verhaltensweisen höherer Ebenen sind nicht durch die Summation der Eigenschaften und Verhaltensweisen ihrer Bestandteile erklärbar, solange man diese isoliert betrachtet. Wenn wir jedoch das Ensemble der Bestandteile und Relationen kennen, die zwischen ihnen bestehen, dann sind die höheren Ebenen von den Bestandteilen ableitbar“[44]. Weitere Wurzeln sind in der Kybernetik, verschiedenen Ansätzen zur Verwissenschaftlichung praktischen Problemlösens sowie der Mathematik zu finden.[45]

Eine mathematische Beschreibung von Bertalanffys Konzept offener Systeme, also jener, die in Interaktion mit ihrer Umwelt stehen, zeigt Gleichung (2.1).

[44] Zitiert von Ropohl (2009) aus Bertalanffy (1949)
[45] Ropohl (2009)

$$\frac{\partial Q_i}{\partial t} = T_i + P_i \tag{2.1}$$

Q_i bezeichnet darin das i-te Element im System, T_i seine Transsportgeschwindigkeit und P_i seine die Produktions- bzw. Vernichtungsrate, jeweils an einem konkreten Punkt im Raum. Die zeitunabhängige Form dieser Gleichung ergibt sich durch Ableitung zu Gleichung (2.2).

$$0 = T_i + P_i \tag{2.2}$$

Diese beiden einfachen Gleichungen beschreiben sowohl die Erhaltungsgesetze der Physik als auch die dynamische Stabilität offener Systeme[46].

EHRLENSPIEL zeigt eine grafische Darstellung zur Definition des Systembegriffs (Bild 2-1), ergänzt um folgende Definition: „Ein System besteht aus einer Menge von Elementen (Teilsystemen), die Eigenschaften besitzen und durch Beziehungen miteinander verknüpft sind. Ein System wird durch eine Systemgrenze von der Umgebung abgegrenzt und steht mit ihr durch Ein- und Ausgangsgrößen in Beziehung (offenes System). Die Funktion eines Systems kann durch den Unterschied der dem Zweck entsprechenden Ein- und Ausgangsgrößen beschrieben werden (…). Nach ihrem Verhalten können statische und dynamische Systeme unterschieden werden.“[47]

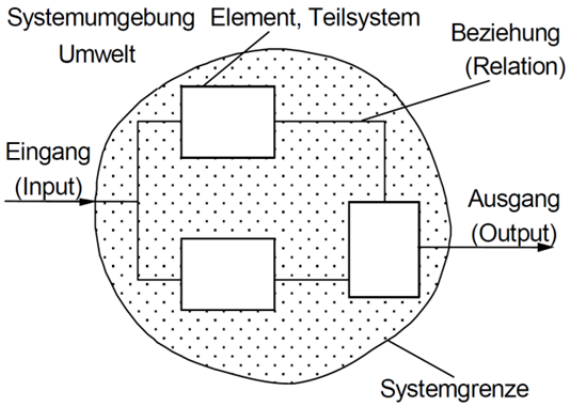


Bild 2-1: Darstellung eines Systems nach EHRLENSPIEL[47]

ROPOHL[48] unterscheidet drei Aspekte, welche das Systemkonzept umfasst: das funktionale, das strukturale und das hierarchische Konzept (Bild 2-2).

[46] Hitchins (2007)
[47] Ehrlenspiel (2009), S. 19f
[48] Ropohl (2009)

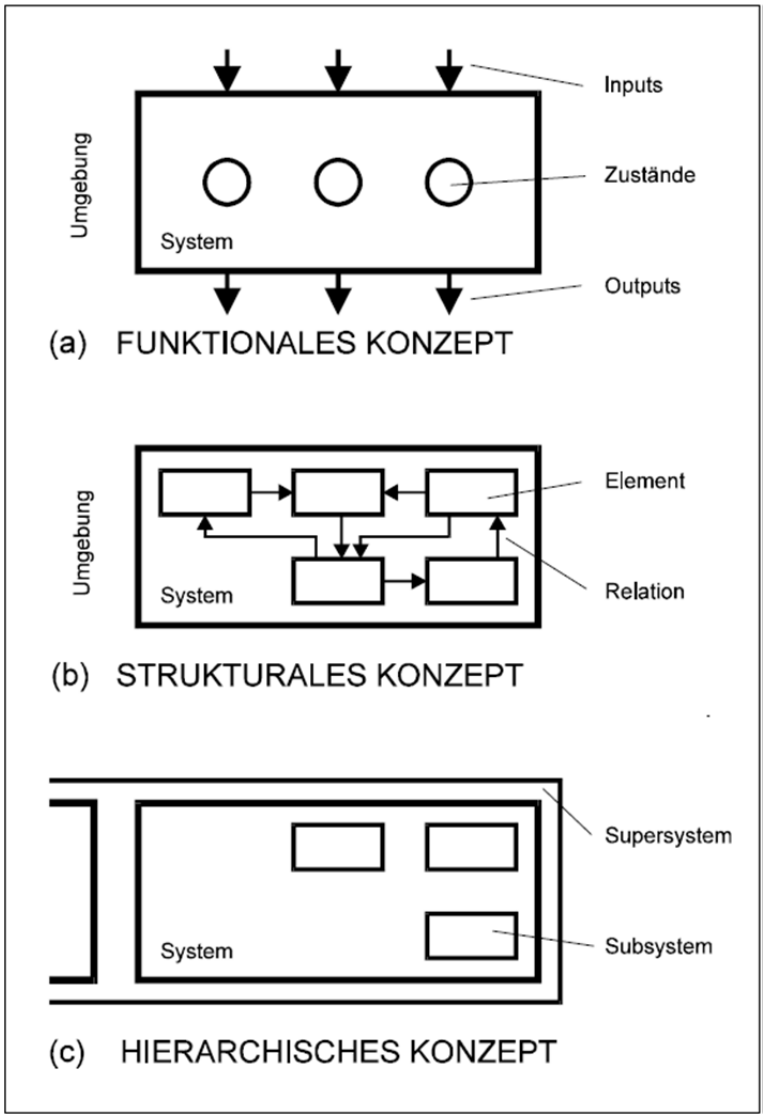


Bild 2-2: Konzepte der Systemtheorie

Das *funktionale Konzept* stellt das System als Black Box dar, interagiert durch Eingangsgrößen und Ausgangsgrößen mit seiner Umwelt und beschreibt seine Verfassung durch Zustände. Dabei sieht das Konzept ausdrücklich von der materiellen Konkretisierung und der Beschreibung des inneren Aufbaus des Systems ab.

Das *strukturale Konzept* betrachtet das System als eine Ganzheit miteinander verknüpfter Elemente. Damit geht einher, dass die Elemente nicht losgelöst von ihrem Kontext, sondern in ihrer Interdependenz mit anderen Elementen innerhalb des Systems betrachtet werden müssen. Dabei rufen einerseits die Vielfalt möglicher Beziehungsgeflechte und andererseits die Beschaffenheit der Elemente charakteristische Systemeigenschaften hervor.

Das *hierarchische Konzept* unterscheidet mehrere Stufen von Ganzheit und Teilen, wobei die Ganzheit einer Betrachtungsebene immer ein Teil der darüber liegenden Ebene ist. MESAROVIC[49] beschreibt eine Art des Erkenntnisgewinns durch Bewegung abwärts durch die hierarchischen Schichten („strata“) zum Erhalt einer detaillierteren Erklärung des Systems, während durch Aufwärtsbewegung ein tieferes Verständnis seiner Bedeutung gewonnen wird. ROPOHL definiert dabei rangniedrigste Sachsysteme als „die elementaren technischen Bauteile“ mit der Begründung „[…] würde man diese wiederum in Subsysteme zerlegen, hätte man es nicht mehr mit technischen, sondern mit mikrophysikalischen bzw. chemischen Sachverhalten zu tun.“[50]

Diese drei Konzepte schließen einander nicht aus, sondern wurden in der Modellbildung von Systemen vielfach kombiniert aufgegriffen[51]. So kombiniert auch ROPOHL alle drei Aspekte in seiner Definition des Systembegriffs, die auch in der vorliegenden Arbeit übernommen wird:

Definition 1: System

> Ein ***System*** ist das Modell einer Ganzheit, die (a) Beziehungen zwischen Attributen (Inputs, Outputs, Zustände etc.) aufweist, die (b) aus miteinander verknüpften Teilen bzw. Subsystemen besteht, und die (c) von ihrer Umgebung bzw. von einem Supersystem abgegrenzt wird.

Die Buchstaben in Klammern beziehen sich auf die 3 zentralen Charakteristika eines Systems, das funktionale Konzept, das strukturale Konzept sowie das hierarchische Konzept in (vgl. Bild 2-2)[52]. Weiterhin klassifiziert ROPOHL verschiedene Arten von Systemen, wobei das Technische System wie folgt charakterisiert wird[53]:

Definition 2: Technisches System

> Ein ***Technisches System*** (Sachsystem) ist ein künstlich entstandenes, zweckorientiertes, offenes und dynamisches System.

---

[49] Mesarovic (1970)

[50] Ropohl (1975), S. 26

[51] Erst die Kombination dieser Aspekte ermöglicht die ganzheitliche Beschreibung von Systemen, weshalb verschiedene Modellierungssprachen eigene Diagramme für jeden dieser Aspekte anbieten (vgl. Kapitel 2.3)

[52] Ropohl (2009), S. 77

[53] Ropohl (1975), S. 34

Ein technisches System dient seinem Anwender zur Unterstützung bei der Durchführung einer gewünschten Handlung. In diesem Zusammenhang spricht man oft vom Zweck eines (technischen) Systems, welcher durch eine menschliche Instanz festgelegt wird und durch die Ausführung von Funktionen erfüllt wird. Daraus leitet sich die folgende Definition ab:

Definition 3: Zweck

> Ein ***Zweck*** eines technischen Systems ist eine von einer menschlichen Instanz erwartete Funktion in Interaktion mit seiner Umgebung mit Hilfe bestimmter Inputs und unter bestimmten Randbedingungen.

Eine „erwartete Funktion“ entspricht dem *teleologischen Funktionsbegriff* nach ROPOHL und deckt sich darüber hinaus auch nach der intendierten Beschreibung des 5-Key-Concepts nach VERMAAS, wobei diese erwartete oder beabsichtigte Funktion hier mit einer umweltzentrierten Sicht beschrieben wird[54].

Im Gegensatz dazu beschreibt der *deskriptive Funktionsbegriff* die Zusammenhänge zwischen den Eingangs- und Ausgangsgrößen durch deren Transformation innerhalb des Systems und gibt damit an, was das System tatsächlich leistet[55]. Diese systemzentrierte Sicht auf den Funktionsbegriff wird ebenfalls im 5-Key-Concept nach VERMAAS aufgegriffen, der darunter die physikalisch-/chemische Beschreibung von Funktionen eines Systems versteht (vgl. Bild 2-3).

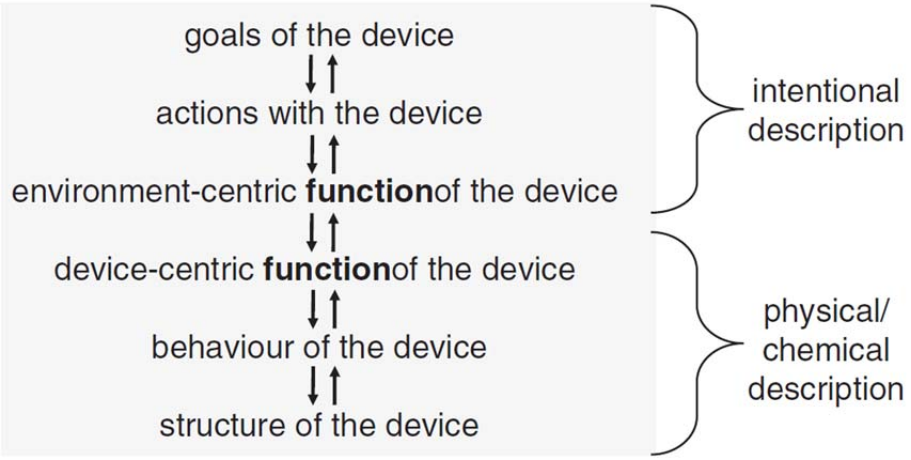


Bild 2-3: 5-Key-Conzept nach VERMAAS[56]

[54] Vgl. Vermaas (2010)
[55] Ropohl (2009), S. 80 sowie S. 126
[56] Aus Vermaas (2010)

Diese differenzierte Sicht auf den Funktionsbegriff mit den jeweiligen Auslegungen entweder aus Sicht der beabsichtigten Funktion eines Anwenders oder der im System realisierten Funktion wurde auch von ECKERT ET AL. sowie ALINK als die am weitesten verbreitete Differenzierung durch Produktentwickler in Studien zum Funktionsverständnis identifiziert[57]. Im Kontext dieser Arbeit wird diese Differenzierung ebenfalls in Form von *Ziel-Funktionen* und *Ist-Funktionen* in Form der nachfolgenden Definitionen der Arten technischer Funktionen aufgegriffen.

Definition 4: Technische Funktion

> Eine ***Technische Funktion*** beschreibt den Zusammenhang zwischen Inputs (Stoff, Energie, Information) - den ***Ursachen*** – und Outputs – den ***Wirkungen***, der auch gleichzeitig bidirektional in Form von ***Wechselwirkungen*** bestehen kann.

Die vorliegende Definition lehnt sich an das in der Literatur häufig aufgegriffene, übergeordnete Verständnis des Funktionsbegriffs an, das eine Funktion als gewollten Zusammenhang bzw. Abhängigkeit zwischen Ein- und Ausgangsgrößen mit Eigenschaftsänderung der Zustandsgrößen definiert[58]. Eingangsgrößen werden in der Modellbildung meist als Input bezeichnet, die Funktionen auslösen. Entsprechend werden Ausgangsgrößen als Output bezeichnet, sie stehen für die Wirkung einer Funktion. Darüber hinaus wurde hier bewusst der Begriff Wechselwirkung aufgenommen, der für eine bidirektionale Abhängigkeit steht. Wechselwirkungen treten vor allem bei der Betrachtung von Funktionen mechanischer Systeme wie beispielsweise Kräfte- und Momentengleichgewichten auf[59].

Aufgrund der nachfolgenden Differenzierung zwischen gewollten und tatsächlichen (möglicherweise ungewollten) Funktionen wurde in dieser Basisdefinition auf die Einordnung hinsichtlich Absicht, Aufgabenerfüllung oder Zweckkonformität bewusst verzichtet, ähnlich wie es auch MATTHIESEN konkludierte[60]. Die hervorgehobenen Begriffe sind Teil der Basisdefinition einer Sprache für den Kontext der Modellbildung technischer Systeme, wofür in Kapitel 5 eigene Definitionen herausgearbeitet werden. Diese Definition einer Technischen Funktion wird in Anlehnung an die zuvor vorgestellten Konzepte weiter in zwei Arten technischer Funktionen konkretisiert:

---

[57] Eckert et al. (2011), Alink (2010)

[58] Bspw.: VDI 2221 (1993), Pahl et al. (2006), Ehrlenspiel (2009), Ponn und Lindemann (2011)

[59] Die Begriffe, Ursache, Wirkung und Wechselwirkung werden in Kapitel 5.2 ausführlich diskutiert und definiert.

[60] Matthiesen (2002): „Eine Funktion stellt einen eindeutigen Bezug zwischen einer Eingangs- und einer Ausgangsgröße dar.“

Definition 4.1: Ziel-Funktion

> Eine ***Ziel-Funktion*** als Teil des Zielsystems beschreibt einen ***Zweck*** bzw. eine Aufgabe des zu entwickelnden technischen Systems aus Sicht des Anwenders (als Teil der Systemumgebung).

Mit dieser die Basisdefinition (siehe Definition 4) erweiternden Definition einer *Ziel-Funktion* werden der zuvor eingeführte teleologische Funktionsbegriff nach ROPOHL bzw. die intendierte Beschreibung von Funktion nach VERMAAS aufgegriffen. Hiermit werden die von Anwendern beabsichtigten Funktionen, folglich der Zweck bzw. die Aufgaben eines technischen Systems als Teil des Zielsystems spezifiziert.

Technische Systeme können jedoch auch ein Verhalten äußern, das vom Anwender oder Entwickler nicht beabsichtigt wurde. Dieses Verhalten resultiert aus Wirkungen realisierter Funktionen, die in dieser Form oder Ausprägung nicht gewünscht bzw. unvermeidbar sind[61]. Dennoch liegt diesen Wirkungen eine Funktion zugrunde, die Inputs (ggf. in einem Teilspektrum) durch zugrundeliegende Effekte sowie Merkmale und Eigenschaften der ausführenden *Wirkstruktur*[62] in diese ungewollten Outputs wandelt. Daraus leitet sich die folgende Definition ab, die auf dem deskriptiven Funktionsbegriff nach ROPOHL bzw. der systemzentrierten (physikalisch/chemischen) Beschreibung nach VERMAAS basiert und eine *Ist-Funktion* als Teil des Objektsystems beschreibt:

Definition 4.2: Ist-Funktion

> Eine ***Ist-Funktion*** als Teil des Objektsystems beschreibt das in ihrer ***Wirkstruktur*** stattfindende Transformationsprinzip von Inputs in Outputs durch zugrundeliegende ***Effekte*** sowie ***funktionsrelevante Merkmale und Eigenschaften***.

Der Definition ist zu entnehmen, dass für die Beschreibung einer Ist-Funktion neben dem Verhältnis von Inputs zu Outputs insbesondere die zugrundeliegenden naturwissenschaftlichen Effekte sowie funktionsrelevante Merkmale und Eigenschaften einbezogen werden. Diese sind erforderlich, um das gewählte Funktionsprinzip für die Realisierung des geforderten Zusammenhangs zwischen Inputs und Outputs zu modellieren. Gleichzeitig wird hiermit deutlich, dass Ist-

[61] Entsprechend bspw. der Definition von „unerwünschte Funktion“ in VDI-2803 (1996), S. 3
[62] Zur Definition von Wirkstruktur vgl. Kapitel 2.4.9, die semantische Einordnung erfolgt in Kapitel 5.2

Funktionen nie losgelöst von der realisierenden Struktur und deren Gestalt zu betrachten sind. So erkennt beispielsweise ROPOHL zwischen Funktion und Struktur eines Systems ein bestehendes „[…] formales Dualitätsverhältnis, weil sie mit den gleichen Termen spiegelbildliche Zusammenhänge beschreiben“[63]. HUBKA UND EDER merken zu diesem Entwicklungsschritt an: „Somit bilden die Beziehungen zwischen Funktion (Verhalten) und Struktur die eigentliche Problematik der technischen Wissenschaften“[64]. Die Notwendigkeit einer integrativen Betrachtung von Funktion und Gestalt ist auch ein Kernkonzept des Contact & Channel – Ansatzes (C&C²-A) nach ALBERS[65], der in Kapitel 2.4.9 vorgestellt wird. Er stellt gleichzeitig die Basis für die funktionszentrierte Modellierung von Systemen in dieser Arbeit dar.

Aus den Ansätzen der Reflektion von Beziehungskonflikten zwischen Theorie und Praxis bei der Analyse und Synthese von technischen Systemen hat sich in den 50er Jahren des letzten Jahrhunderts ein Paradigma auf Basis der Systemtheorie unter dem Namen *Systems Engineering* etabliert. Das Forschungsfeld hat durch die rasante technische Entwicklung und einhergehende Steigerung der Komplexität der zu entwickelnden technischen Systeme in den letzten Jahrzehnten enorm an Bedeutung für die Produktentwicklung gewonnen.

2.2 Systems Engineering

Die Anfänge des modernen Systems Engineering lassen sich auf die Technik des Telekommunikationsunternehmens Bell Laboratories bei der Analyse und Planung großer Telefonnetze und Bemühungen des britischen und US-amerikanischen Militärs bei der Analyse von Luftverteidigungssystemen zurückführen[66]. Die erste bedeutende Veröffentlichung ist das Buch „A Methodology for Systems Engineering“ von ARTHUR DAVID HALL[67] aus dem Jahre 1962. 1990 wurde in den USA die NCOSE – National Council on Systems Engineering mit dem Ziel gegründet, die Entwicklung des Systems Engineering in den USA voranzutreiben. Fünf Jahre später wurde die Organisation in INCOSE – International Council on Systems Engineering umbenannt, um dem internationalen Interesse der Förderung dieser Disziplin Rechnung zu tragen. Ende 2010 zählte die als offizielle Dachorganisation des Systems Engineering geltende Vereinigung aus Forschung und Industrie bereits 7836 Mitglieder und entwickelt zahlreiche Standards für die Anwendung des SE[68]. Das

[63] Ropohl (2009), S. 80
[64] Hubka und Eder (1992), S. 10 und S. 92
[65] Albers et al. (2008a), Albers et al. (2008b), Albers et al. (2009a), Alink (2010)
[66] Haskins et al. (2011)
[67] Hall (1962)
[68] INCOSE Website (2012)

INCOSE ist in nationalen Verbänden organisiert. Der deutsche Ableger ist die GfSE – Gesellschaft für Systems Engineering[69].

Mit der Einführung des internationalen Standards ISO/IEC 15288 „Systems and software engineering - System life cycle processes“ im Jahr 2002 wurde ein Konsens für ein übergeordnetes Vorgehensmodell im Systems Engineering offiziell formuliert und als zu bevorzugende Methodik, insbesondere für unternehmensübergreifende Kooperationen, in der System- und Prozessentwicklung definiert[70].

Das INCOSE definiert Systems Engineering wie folgt: „Systems engineering is an interdisciplinary approach and means to enable the realization of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, and then proceeding with design synthesis and system validation while considering the complete problem: operations, cost and schedule, performance, training and support, test, manufacturing, and disposal. SE considers both the business and the technical needs of all customers with the goal of providing a quality product that meets the user needs.“[71]. In dieser Definition sind alle wesentlichen Aspekte der Disziplin benannt, die sich zusammenfassend mit der effizienten, disziplinübergreifenden Entwicklung von komplexen Systemen beschäftigt. Das Ziel von Systems Engineering ist, das Risiko bei der Entscheidung zur Einführung neuer Systeme oder der Veränderung komplexer Systeme, insbesondere in frühen Entwicklungsphasen, zu minimieren. Dort werden laut einer Studie des US-Verteidigungsministeriums nach 20% bereits aufgewendeten Entwicklungskosten Entscheidungen zur Festlegung von 80% der Gesamtentwicklungskosten getroffen, was die enorme Bedeutung dieser frühen Konzeptphasen hervorhebt. Die Behebung von Fehlern ist dort noch deutlich kostengünstiger durchführbar als deren Behebung in späteren Entwicklungsphasen[72].

HITCHINS definiert Systems Engineering als “[…] the art and science of creating whole solutions to complex problems“ und beschreibt zur Erläuterung in diesem Kontext drei Wege, Probleme anzugehen[73]:

- Probleme beheben: Ein Problem zu beheben bedeutet, eine zufriedenstellende Antwort zu finden, die „gut genug“ ist. Dies ist die Variante mit dem geringsten Aufwand.

[69] GfSE Website (2012)
[70] ISO-15288 (2008)
[71] Haskins et al. (2011), S. 7
[72] Haskins et al. (2011), S. 14
[73] Hitchins (2007)

- Probleme auflösen: Ein Problem aufzulösen bedeutet, die Situation derart zu ändern, dass das Problem verschwindet, also die “Torpfosten zu verschieben”.
- Probleme lösen: Ein Problem lösen bedeutet, die korrekte Antwort zu finden, ähnlich bei der Lösung einer Gleichung. Dieser Weg bedeutet den größten Aufwand, denn er erfordert umfassendes Wissen über das System und die Ursache des Problems, um diese beseitigen zu können. Er ist auf der anderen Seite aber auch der nachhaltigste Weg und daher Ziel des Systems Engineering.

Eine ganzheitliche Herangehensweise an Probleme mit dem Ziel der Beseitigung ihrer Ursache ist insbesondere bei modernen technischen Systemen ein hochkomplexer Prozess. Die Modellbildung eines technischen Systems – oder anders formuliert, die modellbasierte Systementwicklung ist eine hilfreiche Technik zur Reduktion der Systemkomplexität auf relevante Aspekte und stellt auch die Grundlage der in dieser Arbeit vorgestellten Modellierungstechnik dar. Da Systeme auf verschiedenen Ebenen betrachtet werden können, stellt HITCHINS ein 5-Ebenen-Modell vor, das eine grobe Einordnung der Anwendungsebene von Systems Engineering (SE) erlaubt[74]. Dabei ist jede Ebene integraler Bestandteil der jeweils darüber liegenden Ebene.

1) Produkt SE: Diese Ebene betrifft das zu entwickelnde Produkt. Oftmals sind Verständnis und Wahrnehmung von SE auf diese Ebene beschränkt.

2) Projekt SE: Hierunter fallen Referenzmodelle (z.B. das V-Modell XT der Softwareentwicklung[75]) und Vorgehensweisen (z.B. Problemlösungsprozesse wie SIMILAR[76]) zur Planung und Steuerung von Entwicklungsprozessen.

3) Unternehmensbezogenes SE: Auf dieser Ebene findet die Wertschöpfung in einem Unternehmen statt. Insbesondere werden hier parallel laufende oder überlappende Projekte koordiniert.

4) Industriebezogenes SE: Hier steht das Zusammenwirken der einzelnen Unternehmen in (globalen) Wertschöpfungsnetzen im Fokus.

5) Sozio-ökonomisches SE: Gegenstand dieser Ebene ist das Verhältnis eines technischen Systems zu den beeinflussenden Interessensvertretern (Mitarbeiter, Markt, Kapitalgeber, Behörden etc.).

---

[74] Hitchins (2007)
[75] Friedrich et al. (2009), vgl. auch Kapitel 2.9.4
[76] Bahill und Gissing (1998), vgl. auch Kapitel 2.9.2

Die vorliegende Arbeit fokussiert sich auf die 1. Ebene (Produkt) und 2. Ebene (Prozess), betrachtet jedoch auch die Einflüsse aus übergeordneten Ebenen, beispielsweise bei der Vorstellung eines Konzepts für einen Produktportfolio-Lenkungsprozess in Kapitel 6.5.3. Bild 2-4 zeigt ein typisches Systems Engineering Paradigma[77] auf Produktebene mit den beeinflussenden Artefakten.

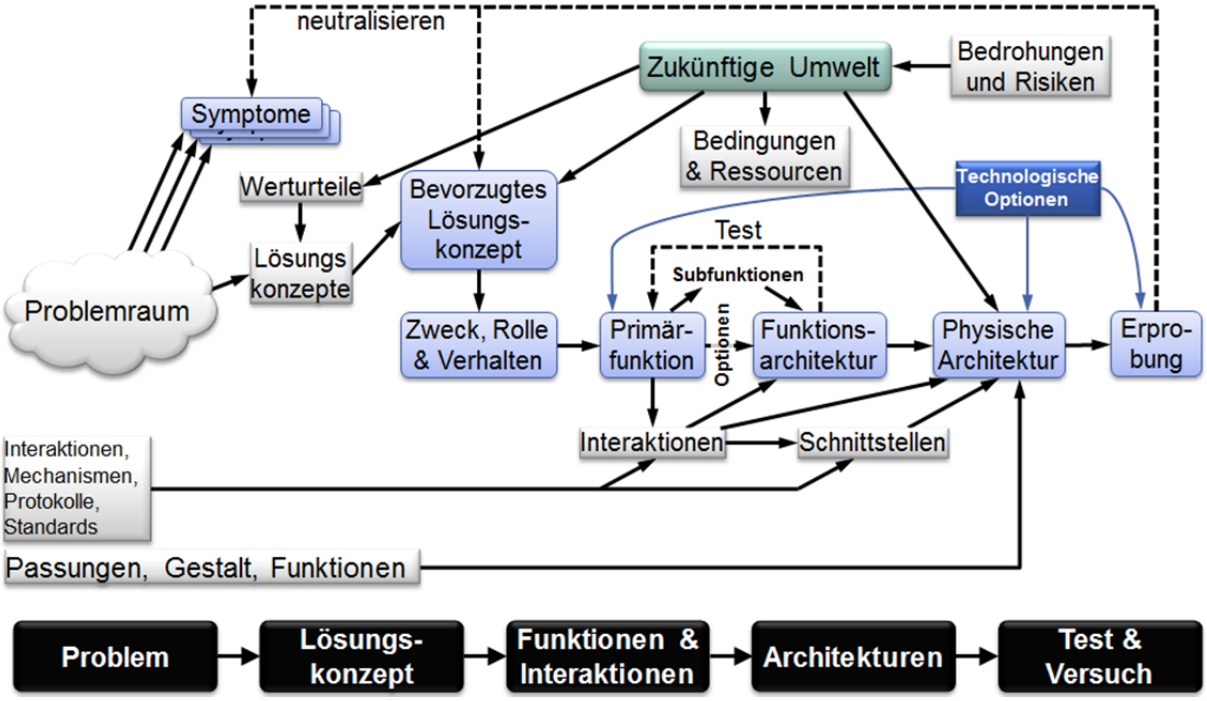


Bild 2-4: Typisches Ebene 1-Systems Engineering-Paradigma (ins Deutsche übersetzt aus HITCHINS[78])

Die in der Grafik aufgeführten, generischen Systembeschreibungsaktivitäten (blau), die entsprechenden Entwicklungsprozessschritte (schwarz) und beeinflussenden Informationen (weiß) vereinen zahlreiche allgemeine Konstrukte der Modellbildung von technischen Systemen, die sich in vielen aktuellen Ansätzen wiederfinden. Das folgende Kapitel beleuchtet den Stand der Forschung im Bereich des Model-Based Systems Engineering, beginnend mit wissenschaftstheoretischen Ansätzen und gefolgt von konkreten Implementierungen in Form von Modellierungssprachen, und diskutiert Stärken und Schwächen der vorgestellten Ansätze und Technologien.

2.3 Model-Based Systems Engineering

Der Wortherkunft nach steht der Begriff “Modell” für ein Abbild von etwas bzw. ein Vorbild für etwas sowie eine Repräsentation eines bestimmten Originals.

[77] Paradigma steht hier für eine Referenzmodell oder Muster, allgemein wird der Begriff jedoch generalisierter als wissenschaftliches Denkmuster verstanden (bspw. das Paradigma SE).

[78] Hitchins (2007)

STACHOWIAK[79] definiert drei Hauptmerkmale des allgemeinen Modellbegriffs: das *Abbildungsmerkmal*, das *Verkürzungsmerkmal* und das *Pragmatische Merkmal*. Ersteres besagt, dass Modelle stets Abbildungen[80] oder Repräsentationen natürlicher oder künstlicher Originale sind, die selbst wiederum Modelle sein können. Dabei erfassen Modelle im Allgemeinen nicht alle Attribute des Originals, sondern nur die aus Sicht von Modellerschaffern und/oder Benutzern relevanten (Verkürzungsmerkmal), da sie einem pragmatischen Zweck dienen: sie sind nicht nur Modelle von etwas, sondern für jemanden (erkennendes und/oder handelndes Subjekt) für bestimmte Operationen über einen bestimmten Zeitraum. Daraus leitet sich das zentrale Fragequadrupel für jeden Modellerschaffer vor Beginn der Modellbildung selbst ab: **Wovon** wird ein Modell **für wen**, **wann** und **wozu** gebildet? Das INCOSE definiert Model-Based Systems Engineering als „formalisierte Anwendung von Modellbildung zur Unterstützung der Aktivitäten der Anforderungsdefinition, des Systemdesigns und der Systemanalyse sowie dessen Verifikation und Validierung, beginnend beim konzeptionellen Entwurf sowie über die Entwicklung und weitere Lebenszyklusphasen hinweg“[81].

Im folgenden Kapitel werden bedeutende wissenschaftliche Modellbildungsansätze vorgestellt, die eine zentrale Grundlage für die Entwicklung rechnerbasierter Modellierungssprachen bilden. Die etabliertesten Modellierungssprachen, Technologien und Methoden werden ab Kapitel 2.6 nach einer kurzen Einführung der Grundkonzepte der Objektorientierung in Kapitel 2.5 vorgestellt und hinsichtlich ihrer Stärken und Schwächen diskutiert.

2.4 Wissenschaftliche Produktmodellbildungsansätze

In der Literatur finden sich zahlreiche wissenschaftstheoretische Ansätze zur Modellbildung von Systemen. All diese Ansätze verfolgen die vereinfachte und verkürzte Repräsentation realer Systeme mit dem Ziel, die inhärente Komplexität der beschriebenen Systeme auf ein für den Menschen verständliches und nachvollziehbares Maß zu reduzieren[82]. Ein zentrales Begleitmerkmal dieser Ansätze ist die Schaffung einer Grundlage für die Bildung rechnerverarbeitbarer Modelle, die

---

[79] Stachowiak (1973)

[80] Eine Abbildung meint die gezielte Beschränkung auf eine Teilmenge aller Informationen des modellierten Systems. Hierbei werden für den Modellierer nicht relevante Informationen bewusst weggelassen. Beispielsweise verzichten CAD-Modelle auf die Abbildung des physikalischen Verhaltens ihrer Konstruktionselemente, Mehrkörpersimulationen hingehen verzichten weitgehend auf die Gestaltmodellierung.

[81] Übersetzt aus INCOSE (2007), S. 15

[82] Diesen Ansätzen liegen die Merkmale der Modelltheorie nach STACKOWIAK (vgl. Stachowiak (1973)) zugrunde.

vor allem durch eine formale Spezifikation charakterisiert sind. LOSSACK stellt beispielsweise wissenschaftstheoretische Grundlagen für die rechnerunterstützte Konstruktion vor[83]. Die wichtigsten dort eingeführten Ansätze sowie weitere bedeutende Modellbildungsansätze werden in diesem Kapitel vorgestellt.

2.4.1 GRM (Generic Reference Model)

Das Generic Reference Model (GRM) nach HITCHINS verfolgt das Ziel, entstehende Eigenschaften, Fähigkeiten und Verhaltensweisen beliebiger Systeme technologie- und werkzeugneutral durch die Gesamtheit von Teilen und deren Interaktionen zu beschreiben[84]. Dabei soll das generische Referenzmodell mehr als eine Art Gerüst oder Rahmenwerk dienen, um daraus konkrete Instanzen realer Systeme gestalten zu können. Es beschreibt die externe Sicht auf ein System auf der allgemeinsten Ebene durch die drei Aspekte „Verhalten“ (thinking), „Funktion“ (doing) und „Gestalt“ (being). Eine interne Betrachtung setzt diese Aspekte in Relation zueinander, immer mit dem Ziel einer gewissen Vollständigkeit durch Erwähnung aller notwendigen und hinreichenden Teilaspekte. Beispielsweise implizieren Funktionen immer eine ausführende reale (greifbare) Gestalt (engl. form), die Bestandteil eines auf Aspekten wie Kognition, Stimulation oder Selektion beruhenden dynamischen Verhaltens sind. Das GRM ist mit seiner ganzheitlichen Dekomposition der Aspekte zur Beschreibung von Systemen eine wertvolle Grundlage für die Formalisierung von Informationen.

2.4.2 General Design Theory (GDT)

Die General Design Theory (GDT) ist eine domänenunabhängige Formulierung der Aktivität des Designens von Objekten durch die Verbindung der Zielspezifikation mit der Designlösung. Sie fasst ihre wesentlichen Aussagen in drei Axiomen, daraus abgeleiteten Theoremen und diversen mathematischen Definitionen zusammen[85]. Die Theorie unterscheidet zwischen idealem (vollständigem) und realem (unvollständigem) Wissen über die zu spezifizierenden Objekte (Entitäten), wobei die Anwendbarkeit der Axiome ein ideales Wissen voraussetzt, was als das „Paradoxon der General Design Theory“ bezeichnet wird[86]. Das ideale Wissen wird durch zwei topologische Räume repräsentiert, dem funktionalen Raum (function space) und dem Attributraum (attribute space) sowie einer Vernetzung der beiden Räume. Dabei werden konkrete Objekte in Objektdomänen (entity sets) eingeordnet, und eindeutig klassiert (abstract concepts). Das Ziel der Design-Aktivität ist eine eindeutige

[83] Lossack (2006)
[84] Hitchins (2007)
[85] Yoshikawa (1981)
[86] Kikuchi und Nagasaka (2002)

Zuordnung von Attributen aus dem Attributraum für die Designlösung zu den spezifizierten Funktionen aus dem funktionalen Raum.

Die Natur von Objekten und deren Potential, eine gewünschte Funktionalität zu erreichen, werden in drei Axiomen formuliert. Das **Axiom der Wahrnehmung** (*Axiom of recognition*) besagt, dass jede Entität (Objekt) durch seine Attribute und/oder durch weitere Klassifizierungen eindeutig wahrgenommen und beschrieben und somit von anderen unterschieden werden kann. Dies setzt jedoch voraus, dass eine ausreichende Anzahl an Attributen vorhanden ist, um ein Objekt eindeutig wahrnehmbar von anderen zu unterscheiden, was wiederum sehr viele Attribute erfordern kann. Das **Axiom der Korrespondenz** (*Axiom of correspondence*) besagt, dass eine Objektdomäne (die Gesamtheit aller Objekte einer bestimmten Art – das „entity set“) und seine Repräsentation eindeutig korrespondieren müssen. Auch hieraus folgt, dass die Repräsentation sehr umfangreich – bis hin zu Unendlichkeit – werden kann, um dem Axiom gerecht zu werden. Das **Axiom der Verknüpfung** (*Axiom of Operation*) besagt, dass die Summe aller Klassierungen die Topologie einer Objektdomäne spezifiziert (die Summe aller Quervernetzungen zwischen verschiedenen Objekten, wie beispielsweise das zugehörige Verhalten zur Struktur eines Subsystems)[87].

2.4.3 Universal Design Theory (UDT)

GRABOWSKI ET AL. stellten 1999 die am Institut für Rechneranwendung in Planung und Konstruktion (RPK) entwickelte Universal Design Theory vor, welche konstruktionsrelevante Informationen verschiedener Disziplinen in einer konsistenten, kohärenten und kompakten Form beschreibt[88]. Sie dient als wissenschaftliche Basis zur Förderung der Effizienz und Nachhaltigkeit interdisziplinärer Zusammenarbeit. Im Gegensatz zur General Design Theory (GDT, vgl. Kapitel 2.4.2) umfasst sie nicht nur generisches, disziplinunabhängiges Wissen, sondern auch disziplinspezifisches, konstruktionsrelevantes Wissen, indem sie beispielsweise auch die Schnittstellen verschiedener Konstruktionsdisziplinen beschreibt. Sie verfolgt den Ansatz, einen Kompromiss aus Universalität und praktischer Anwendbarkeit zu etablieren, indem sie einerseits eine generische Design-Sprache bereitstellt und andererseits damit verbundene, konstruktionsrelevante und anwendbare Lösungsmuster beinhaltet. Dies wird ähnlich zu anderen Ansätzen durch entsprechende, grafisch in Diagrammen aufbereitete Partialmodelle realisiert.

---

[87] Kikuchi und Nagasaka (2002)
[88] Grabowski et al. (1999)

2.4.4 Design for X

STÖBER ET AL. beschreiben Design for X (DfX) als „…die Bestrebungen in der Produktentwicklung, die in den meisten Fällen widersprüchlichen Anforderungen an ein zu entwickelndes Produkt gleichzeitig zu berücksichtigen und den bestmöglichsten Kompromiss zwischen ihnen zu finden.“[89]. Die Anfänge dieses Ansatzes finden sich in den 1970er Jahren, wo das Hauptaugenmerk auf dem fertigungs- und montagegerechten Konstruieren lag. Durch den kontinuierlich wachsenden Umfang der Gerechtheiten entstand ein enormes Netzwerk mit häufig unbekannten Wechselwirkungen zwischen den einzelnen Gesichtspunkten. Der DfX-Ansatz dient der Bereitstellung von Entscheidungsgrundlagen hinsichtlich definierter Ziele, Eigenschaften und/oder Lebenszyklusphasen eines Produktes. Die Grundlage zur Einbettung des DfX in Unternehmen und deren Produktentwicklungsprozesse stellt die Integrierte Produktentwicklung durch die ganzheitliche Betrachtung der Wechselwirkungen zwischen Mensch, Methodik, Organisation und Technik dar. Die Beschreibung der für die Produktentwicklung relevanten Informationen geschieht einerseits durch Ziele und Strategien und andererseits durch Produktmerkmale und -eigenschaften. Bei deren Findung soll der Entwickler durch DfX-Richtlinien, -methoden und –werkzeuge unterstützt werden (siehe Bild 2-5).

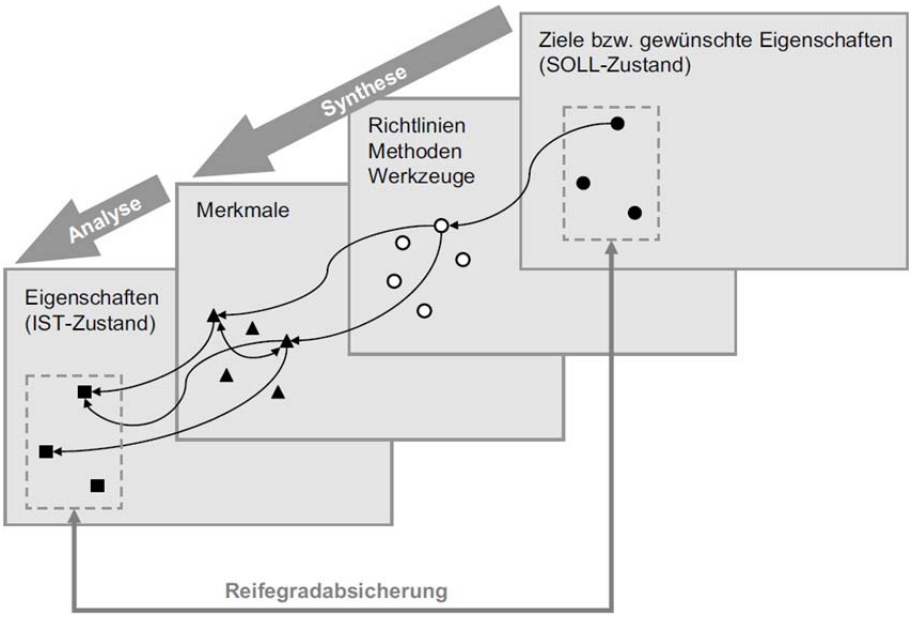


Bild 2-5: Abhängigkeiten zwischen Zielen, Richtlinien, Merkmalen und Eigenschaften[90]

Die Synthese von Zielen, die beispielsweise gewünschte Eigenschaften ausdrücken, wird durch den Einsatz situations- und rollengerechter Richtlinien, Methoden und

[89] Stöber et al. (2009), S. 101
[90] aus Stöber et al. (2009)

Werkzeuge unterstützt. So entstehen zunächst die entsprechenden Merkmale eines Produkts, die im DfX als Gestaltmerkmale verstanden werden und durch den Entwickler festgelegt werden können. Aus einem Geflecht an Abhängigkeiten zwischen diesen Merkmalen resultiert das Produktverhalten, welches somit nicht durch den Entwickler definiert werden kann. Der Findungsprozess der gewünschten Eigenschaften kann durch iterative Variation und Simulation definierter Merkmale erfolgen.

Da der DfX-Ansatz auf die zielgerechte, integrierte Produktentwicklung abzielt, wird er sehr häufig in wissenschaftlichen Publikationen aufgegriffen oder dient als Grundlage für weitere Produktentwicklungsmethoden. Der CPM/PDD-Ansatz nach WEBER[91] (vgl. auch Kapitel 2.4.7) und die softwarebasierte Umsetzung aspektorientierter Visualisierungen von BAUER UND MEERKAMM[92] basieren beispielsweise auf dem DfX-Ansatz. Jedoch existieren keine standardisierten Modellierungssprachen und auch kein standardisiertes Vorgehen für dessen Anwendung, weshalb er auch nur in vereinzelten Softwareprototypen umgesetzt wurde. Mögliche Gründe sind die sehr breite und gleichzeitig generische Definition sowie die häufig divergierende Interpretation von DfX. Der Ansatz dient in der vorliegenden Arbeit daher als eine Forschungsgrundlage, wird jedoch nicht weiter namentlich aufgegriffen.

2.4.5 Axiomatic Design

Axiomatic Design nach SUH ist eine Theorie für das strukturierte Entwerfen von Systemen und kann auf soziotechnische Systeme jeder Art angewandt werden[93]. Sie beschreibt eine iterative Vorgehensweise (*zig-zagging*) zur Entwicklung von Lösungen aus Anforderungen. Hierzu wird zwischen den vier Domänen *Customer Domain*, *Functional Domain*, *Physical Domain* und *Process Domain* unterschieden, die schematisch gemäß Bild 2-6 in Relation zu einander stehen.

[91] Weber (2005), Weber (2007), Weber (2013)
[92] Bauer und Meerkamm (2007)
[93] Suh (1990)

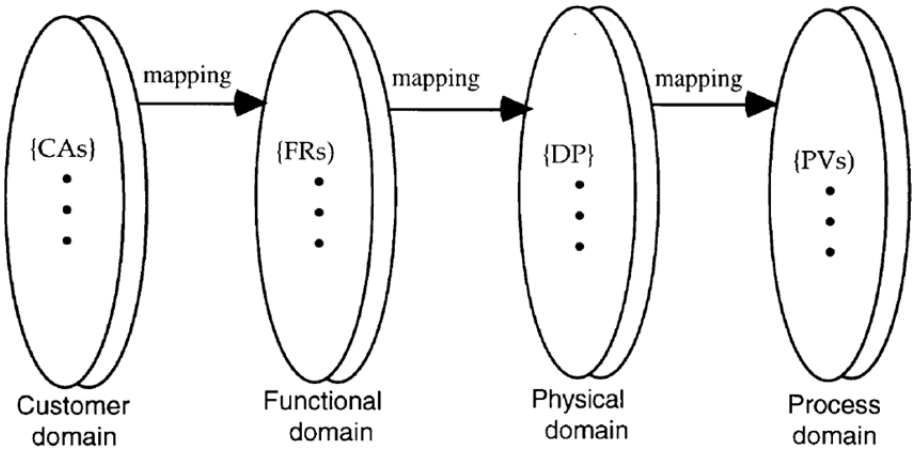


Bild 2-6: Domänen der Design-Welt im Axiomatic Design[94]

Dabei beschreibt die *Customer Domain* (engl. für Kundendomäne) die zu erreichenden Kundenziele, aus denen *Functional Requirements* (FRs, engl. für funktionale Anforderungen) und *constraints* (Cs, engl. für Randbedingungen bzw. zulässige Grenzen) abgeleitet werden. Diese werden durch entsprechende *Design Parameters* (DPs, engl. für Entwurfsparameter) erfüllt, deren Identifikation den eigentlichen Gestaltungsprozess beinhaltet. Schließlich erfolgt die Ableitung des zugehörigen Prozesses in Form von *Process Variables* (PVs, engl. für Prozessvariablen) aus den *Design Parameters*.

Dieser Prozess aus Anforderungsdefinition und *mapping* (engl. für Zuordnung) korrespondierender Lösungen kann durch Matrizen abgebildet werden. Bild 2-7 zeigt das generelle Format einer Design Matrix.

$$\begin{bmatrix} FR_1 \\ FR_2 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} DP_1 \\ DP_2 \end{bmatrix}$$

Bild 2-7: Design Matrix mit 2 Functional Requirements (FR) und 2 Design Parameters (DP)

Werden solche Zuordnungsentscheidungen getroffen, können daraus jeweils neue Anforderungen entstehen, für die wiederum entsprechende Lösungen gefunden werden müssen, wodurch der iterative Charakter des Prozesses, der als *zig-zagging* (engl. für hin- und herspringen) bezeichnet wird, entsteht. Beispielsweise wurde für die funktionale Anforderung „Drehmoment und Drehzahl wandeln“ das Lösungsprinzip „Wandlung über Gangstufen“ ausgewählt. Darauf ergeben sich aber als Voraussetzung für seine Realisierung neue Anforderungen wie beispielsweise ein erforderlicher Bauraum und eine geeignete Schmierung zur Erfüllung der nichtfunktionalen Anforderung „Lebensdauer von 100.000 Betriebsstunden“.

[94] Suh (1998)

Die beiden namensgebenden Axiome[95] sind das *Independence Axiom* (engl. für Unabhängigkeitsaxiom) und das *Information Axiom* (engl. für Informationsaxiom). Sie schreiben vor, dass sämtliche funktionalen Anforderungen zum Zeitpunkt Ihrer Definition voneinander unabhängig sein müssen und dass der Informationsgehalt des Designs unter Wahrung der Vollständigkeit minimiert werden muss. Aus diesen Axiomen leiten SUH[96] und STROGATZ[97] auf Basis von Anwendungserfahrungen eine Vielzahl an Schlussfolgerungen und konkretisierenden Leitsätzen für verschiedene Anwendungsbereiche ab. Die Theorie wurde in den folgenden zwei Jahrzehnten seit ihrer Veröffentlichung auf vielfältige Weise eingesetzt und als Grundlage für weitere entwicklungstheoretische Ansätze, beispielsweise von KULAK ET AL.[98], aufgegriffen.

2.4.6 Function – Behaviour – Structure (FBS)

Der Function-Behaviour-Structure Ansatz nach GERO besteht aus drei Variablenklassen zur Beschreibung der verschiedenen Aspekte eines Designobjekts[99]:

Function (Funktion): Beschreibt die teleologischen Funktionen des Objekts, also dessen Zweck.

Behaviour (Verhalten): Beschreibt die Verhaltensattribute des Objekts, die sich aus den Strukturvariablen ableiten, also dessen Verhalten.

Structure (Struktur): Beschreibt die Komponenten des Objekts und deren Beziehungen, also dessen Struktur.

Nach GERO besteht zwischen Funktionen und Verhalten kein direkter, sondern nur der indirekte Zusammenhang über die Systemstruktur, die Funktionen ausübt und die sich in einem Verhalten äußern. Der FBS-Designprozess besteht aus 8 Schritten, die grafisch in Bild 2-8 dargestellt sind.

---

[95] Ein Axiom ist eine fundamentale Wahrheit ohne Gegenbeispiele oder Ausnahmen, die nicht aus anderen Gesetzmäßigkeiten oder Prinzipien hergeleitet werden dürfen (nach Suh (1998)).
[96] Suh (1990), Suh (1995), Suh und Do (2000)
[97] Strogatz (2001)
[98] Kulak et al. (2010)
[99] Gero (1990)

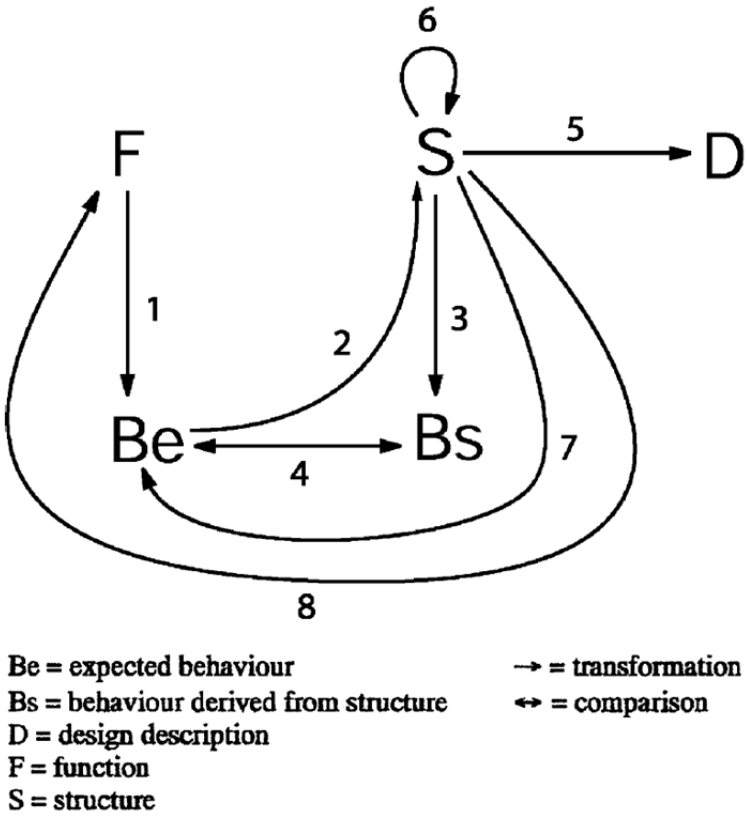


Bild 2-8: Das FBS-Framework nach GERO[100]

Zu Beginn steht die Formulierung (1), die Anforderungen, die in Form von Funktionen (F) ausgedrückt wurden, in erwartetes Verhalten (Be) transformiert. Anschließend erfolgt die Synthese (2), in der das erwartete Verhalten (Be) in eine Lösungsstruktur (S) überführt wird. Eine Analyse (3) leitet daraus deren reales Verhalten (Bs) ab. Die Evaluation (4) vergleicht erwartetes und reales Verhalten und liefert die Entscheidungsgrundlage, ob die Designlösung akzeptiert wird. In der Dokumentation (5) werden die Designbeschreibungen (D) zur Herstellung bzw. Fertigung erarbeitet. Die drei weiteren Schritte der Reformulierung beziehen sich auf eventuelle Änderungen und deren Auswirkungen im Falle nicht erreichter Ziele bezüglich der Struktur (6), dem Verhalten (7) oder den Funktionen (8). Die Autoren argumentieren, dass das FBS-Framework insbesondere durch diese drei letzten Schritte im Gegensatz zu den meisten anderen Entwicklungsprozessmodellen die Dynamik realer Entwicklungsprozesse berücksichtigt. GERO UND KANNENGIESSER betrachten in diesem Zusammenhang die Unterschiede zwischen erwarteter Welt, interpretierter Welt und externer Welt und leiten daraus und auf Basis des FBS-Frameworks ein erweitertes, situationsbedingtes Framework des Designens für die jeweiligen oben

[100] Gero (1990)

genannten Schritte ab[101]. Die jeweiligen Schritte wurden in einem Gesamtframework vereint, das in Bild 2-9 dargestellt ist.

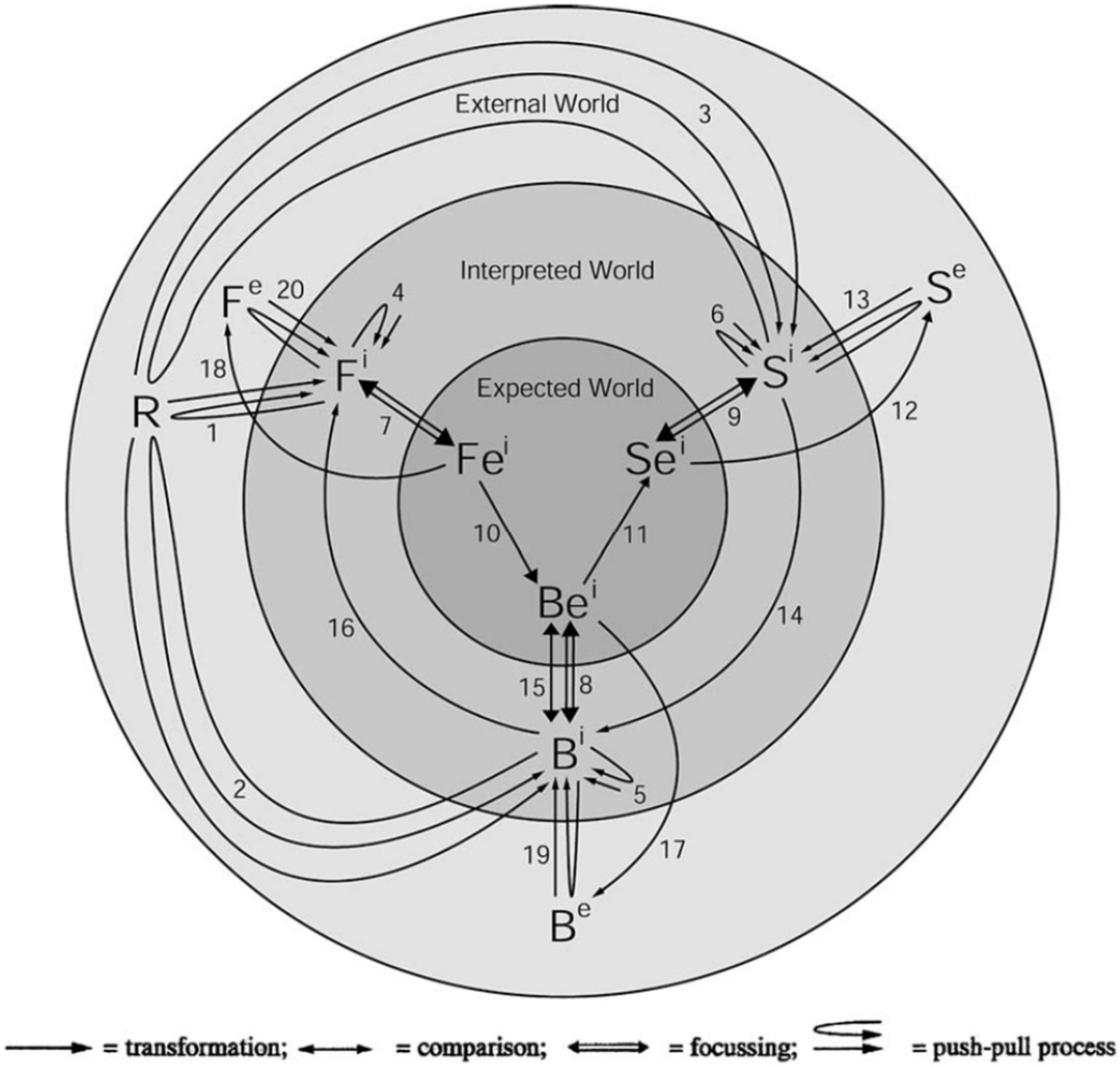


Bild 2-9: Situationsspezifisches FBS-Framework nach GERO UND KANNENGIESSER[101]

Das hier gezeigte Framework besteht nun aus insgesamt 20 möglichen Schritten, welche die Autoren ausführlich herleiten. Der Mehrwert dieses erweiterten Frameworks liegt laut den Autoren insbesondere in der besseren Unterstützung in der Konzeptphase (in der Lösungsprinzipien erarbeitet werden) eines in einer dynamischen Welt ablaufenden Designprozesses. UMEDA UND TOMIYAMA[102] stellen ebenfalls einen FBS-Ansatz in zahlreichen Publikationen vor, der sich bis in die 80er Jahre des letzten Jahrtausends insbesondere auf die General Design Theory

[101] Gero und Kannengiesser (2004)
[102] Bspw. in Umeda und Tomiyama (1995), Umeda und Tomiyama (1997)

zurückführen lässt[103]. In ihrer Publikation aus 1990 stellen UMEDA ET AL. das Function – Behaviour – State-Diagram vor, das in Bild 2-10 zu sehen ist[104].

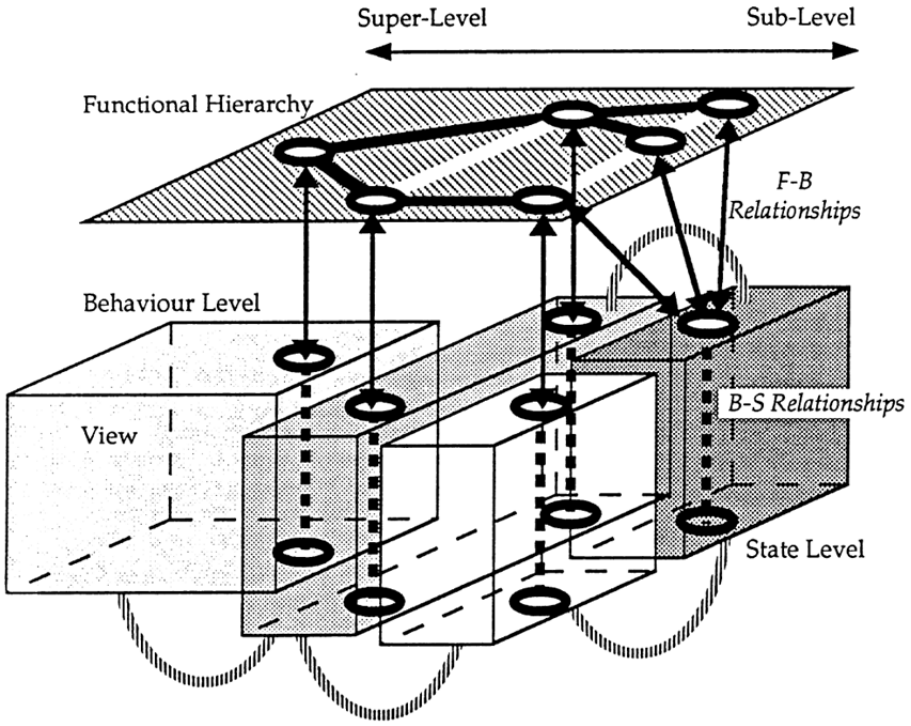


Bild 2-10: Function-Behaviour-State-Diagram nach UMEDA ET AL.[104]

Ziel dieses Diagramms ist die Modellierung von Systemen auf Basis seiner funktionalen Ziele. Die Autoren unterscheiden hierbei einen subjektiven und einen objektiven Anteil. Analog zu Gero's Aussagen resultieren das (objektive) Verhalten und auch die Zustände aus den (subjektiven) Funktionen. Sichten ermöglichen die Betrachtung unterschiedlicher Detaillierungsebenen in der Hierarchie. Die Zusammenhänge zwischen den Ebenen ermöglichen die Herleitung der Funktionshierarchie aus den verschiedenen Sichten auf das Verhalten.

2.4.7 Characteristics-Properties Modelling (CPM)

Characteristics-Properties Modelling (CPM) ist eine maßgeblich von WEBER[105] entwickelte Methode zur Produktmodellierung. Der Grundgedanke im CPM ist die klare Unterscheidung zwischen Merkmalen (engl. *Characteristics*) und Eigenschaften (engl. *Properties*) in der Betrachtung eines Produktes. Merkmale beschreiben die Struktur, Gestalt und Beschaffenheit eines Produktes. Sie können vom Entwickler

[103] Der Bezug wird in Tomiyama et al. (1989) hergestellt, worin u.a. auch Gero referenziert wurde. Weiterhin nennen die Autoren darin auch explizit die GDT (vgl. Kapitel 2.4.2)

[104] Umeda et al. (1990)

[105] Weber (2005), Weber (2007), Weber (2013)

direkt beeinflusst werden. Beispiele für Merkmale sind Geometrie, Materialien, Dimensionen oder Oberflächen. Eigenschaften beschreiben das Verhalten eines Produktes. Sie können nicht direkt vom Entwickler beeinflusst werden. Beispiele für Eigenschaften sind Gewicht, Sicherheit, Ästhetik, Fertigungs-, Montage-, Prüfgerechtheit, Umwelteigenschaften oder Kosten. Der Verknüpfung von Merkmalen und Eigenschaften liegen die zwei Aktivitäten Analyse und Synthese zugrunde. Bei der Analyse werden Produkteigenschaften auf Grundlage von bekannten bzw. gegebenen Merkmalen bestimmt oder vorhergesagt, falls diese noch nicht real existieren. Analysen können sowohl an realen Objekten, wie beispielsweise Prototypen, als auch an virtuellen Modellen durchgeführt werden. Bei der Synthese werden Produktmerkmale ausgehend von vorgegebenen bzw. geforderten Eigenschaften festgelegt. Die Synthese stellt hierbei die Kernaktivität der Produktentwicklung dar, da sich bei einem Produkt letztlich die Merkmale aus den Eigenschaften ergeben, die den Anforderungen genügen müssen[106].

Das dem CPM zugrunde liegende Metamodell ist sehr rudimentär durch nur wenige Artefakte aufgebaut und daher nicht in der Lage, die differenzierten Informationen der Produktentwicklung angemessen zu beschreiben. Der Fokus der Methode liegt auf der Herstellung von Beziehungen zwischen Datenfragmenten, die jedoch noch sowohl einer Formalisierung als auch einer angemessenen Repräsentation bedürfen, um sie nutzbringend in Entwicklungsumgebungen integrieren zu können. Ein erster Softwareprototyp erweist sich als noch relativ unausgereift, da er einerseits über nur eine Netzdarstellung verfügt, die schnell unübersichtlich wird, und andererseits die Anbindung an Fremdsoftware technisch sehr aufwändig ist und bisher nicht methodisch erforscht wurde.

2.4.8 Münchener Produktkonkretisierungsmodell (MKM)

Ebenso wie EHRLENSPIEL referenzieren auch PONN UND LINDEMANN auf die VDI 2221 und stellen mit dem Münchner Produktkonkretisierungsmodell (MKM) eine Weiterentwicklung vor[107]. Dabei heben sie neben der Produktkonkretisierung auch den Zerlegungsgrad und den Variationsgrad eines Systems als wichtige Dimension bei der Entwicklung und Konstruktion von Produkten hervor. Erstere bezeichnet die Zerlegung eines komplexen Problems in Teilprobleme, wodurch sie überschaubarer und leichter zu bearbeiten sind. Das Ziel der Variantenbildung (oder dem Denken in Alternativen[108]) ist es, realistische Alternativen zur vorhandenen Lösung zu

[106] Conrad (2010)
[107] Ponn und Lindemann (2011)
[108] nach Daenzer und Huber (2004)

generieren, um dadurch die Chance auf innovative Lösungen zu erhöhen. Diese drei Dimensionen spannen einen Modellraum des Konstruierens auf[109] und dienen so der Ordnung von Ergebnissen des Entwicklungsprozesses (siehe Bild 2-11).

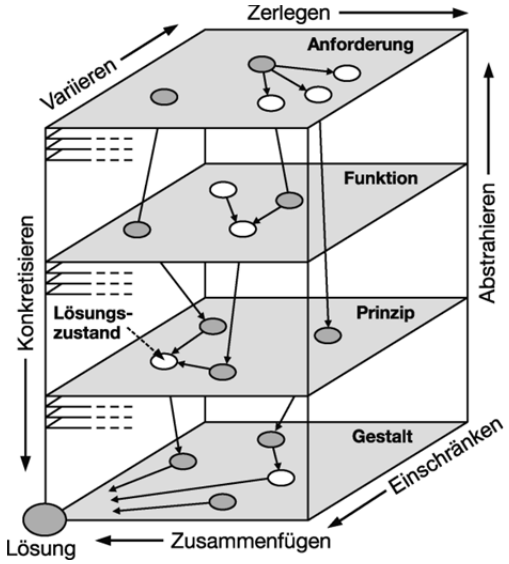


Bild 2-11: Modellraum nach RUDE (aus PONN UND LINDEMANN[107])

Innerhalb des Modellraums werden den vier Konkretisierungsstufen Anforderung, Funktion, Prinzip und Gestalt jeweils Partialmodelle eines integrierten Produktmodells zugeordnet. Der Entwicklungsprozess selbst dient der Navigation durch den Modellraum. Aus diesem Konzept und diversen Forschungsprojekten leiten PONN UND LINDEMANN das Münchner Produktkonkretisierungsmodell (MKM) mit den beiden Hauptkomponenten Anforderungsraum und Lösungsraum ab, das in Bild 2-12 gemeinsam mit den wichtigsten Entwicklungs- und Konstruktionstätigkeiten innerhalb des Lösungsraums dargestellt ist.

[109] nach Rude (1998)

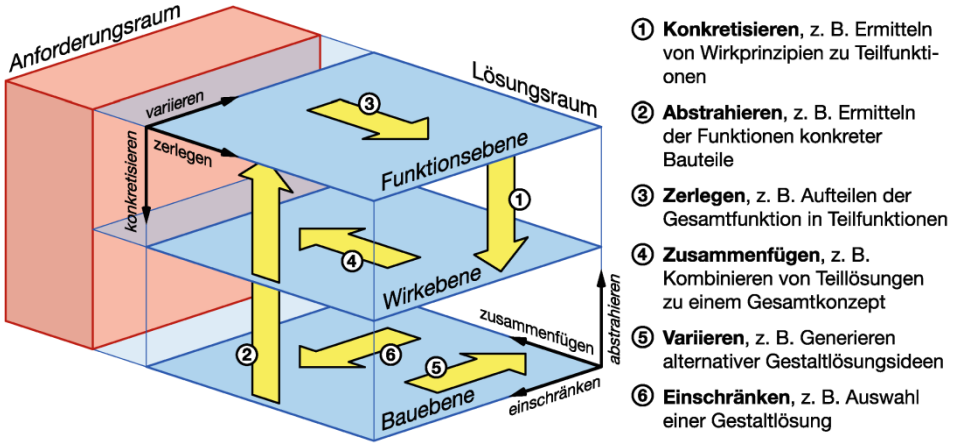


Bild 2-12: Das Münchener Produktkonkretisierungsmodell (MKM) mit Entwicklungs- und Konstruktionstätigkeiten[110]

Der Lösungsraum ist in die drei Konkretisierungsebenen „Funktionsebene“, „Wirkebene“ und „Bauebene“ untergliedert. Durch diese Gestaltung des MKM wird der besonderen Rolle der Anforderungen Rechnung getragen, welche über den gesamten Produktentwicklungsprozess ergänzt, detailliert und konkretisiert sowie mit dem Lösungsraum abgeglichen werden müssen. Auf der Funktionsebene werden zur Problemzerlegung in handhabbarere Bestandteile drei Sichten unterschieden[111]:

- Umsatzorientiertes Funktionsmodell (mit Fokus auf den Stoff-, Energie- und Informationsumsätzen im System)
- Relationsorientiertes Funktionsmodell (zur Identifikation von Schwachstellen und Zielkonflikten)
- Nutzerorientiertes Funktionsmodell (beschreibt die Interaktionen der Nutzer mit dem System im Verlauf seines Lebenszyklus)

DEUBZNER UND LINDEMANN[112] stellen ein Framework zur graduellen funktionalen Dekomposition technischer Systeme mit eigener Notation auf Basis des umsatzorientierten Funktionsmodells vor, welches am Beispiel der Architekturbeschreibung von Hybridantriebsstrangvarianten evaluiert wurde.

Auf der Wirkebene werden die für die Funktionserfüllung relevanten Aspekte einer Lösung abgebildet und nicht funktionsrelevante Details ausgeblendet. Dies entspricht dem Konzept des Contact & Channel-Ansatzes nach ALBERS[113], dessen Elemente auch für das Wirkmodell zur Beschreibung der Wirkgeometrie als Teil des

[110] aus Ponn und Lindemann (2011)
[111] Vgl. Ponn und Lindemann (2011), S. 66
[112] vgl. Deubzer und Lindemann (2009)
[113] Vgl. Kapitel 2.4.7

Wirkmodells aufgegriffen wurden. Um Barrieren, wie sie u.a. durch das Denken in bekannten Lösungsmustern auftreten, zu überwinden, schlagen PONN UND LINDEMANN das Erzeugen einer abstrahierten Beschreibung der konkreten Problemstellung vor, das Problemmodell[114]. Dieses soll als Basis zur Ermittlung von Lösungsideen dienen, die in den ursprünglich betrachteten Bereich zu übertragen sind, um eine Lösung für das eigentliche Problem zu erzeugen. Diese Vorgehensweise zur technischen Problemlösung ist in Bild 2-13 dargestellt.

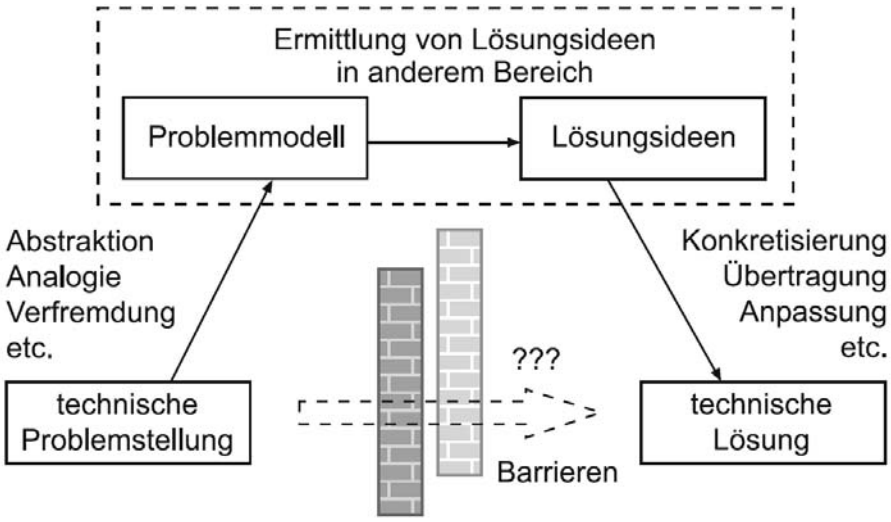


Bild 2-13: Vorgehen bei der Lösung technischer Problemstellungen[273]

Ausgehend von der Wirkebene, in der die funktionsrelevanten Eigenschaften festgelegt wurden, erfolgt die weitere Konkretisierung der Produktgestalt durch Ausarbeitung der Systemkomponenten sowie ihrer Schnittstellen untereinander auf der Bauebene. Hier kann eine Modularisierung beispielsweise durch Gruppierung besonders stark vernetzter Komponenten erfolgen, die sich in Form einer Einflussmatrix bzw. Design Structure Matrix[115] durchführen und grafisch in Wirkungsnetzen darstellen lässt. Weiterhin erfolgen auf der Bauebene zusätzliche Konkretisierungen von Eigenschaften durch die Festlegung von Toleranzen, Passungen und weiteren Gerechtheiten (z.B. für die fertigungsgerechte Konstruktion) auf Basis der Grundregeln der Gestaltung (einfach, eindeutig und sicher[116]).

[114] Vgl. Ponn und Lindemann (2011), S. 92
[115] Vgl. Maurer (2007), Lindemann et al. (2009)
[116] Vgl. Pahl et al. (2007)

2.4.9 Contact & Channel – Ansatz (C&C²-A)

Der Vorläufer des Contact & Channel – Ansatzes (C&C²-A) nach ALBERS wurde erstmals 1999 als Bestandteil des Karlsruher Lehrmodells KaLeP vorgestellt[117] und in seiner Basisdefinition 2002 durch MATTHIESEN[118] als Elementmodell "Wirkflächenpaare & Leitstützstrukturen" beschrieben. Der Ansatz basiert unter anderem auf Vorarbeiten von RODENACKER, ROTH sowie HUBKA UND EDER[119]. Der C&C²-A ist eine Systematik zur Modellbildung technischer Systeme[120]. Der Zweck des Ansatzes liegt in der durchgängigen Beschreibung des Zusammenhangs von beobachteter Funktionsweise und Ausprägung von Gestalt (Analyse) bzw. der beabsichtigten Funktion und Gestalt (Synthese). Er beruht auf den folgenden drei Grundhypothesen nach ALBERS ET AL.[121]:

Grundhypothese 1 (Wechselwirkung): *Leitstützstrukturen* (LSS) eines technischen Systems stehen bei der Funktionserfüllung in Wechselwirkung mit anderen *Leitstützstrukturen* (Wechselwirkung innerhalb des Designraums) bzw. *Connectoren* (Wechselwirkungen mit Nachbarsystemen). Wechselwirkungen finden nur statt bei Kontakt von *Wirkflächen* (WF), die gemeinsam *Wirkflächenpaare* (WFP) bilden. Die wirkungsrelevanten Merkmale, Eigenschaften und *Wirkflächen* (WF) der interagierenden Nachbarsysteme werden durch *Connectoren* (C) dargestellt (siehe Bild 2-14 links).

Grundhypothese 2 (Funktion): Ein System kann seine Funktion(en) nur in Wechselwirkung mit seiner Umgebung erfüllen – ein Bauteil allein hat keine Funktion! Die Beschreibung einer technischen Funktion benötigt immer mindestens eine *Leitstützstruktur* (LSS) mit zwei *Wirkflächen* (WF), die *Wirkflächenpaare* (WFP) mit den *Wirkflächen* (WF) interagierender *Leitstützstrukturen* (LSS) innerhalb des Systems oder mit *Wirkflächen* (WF) interagierender *Connectoren* (C) bilden (siehe Bild 2-14 rechts). Funktionsbestimmend sind dabei die in den Elementen während der Interaktion stattfindenden Effekte sowie ihre Merkmale und Eigenschaften.

Grundhypothese 3 (Fraktaler Charakter): Jedes Teilsystem kann mit C&C²-Modellen auf verschiedenen Abstraktions- und Detaillierungsstufen beschrieben werden. Dazu ist eine Variation der Anzahl, Anordnung und/oder der Eigenschaften der dargestellten Grundelemente erforderlich.

---

[117] Albers und Matthiesen (1999)
[118] Matthiesen (2002)
[119] U.a. Rodenacker (1991), Roth (2000), Hubka und Eder (1992)
[120] Albers et al. (2008c)
[121] Vgl. z.B. Albers und Sadowski (2013), Wiedner (2013), Matthiesen und Ruckpaul (2012), Alink (2010)

Bild 2-14 zeigt die Modellelemente des C&C²-A zur Beschreibung einer Wechselwirkung und einer Funktion in abstrakter Form. Der *Betrachtungsraum* umfasst die für die Betrachtung der Funktion relevanten Elemente, wohingegen der durch eine Systemgrenze abgegrenzte *Designraum* die durch den Entwickler gestaltbaren Elemente beinhaltet. Der Betrachtungsraum schließt dabei auch die als Connector repräsentierten wirkungsrelevanten Merkmale und Eigenschaften sowie die zugehörigen *Wirkflächen* (WF) der interagierenden Nachbarsysteme.

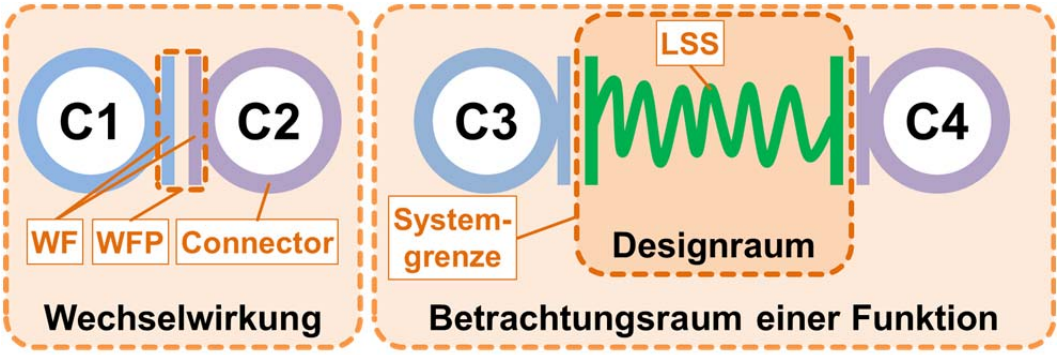


Bild 2-14: Abstrakte Contact & Channel-Modelle von Wechselwirkung und Funktion

Die Modellelemente zur Bildung von Contact- und Channel-Modellen im Rahmen von Analyse und Synthese technischer Systeme sind folgendermaßen definiert[122]:

Wirkflächen (WF) sind feste Oberflächen von Körpern oder generalisierte Grenzflächen von Flüssigkeiten, Gasen oder Feldern, die dauernd oder zeitweise im Kontakt zu einer weiteren Wirkfläche stehen und am Energie-, Stoff- und/oder Informationsaustausch des technischen Systems beteiligt sind.

Wirkflächenpaare (WFP) werden gebildet aus zwei beliebig geformten Wirkflächen, die in Wirkkontakt stehen, in dem Energie, Stoff und/oder Information übertragen werden.

Wirkkontakt (WK) ist der Teil eines Wirkflächenpaares, in dem aktuell die Wechselwirkungen stattfinden.

Leitstützstrukturen (LSS) leiten während der Funktionserfüllung zwischen genau zwei Wirkflächenpaaren Energie, Stoff und/oder Information. Eine Leitstützstruktur kann sich dabei abhängig vom Detaillierungsgrad der Modellbildung über Systeme oder Subsysteme hinweg erstrecken. Leitstützstrukturen existieren gemeinsam mit den zugehörigen Wirkflächenpaaren ausschließlich im Zeitraum der Funktionserfüllung. Leitstützstrukturen können nur in Volumina von Festkörpern,

[122] Nach Albers und Sadowski (2013) und Matthiesen (2002), präzisierte Definitionen von WFP, WK und LSS nach Wiedner (2013)

Flüssigkeiten, Gasen oder felddurchsetzten Räumen oder deren Kombination realisiert werden.

Connectoren (C) integrieren die wirkungsrelevanten Merkmale, Eigenschaften und Wirkflächen durch Modelle der mit dem System interagierenden Systemumgebung. Sie liegen im Betrachtungsraum, jedoch nicht im Designraum.

Begrenzungsflächen (BF) sind feste Oberflächen von Körpern oder generalisierte Grenzflächen von Flüssigkeiten, Gasen oder Feldern, die nie Wirkflächen sind.

Tragstruktur (TS) ist die Menge aller möglichen Leitstützstrukturen.

Reststruktur (RS) sind Volumina von Körpern, Flüssigkeiten, Gasen oder felddurchsetzte Räume, die nie Tragstruktur werden.

Der Zweck des C&C²-A in der Analyse ist die durchgängige Beschreibung des Zusammenhangs von beobachteter Funktionsweise und Ausprägung von Gestalt. Beobachtet (bzw. gemessen) werden können jedoch nur die an Wirkflächenpaaren unter konkreten Betriebsbedingungen auftretenden Outputs von Stoff, Energie und/oder Information. Diese Betriebsbedingungen setzen sich aus konkreten Verursachern[123] von Funktionen durch interagierende Nachbarsysteme[124] sowie den aktuellen Zuständen[125] der betroffenen Teilsysteme und Komponenten innerhalb des beobachteten Systems zusammen.

Unter konkreten Betriebsbedingungen bildet sich ein *Wirknetz* über mittels Wirkflächenpaaren interagierende Connectoren und Leitstützstrukturen aus. Die aus der Ausführung einer Funktion unter diesen konkreten Bedingungen resultierenden und beobachtbaren (bzw. messbaren) Outputs werden als *Wirkung* bezeichnet[126]. Verursachen diese Wirkungen gleichzeitig Funktionen in den benachbarten Systemen, liegen *Wechselwirkungen*[127] vor. Bild 2-15 zeigt beispielhaft ein Contact & Channel-Modell einer Schraube auf Bauteilbetrachtungsebene, bestehend aus dem sich während des Einschraubvorgangs ausbildenden Wirknetzes sowie einer Darstellung der detaillierten physischen Struktur im Hintergrund.

---

[123] Man spricht hier auch oft von Auslösern oder Triggern

[124] Diese werden bei der Durchführung einer Analyse (z.B. FTA, Simulation oder Experiment) beaufschlagt (z.B. durch Definition einer Fehlerquelle, Restsystemsimulationsmodelle oder Belastungsmaschinen)

[125] Diese können entweder definiert werden oder ergeben sich aus vorausgegangenen Analysen. Zur Definition von Zustand siehe Kapitel 5.2

[126] Vgl. Albers und Sadowski (2013)

[127] Bei Energieübertragung ist das immer der Fall, da stets ein Gleichgewicht herrschen muss (vgl. Ausführungen von Matthiesen (2002), Kapitel 3.2 sowie 3.6f)

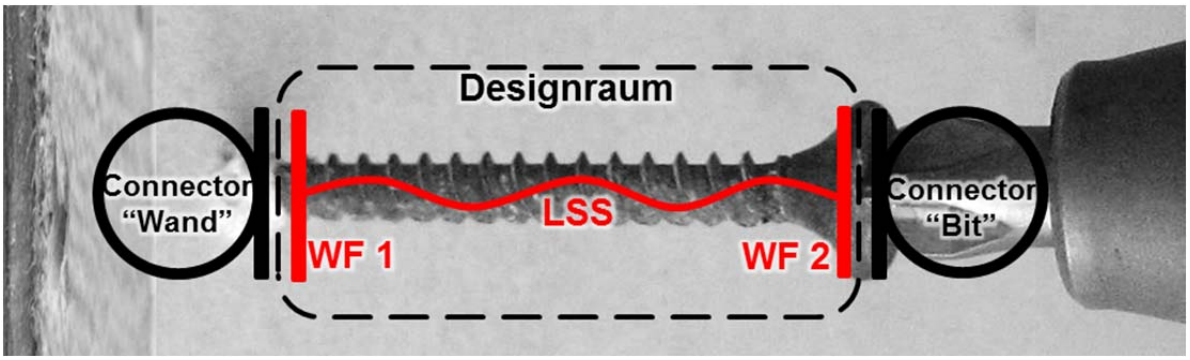


Bild 2-15: C&C²-Modell, bestehend aus dem Wirknetz und der entsprechenden, detaillierten physischen Struktur einer Schraube[132]

Das Ziel der Analyse in der Produktentwicklung ist, möglichst alle denkbaren Betriebsbedingungen des zweckgebunden betrachteten technischen (Teil-)Systems zu untersuchen und somit sämtliche möglichen Wirknetze zu ermitteln und deren (sowohl gewollten als auch ungewollten) Funktionen sowie ihre funktionsrelevanten Eigenschaften zu bestimmen. Die Summe aller (vielfach überlappenden) Wirknetze einer zweckgebundenen Analyse ergibt die *Wirkstruktur* des betrachteten (Teil-)Systems. Bild 2-16 zeigt ein Analyseergebnis in Form eines C&C²-Modells eines Hybridantriebsstrangs mit relativ geringem Detaillierungsgrad. Hierin werden zwei Wirknetze für die Betriebszustände „Rekuperation“ und „Verbrennungsmotorischer Betrieb“ als Ausprägungen der Hauptfunktion „Antriebsenergie bereitstellen“ des Antriebsstrangs hervorgehoben dargestellt. Nicht aktive Leitstützstrukturen und Wirkflächen sind grau dargestellt. Diese werden erst für andere Betriebszustände wie „Boost“ oder „Lastpunktanhebung“ in eigenen Wirknetzen relevant.

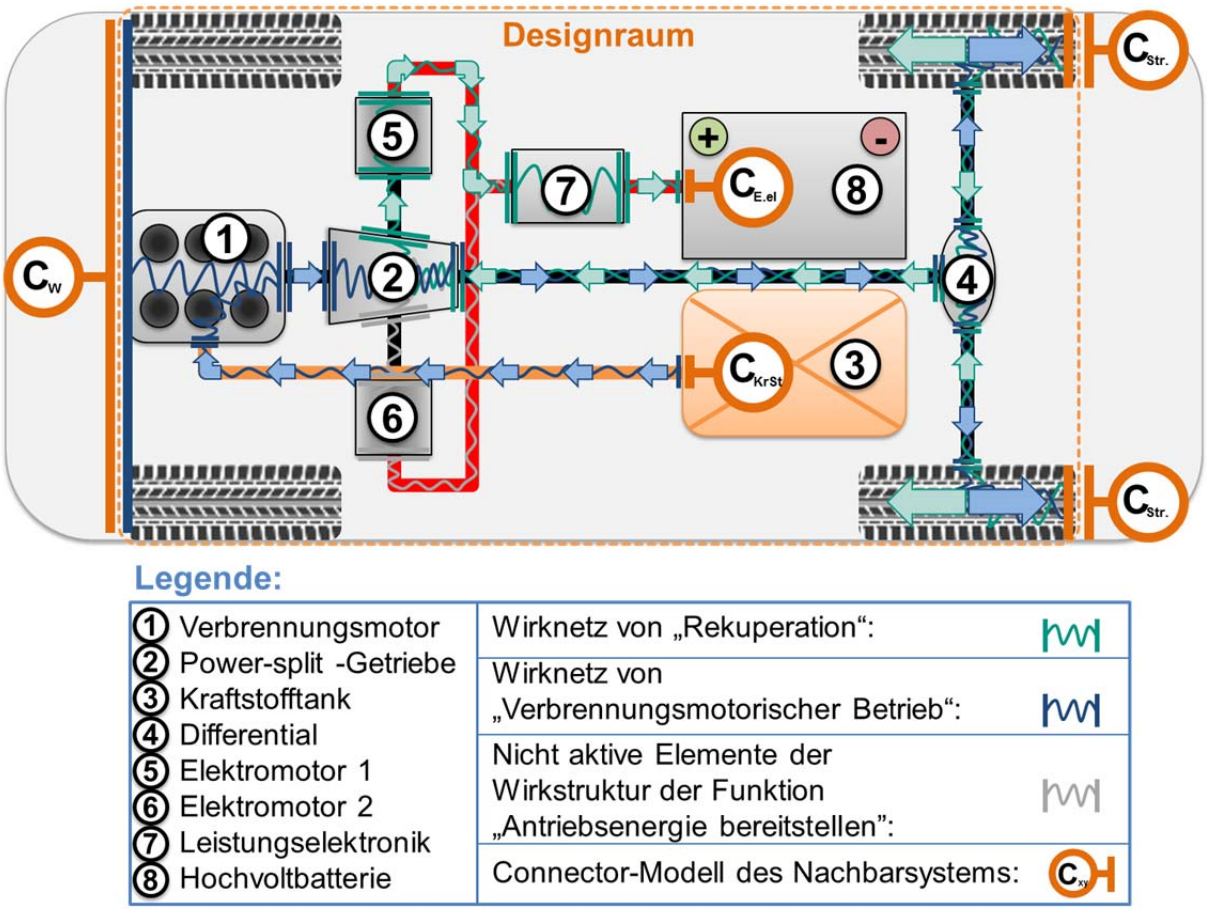


Bild 2-16: C&C-Modell, bestehend aus der Wirkstruktur, zwei hervorgehobenen Wirknetzen sowie einer Prinzipdarstellung der Topologie eines Hybridantriebsstrangs[128]

Abhängig vom Betrachtungsraum ist es möglich, die Analyse der Wirkstruktur im Hinblick auf die Analyse einer einzelnen Funktion oder auch eines Verbunds mehrerer Funktionen gleichen Detaillierungsgrades (bei der Untersuchung von Haupt- und Nebenfunktionen[129]) in gleicher Weise durchzuführen. In beiden Fällen werden hierbei auftretende Teilfunktionen[130] der Strukturelemente betrachtet. Die gesamte Wirkstruktur eines technischen Systems ist die Summe aller Wirkstrukturen aller Funktionen. Sie beinhaltet damit alle an einer Funktion beteiligten Strukturelemente des technischen Systems. Diese ist insbesondere bei sehr komplexen Systemen nicht vollständig ermittelbar, weil auch Aspekte wie unsachgemäßer Gebrauch vollständig analysiert werden müssten.

In der Synthese ist die Herausforderung noch größer, da hier sämtliche mögliche Betriebsbedingungen (auch sämtliche Extrembedingungen) „vorausgedacht“ werden

[128] Basierend auf Albers und Sadowski (2013)
[129] Angelehnt an die Klassierung von Funktionen nach Pahl et al. (2006), S. 44f und S. 664f
[130] Angelehnt an die Gliederung von Gesamtfunktionen in Teilfunktionen nach Pahl et al. (2006), S. 44

müssten, um die gesamte Wirkstruktur eines technischen Systems zu antizipieren. Dies wäre sowohl für gewollte Funktionen (*Ziel-Funktionen*) als auch für ungewollte Funktionen der Fall. Insbesondere Letztere identifiziert man meist erst bei Tests unter Extrembedingungen. Da bereits hergeleitet wurde, dass die gesamte Wirkstruktur aus einem vorhandenen System kaum analysiert werden kann, ist es einleuchtend, dass diese noch schwieriger vollständig vorausgedacht werden kann. Daher ist die zweckgebundene Modellbildung mit dem Contact & Channel-Ansatz nicht nur im Sinne der Reduktion der Komplexität, sondern insbesondere im Sinne eines strukturierten Vorgehens in Analyse und Synthese technischer Systeme geeignet. Daraus ergibt sich zusammenfassend:

Bei der **Analyse** mit dem C&C²-A werden bestehende, in der Produktentwicklung eingesetzte gestaltvisualisierende Modelle[131] genutzt, um die Funktionen und die zugehörigen Wirkstrukturen bei der Interaktion des untersuchten Systems mit Nachbarsystemen zu erkennen. Für diese Strukturen können Merkmale identifiziert werden, welche die Ausprägung der Funktionserfüllung[132] und ihre Eigenschaften beeinflussen[133].

In einer anschließenden **Synthese** können einerseits bestehende Designmerkmale optimiert werden, um die Qualität der Funktionserfüllung und der Eigenschaften zu verbessern. Andererseits können alternative Lösungsprinzipien durch Erweiterung oder Reduktion der Wirkstruktur erarbeitet werden[134].

Weiterhin unterstützen Contact & Channel-Modelle die **Zusammenarbeit** in der Produktentwicklung durch Speicherung (Dokumentation), Bereitstellung und Kommunikation von Informationen bezüglich Funktionen und Eigenschaften interagierender technischer Systeme oder Teilsysteme auf Basis etablierter Produktmodelle[134].

Seit seiner Vorstellung wird der Ansatz kontinuierlich validiert[135], weiterentwickelt[136] und mit weiteren Modellbildungsansätzen wie Axiomatic Design gekoppelt[137]. Beispielsweise wurden von ALBERS ET AL. die Beschreibung von Sequenzmodellen[138]

---

[131] z.B. Handzeichnungen, Prinzipskizzen, 2D-CAD-Schnittdarstellungen, 3D-CAD-Modelle etc.

[132] Das meint die Art und Weise, in welcher Form die Funktion erfüllt wird, also die quantitative Ausprägung ihrer Wirkungen (Outputs)

[133] Vgl. Albers und Sadowski (2013); zu den Definitionen von „Merkmale“ und „Eigenschaften“ vgl. Kapitel 5.2.3 bzw. 5.2.7

[134] Vgl. Albers und Sadowski (2013)

[135] z.B. Albers et al. (2008b), Matthiesen (2011)

[136] z.B. Albers et al. (2011a)

[137] z.B. Marques et al. (2009)

[138] Albers et al. (2008c)

und von MATTHIESEN UND RUCKPAUL die Modellierung verschiedener Zustände mit dem C&C²-A eingeführt[139].

Neben der Weiterentwicklung des C&C²-A wurden in den vergangenen Jahren auch zwei prototypische Softwarewerkzeuge zur computergestützten Anwendung des Ansatzes entwickelt, die in den folgenden Unterkapiteln vorgestellt werden.

2.4.9.1 Der Contact & Channel Model Coach – C3

Der Softwareprototyp Contact & Channel Model Coach – C3, kurz ConChaCoach, ist ein Werkzeug des IPEK zur Erstellung von Contact & Channel – Modellen auf Basis des C&C²-A[140]. Es ermöglicht die Darstellung der Elemente WF, WFP und LSS sowie eine Anbindung an Nachbarsysteme, die inzwischen durch den Connector repräsentiert wird. Das Datenformat basiert auf dem XML-Standard[141] und erlaubt eine einfache Adaptier- und Erweiterbarkeit. Eine Integration mit CAD-Umgebungen wird angestrebt, um je nach Bedarf den Zusammenhang zwischen Funktion, Basiselementen und konkreter Geometrie zu ermöglichen[142]. Bild 2-17 zeigt die Arbeitsoberfläche des derzeitigen Prototyps des Softwarewerkzeugs.

[139] Matthiesen und Ruckpaul (2012)
[140] Albers et al. (2009a), Albers et al. (2011a)
[141] Vgl. Kapitel 2.8.6
[142] Enkler (2010)

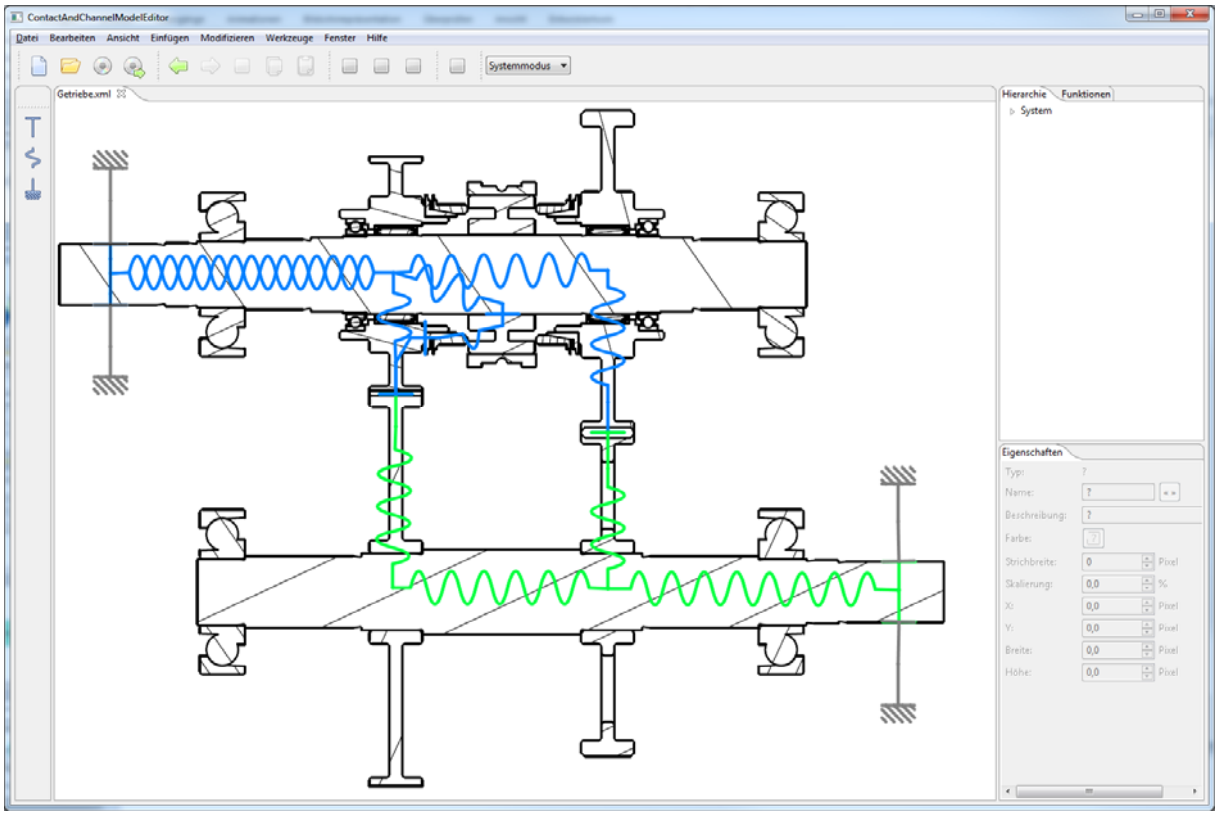


Bild 2-17: Arbeitsoberfläche des ConChaCoaches

Das Softwarewerkzeug wird kontinuierlich weiterentwickelt und liefert somit wichtige Erkenntnisse bei der Modellierung und zweidimensionalen Visualisierung von Funktion und Gestalt technischer Systeme, die auch in die Ergebnisse der vorliegenden Arbeit einflossen. Dennoch ist das Werkzeug eine reine Forschungsplattform und weist weder den Funktionsumfang noch den Reifegrad professioneller Softwareprodukte auf.

2.4.9.2 Cambridge Advanced Modeler

In einer Kooperation mit dem Engineering Design Centre (EDC) der Universität Cambridge wurde eine prototypische Implementierung des C&C²-A in deren Prozessmodellierungswerkzeug Cambridge Advanced Modeler (CAM) realisiert[143]. Neben der Erstellung von Contact & Channel – Modellen wurde in diesem Werkzeug auch eine Fluss- und Funktionsbibliothek auf Basis der NIST-Funktionsdatenbank[144] implementiert, wodurch die Auswahl von Standardflüssen und –funktionen für die Basiselemente des C&C²-A ermöglicht wird. Bild 2-18 zeigt die Arbeitsoberfläche des Werkzeugs sowie die Möglichkeit, den Modulen aus Leitstützstrukturen und

[143] Albers et al. (2009a); Albers et al. (2010a), Albers et al. (2011b)
[144] nach Hirtz et al. (2002)

Wirkflächen auf Basis definierter Flüsse und Funktionen passende Bauteile bzw. Systeme aus einer Bauteilbibliothek zuzuweisen.

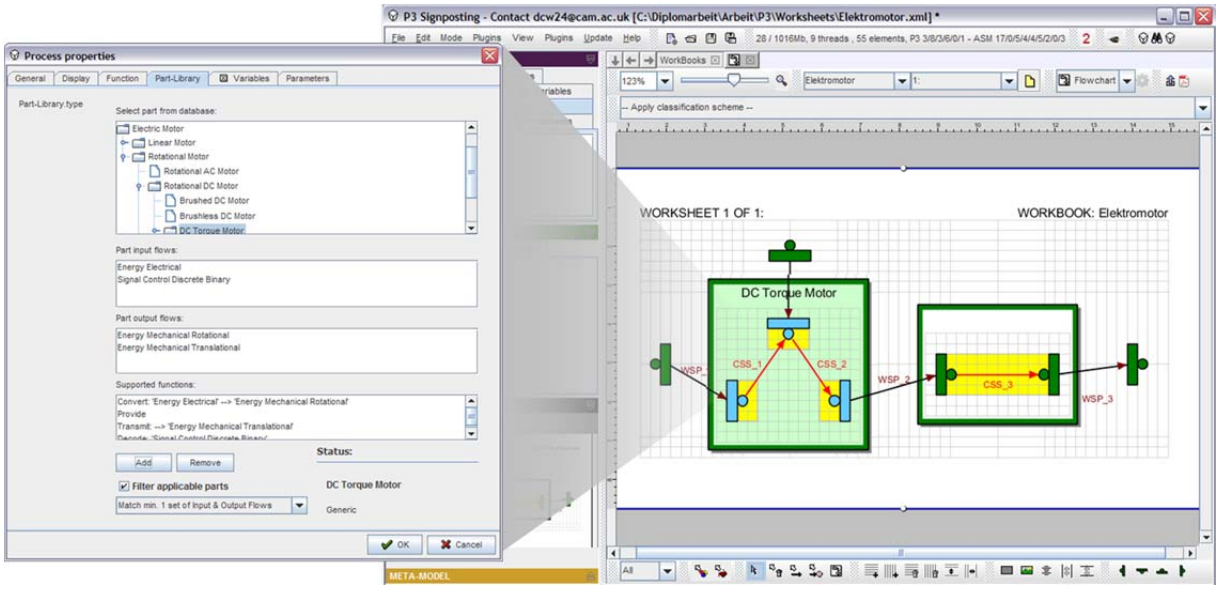


Bild 2-18: Arbeitsoberfläche des Cambridge Advanced Modelers

Derzeitige Forschungsarbeiten beschäftigen sich mit der Verwendung des CAM zur integrierten, softwarebasierten Modellierung von Produkt- und Prozessinformationen mit dem Ziel, Informationen aus Ziel-, Objekt- und Handlungssystem modellbasiert zu koppeln und bereitzustellen[145].

2.4.10 Zwischenfazit

Die wissenschaftstheoretischen Ansätze bilden eine zentrale Grundlage zur formalisierten (mathematischen) Beschreibung und Klassierung von Objekten und trugen dadurch unter anderem entscheidend zur Entwicklung von Computer-Aided Design (CAD)-Werkzeugen bei[146]. Diese sich in vielen Aspekten stark ähnelnden Theorien wurden inzwischen auf vielfältige Weise diskutiert und weiterentwickelt, deren Grundideen sind jedoch nach wie vor eine wichtige Basis bei der Entwicklung von computergestützten, formalen Modellen. Die Systemtheorie der Technik sowie der Contact & Channel – Ansatz bilden die zentralen wissenschaftlichen Grundlagen für die Sprache zur Modellierung technischer Systeme als Teil der in dieser Arbeit vorgestellten Modellierungstechnik. Darüber hinaus werden jedoch auch Konzepte und Einflüsse aus anderen Modellbildungsansätzen aufgegriffen. Die zentrale Herausforderung bei der Entwicklung einer durchgängigen, flexiblen Systemmodellierungstechnik besteht in den teilweise unterschiedlichen Termini für

[145] Albers et al. (2012a)
[146] Vgl. Lossack (2006)

gleichartige Aspekte / Elemente oder auch abweichende Definitionen von verbreitet in verschiedenen Fachdisziplinen verwendeten Fachbegriffen. Beispielsweise sprechen Gausemeier von der Baustruktur, Lindemann von der Bauebene und Ehrlenspiel sowie Rude von der Gestalt[147]. Auch Begriffe wie physikalisches Prinzip, Wirkprinzip, Wirkstruktur oder Funktionsstruktur wirken grundsätzlich verständlich, sind aber sehr unscharf voneinander und von den anderen Abstraktionsebenen trennbar, wodurch diesen auch von vornherein oft ein „fließender Übergang“ bekundet wird. Weiterhin gehen sämtliche Ansätze von Anforderungen aus, die in Funktionen, prinzipielle Lösungen und schließlich in ein ausgestaltetes Produkt münden. Auch hier überwiegt ein sequentieller Charakter, denn Iterationen zwischen den Schritten werden zwar benannt, jedoch erfolgt häufig keine explizite Rückspiegelung auf die Anforderungen, die ebenfalls dynamischen Änderungen und Ergänzungen unterliegen. Auch die Durchgängigkeit wird in weiten Teilen nicht explizit benannt. Die Beantwortung der Frage, wie konkret denn welche Informationen welcher Ebenen zusammenhängen, und wie man diese durch das Systemmodell zurückverfolgen kann, wird nicht vollständig gegeben. Dieser Herausforderung soll insbesondere durch die Basisdefinition einer gemeinsamen Sprache zur Modellbildung technischer Systeme (vgl. Kapitel 5) und ihrer Umsetzung in Form einer Modellierungstechnik auf Basis der Sprache SysML (vgl. Kapitel 6) begegnet werden.

Es gibt nach wie vor keine rechnerimplementierte Lösung für die vorgestellten Modellierungsansätze, die sich in der industriellen Anwendung durchsetzen konnte. Ein möglicher Grund ist die bisher entweder nur prototypische oder nur teilweise Umsetzung der Ansätze in formalen Modellen und den entsprechenden Rechnerwerkzeugen. Die anschließende Kurzeinführung wesentlicher Grundkonzepte der Objektorientierung dient dem Leser als Unterstützung zum Erlangen eines besseren Verständnisses der in den darauf folgenden Kapiteln vorgestellten, bisher realisierten objektorientierten Modellierungssprachen.

2.5 Grundkonzepte der Objektorientierung

Dieses Kapitel stellt die Grundkonzepte der Objektorientierung vor, da sie die Grundlage nahezu sämtlicher, rechnerbasierter Modellierungssprachen darstellen und dort immer wieder Anwendung finden. Somit unterstützt dieses Kapitel das Verständnis grundlegender Konzepte der in dieser Arbeit vorgestellten Modellierungstechnik, insbesondere wenn der Leser kein

[147] Vgl. auch Kapitel 2.9

informationstechnologisches Hintergrundwissen mitbringt. Im Folgenden werden die wichtigsten Begriffe und Konzepte der Objektorientierung kurz erläutert.

2.5.1 Klassen und Objekte

Der Begriff *Klasse* leitet sich aus ihrem Zweck, der Klassifizierung von Objekten mit gleichen Merkmalen, ab. Klassen können sowohl Dinge, Personen oder Daten sein. Ihnen können folglich konkrete *Objekte* anhand ihrer Parameter (*Attribute*) zugeordnet werden und diese somit Mitglied einer Klasse werden. Der Vorgang der Zuordnung eines Objektes zu einer Klasse wird als *Instanziierung* bezeichnet. Beispielsweise könnte eine Klasse „Fahrzeug“ gebildet werden, der die Merkmale „Seriennummer“, „Fortbewegungsmedium“ und „Einsatzzweck“ zugeordnet werden. Die Seriennummer ist ein freier Text im Format „String“ (eine Folge von Zeichen), als Fortbewegungsmedium kann „Wasser“, „Luft“, „Straße“ oder „Schiene“ gewählt werden und der Einsatzzweck kann „Gütertransport“ oder „Personentransport“ sein. Bild 2-19 zeigt auf der linken Seite die entsprechende Klasse und auf der rechten Seite zwei beispielhafte Instanzen dieser Klasse.

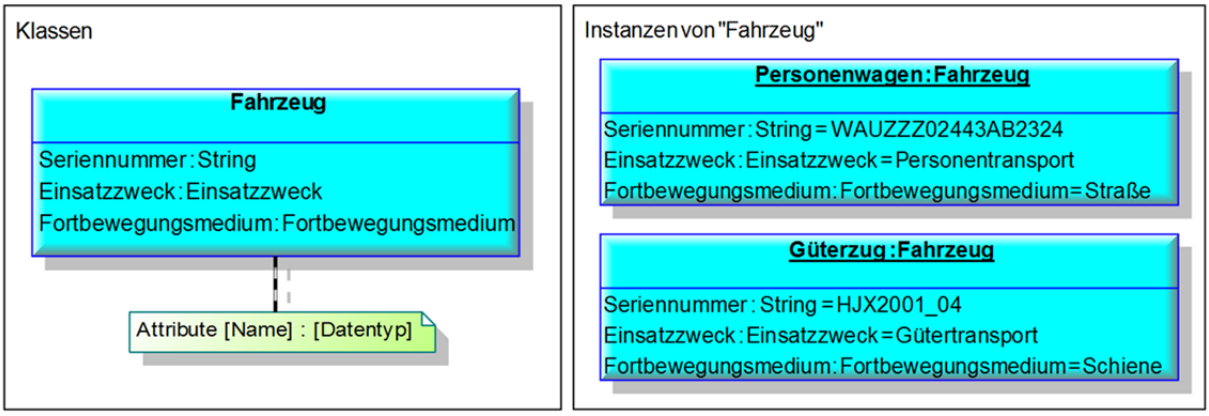


Bild 2-19: Beispiel für Klassen und Instanzen

Die Instanzen werden in UML-Modellierungswerkzeugen durch ihre individuelle Bezeichnung und deren Typ (also deren Klassenzugehörigkeit), getrennt durch einen Doppelpunkt, dargestellt (siehe Bild 2-19 rechts). Gleichzeitig kann man nun konkrete Werte für die von der Klasse erhaltenen Parameter definieren. Dies ist ebenfalls beispielhaft in der Abbildung rechts geschehen. Beide Instanzen haben nun individuelle Parameterwerte, sind jedoch beide vom gleichen Typ, nämlich der Klasse „Fahrzeug“.

2.5.2 Vererbung

Viele allgemeine Merkmale von Klassen überschneiden sich auch klassenübergreifend. Häufig macht es daher Sinn, Objekte verschiedenen Typs in mehr als nur einer Ebene zu unterscheiden. So könnten Fahrzeuge nochmals nach ihrem Fortbewegungsmedium in eigene Klassen unterteilt werden, um weitere, für

die jeweiligen Fortbewegungsmedien spezifische Merkmale in separate Klassen auszulagern. Um eine solche Klassenhierarchie konsistent aufbauen zu können, wurde das Konzept der *Vererbung* eingeführt. Hier *erben* die *Kind-Klassen* die Merkmale der *Eltern-Klasse*. Man spricht hier daher von einer *Eltern-Kind-Beziehung* oder von einer *Spezialisierung*. Somit haben die Kind-Klassen sowohl gemeinsame Merkmale (nämlich die von der Eltern-Klasse geerbten) als auch weitere, individuelle Merkmale, die bei der Bildung von Instanzen wiederum alle mit konkreten Werten belegt werden können. Bild 2-20 zeigt hierzu ebenfalls ein Beispiel.

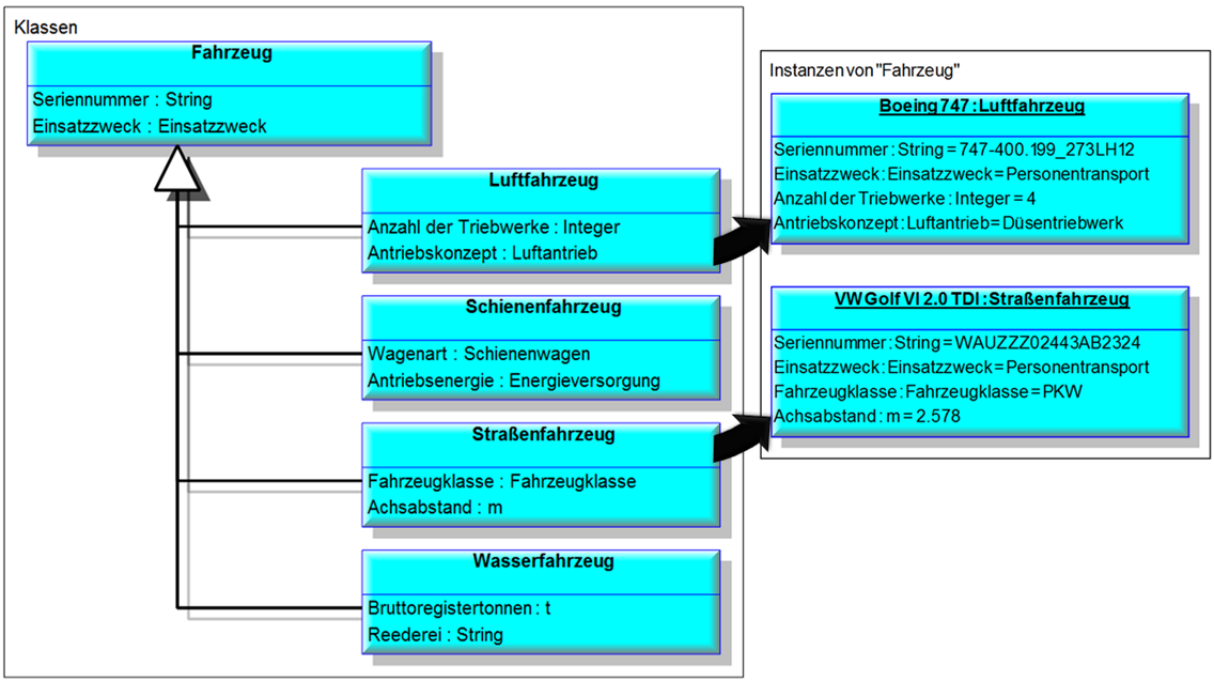


Bild 2-20: Hierarchisierung von Klassen und Instanzbildung

Links im gezeigten Beispiel wurde eine Eltern-Klasse „Fahrzeug“ in die Kind-Klassen „Luftfahrzeug“, „Schienenfahrzeug“, „Straßenfahrzeug“ und „Wasserfahrzeug“ unterteilt, wodurch das ehemalige Attribut „Fortbewegungsmedium“ obsolet wurde, da dies nun als eigenes Klassierungsmerkmal dient. Die Hierarchisierung wird durch den Beziehungstyp *Spezialisierung* modelliert, die im Diagramm als weißes Dreieck dargestellt ist. Die Seriennummer und der Einsatzzweck sind gemeinsame Merkmale aller Klassen und daher der Eltern-Klasse zugeordnet, die diese an ihre Kind-Klassen vererbt. Darüber hinaus haben die Kind-Klassen weitere, spezifischere Merkmale erhalten, die in den anderen Kind-Klassen keinen Sinn ergeben würden. Rechts in der Abbildung wurden nun je eine Instanz von „Luftfahrzeug“ und „Straßenfahrzeug“ gebildet. Erstere ist eine neue Instanz, Letztere entspricht der Instanz „Personenwagen“ aus Bild 2-19. Beiden wurden wiederum konkrete Werte für ihre geerbten und ihre speziellen Merkmale zugewiesen. An dem übereinstimmenden Merkmal „Seriennummer“ von „VW Golf VI 2.0 TDI“ aus Bild 2-20 und „Personenwagen“ aus Bild 2-19 ist zu erkennen, dass es sich hier um die

gleiche Instanz handelt, die jedoch aufgrund der neuen Klassenhierarchie neu angelegt wurde und dadurch die weiteren Merkmale „Fahrzeugklasse“ und „Achsabstand“ erhielt.

In welcher Ausprägung das Vererbungsprinzip oder Attribute zur eindeutigen Unterscheidung von Objekten angewendet werden, hängt vom jeweiligen Anwendungsfall und dem Modellierungszweck ab und muss daher individuell entschieden werden.

2.5.3 Parts – Klassen, die eine Rolle erhalten

Die in den vorangegangenen Kapiteln eingeführten Objekte sind als konkrete Individuen zu verstehen, mit denen eine reale Person oder ein realer Gegenstand beschrieben wird. Da Systeme aber oft nicht nur durch Attribute beschrieben werden, sondern auch eine eigene interne Struktur aufweisen können, wurde hierfür ein weiteres Konzept der Objektorientierung eingeführt: die Bildung von *Parts* (deutsch: Teil). Ein Part ist eine *Sub-Klasse*, die selbst Teil einer *Super-Klasse* ist und darin eine konkrete Rolle ausübt. Der Unterschied eines Parts zu einer Klasse ist insbesondere, dass seine Schnittstellen (*Ports*) mit denen von Nachbarrollen verbunden werden. Eine Klasse hat zwar immer Schnittstellen, kennt aber keine Verbindungen zu Nachbarklassen, ein Part in seiner Rolle hingegen schon. Will man eine Sub-Klasse als Teil einer Super-Klasse deklarieren (beispielsweise einen Motor als Teil (Part) eines Straßenfahrzeugs), so verwendet man hierzu eine *Komposition*. Bildlich ausgedrückt wird die Struktur einer Super-Klasse (eines Systems) durch die Anordnung von Sub-Klassen (Subsystemen) „komponiert“. Man spricht hier auch von einer *Ganzes-Teil-Beziehung*. Es gibt einen weiteren, schwächeren Beziehungstyp, die *Aggregation*. Diese ordnet einer Super-Klasse ebenfalls eine Sub-Klasse zu, jedoch ist dieses entstehende Part optional, die Super-Klasse kann also im Gegensatz zu einer per Komposition zugeordneten Sub-Klasse auch ohne diese existieren.

Bild 2-21 zeigt ein *Klassendiagramm* eines Straßenfahrzeugs, dem verschiedene Sub-Klassen als Part per Komposition (schwarz ausgefüllte Raute) und die Klasse „Klimaanlage“ per Aggregation (also als optionales Part, Raute mit weißer Füllung) zugeordnet wurden.

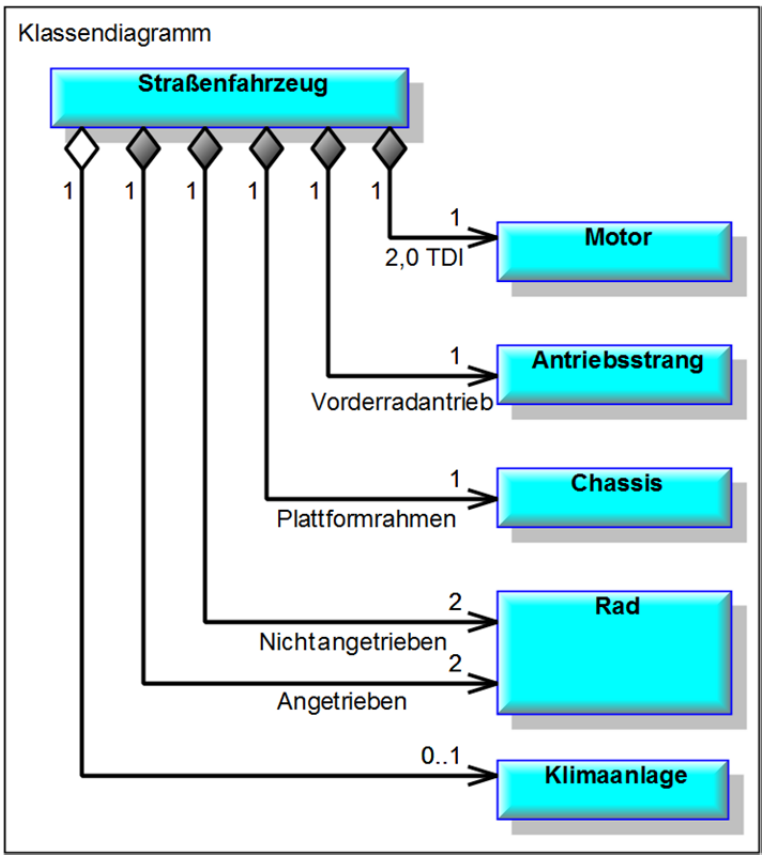


Bild 2-21: Klasse mit Parts in einem Klassendiagramm

In dem gezeigten Diagramm sind die Klassen selbst in Form von blauen Kästen dargestellt sowie die Parts in Form der Beziehungspfeile (die Kompositionen und die Aggregation). Weiterhin werden die Namen der Rollen als Beschriftung der Beziehungspfeile und deren Multiplizität, also deren Anzahl angezeigt. Es existieren beispielsweise insgesamt 4 Räder, wovon zwei die Rolle „Angetrieben“ und zwei weitere die Rolle „Nicht angetrieben“ einnehmen.

Um nun auch die Art der Interaktionen der Parts modellieren zu können, verwendet man ein Diagramm zur Modellierung der internen Struktur einer Klasse, welches in der UML *Kompositionsstrukturdiagramm* heißt. Die interne Struktur der Klasse „Straßenfahrzeug“ wird in Bild 2-22 dargestellt.

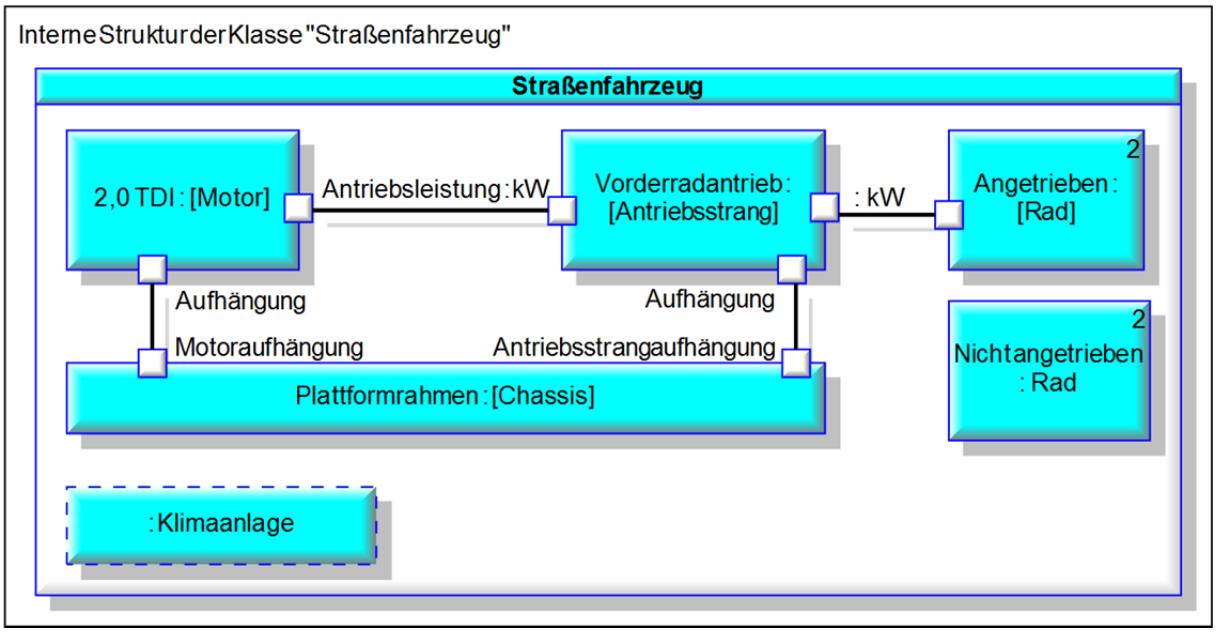


Bild 2-22: Interne Struktur der Klasse „Straßenfahrzeug“ im Kompositionsstrukturdiagramm

In diesem Diagramm ist die Klasse „Straßenfahrzeug“ zu sehen (der „geöffnete“ Kasten) sowie deren Parts (die blauen Kästen). Weiterhin wurden einige Schnittstellen (in UML *Port* genannt) modelliert, welche durch die kleinen Vierecke an den Rändern der Parts dargestellt werden. Diese sind mit Ports anderer Parts über *Assoziationen* verbunden. Damit können die Parts in ihrer ausübenden Rolle mit anderen Parts interagieren und Informationen übermitteln. In der UML sind mit „Informationen“ Daten gemeint. Die SysML versteht darunter allgemein Objektflüsse (Stoff, Energie oder Information), die über diese Schnittstellen übertragen werden können. Weiterhin erkennbar ist, dass das Part „Klimaanlage“ einen gestrichelten Rahmen hat, womit es als Optional kenntlich gemacht wird. Auch dieses Part könnte dennoch mit den anderen über per Assoziation verbundene Ports interagieren. Würde diese optionale Interaktion wegfallen, würde das Straßenfahrzeug trotzdem einen logischen Sinn ergeben (es würde ja auch ohne Klimaanlage funktionieren). Würde dagegen der Motor weggelassen, fehlte die Antriebsleistung, was für ein Straßenfahrzeug dann keinen Sinn ergäbe und weshalb er daher auch per Komposition eingebunden wurde.

Das Prinzip der Rollenbildung durch die Erzeugung von Parts ist ein zentrales Element objektorientierter Modellierung und ist gleichzeitig das fehlerträchtigste Konzept aus Sicht eines Modellierers. Würde man die Klasse „Straßenfahrzeug“ aus dem vorangegangenen Beispiel so modellieren, bedeutet das, **jedes** Straßenfahrzeug hat zwei angetriebene Räder, einen Motor usw. Dies ist aber nicht der Fall, das gilt nur für Fahrzeuge ohne Allradantrieb. Genauso verfügt eine Vielzahl an Straßenfahrzeugen eben nicht über genau 4 Räder, sondern mehr (z.B. LKW) oder weniger (z.B. Motorrad). Hier müsste man korrekterweise eine weitere

Hierarchisierung von Klassen durch Vererbung vornehmen (wie in Kapitel 2.5.2 beschrieben), bis man Klassen mit einer eindeutigen internen Struktur unterscheiden kann. Klassen, die unterschiedliche interne Strukturen aufweisen können, wie es für „Fahrzeug“ oder „Straßenfahrzeug“ der Fall wäre, kann man daher auch als *abstrakt* deklarieren, um zu verdeutlichen, dass es nicht sinnvoll ist, daraus konkrete Objekte zu erzeugen. Instanzen kann man jedoch auch aus abstrakten Klassen bilden, der Modellierer muss sich jedoch im Klaren darüber sein, dass die Struktur einer Eltern-Klasse immer auf seine Kind-Klassen mitvererbt wird. Die korrekte Anwendung der Konzepte objektorientierter Modellierung kann daher durchaus komplex sein und erfordert einige Erfahrung, bringt jedoch automatisch eine konsequente Formalisierung von Systemhierarchien und –strukturen mit sich.

2.6 Objektorientierte Modellierungssprachen

Die Ursprünge der heute weit verbreiteten Sprachen zur objektorientierten Modellierung von Software reichen zurück in die 80er und 90er Jahre des letzten Jahrtausends. Zunächst wurden die kleinsten Bausteine von Software – Nullen und Einsen – durch Assembler und Makroassembler zu größeren Einheiten zusammengefasst, bis sie in *Klassen* und *Objekten* mündeten, die in objektorientierten Programmiersprachen der 1980er wie Smalltalk und C++ zum Einsatz kamen. Sie wurden im Programmcode als abgeschlossene Einheiten mit definierten Eingabe- und Ausgabewerten abgelegt und konnten so von anderen Codebausteinen aufgerufen werden. Dies ermöglichte erstmals die Strukturierung und Wiederverwendung von Programmteilen für regelmäßig wiederkehrende Verarbeitungsschritte innerhalb eines Softwareprogramms.

Da aber ein Programmcode mit zunehmendem Umfang über zigtausende Codezeilen nur schwer überschaubar ist, begann man früh mit der grafischen Visualisierung. So wurden vor allem Algorithmen mit Flussdiagrammen (z.B. dem Nassi-Shneiderman-Diagramm[148]) beschrieben. In den 1990er Jahren entstand eine Vielzahl grafischer Darstellungen, die gleichzeitig auch eine Methode zum planvollen Vorgehen bei der Softwareentwicklung umfassten. Diese waren jedoch so heterogen, dass Begriffe wie „Methodenkrieg“ aufkamen. Daher taten sich drei der prominentesten Vertreter mit jeweils eigenen Ansätzen, GRADY BOOCH (*Booch*[149]), IVAR JACOBSON (*Objectory*[150]) und JAMES RUMBAUGH (*Object Modeling Technique*[151]),

[148] DIN 66261 (1985)
[149] Booch (1993)
[150] Jacobson et al. (1992)
[151] Rumbaugh et al. (1991)

zusammen, um ein gemeinsames Vorgehen zu entwickeln. Die drei Protagonisten, welche zu dieser Zeit alle bei dem gleichen Arbeitgeber Rational Software arbeiten, wurden fortan unter dem Namen „Die drei Amigos“ bekannt. Aufgrund von Problemen bei der Einigung auf eine gemeinsame Methode wurde diese auf die Spezifikation einer gemeinsamen Sprache beschränkt, die Unified Modeling Language (UML). Sie wurde 1997 von der *Object Management Group (OMG)*, einem internationalen Industriekonsortium zur Nutzung objektorientierter Technologien, in der Version 1.1 als Standard anerkannt[152]. Die UML wurde im Jahr 2004 von der *ISO – International Organisation for Standardization* international standardisiert[153]. Die Entstehungsgeschichte der objektorientierten Modellierungssprachen ist in Bild 2-23 dargestellt, wo einerseits die Vielfalt an Sprachen und Methoden und andererseits die Herauskristallisation der UML als Standard deutlich werden.

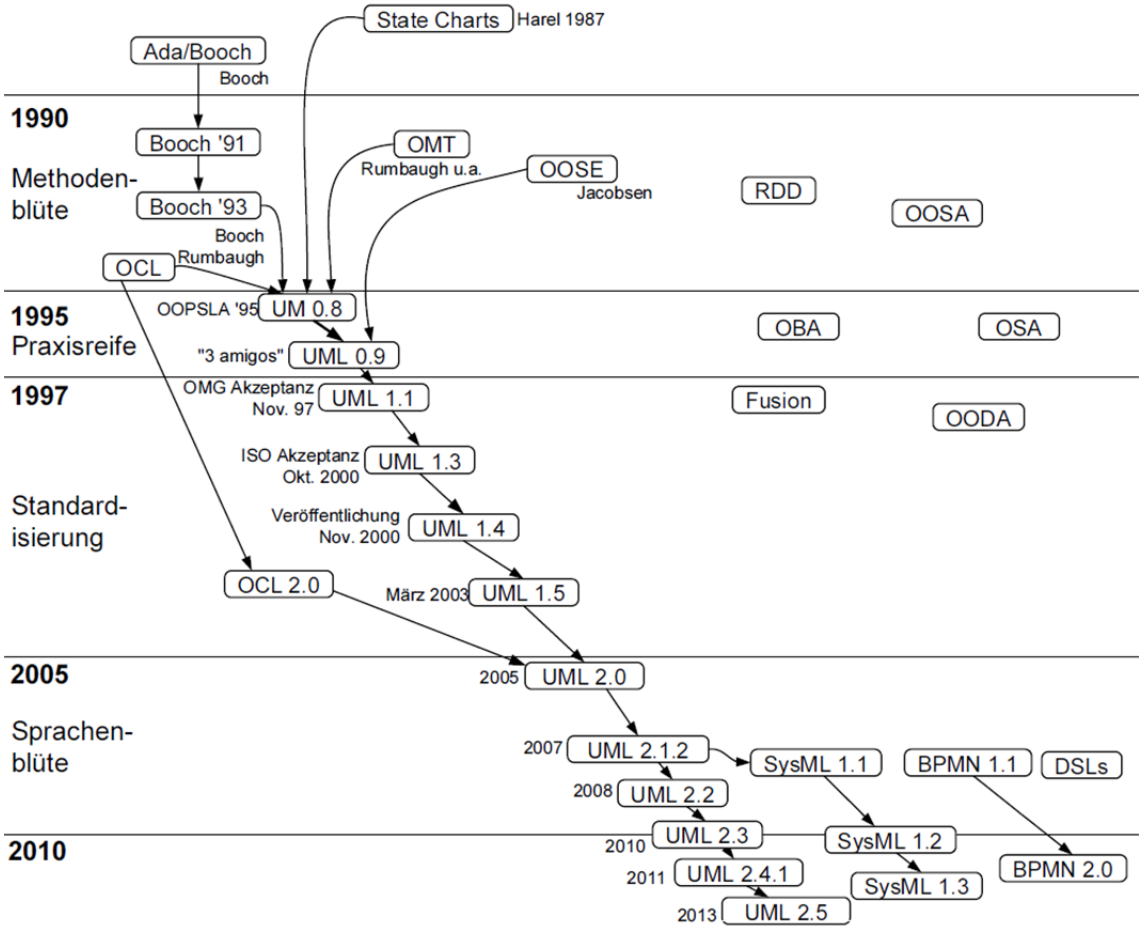


Bild 2-23: Historie der Objektorientierten Modellierungssprachen[154]

[152] Korff (2008), Weilkiens (2008)
[153] ISO 19501 (2005)
[154] Oesterreich et al. (2012)

In den letzten Jahren, die in Bild 2-23 auch als „Sprachenblüte“ bezeichnet werden, entstanden auf Basis der UML weitere Modellierungssprachen, die das Ziel hatten, den erfolgreichen Ansatz zur Beschreibung von Softwaresystemen auch auf andere Systeme zu übertragen. Die bekanntesten Vertreter dieser Modellierungssprachen werden in den folgenden Kapiteln vorgestellt, beginnend mit der Mutter all dieser Sprachen selbst, der UML.

2.6.1 Unified Modeling Language (UML)

Die Entstehungsgeschichte der UML und auch einige ihrer Diagramme, wie das Klassendiagramm und das Kompositionsstrukturdiagramm, wurden bereits in den vorgegangenen Kapiteln vorgestellt. Dieses Kapitel gibt einen kurzen Überblick über den Aufbau der Modellierungssprache. Für weiterführende Informationen sei auf die zahlreiche Literatur wie bspw. das UML Reference Manual in der zweiten Auflage von RUMBAUGH ET AL.[155] verwiesen.

2.6.1.1 Aufbau der UML

Die Spezifikation der UML 2 umfasst vier Teile:

- Superstructure Specification[156]
- Infrastructure Specification[157]
- Object Constraint Language (OCL)[158]
- XML Metadata Interface (XMI)[159]

Der Sprachkern ist in der Infrastructure beschrieben, auf dem die Superstructure mit weiteren Sprachelementen aufbaut. Die Object Constraint Language ist eine eigene Sprache und XMI ein Datenaustauschformat, um UML-Modelle zwischen verschiedenen Werkzeugen austauschen zu können. Seit der UML 2.0 können damit auch Diagramme, also konkrete Sichten auf ein Modell, ausgetauscht werden. Die UML kennt 14 verschiedene Diagrammtypen, die – mit Ausnahme des erst in Version 2.2 hinzugekommenen Profildiagramms – in Bild 2-24 aufgeführt sind.

[155] Rumbaugh et al. (2004)
[156] OMG UML (2011a)
[157] OMG UML (2011b)
[158] OMG OCL (2012), vgl. auch Kapitel 2.8.3
[159] OMG XMI (2011), vgl. auch Kapitel 2.8.7

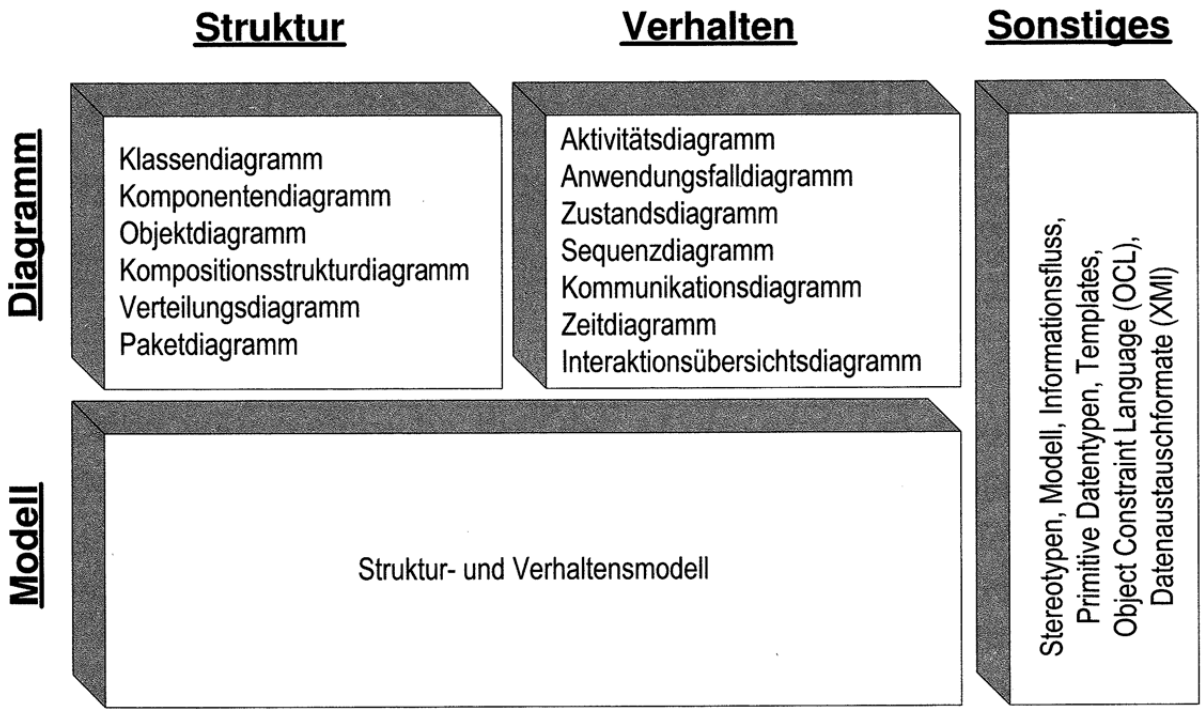


Bild 2-24: Aufbau der UML[160]

Die Modellierungssprache beruht auf einigen grundlegenden Designprinzipien, die im folgenden Kapitel vorgestellt werden.

2.6.1.2 Designprinzipien der UML

Die nachfolgend vorgestellten Designprinzipien[161] repräsentieren gleichzeitig auch die Ziele der UML hinsichtlich der Modellierung von Software und Systemen.

Modularität: Durch die Definition von Sichten in Paketen und die Definition von Eigenschaften in Metaklassen und Klassen wird einerseits eine starke Kohäsion und andererseits eine lose Kopplung einzelner Elemente erreicht. Starke Kohäsion bedeutet hierbei, dass immer ein Element für eine definierte Aufgabe verantwortlich ist. Die Lose Kopplung meint, dass Elemente eines Designs möglichst unabhängig von anderen Elementen des gleichen Designs funktionieren und diese über möglichst schmale, definierte Schnittstellen miteinander gekoppelt werden.

Schichtung: Dieses Prinzip sieht vor, dass eine Modellierungssprachspezifikation Pakete für Basiselemente sowie für Konstrukte enthält, welche diese Basiselemente nutzen. Das Prinzip der Schichtung wird auch im 4-Ebenen-Modell der Hierarchie der Metamodellierung (siehe Bild 2-35 in Kapitel 2.8.2) und in der UML angewendet.

---

[160] Aus Weilkiens (2008)
[161] Nach Korff (2008)

Partitionierung: Die Partitionierung dient der Einteilung der Sprachkonstrukte in konzeptionelle Bereiche innerhalb der Schichtung, was auch als Bildung kohärenter Partialmodelle bezeichnet wird. Dieses Prinzip findet neben der UML und weiteren auf ihr basierenden Sprachen beispielsweise auch in der nicht MOF-konformen[162] Sprache CONSENS[163] Anwendung.

Erweiterbarkeit: Die UML ist eine umfangreiche Modellierungssprache und umfasst sehr viele Aspekte zur Modellierung von Software und Systemen. Dennoch kann sie nicht alle Aspekte sämtlicher Fachdisziplinen abdecken. Daher sieht sie ein Erweiterungskonzept vor, um erweiternde Sprachprofile („Dialekte“) in Profildiagrammen der UML oder die Bildung eigener, mit der UML verwandter Sprachen für die domänenspezifische Modellierung zu definieren. Diese können mit Domain-Specific Languages (DSL)[164] beschrieben werden. Dieses Prinzip sorgt für die hohe Flexibilität der UML und unterstreicht damit ihren Anspruch, „unified“ zu sein. Beispielsweise ist auf diesem Wege auch die SysML entstanden. Auch in der vorliegenden Arbeit wurde das Prinzip zur Realisierung der neuen Modellierungstechnik angewendet.

Wiederverwendbarkeit: Die Metamodell-Elemente der UML können aufgrund ihrer flexiblen modularen Beschreibung in anderen Sprachen wiederverwendet werden.

Diese Prinzipien der Gestaltung von Modellierungssprachen lassen sich auch unmittelbar auf die Ziele bei der Gestaltung konkreter Modelle übertragen. Auch hier sind Modularität, Schichtung etc. anzustreben. Da die Erweiterbarkeit der UML eine zentrale Rolle bei der Entwicklung der SysML, aber auch der in der vorliegenden Arbeit vorgestellten Modellierungstechnik spielt, werden im folgenden Kapitel kurz die Grundlagen zum Profilmechanismus der UML vorgestellt.

2.6.1.3 Profilmechanismus der UML

Die Definition einer komplett eigenen, objektorientierten Modellierungssprache ist sehr aufwändig. Daher bietet die UML als Basissprache bereits eine breite Palette wiederverwendbarer Konstrukte wie beispielsweise Klassen, Attribute oder Beziehungen an, um fachdisziplinspezifische Elemente durch eine Erweiterung mittels des Profilmechanismus hinzuzufügen. Dieser Mechanismus wird auch bereits durch zahlreiche UML-Modellierungswerkzeuge (z.B. MagicDraw von NoMagic oder Artisan Studio von Atego) unterstützt, was die äußerst aufwändige Programmierung einer eigenen Modellierumgebung erspart. Seit Erscheinung der UML 2.2 haben

[162] Siehe Kapitel 2.8.2
[163] Gausemeier et al. (2009a), siehe auch Kapitel 2.6.5
[164] Siehe Kapitel 2.6.3

viele Modellierungswerkzeuge das Profildiagramm aufgenommen, was eine grafische Modellierung der erweiternden Konstrukte des Profilmechanismus erlaubt. Der Funktionsumfang des Mechanismus variiert je nach Modellierungswerkzeug sehr stark, weshalb es sinnvoll ist, dessen Funktionsumfang bei der Auswahl eines geeigneten Modellierungswerkzeugs zu berücksichtigen.

Die Erweiterung vorhandener UML-Metamodellelemente wird durch *Stereotypen* vollzogen. Das in Bild 2-25 gezeigte Beispiel verdeutlicht die Funktionsweise dieser Erweiterung am Beispiel der hinzugefügten „anforderung“.

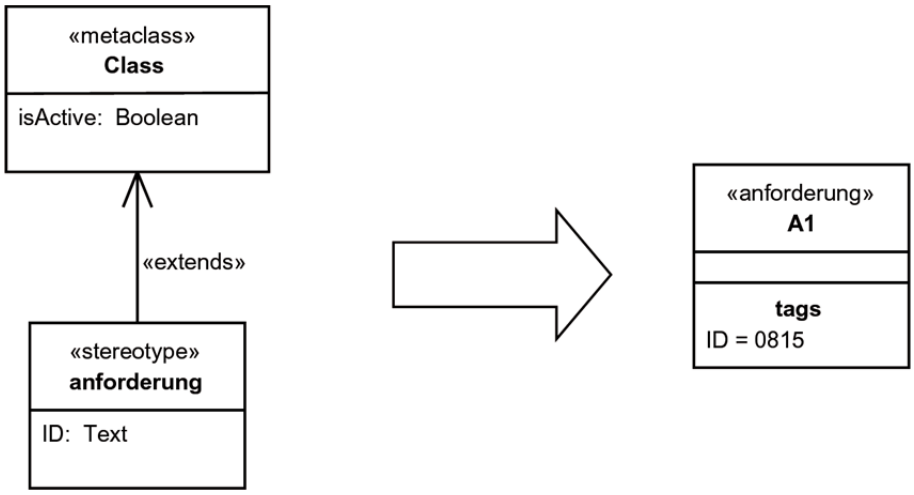


Bild 2-25: Prinzip des Profilmechanismus der UML[165]

Die UML-Metaklasse „Class“ wird durch den Stereotyp „anforderung“ erweitert. Dieser erbt die Merkmale und möglichen Beziehungen seiner Metaklasse. Weiterhin hat der Stereotyp ein neues Merkmal, das *Attribut* „ID“ mit dem Datentyp „Text“, erhalten. Wird nun in einem Modell ein Element des Stereotyps „anforderung“ in einem Modell erzeugt, kann man für diesen neben einem Namen auch einen konkreten Wert für sein Attribut „ID“ vergeben, wie beispielhaft rechts in der Abbildung gezeigt. Attribute werden als sogenannte *tagged values* (kurz: *tags*) in einem eigenen Bereich (engl. *compartment*) angezeigt. Diese kann man zum Hinzufügen von Merkmalen wie die in der Abbildung gezeigte „ID“ oder bspw. auch „Gewicht“ nutzen. Stereotypenbezeichner erkennt man in konkreten Modellen an den französischen Anführungszeichen («»). Neben den Attributen kann man in vielen Modellierungswerkzeugen auch deren Erscheinungsbild in Diagrammen (z.B. Form und Farbe) zur besseren visuellen Unterscheidbarkeit verändern. Diese Funktionalität ist derzeit noch nicht Teil des Profilmechanismus, hat sich jedoch in der Praxis vielfach bewährt und findet auch in der vorliegenden Arbeit umfassend

[165] aus Alt (2012)

Anwendung, insbesondere um der Anforderung der anwendergerechteren Darstellung von Elementen in Diagrammen gerecht zu werden.

Beispiele für Profile der UML sind neben der SysML auch die ModelicaML (siehe Kapitel 2.6.10), BPMN (siehe Kapitel 2.6.4) oder die Erweiterungen des SysML-Profils, die in Kapitel 2.6.2.6 vorgestellt werden.

2.6.2 Systems Modeling Language (SysML)

Die Systems Modeling Language (SysML) ist ein Dialekt der in der Softwareentwicklung weit verbreiteten UML. Ihre Entwicklung wurde 2001 von der INCOSE[166], später in Zusammenarbeit mit der OMG[167] mit dem Ziel begonnen, die UML zur Standardsprache des Systems Engineering zu machen. Hierzu wurden notwendige Artefakte (Entitäten und Relationen) in die Sprache aufgenommen sowie einige UML-Diagramme modifiziert und gleichzeitig einige zu softwarespezifische Artefakte und Diagramme ausgeschlossen. Das Ergebnis wurde SysML getauft. Die Modellierungssprache unterstützt die Spezifikation, Analyse, Design, Verifikation und Validierung eines breiten Bandes von Systemen und Subsystemen. Diese Systeme können Hardware, Software, Informationen, Prozesse, Personal und Anlagen/Einrichtungen enthalten. Die SysML wurde im September 2007 als Version 1.0 erstmals als OMG-Standard angenommen und gewinnt seitdem insbesondere durch die Möglichkeit der benutzerspezifischen Erweiterbarkeit des Sprachprofils und der Kompatibilität mit zahlreichen weiteren Standards wie MOF, STEP oder XML immer mehr an Bedeutung in Forschung und Industrie.

2.6.2.1 Zweck und Ziel der SysML

Der Zweck der SysML ist einerseits die Vereinheitlichung der Sprache der Systemingenieure mit dem Ziel einer verbesserten Kommunikation und Zusammenarbeit in multidisziplinären Entwicklerteams und andererseits die formale Beschreibung von systemübergreifenden Zusammenhängen mit dem Ziel, Arbeitsprodukte wie Code, Testfälle oder Dokumentationen (teil-)automatisiert erzeugen zu können. Die Sprache soll in diesem Kontext jedoch keine bestehenden Entwicklungswerkzeuge ersetzen, sondern sinnvoll ergänzen, um das Fundament für eine durchgängige Entwicklungsumgebung bereitzustellen (siehe Bild 2-26)[168].

[166] International Council on Systems Engineering
[167] Object Management Group
[168] Alt (2012)

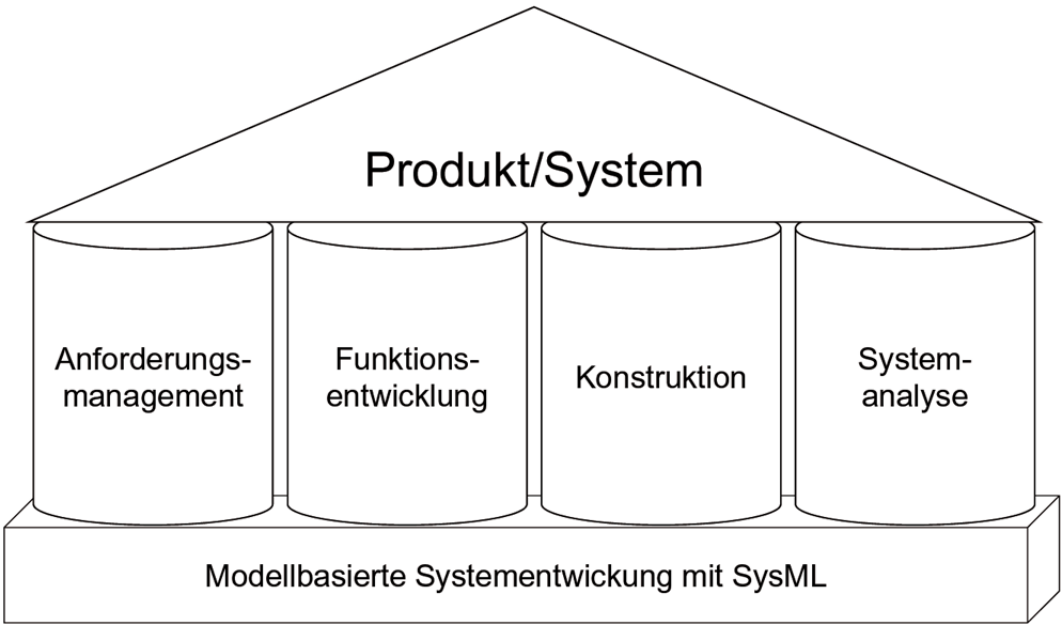


Bild 2-26: SysML als Fundament der Systementwicklung[169]

Auch wenn es prinzipiell möglich wäre, Systeme bis ins kleinste Detail mit SysML zu beschreiben, ist es nicht sinnvoll, da es dafür meist deutlich spezialisiertere und leistungsfähigere Werkzeuge gibt. Der Grad der Detaillierung in der Modellierung mit SysML ist vor jedem Einsatz abzuwägen, wobei die Modellierung von disziplinübergreifenden Interaktionen und Wechselwirkungen maßgebend ist: sind Informationen bezüglich des zu modellierenden Systems zwischen verschiedenen Werkzeugen oder Entwicklerteams zu kommunizieren, sollten diese auch in SysML modelliert werden, um den erforderlichen Übertragungspfad (engl. traceability) und ihre Nachvollziehbarkeit herzustellen. Da die SysML die Basis für die Entwicklung von Systemen darstellt und Informationen anwendergerecht distributiert und synchronisiert, kann mit ihrer Verwendung auch eine parallelisierte Entwicklung (engl. Simultaneous Engineering oder Concurrent Engineering) in den Fachdisziplinen stattfinden. Das SysML-Modell sorgt hierbei für Konsistenz und Aktualität der für die beteiligten Entwickler relevanten Informationen. Dies sorgt letztlich für das gewünschte, gemeinsame Systemverständnis aller Beteiligten des zu entwickelnden Produkts.

In den folgenden Kapiteln werden das Verhältnis der SysML zur UML, die durch die SysML bereitgestellten Diagrammarten, die aktuellen Entwicklungsfortschritte des Standards und relevante Fachliteratur vorgestellt. Darüber hinaus werden weiterführende, wissenschaftliche Ansätze unter Verwendung der SysML und die Verbreitung der Sprache in Forschung und Industrie diskutiert.

[169] Alt (2012)

2.6.2.2 Verhältnis von SysML und UML

Die SysML wurde auf Basis der UML[170] entwickelt. Hierzu wurden einige Konstrukte der UML direkt wiederverwendet, weitere modifiziert und wieder andere ganz weggelassen. Diese Maßnahmen wurden technisch durch ein Profil realisiert, das aus Stereotypen und Bezeichnern (*tags* zur Bildung von Attributen und Relationen) besteht und unter Anwendung des Meta-Object-Facility (MOF)-Standards[171] beschrieben wurde (siehe Bild 2-27).

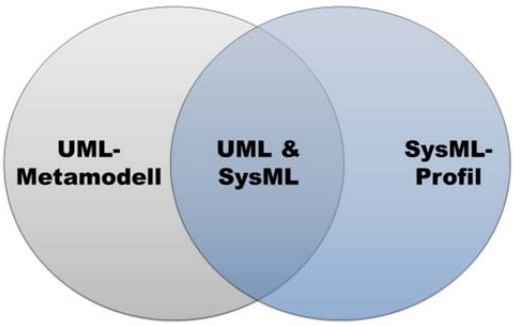


Bild 2-27: Zusammenhang zwischen SysML und UML

Der Zweck der SysML war eine Erweiterung des Anwendungsbereichs der etablierten Softwaremodellierungssprache UML auf Systeme beliebiger Art. Daher wurden insbesondere jene Konstrukte, die softwarespezifisch waren, angepasst oder aus der Spezifikation ausgegrenzt. Beispielsweise gibt es in der UML 14 verschiedene Diagrammarten, in der SysML nur noch 9 (vgl. Kapitel 2.6.2.3), wodurch sie auch etwas leichter erlernbar ist als die UML. Das Weglassen von Konstrukten wurde in der SysML-Spezifikation zwar beschrieben, jedoch ist die technische Umsetzung als Profil der UML dazu eigentlich nicht in der Lage. Tatsächlich sind alle am Markt erhältlichen SysML-Modellierungswerkzeuge auch UML-Modellierungswerkzeuge, womit die darin erstellbaren Modelle nie reine SysML-Modelle sind, sondern eine Mischung aus SysML und UML[172].

Beispiele für modifizierte UML-Artefakte in der SysML sind in Tabelle 2-1 aufgeführt.

[170] OMG UML (2011a), OMG UML (2011b)
[171] Siehe Kapitel 2.8.2
[172] Weilkiens (2008), Alt (2012)

Tabelle 2-1: Modifikation von UML-Artefakten in SysML (Auszug)

UML - Modellartefakt	modifiziertes SysML - Modellartefakt	Technische Beispiele
Class (Softwarekomponente)	*Block* (Systemkomponente)	Zahnrad, Welle, Steuergerät…
Role (Class im Systemkontext)	*Block Property (Part)*	Zahnrad (auf einer Welle und im Eingriff mit einem weiteren ZR)
Attribute (Merkmal)	*Block Property (Value)*	Gewicht [kg], Modul etc.
Standard Port (Softwareschnittstelle)	*Port* (Systemschnittstelle)[173]	Keilwellenprofil, Netzanschluss, CAN-Bus-Buchse

2.6.2.3 Diagrammarten der SysML

Die SysML ist eine grafische Modellierungssprache und verfügt über insgesamt neun Diagrammarten zur Modellierung verschiedener Aspekte der Struktur und des Verhaltens eines Systems sowie zur Anforderungsmodellierung. Die Verhaltensdiagramme wurden fast komplett aus der UML 2 übernommen, lediglich das Aktivitätsdiagramm wurde modifiziert. Völlig neu hinzugekommen ist das Anforderungsdiagramm, was technisch eine spezialisierte Form eines UML-Klassendiagramms darstellt. Somit ist es auch möglich, eine Vielzahl an SysML-Elementen in einem Anforderungsdiagramm darzustellen und mit Anforderungen zu vernetzen (beispielsweise mit der Erfüllungsbeziehung *satisfy*). Die Strukturdiagramme wurden – mit Ausnahme des Paketdiagramms - umfassend modifiziert, um dem breiteren Spektrum an Systemen gerecht zu werden, dessen Strukturen nun auch modellierbar sein sollen. Ebenfalls neu hinzugekommen ist das Zusicherungsdiagramm, mit dem es möglich ist, Abhängigkeiten zwischen verschiedenen Elementen abzubilden sowie parametrische Zusammenhänge zu beschreiben. Bild 2-28 zeigt die Diagrammarten in einer Baumstruktur.

[173] Seit SysML V1.3, vorher *Flow Port*. Vgl. Kapitel 2.6.2.4 und OMG SysML (2012)

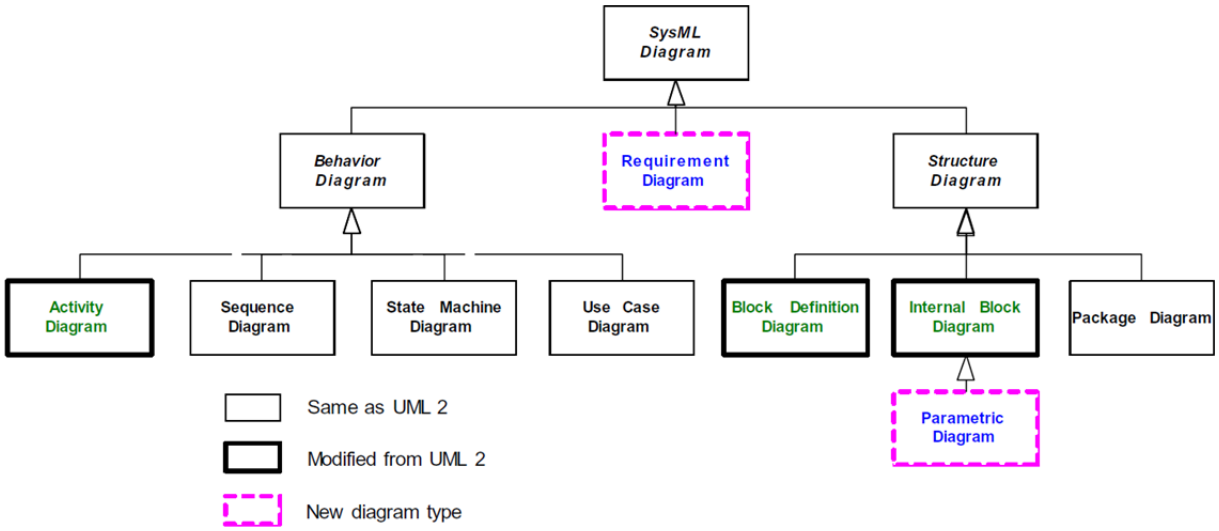


Bild 2-28: Diagrammarten der SysML[174]

Die Verwendung der einzelnen Diagrammarten wird in Kapitel 5 im Rahmen der Beschreibung des Vorgehensmodells der neuen Modellierungstechnik näher vorgestellt. Detaillierte Informationen zu den Diagrammarten und deren Anwendung finden sich in der in Kapitel 2.6.2.5 benannten Fachliteratur.

2.6.2.4 Aktuelle Weiterentwicklungen des Standards

Im Juni 2012 wurde die Version 1.3 der Spezifikation von der OMG freigegeben[174]. Die wesentlichen Neuerungen darin beziehen sich auf ein neues Konzept zur Schnittstellenmodellierung. Statt *Standard Ports* (UML) und *Flow Ports* (SysML) kommen nun *Full Ports* zur Definition von Schnittstellenmerkmalen und Flussgrößen ihres besitzenden *Blocks* und *Proxy Ports* als Stellvertreter für Schnittstellenmerkmale interner Bausteine (*Parts*) ihres besitzenden *Blocks*. Als Typ kann der neu eingeführte *Interface Block* zugewiesen werden, der neben der Spezifikation von Flussgrößen (z.B. ihrem Wertetyp, der Einheit und der Dimension) auch die Modellierung von Schnittstellenmerkmalen erlaubt, die keine Flussgröße sind. Vor der Einführung des neuen Port-Konzepts mangelte es SysML einerseits an der Möglichkeit, Merkmale von Schnittstellen zu definieren, die nicht übermittelt werden. Andererseits fehlte die Möglichkeit, zwischen der „echten“ Schnittstelle der kommunizierenden Komponenten und „virtuellen“ Schnittstellen zu unterscheiden. Dies kann für das Systemverständnis von hoher Bedeutung sein, würde beispielsweise ein größeres Teilsystem durch ein anderes ersetzt, und man müsste die „inneren Schnittstellen“ berücksichtigen (siehe Beispiel in Bild 2-29).

[174] OMG SysML (2012)

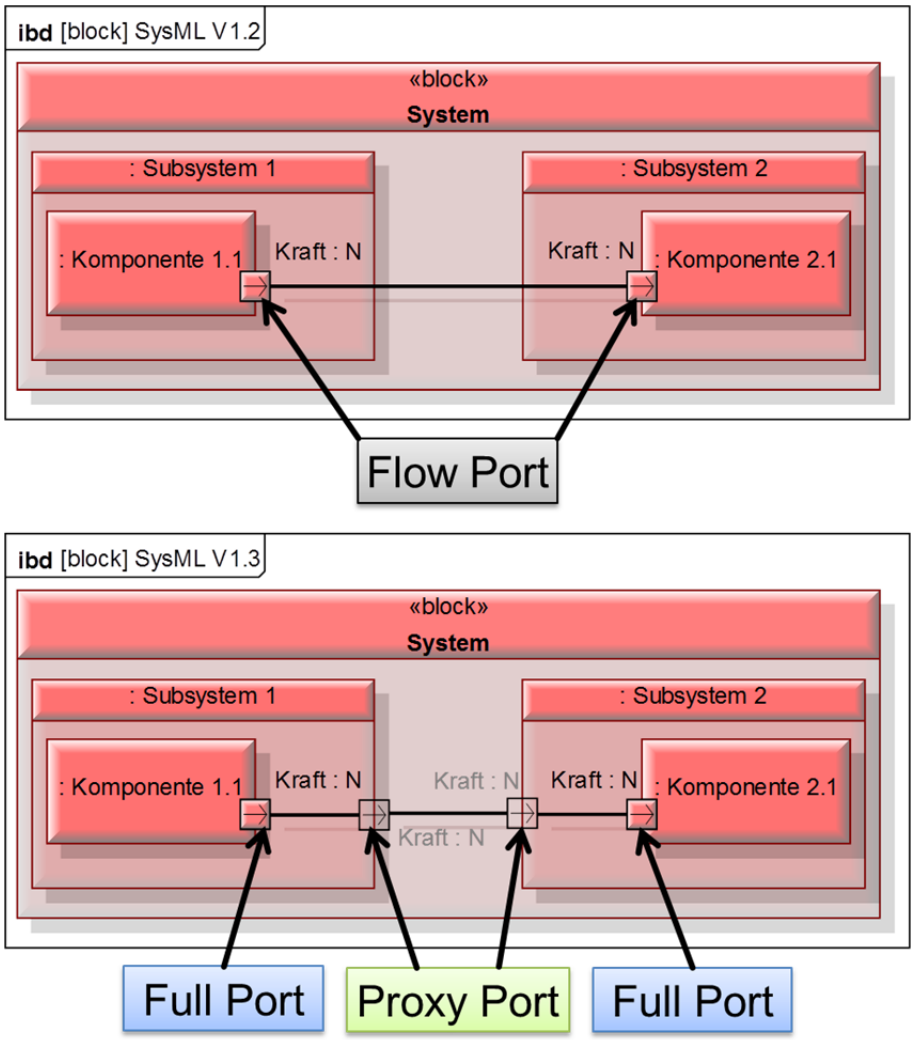


Bild 2-29: Strukturdiagramm ohne (oben) und mit Proxy Ports (unten)

Im gezeigten Beispiel haben die Komponenten 1.1 und 2.1 tatsächlich Schnittstellen, über die sie verbunden sind. Die Subsysteme 1 und 2 wissen davon in der alten Darstellung (links) nichts, mit SysML 1.3 jedoch wird diese Kommunikation ihrer Subsysteme durch Proxy Ports (rechts) dargestellt. Somit ist der Entwickler in der Lage, diese Interaktion auch bei einer Darstellung der Subsysteme 1 und 2 ohne deren innerer Struktur zu erkennen. Beispielsweise wird so die Kommunikation zweier Steuergeräte (Komponenten) erkennbar, die zwei verschiedenen Subsystemen (z.B. Motor und Getriebe) zugeordnet sind, auch wenn ein Diagramm nur die Subsystemebene darstellt.

Darüber hinaus ist nun auch die Modellierung und Darstellung von eingebetteten Ports (*embedded Port*) möglich, was die Kombination mehrerer auch verschiedenartiger Flüsse, wie beispielsweise die eines Signals, das aus Strom (Energiefluss) und Daten (Informationsfluss) besteht, erlaubt. Diese Funktionalität wurde jedoch bisher nicht im verwendeten Werkzeug (Artisan Studio) implementiert. Derzeit befindet sich die Version 1.4 der SysML-Spezifikation in der Entwicklung, die

sich vor allem auf die Weiterentwicklung der Variantenmodellierung mit SysML fokussiert. Mit ihrer Erscheinung wird in OMG-Kreisen derzeit frühestens im dritten Quartal 2013 gerechnet.

2.6.2.5 Fachliteratur und Zertifizierungsprogramm der OMG

In den letzten Jahren sind diverse Fachbücher zur SysML erschienen. Im deutschen Raum sind die bekanntesten Vertreter die Werke von WEILKIENS[175], KORFF[176] und ALT[177]. Das englischsprachige Fachbuch von FRIEDENTHAL, MOORE UND STEINER[178] ist gleichzeitig das Referenzwerk von INCOSE und OMG für das seit 2010 angebotene Zertifizierungsprogramm OCSMP (OMG-Certified Systems Modeling Professional)[179].

2.6.2.6 Weiterführende, wissenschaftliche Ansätze unter Verwendung der SysML

Der steigende Bekanntheitsgrad in Verbindung mit ihrem Neuheitswert, multidisziplinäre Systeme modellbasiert beschreiben zu können, führte in der jüngeren Vergangenheit zu zahlreichen wissenschaftlichen Arbeiten mit Verwendung oder Weiterentwicklung der SysML. Dieses Kapitel beschränkt sich daher vorwiegend auf jene Ansätze, die eine Erweiterung der SysML hinsichtlich funktionaler Modellierung vorstellen, da eine solche Erweiterung auch Teil der Implementierung des Contact & Channel – Ansatzes in SysML ist, die in Kapitel 4.3.3 vorgestellt und in Kapitel 6.3 ausführlich erläutert wird.

LAMM UND WEILKIENS[180] stellen die Methode „Funktionale Architekturen für Systeme (FAS)“ vor, die eine Erweiterung der SysML durch ein FAS-Profil sowie eine Handlungsanweisung zur funktionalen Modellierung in SysML umfasst. Hierzu werden zunächst funktionale und nichtfunktionale Anforderungen modelliert. Die funktonalen Anforderungen werden anschließend durch Anwendungsfälle verfeinert, deren interner Ablauf durch Aktivitäten ausdetailliert wird. Diese können unter Verwendung von Heuristiken in funktionale Gruppen zusammengefasst werden, für die entsprechende *funktionale Blöcke* (Spezialisierung eines SysML-Blocks) erzeugt werden. Weiterhin werden für die an den Aktivitäten modellierten Objektflüsse automatisch die passenden Schnittstellen (*Ports*) der funktionalen Blöcke generiert. Diese funktionalen Blöcke können nun in einem *Internal Block Diagram* zu einer funktionalen Struktur vernetzt werden. Diese funktionalen Elemente können nun

[175] Weilkiens (2008)
[176] Korff (2008)
[177] Alt (2012)
[178] Friedenthal et al. (2011)
[179] Weitere Informationen finden sich auf http://ww.omg.org
[180] Lamm und Weilkiens (2010)

physischen Elementen, modelliert durch „normale“ SysML-Blöcke, zugewiesen werden, was jedoch bereits nicht mehr Teil der Methode ist. Diese Schritte sind schematisch in Bild 2-30 in einem Aktivitätsdiagramm dargestellt.

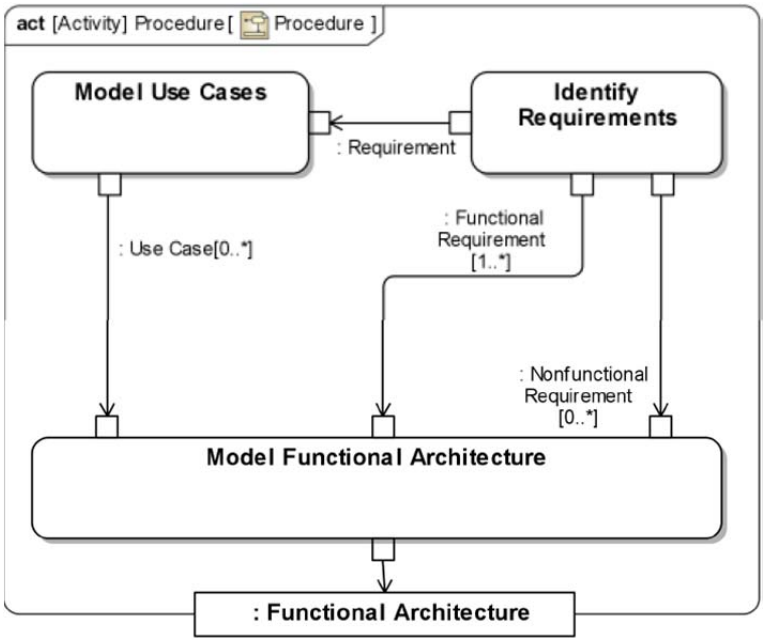


Bild 2-30: Ablauf der Modellierung nach der FAS-Methode[181]

Inzwischen wurden für die Weiterentwicklung der Methode, die in der Fachwelt auf weitreichende positive Resonanz stößt, eine eigene GfSE-Arbeitsgruppe gegründet und eine eigene Website eingerichtet[182]. Weiterhin wurden bereits Implementierungen für mehrere Modellierungswerkzeuge vorgestellt, beispielsweise für Artisan Studio von atego[183].

EIGNER ET AL.[184] stellen eine Methode, basierend auf dem um den AFLP[185]-Ansatz erweiterten V-Modell, vor (siehe Kapitel 2.10.4) und verwenden die um ein anwenderspezifisches Profil erweiterte SysML zur funktionalen Beschreibung von Produkten. In der Methode werden drei Ebenen der Modellierung unterschieden:

- Modellbildung und Spezifikation: hier kommen qualitative Beschreibungsmodelle zum Einsatz, die u.a. Anforderungen, Funktionen und Strukturen umfassen. Als favorisiertes Beschreibungswerkzeug wird die SysML benannt.

[181] Lamm und Weilkiens (2010)
[182] http://www.fas-method.org
[183] Korff et al. (2011)
[184] Eigner et al. (2012a), Eigner et al. (2012b)
[185] Anforderungen, Funktionen, Logische Struktur, Physische Struktur

- Modellbildung und erste Simulation: das Ziel auf dieser Ebene ist die Erstellung quantitativer, simulierbarer Modelle unter Einbezug mehrerer Fachdisziplinen. Geeignete Modellierungswerkzeuge sind bspw. Dymola oder Matlab/Simulink.
- Disziplinspezifische Modellbildung: hier werden konkrete physische Modelle in CAx-Werkzeugen modelliert. Hierzu zählen sowohl Softwarecodes als auch Platinenlayouts, Stromlaufpläne, CAD- oder Finite-Elemente-Modelle.

Das im interdisziplinären Systementwurf entstehende Wissen in der Anforderungsdefinition, dem Entwurf von funktionaler und logischer Struktur sowie deren iterativer Verfeinerung durch erste virtuelle Tests ist nur schwer festzuhalten und zu vernetzen. Hierzu soll ein PDM-System mit einem System-Spezifikationsmodell in SysML und XMI als Datenaustauschformat eingesetzt werden. Damit die SysML der angestrebten Modellierungsmethode gerecht wird, wurden neue Entitäten (z.B. der Stereotyp „Function“ als Spezialisierung von *Block*) und Relationen (z.B. „realisiert“ zur Zuweisung von logischen Elementen wie eines Controllers zu funktionalen Elementen wie der Funktion „Intervall berechnen“[186]) in Form eines Profils hinzugefügt. Hierarchien und Querverweise innerhalb des SysML-Modells sowie typisierte Verbindungen zwischen verschiedenen Modellen können per XML-Dateien in einem PDM-System hinterlegt werden. Hier wird jedoch vorausgesetzt, dass das zu verwendende PDM-System bereits die Verwaltung von Anforderungen und Stücklistenstrukturen unterstützt, zwischen denen die funktionale Produktbeschreibung integriert wird. Bild 2-31 zeigt den schematischen Aufbau dieses Konzepts.

[186] Beispiel aus Eigner et al. (2012a)

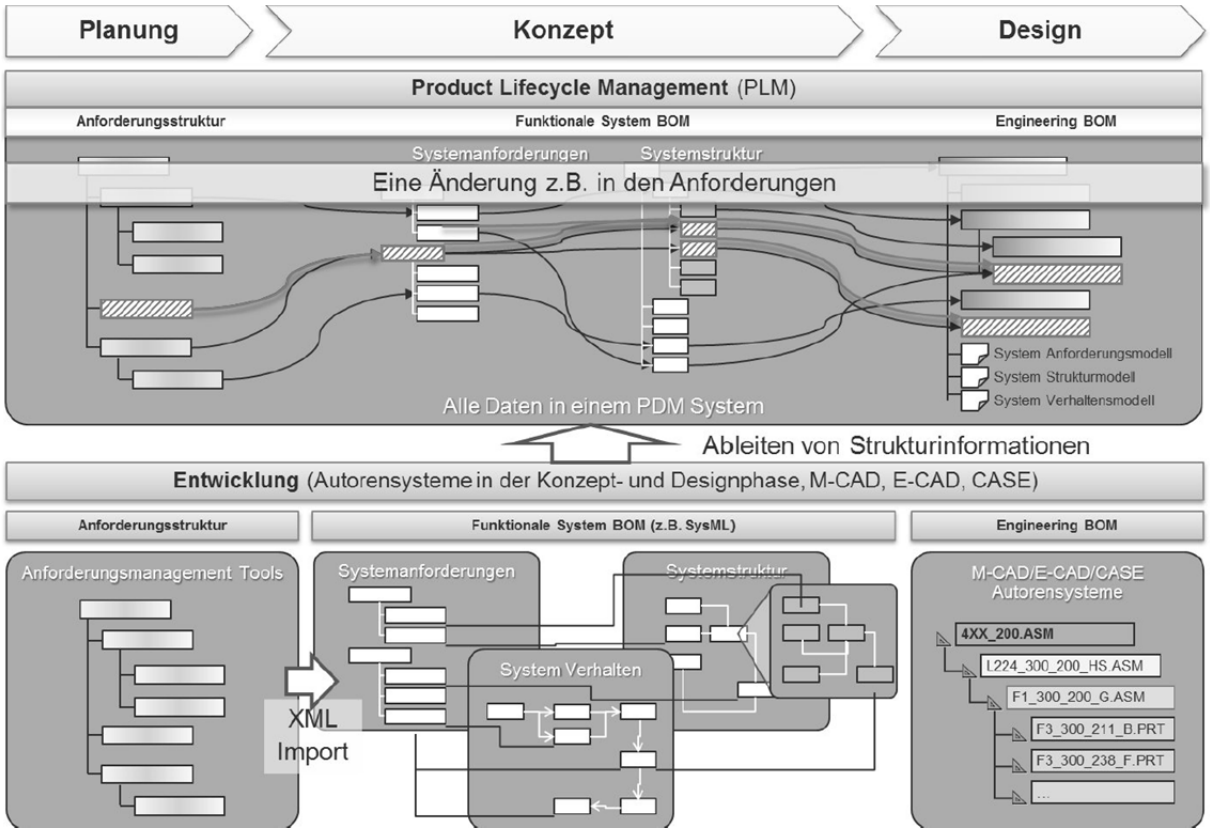


Bild 2-31: Funktionale Produktbeschreibung als Teil eines PLM Konzepts[187]

In diesem Konzept ist vorgesehen, dass logische Systemelemente die Realisierung von Funktionen mit einem physikalischen Effekt darstellen und hierzu Merkmale definieren. Weiterhin können hierüber externe Modelle (z.B. Modelica oder Matlab/Simulink) referenziert werden. Diese logischen Systemelemente sind mit der physikalischen Stückliste (BOM) verknüpft, welche wiederum M-CAD oder E-CAD-Modelle sowie Softwaremodelle aufnimmt.

Der Ansatz nach EIGNER ET AL. greift das Konzept der FAS-Methode nach LAMM UND WEILKIENS auf. Weiterhin ist die Spezifikation physikalischer Effekte und Merkmale in logischen Systemelementen ein Konzept, dass auch bei der Integration des Contact & Channel – Ansatzes in SysML angewendet wird, jedoch steht hier nicht die Integration in PDM-Systeme im Fokus, sondern die vollständige Beschreibung von Funktionen multidisziplinärer Systeme und die Transformation auf resultierende physische Strukturen.

WÖLKL UND SHEA stellen einen Ansatz zur Verwendung der SysML für die Entwicklung von computergestützten Modellen für die frühe Phase der

[187] Eigner et al. (2012b)

Produktentwicklung mit Fokus auf mechanische Systeme vor[188]. Deren Ziele sind der Übergang von der dokumentenzentrierten hin zur modellbasierten Entwicklung und die Verbesserung der Zusammenarbeit der Ingenieure verschiedener Disziplinen. Hierzu wurde die SysML als mögliche Sprache hinsichtlich ihrer Eignung zur Modellierung mechatronischer Systeme untersucht und erweitert. Basierend auf der VDI 2221[189] als aus deren Sicht etabliertestes Vorgehensmodell für den Maschinenbau werden Anforderungen an Modelle zur Unterstützung der Aufgabenklärung und der Produktkonzipierung definiert. Diese umfassen u.a. Anforderungs-, Funktions-, Struktur- und Verhaltensmodellierung in Anlehnung an Modellbildungsansätze wie FBS (Function-Behavior-State)[190].

Funktionen werden dort zunächst aus Anwendersicht durch Use-Cases und aus Systemsicht durch Aktivitätsdiagramme beschrieben. Gemäß dem Ansatz zur Beschreibung von Funktionen nach PAHL ET AL. werden Funktionen hierbei lösungsneutral als Substantiv-Verb-Paar zur Transition von Eingangsgrößen in Ausgangsgrößen beschrieben[191]. Weiterhin empfehlen WÖLKL UND SHEA die Implementierung der NIST-Funktionsdatenbank nach HIRTZ ET AL. zur einheitlichen Dekomposition in Basisfunktionen[192]. In einem weiteren Schritt werden Wirkprinzipien in Blockdiagrammen modelliert und auf die Aktivitäten analog dem „Behavior“-Level des FBS-Ansatzes allokiert. Darin repräsentieren Blöcke (*Block*) Klassen von Wirkprinzipien und Zusicherungen (*Constraints*) beschreiben physikalische Zusammenhänge. Wirkstrukturen werden durch vernetzte Wirkprinzipien repräsentiert. Die eigentliche Zuordnung von Funktionen zu Wirkprinzipien findet hierbei auf Basis von Konstruktionskatalogen statt, folglich unterstützt das Modell hierbei nicht aktiv (es wird keine Beziehung zwischen diesen Elementen im Modell hergestellt). Auch stellen die Autoren fest, dass aus Sicht eines Maschinenbauers eine vereinfachte, grafische Darstellung mechanischer Systeme fehlt.

Dieser Ansatz einer Implementierung des FBS-Ansatzes und der NIST-Datenbank in SysML wurde von KRUSE ET AL. weiterentwickelt[193]. Durch eine Profilerweiterung können benutzerspezifische Funktionen oder Elementarfunktionen der implementierten NIST-Datenbank in Form stereotypisierter Activities verwendet werden. Die Zuordnung von Activities auf Strukturen wird über Allokationen realisiert,

[188] Wölkl und Shea (2009)
[189] VDI-2221 (1993)
[190] Vgl. Kapitel 2.4.6
[191] Vgl. Pahl et al. (2007)
[192] Vgl. Hirtz et al. (2002)
[193] Kruse et al. (2012)

die auch in einer entsprechenden Matrix dargestellt werden können, eine Standardfunktion vieler SysML-Modellierungswerkzeuge. Die NIST-Flussdatenbank wurde ebenfalls per Profilerweiterung implementiert und um einige Mechanik-spezifische Flüsse wie bspw. translatorisch-mechanische Energie erweitert. Die Konkretisierungsstufen der Modellierung sind in Bild 2-32 dargestellt.

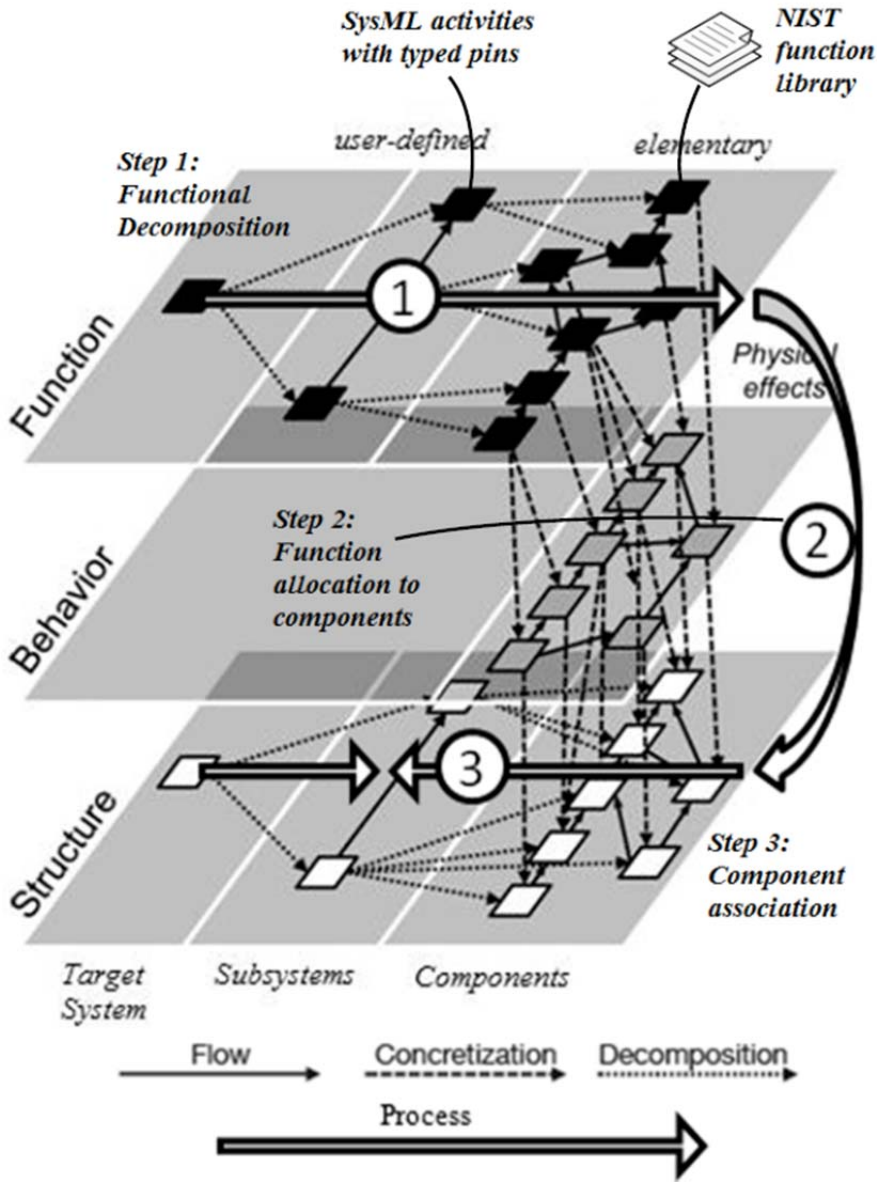


Bild 2-32: Function-Behavior-Structure – Methode für SysML[194]

Die Modellierung des Systemverhaltens wurde bisher nicht realisiert. Die Modellierungsmethode wird am Beispiel eines einfachen Modells eines Elektrofahrzeugs evaluiert. Die funktionalen Modelle in Aktivitätsdiagrammen wurden bisher ohne Kontrollflüsse modelliert und werden als statisch angesehen. Das Fehlen

[194] Aus Kruse et al. (2012)

funktionaler Blöcke im Gegensatz bspw. zur FAS-Methode nach LAMM UND WEILKIENS[195] wird durch eine höhere Benutzerfreundlichkeit aufgrund der Beschränkung auf einen Diagrammtyp zur Modellierung von Funktionen argumentiert. Hieraus ergibt sich auch die Schwäche des vorgestellten Ansatzes. Es ist weder möglich, dynamische Strukturen noch Zustandsabhängigkeiten der Funktionen zu modellieren. Darüber hinaus wird die Konkretisierung zu Strukturen ausschließlich über Allokationen vorgenommen, die eine erneute Modellierung der Schnittstellen zur Übertragung der Objektflüsse erfordert und damit fehleranfällig ist. Die in dieser Arbeit vorgestellte Modellierungstechnik stellt hierfür Vererbungsmechanismen und Automatismen für die Erstellung der passenden Schnittstellen für die Objektflüsse bereit. Die Integration der NIST-Datenbanken für Funktionen und Objektflüsse bzw. Schnittstellen wurde u.a. bereits von ALBERS ET AL. in einer anderen Werkzeugumgebung realisiert[196].

SHAH ET AL.[197] stellen einen Ansatz zur Modellierung von Eingebetteten Systemen (eigenständige Softwaresysteme innerhalb eines übergeordneten mechatronischen Systemverbunds) in SysML vor. Hier liegt der Fokus vor allem in der Beschreibung der Wechselwirkungen zwischen verschiedenen Disziplinen und der automatischen Generierung fachdisziplinspezifischer Modelle aus einem übergeordneten, multidisziplinären Systemmodell in SysML. Hierzu sollen verschiedene Sichten in der SysML durch eine Erweiterung des SysML-Profils implementiert werden sowie durch domänenspezifische Modellierung und Modelltransformationen verschiedene Domänen und deren Beziehungen beschrieben werden. Das dort vorgestellte Framework ist kompatibel zu den Standards MOF[198], MDA[199] und SysML, um es aufwandsarm in eine Vielzahl existierender Werkzeuge und Frameworks integrieren zu können. Das Schema des Frameworks ist in Bild 2-33 visualisiert.

[195] Lamm und Weilkiens (2010)
[196] Vgl. Albers et al. (2011b) oder auch Kapitel 2.4.9.2
[197] Shah et al. (2009), Shah et al. (2010)
[198] Siehe Kapitel 2.8.2
[199] Siehe Kapitel 2.8.1

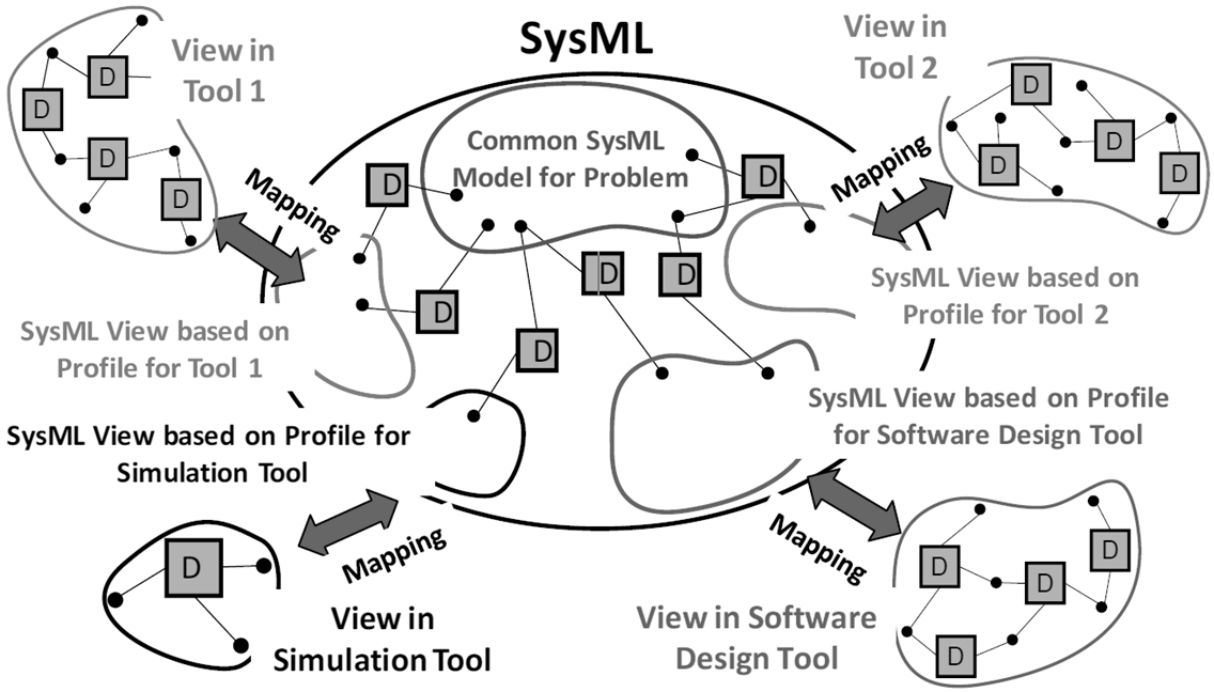


Bild 2-33: Schema eines Frameworks zur Kopplung disziplinspezifischer Modelle über SysML nach SHAH ET AL.[200]

Die in der Grafik dargestellte Zuordnung erfolgt über Modelltransformationen, welche die Existenz MOF-konformer Metamodelle der disziplinspezifischen Modelle voraussetzt (hier wurden in einem Anwendungsbeispiel u.a. das E-CAD-Werkzeug EPLAN und die Simulationssprache Modelica mit SysML vernetzt). Diese wurden ebenso wie die Transformationsregeln mit dem Metamodellierungswerkzeug MOFLON spezifiziert. Für die Erweiterung der SysML um ein Profil zur Darstellung der disziplinspezifischen Sichten wurden DSL's (siehe Kapitel 2.6.3) eingesetzt.

Die Erweiterung der SysML wurde hier weniger mit dem Fokus funktionaler Modellierung als vielmehr mit dem Ziel, disziplinspezifische Elemente in SysML explizit modellieren zu können, vorgenommen. Dennoch wurde hier erfolgreich demonstriert, wie man die SysML durch Erweiterungen in eine Werkzeugkette mit automatischer Modelltransformation einbetten kann. Diese Ansätze werden in der vorliegenden Arbeit in Kapitel 6.5 aufgegriffen, um die technische Machbarkeit einer Einbettung der vorgestellten Modellierungstechnik in eine IT-Entwicklungsumgebung aufzuzeigen.

2.6.2.7 Verbreitung der SysML in Forschung und Industrie

Trotz ihrer zunehmenden Verbreitung, vor allem in Forschungs- und industriellen Pilotprojekten[201], hat die SysML bis heute keine zufriedenstellende Akzeptanz in

[200] Shah et al. (2010)

allen Fachdisziplinen erlangt[202]. Dies begründet sich insbesondere auf ihrer UML-Abstammung und dem generischen Charakter ihrer Modellkonstrukte, was die Sprache und ihre Repräsentationen besonders Entwicklern der klassischen Maschinenbaudisziplinen wie Konstruktionstechnik, Fluidtechnik, Werkstofftechnik oder Technischer Mechanik nur schwer zugänglich macht. Darüber hinaus klafft weiterhin eine Lücke zwischen der generischen Systembeschreibung in SysML und den spezialisierten, sich im produktiven Einsatz befindlichen Computerwerkzeugen, was sich besonders deutlich im Bereich der 3D-CAD-Konstruktion äußert. Darüber hinaus bieten die SysML-Diagramme auch keine geeignete Visualisierung management-relevanter Informationen (z.B. Kosten, Entwicklungsstatus, Ressourcen etc.) zum modellierten System. Diese Informationen können zwar im Modell hinterlegt, jedoch nicht angemessen dargestellt werden. BROY ET AL. diskutieren darüber hinaus ausführlich, dass unzureichend formale Spezifikationen der Sprachen und Schnittstellen, fehlende Vorgehensmodelle und insbesondere auch unzureichend leistungsfähige Softwarewerkzeuge einen höheren Verbreitungsgrad verhindern[203].

2.6.3 Domain-Specific Languages (DSL)

Domain-Specific Language (DSL) ist die Bezeichnung für eine formale Sprache, die speziell auf die Bedürfnisse eines konkreten Fachbereichs zugeschnitten ist. Dies gilt für viele Modellierungssprachen, beispielsweise ist EAST-ADL (vgl. Kapitel 2.6.7) eine Sprache zur Modellierung von eingebetteten Systemen und wird meist von Elektrotechnikern eingesetzt. Die Verwendung von DSL's kann Anwendern helfen, generischere Modellierungssprachen wie die UML hinsichtlich ihrer Belange zu erweitern und gleichzeitig nicht benötigte Konstrukte (Diagrammarten, Entitäten oder Relationen) weglassen. Es wird zwischen zwei Arten von DSL's unterschieden: den internen DSL's wie das XML-Schema oder der UML2-Profilmechanismus, die eine echte Untermenge einer generischeren Sprache sind, und den externen, also eigenständigen DSL's, wie beispielsweise die Datenbanksprache SQL[204].

Domain-Specific Languages wurden unter anderem eingesetzt, um die SysML, MARTE und viele weitere Modellierungssprachen zu definieren. Auch in dieser Arbeit werden Profile und DSL's eingesetzt, um die SysML bedarfsgerecht zu erweitern.

---

[201] z.B. Andersson et al. (2009), Alt (2009)
[202] Vgl. Albers und Zingel (2013a)
[203] Broy et al. (2010)
[204] Stahl et al. (2007)

2.6.4 Business Process Model and Notation (BPMN)

Business Process Model and Notation (BPMN) ist ein Standard der OMG zur Modellierung von Geschäftsprozessen und deren Implementierung. Das Ziel ist die Bereitstellung einer für alle Anwender klar verständlichen Notation, beginnend mit Geschäftsanalysten beim Prozessentwurf über technische Entwickler bei der Ausführung der Prozesse bis hin zum Manager bei der Überwachung der Prozesse. Gleichermaßen soll auch die Kommunikation über Unternehmensgrenzen hinweg, also beispielsweise mit Kunden oder Zulieferern, unterstützt werden. BPMN soll folglich die Lücke zwischen Prozessdefinition und Implementierung schließen. Hierzu bedient sich die Notation der Kollaborationsdiagramme, der Prozessdiagramme und der Choreographiediagramme. Durch die Formalisierung soll weiterhin eine Basis für den Austausch von Geschäftsprozessmodellen zwischen verschiedenen Softwarewerkzeugen ermöglicht werden[205].

Die verwendeten Diagramme ähneln allesamt dem Aktivitätsdiagramm der UML/SysML, beispielsweise wurden die Konzepte „Activity“ und „Swimlane“ übernommen. In der BPMN wurden jedoch zahlreiche weitere Elemente mit eigenem Erscheinungsbild eingeführt, um die Lesbarkeit deutlich gegenüber einem Aktivitätsdiagramm zu verbessern. Auch wenn die Spezifikation erweiterbar ist, so bleibt sie dennoch sehr auf Geschäftsprozesse spezialisiert. Produktspezifische Informationen müssten im Fall einer modellbasierten Vernetzung aus anderen Modellen (z.B. SysML) bezogen werden.

2.6.5 Conceptual Design Specification Technique for the Engineering of Complex Systems (CONSENS)

Die im Rahmen des SFB 614 “Selbstoptimierende Systeme des Maschinenbaus” entwickelte Spezifikationstechnik CONSENS (Conceptual Design Specification Technique for the Engineering of Complex Systems) beschreibt die Prinziplösung als Ergebnis der Konzipierungsphase mechatronischer und selbstoptimierender Systeme[206]. Sie hat das Ziel, die Lücke zwischen dem Anforderungskatalog, der eine eher grobe Spezifikation des Gesamtsystems darstellt und naturgemäß weit interpretierbar ist, und den etablierten Spezifikationstechniken der einzelnen Domänen zu schließen. Die Spezifikationstechnik besteht aus mehreren Aspekten, aus denen sich die Beschreibung eines Produkts und des zugehörigen Produktionssystems zusammensetzen kann. Diese sind Anforderungen, Umfeld, Anwendungsszenarien, Funktionen, Wirkstruktur, Gestalt und die Gruppe Verhalten

[205] OMG BPMN (2011)
[206] Gausemeier et al. (2009a), Gausemeier et al. (2012)

für das Produkt sowie Anforderungen, Prozesse, Ressourcen und Gestalt für das Produktionssystem. Sie werden rechnerintern durch ein kohärentes System von Partialmodellen repräsentiert (siehe Bild 2-34).

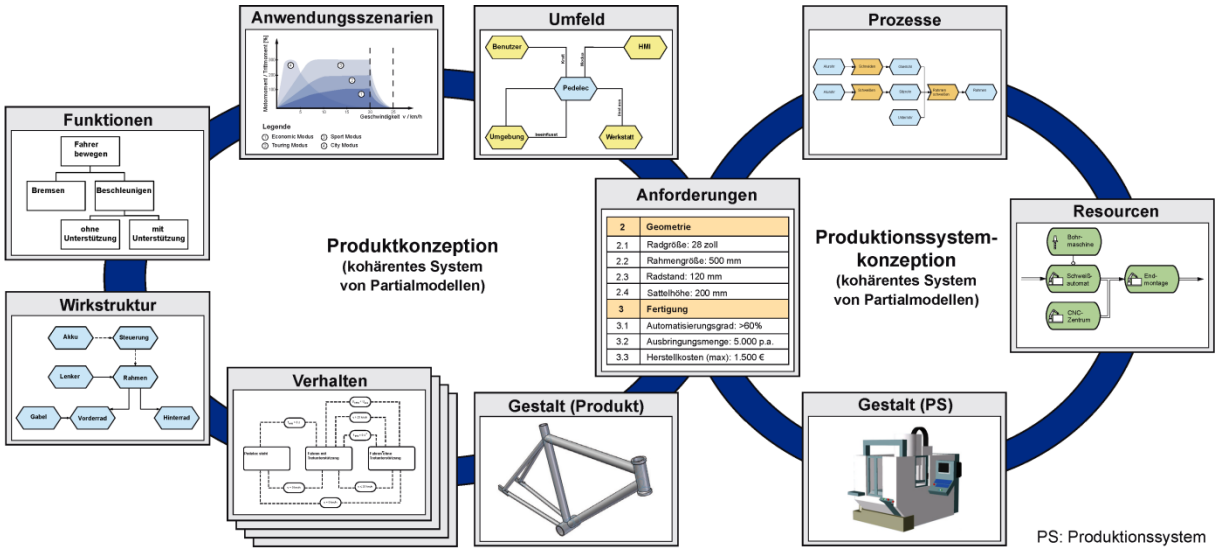


Bild 2-34: Partialmodelle der Spezifikationstechnik CONSENS

Ein zentrales Element der Sprache ist die *Wirkstruktur*, welche das Konstrukt *Systemelement* einführt, um einen Zwischenschritt im Zuge der Konkretisierung von Lösungsmustern auf der einen Seite zu Bauteilen bzw. Softwarekomponenten auf der anderen Seite zu bilden. Funktionen werden in CONSENS nur als Funktionsbaum, oft auch als *Funktionsstruktur* bezeichnet, modelliert. Der interne Aufbau und die Wechselwirkungen von Funktionen wird im Partialmodell „Aktivitäten“ abgebildet, das Teil der Verhaltensgruppe ist. Sie umfasst darüber hinaus auch Zustände und Sequenzen. Anwendungsszenarien, Anforderungen und Systemumfeld werden jeweils durch ein eigenes Partialmodell beschrieben und können durch entsprechende Beziehungen auch mit anderen Partialmodellen verknüpft werden.

Grundsätzlich hat die Spezifikationstechnik in Bezug auf die beschreibbaren Aspekte große Ähnlichkeit mit der SysML, bietet jedoch mit der Wirkstruktur einen zusätzlichen logischen Schritt zwischen der rein funktionalen Modellierung und der Transition in die Gestalt an, um Integralbauweisen (mehrere Wirkprinzipien werden durch ein Bauteil realisiert) und Differentialbauweisen (mehrere Bauteile realisieren ein Wirkprinzip) beschreiben zu können. Der Vorteil gegenüber der SysML ist aber nicht klar erkennbar, zumal es keinerlei Standardkonformität[207] der Spezifikationstechnik gibt, wodurch Implementierungen in Entwicklungsumgebungen

[207] Beispielsweise zur Meta-Object-Facility (MOF), siehe Kapitel 2.8.1

bzw. Kopplungen mit anderen Softwarewerkzeugen manuell und mit hohem Aufwand umgesetzt werden müssen. Die Gestaltmodellierung findet in CAD-Werkzeugen statt. Eine Werkzeugkopplung mit dem im BMBF-Verbundprojekt „VireS“ entwickelten Modellierungswerkzeug Mechatronic Modeller[208] der Spezifikationstechnik wurde bisher nicht realisiert. Weiterhin ist durch die fehlende Konformität des Metamodells automatisch eine Beschränkung auf dieses einzige, verfügbare Softwarewerkzeug gegeben. Erste Erkenntnisse über den Nutzen der Spezifikationstechnik für die Industriepartner des Verbundprojekts VireS werden zwar in GAUSEMEIER ET AL. vorgestellt[209], jedoch sind dem Autor der Arbeit keine Anwendungserfahrungen der Spezifikationstechnik und insbesondere des Modellierungswerkzeugs über die Grenzen des SFB 614 und das Verbundprojekt VireS hinaus trotz eines relativ hohen Bekanntheitsgrades der Spezifikationstechnik bekannt.

2.6.6 Modeling and Analysis of Real-Time and Embedded Systems (MARTE)

MARTE (Modeling and Analysis of Real-Time and Embedded Systems) ist ein die UML erweiterndes Sprachprofil zur Modellierung und Analyse von Echtzeitsystemen und eingebetteten Systemen in den Entwicklungsaktivitäten Spezifikation, Design und Verifikation/Validierung[210]. Das neue Profil ersetzt das bisherige UML SPT Profil (UML Profile for Schedulability, Performance and Time specification) mit dem Ziel einer leistungsfähigeren Modellierung von zeitkritischen Aspekten, Performances und Ablaufkontrollen (schedulability). Die wichtigsten angestrebten Nutzen sind eine verbesserte Kommunikation der Entwickler durch Bereitstellung einer integrierten Modellierung von Software und Hardware, die Befähigung zur Interaktion des Modells mit anderen Entwicklungswerkzeugen und die Ermöglichung quantitativer Analysen und Voraussagen zu Echtzeitfähigkeiten von Systemen unter Einbezug von Hardware- und Softwarecharakteristiken. Neben der UML-Basis gliedert sich das MARTE-Profil auch in weitere OMG-Standards durch Wiederverwendung von Artefakten, beispielsweise von Stereotypen aus der SysML, nahtlos ein.

Aufgrund der etablierten technologischen Basis (UML) ist die Einbettung des MARTE-Profils in UML-Modellierungswerkzeugen schnell erfolgt. In Kombination mit anderen standardisierten OMG-Modellierungssprachen wie der OCL (Object Constraint Language[211]) sowie entsprechenden Modelltransformationsstandards

[208] Gausemeier et al. (2010), Gausemeier et al. (2012)
[209] Gausemeier et al. (2012)
[210] OMG MARTE (2011)
[211] vgl. Kapitel 2.8.3

(MOF 2.0 QVT – Query View Transformations[212]) stellt MARTE ein leistungsfähiges Werkzeug zur Modellierung von E/E-Systemen dar. Allerdings fehlt die Anbindung an weitere im Entwicklungsprozess beteiligte Fachdisziplinen, weshalb die Sprache auch im Paket dem Anspruch der Ganzheitlichkeit nicht hinreichend gerecht wird. Weiterhin ist eine Anwenderakzeptanz nur im Bereich der IT-affinen Entwickler gegeben, die mit den zugrunde liegenden Prinzipien (wie beispielsweise Vererbung und Klassen-Rollen-Instanzenbildung) vertraut sind.

2.6.7 Electronics Architecture and Software Technology - Architecture Description Language (EAST-ADL)

Seit 2003 wurde die “Electronics Architecture and Software Technology - Architecture Description Language (EAST-ADL)” in den EU-Projekten EAST-EEA und ATESST durch ein breites Konsortium aus Automobilindustrie und Hochschulen entwickelt und bis zur heute aktuellen Version 2.1.5 verfeinert. Die Modellierungssprache ist ein domänenspezifisches Sprachprofil zur Spezifikation von Automobilelektrik- und –elektroniksystemen, das aus dem UML-Metamodell abgeleitet wurde und sich Konzepten des AUTOSAR-Metamodells bedient. EAST-ADL ermöglicht die Beschreibung und Vernetzung von E/E-Systemstrukturen auf verschiedenen Abstraktionslevels, Verhalten, Anforderungen sowie Verifikations- und Validierungsaspekten[213].

Zwar hat die Modellierung von E/E- und Softwaresystemen inzwischen einen sehr hohen Reifegrad erreicht, jedoch fehlt bis heute die Anknüpfung an eine ganzheitliche Beschreibungssprache, die auch Belange weiterer Fachdisziplinen, wie der Konstruktions- und Fertigungstechnik, Werkstofftechnik oder der Technischen Mechanik, einbezieht.

2.6.8 AUTomotive Open System ARchitecture (AUTOSAR)

AUTOSAR steht für AUTomotive Open System ARchitecture und ist eine Entwicklungspartnerschaft von Automobil-, Steuergeräte-, Mikrocontroller- und Entwicklungssoftwareherstellern, die eine einheitliche Beschreibungs- und Konfigurationssprache für die Architektur eingebetteter (embedded) Software im Automobil entwickelt haben. Die modellbasierte Sprache umfasst die Beschreibung von Software-Komponenten, Steuergeräten und Systemkonfigurationen und verfolgt das Prinzip funktionsorientierter Entwicklung. Damit ist sie auch in der Lage, weitere

[212] vgl. Kapitel 2.8.5
[213] ATESST2 Consortium (2010a), ATESST2 Consortium (2010b)

Anwendungsbereiche der modellbasierten E/E-Systementwicklung, wie virtuelle Integration, Architekturbewertung und Variantenmanagement, zu unterstützen[214].

2.6.9 Matlab / Simulink / Stateflow

MATLAB steht für MATrix LABoratory und ist eine proprietäre Entwicklungsumgebung und Programmiersprache zur Visualisierung, numerischen Berechnung und Programmierung mathematischer Ausdrücke auf Basis von Matrizen. Neben dem Basismodul erweitern zahlreiche Toolboxen den Funktionsumfang z. B. um Regelungstechnik, Signalverarbeitung und Optimierung. Simulink ist eine Erweiterung zur grafischen Modellierung, Simulation und Analyse dynamischer Systeme in Blockschaltbildern (Signalflussgraphen) und zugehörigen Funktionsbibliotheken. Stateflow ist eine zusätzliche Erweiterung zur Modellierung und Simulation ereignisdiskreter, endlicher Zustandsautomaten. Durch die Kombination mit Simulink lassen sich gemischt kontinuierliche und ereignisdiskrete Modelle erstellen[215].

Das Werkzeugpaket wird verbreitet in Industrie und an Hochschulen eingesetzt, hat jedoch den entscheidenden Nachteil, dass es ein proprietäres Werkzeug und kein Standard ist und somit die Integration in Werkzeugketten und Entwicklungsumgebungen immer herstellergebunden stattfinden muss. Weiterhin beschränkt sich der Funktionsumfang auf numerische Berechnung, Simulation und Analyse, wodurch zahlreiche darüber hinausgehende, fachdisziplinübergreifend relevante Informationen nicht beschrieben und vernetzt werden können.

2.6.10 Modelica und ModelicaML

Die objektorientierte Sprache Modelica wurde im Rahmen des ESPRIT-Projekts „Simulation in Europe Basic Research Working Group“ zur Modellierung physikalischer heterogener Systeme entwickelt. Sie wird durch die Modelica Association vorangetrieben und vermarktet. Modelica verfolgt das Ziel, Sprachen wie Dymola, VHDL-AMS und ObjectMath zu vereinheitlichen. Sie unterstützt den fachdisziplin- und werkzeugübergreifenden Austausch von Modelldaten und die Wiederverwendung von Modellen. Die Modellierung mit Modelica erfolgt auf Komponenten- oder Gleichungsebene und ermöglicht die Beschreibung komplexer physikalischer Systeme unter Einbezug von Mechanik, Elektrik, Elektronik,

[214] Hardt und Große-Rhode (2008)
[215] Angermann et al. (2011)

Regelungstechnik, Hydraulik und Pneumatik. Die Sprache ist in der Industrie bereits sehr etabliert, zahlreiche Modellierungswerkzeuge sind am Markt erhältlich[216].

Die ModelicaML ist ein UML-Profil, das die Stärken von UML und SysML für die grafische Modellierung von Systemen und Software mit denen der Modelica in der Systemsimulation kombiniert[217]. 2008 wurde hierzu eine Arbeitsgruppe der OMG eingerichtet, die sich seither der Kopplung von Modelica mit SysML über QVT[218] sowie der Weiterentwicklung des zugehörigen Sprachprofils ModelicaML widmet[219].

Ein weiterer, der ModelicaML sehr ähnlicher Ansatz zur Kopplung von Modelica und UML ist die Mechatronic UML zur Modellierung sicherheitskritischer, rekonfigurierbarer mechatronischer Systeme[220].

2.6.11 Zwischenfazit

Die vorgestellten Modellierungssprachen basieren weitgehend auf der gleichen technologischen Basis zur Metamodellierung, der Meta-Object Facility (MOF)[221]. Damit sind sie grundsätzlich zueinander kompatibel und Teilmodelle können untereinander ausgetauscht werden. Viele der Modellierungssprachen sind teilweise disziplinübergreifender Natur, da mehrere in der Entwicklung multidisziplinärer Systeme relevante Aspekte mit ihnen modelliert werden können. Mit Ausnahme von CONSENS, die jedoch nicht auf MOF basiert, ist SysML die einzige Modellierungssprache, die in der Lage ist, Aspekte sämtlicher Disziplinen zu modellieren, jedoch musste hierzu auch der Kompromiss eingegangen werden, dies auf einem hohen Abstraktionsniveau zu realisieren, um den Umfang der Sprache nicht zu sprengen. Für die Fachdisziplinen der Softwaretechnik und der Elektrik/Elektronik wurden bereits geeignete Lösungen entwickelt, spezifischere Modellierungssprachen konsistent an die SysML zu koppeln und die dort entstehenden, generischen Modelle weiter ausdetaillieren und synchronisieren zu können. Dies ist jedoch nicht der Fall für die Fachdisziplin Maschinenbau, insbesondere die Konstruktion, die nach wie vor mit relativ isolierten 3D-CAD-Modellen arbeitet. Hier besteht ein hoher Forschungsbedarf bei der funktionsbasierten Modellierung unter Berücksichtigung funktionsrelevanter physischer Merkmale sowie deren modell- und methodengestützten Transformation auf die realisierende physische Struktur und Gestalt. Weiterhin existiert bisher keine

---

[216] Modelica Association (2007)
[217] Johnson et al (2008), Schamai et al. (2009), Paredis et al. (2010)
[218] Query View Transformation, siehe Kapitel 2.8.5
[219] OMG ModelicaML (2012)
[220] Burmester et al. (2005)
[221] Vgl. Kapitel 2.8.2

anwendergerechte Sprache zur modellbasierten Beschreibung aller Aspekte und Wechselwirkungen der kontinuierlichen Aktualisierung und Konkretisierung von Zielsystem und Objektsystem im Verlauf der Entwicklung multidisziplinärer Produkte. Auch die verwendeten Begriffe für Modellelemente sowie deren grafische Repräsentation haben sich in der praktischen Anwendung als nicht für alle Fachdisziplinen hinreichend intuitiv verständlich erwiesen.

2.7 Semantische Modellierungssprachen

Im Gegensatz zu den zuvor vorgestellten Modellierungssprachen ist das Ziel semantischer Sprachen nicht die Modellierung verschiedener Aspekte technischer Systeme, sondern die semantische Kopplung von Zusammenhängen heterogener Informationen. Der Ansatz, Informationen maschinenverarbeitbar (also streng formal mittels Objekten und nicht durch unformalen Freitext) zur Verfügung zu stellen, ist die Kernidee des *Semantic Web*. Die Schaffung der hierzu notwendigen Standards und Vereinbarung ist das Ziel des W3C[222], zu denen beispielsweise die Extensible Markup Language (XML), das Resource Description Framework (Schema) (RDF(S)) und die Web Ontology Language (OWL) als Informations-Beschreibungssprachen gehören. Die letzten beiden sind sogenannte Ontologiesprachen. Unter einer Ontologie wird ein Set aus Artefakten (Entitäten, Attributen und Relationen) verstanden, mit denen eine Wissensdomäne modelliert werden kann. Aus diesem Wissen können Schlussfolgerungen zu neuen Informationszusammenhängen gezogen werden, um Informationen zu „verstehen“ und sie anwendergerecht zu kombinieren[223]. Die beiden Sprachen RDF und OWL sowie eine weitere Ontologiesprache, Frame Logic (F-Logic), werden in den folgenden Kapiteln kurz vorgestellt.

2.7.1 Resource Description Framework (RDF)

Das Resource Description Framework (RDF) ist eine formale Sprache zur Beschreibung strukturierter Informationen[224]. Im Gegensatz zur XML oder der Hypertext Markup Language (HTML) dient RDF nicht nur der korrekten Darstellung von Dokumenten (ihrer Syntax), sondern auch der bedeutungsgemäßen Kombination und Weiterverarbeitung der enthaltenen Informationen (Semantik). Die Erweiterung RDF Schema (RDFS) bietet die Möglichkeit, allgemeine schematische Informationen über einen Datensatz ausdrücken zu können, die über das Beschreiben einfacher Daten hinausgeht. RDF basiert auf einem sehr einfachen graph-orientierten

[222] World Wide Web Consortium, http://www.w3.org
[223] Weitere Informationen zu RDF und RDFS finden sich in Hitzler et al. (2008)
[224] Manola und Miller (2004)

Datenschema. Damit können die beispielsweise in XML hierarchisch abgelegten Daten (in RDF Ressourcen genannt) zueinander in Beziehung gesetzt werden. Dies geschieht durch Tripel, bestehend aus Subjekt, Prädikat und Objekt, die jeweils eindeutig durch ihre URI[225] gekennzeichnet werden. Beispielsweise können so ein Buch und ein Verlag, welche in zwei verschiedenen XML-Bäumen abgelegt sind, durch das Tripel „Buch“ (Subjekt) „wird verlegt bei“ (Prädikt) „Verlag“ (Objekt) zueinander in Beziehung gesetzt werden[226].

2.7.2 Web Ontology Language (OWL)

Da das RDF ausschließlich der Beschreibung einfacher Tripel-Zusammenhänge dient, ist die Abbildung komplexen Wissens damit oft nicht hinreichend genau möglich[227]. Hierzu sind auf formaler Logik basierende Repräsentationssprachen erforderlich, die zudem durch logisches Schlussfolgern auch den Zugriff auf implizites Wissen ermöglichen. Eine solche Sprache ist die im Februar 2004 vom W3C als Ontologiesprache standardisierte Web Ontology Language (OWL)[228]. Um dem Anwender je nach Anforderungen der Praxis die Wahl zwischen verschiedenen Ausdrucksstärken zu ermöglichen, gibt es drei verschiedene Teilsprachen: OWL Full, OWL DL, und OWL Lite. Dabei ist OWL Lite eine echte Teilsprache von OWL DL, welches wiederum eine echte Teilsprache von OWL Full ist. Es existieren zwei Syntaxen der OWL, von denen eine auf dem RDF basiert und für den Datenaustausch bestimmt ist. Die zweite Syntax ist eine abstrakte OWL-Syntax, die für den Menschen leichter verständlich, aber nur für OWL DL verfügbar ist und bevorzugt im wissenschaftlichen Umfeld Anwendung findet. Mit OWL können Klassen und Properties in komplexe Beziehungen zueinander gesetzt werden. Diese Fähigkeit der Regeldefinition ist eine sinnvolle Ergänzung zu objektorientierten Modellierungssprachen, wie beispielsweise GRAVES durch seinen Ansatz zur Integration SysML und OWL feststellt[229].

2.7.3 Frame Logic (F-Logic)

Frame Logic (F-Logic) ist ebenfalls eine formale Sprache zur Wissensrepräsentation. Sie ist eine Erweiterung der Datalog[230], einer leicht implementierbaren und ausdrucksstarken Datenbankregelsprache, der jedoch Möglichkeiten zur grafischen

---

[225] Unique Resource Indentifier (Eindeutiger Ressourcenidentifikator, ähnlich einer ID)
[226] Hitzler et al. (2008)
[227] Beispielsweise natürliche Sätze mit Adjektiven oder Adverbien wie „Jedes Projekt hat mindestens drei technische Mitarbeiter und genau einen Projektleiter.“
[228] W3C OWL (2004)
[229] Graves (2009)
[230] Abiteboul et al (1995)

Modellierung von Regeln fehlen. Dies ermöglicht F-Logic, die leistungsfähigere Arten der Negation, Funktionssymbole, abstrakte Konstrukte zur Modellierung und eine framebasierte Syntax bietet. Diese Sprache findet unter anderem in ALBERS ET AL. bei der Kopplung von produkt- und prozessrelevanten Informationen in Konzepten der softwaregestützten Modellierung des integrierten Produktentstehungsmodells (iPeM) Anwendung[231]. Für weiterführende Informationen zu F-Logic und anderen Ontologiesprachen wird hier auf entsprechende Fachliteratur verwiesen[232].

2.7.4 Zwischenfazit

Semantische Sprachen können komplexe Informationszusammenhänge beschreiben und darüber hinaus auch Rückschlüsse auf weitere vorhandene Informationszusammenhänge in Modellen, die bisher nicht explizit modelliert wurden, ziehen. Der Aufbau einer Ontologie unter Verwendung einer semantischen Sprache unterstützt die eindeutige und für den Anwender verständlichere Aufbereitung von Wissen. Weiterhin können auf diesem Wege Informationen, die in verschiedenen Modellen und Dokumenten vorliegen, semantisch verbunden und damit deren Informationsgehalt und das vorhandene Wissen deutlich gesteigert werden. In Kapitel 6.5.3 wird ein Ansatz zur Kopplung von Ontologien und SysML zum Aufbau einer Wissensdomäne in einem Framework zur funktionalen Lenkung mechatronischer Produkte vorgestellt.

2.8 Technologien der modellbasierten Entwicklung

In diesem Kapitel werden die für diese Arbeit relevanten, softwaretechnischen Basistechnologien und die wichtigsten Standards zur Anwendung modellbasierter Entwicklung kurz vorgestellt.

2.8.1 Model-Driven Architecture (MDA)

Die Object Management Group (OMG) als Dachorganisation der objektorientierten Technologien hat mit dem Standard Model-Driven Architecture (MDA)[233] einen modellgetriebenen Softwareentwicklungsansatz geschaffen, der auf einer klaren Trennung von Funktionalität und Technik von Software beruht. Das Ziel ist die Steigerung der Entwicklungseffizienz und bessere Handhabung von Systemen durch Bereitstellung verschiedener hersteller- und plattformneutraler Technologien für die Handhabung von rechnerverarbeitbaren Modellen. Das Ziel wird durch Definition von abstrakten Standards erreicht, die zueinander in Relation gesetzt werden. Diese

[231] Albers et al. (2012b)
[232] Hitzler et al. (2008), Staab und Studer (2009)
[233] OMG MDA (2003)

Standards beschreiben jeweils bestimmte Anwendungsfälle in der modellbasierten Entwicklung, wie beispielsweise die Erstellung von Metamodellen mit der Meta-Object Facility (siehe Kapitel 2.8.2), dem Austausch von Metadaten mit XML Metadata Interchange (siehe Kapitel 2.8.7) oder der Modelltransformation mit Query View Transformation (siehe Kapitel 2.8.5). Die Nutzung der Standards soll einen möglichst hohen Automatisierungsgrad von Arbeitsabläufen und die Bildung von Werkzeugketten, also eine Reihe zusammenarbeitender Softwarewerkzeuge, ermöglichen[234].

2.8.2 Meta-Object Facility (MOF)

Dieser OMG-Standard[235] definiert, wie die abstrakte Syntax und die statische Semantik von Modellierungssprachen in Form eines Metamodells zu beschreiben sind. Darin wird festgelegt, welche Modellelemente eine Modellierungssprache umfasst und wie diese verwendet werden dürfen. Die notwendigen Elemente zur Spezifikation eines Metamodells werden wiederum in einem Metamodell definiert, was auch als Meta-Meta-Ebene bezeichnet wird. Diesen Sachverhalt veranschaulicht das 4-Ebenen-Modell der Hierarchie der Metamodellierung in Bild 2-35.

[234] Alt (2012), OMG MDA (2003)
[235] OMG MOF (2011)

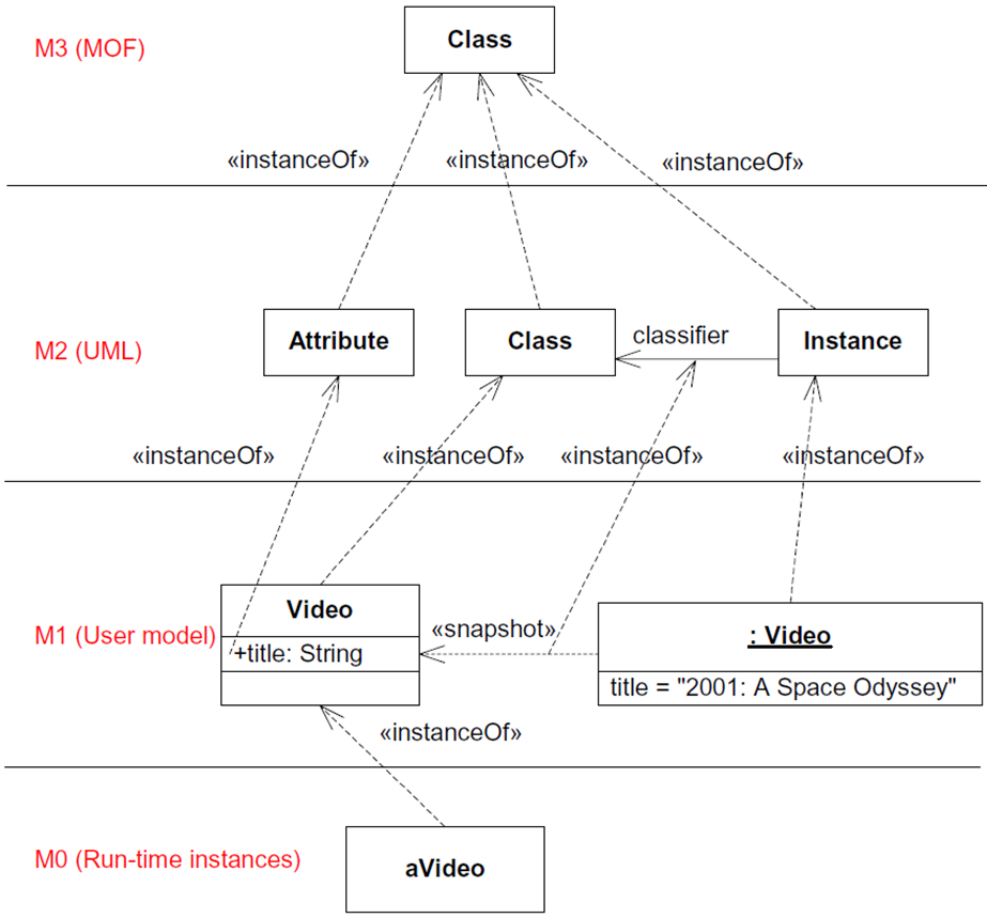


Bild 2-35: 4-Ebenen-Modell der Hierarchie der Metamodellierung[236]

Auf der untersten Ebene M0 befinden sich reale Objekte (z.B. eine Videokassette). Modelliert man dieses Objekt, so abstrahiert man es auf eine Klasse (hier: Video) mit Attributen (z.B. einem Titel) auf der Ebene M1. Innerhalb des Modells kann man Instanzen bilden, um konkrete Werte für dessen Attribute zu bestimmen. Die Definition, dass „Video“ eine Klasse, der Titel ein Attribut und das konkrete Video eine Instanz ist, wird im Metamodell des Modells (hier UML) auf der Ebene M2 beschrieben. Die Vorgabe, wie Modellelemente definiert werden, hinterliegt auf der obersten Ebene M3[237].

2.8.3 Object Constraint Language (OCL)

Bei der Erstellung einer neuen Modellierungssprache müssen neben der mittels der MOF definierbaren abstrakten Syntax für Entitäten und Relationen häufig auch essentielle (allgemeingültige) Regeln oder Bedingungen eingebunden werden. Beispielsweise könnte eine Voraussetzung zum Verleih der Videokassette aus Bild

[236] Aus OMG UML (2011b)
[237] Zum besseren Verständnis vgl. Kapitel 2.5

2-35 sein, dass diese nur an Kunden ab 18 Jahren verliehen werden darf. Zu diesem Zweck hat die OMG die MOF um die textuelle Object Constraint Language (OCL) ergänzt, mit der die Beschreibung solcher essentiellen Regeln und Bedingungen möglich ist[238]. Das Metamodell der OCL selbst ist Teil der UML-Spezifikation, die bereits in Kapitel 2.6.1.1 vorgestellt wurde.

2.8.4 Parser und Abstract Syntax Tree (AST)

Informationen werden in Modellen formal abgelegt. Der Informationsaustausch zwischen Modellen erfordert eine Softwareroutine, die in der Lage ist, in sequentieller Textform aus Modellen exportierte Informationen (z.B. XML) mittels einer Syntaxanalyse in einen abstrakten Syntaxbaum (AST) zu überführen. Dadurch werden z.B. Hierarchien zwischen Elementen ausgedrückt und die Informationen können leichter durch Fremdsoftware verarbeitet, also beispielsweise in andere Modelle transformiert werden. Dieser Vorgang entspricht einer grammatikalischen Analyse natürlicher Sprachen. In der Softwaretechnik übernehmen Parser (lat. „(zer)teilen“) genannte Softwareroutinen unter Verwendung von AST diese Aufgabe[239]. Diese Technologie fand beispielsweise bei der Kopplung des E/E-Werkzeugs PREEvision mit dem CAD-Werkzeug CATIA im Rahmen der Studienarbeit von BULL Anwendung[240].

2.8.5 Query View Transformation (QVT)

Die in den vorangegangenen Kapiteln vorgestellten Standards dienen der Spezifikation von Metamodellen zur Bildung neuer Modelliersprachen oder der Erweiterung bestehender Sprachen. Darüber hinaus hat die OMG auch den Standard Query / View / Transformation (QVT) entwickelt und an MOF und OCL gekoppelt, um verschiedene auf MOF basierende Modelle untereinander transformieren zu können[241]. Die QVT besteht aus den drei (Sub-)Sprachen „Relational“, „Core“ und „Operational“. Die ersteren Beiden dienen der deklarativen Beschreibung von Abfragen (queries) und Transformationen, während der „Operational“-Teil ausführbare Befehle (Operations) umfasst[242].

2.8.6 Extensible Markup Language (XML)

Die Extensible Markup Language (engl. für „erweiterbare Auszeichnungssprache“), abgekürzt XML, ist eine Auszeichnungssprache zur Darstellung hierarchisch

[238] OMG OCL (2012)
[239] Hohagen und Naumann (1994)
[240] Bull (2012), vgl. auch Kapitel 6.5.2
[241] OMG QVT (2011)
[242] Königs (2009)

strukturierter Daten in Form von Textdaten. XML wird u. a. für den plattform- und implementierungsunabhängigen Austausch von Daten zwischen Computersystemen eingesetzt, insbesondere über das Internet. Ein Grundgedanke hinter XML ist es, Daten und ihre Repräsentation zu trennen, um Daten beispielsweise einmal als Tabelle und einmal als Grafik auszugeben, aber für beide Arten der Auswertung die gleiche Datenbasis im XML-Format zu nutzen. Für den Datenaustausch ist einerseits die Einhaltung von Regeln („Wohlgeformtheit“) und andererseits eine entsprechende Grammatik (z. B. ein XML-Schema) notwendig („Gültigkeit“). Programme, die XML-Daten auslesen, interpretieren und auf Gültigkeit prüfen, heißen Parser (siehe Kapitel 2.8.4). Diese werden beispielsweise bei der Transformation von Daten zwischen zwei Modellier- oder Programmiersprachen über API’s (application programming interfaces) eingesetzt[243].

2.8.7 XML Metadata Interchange (XMI)

Dieser OMG-Standard beschreibt ein Format zur Kommunikation von Modellen unter Verwendung der Extensible Markup Language (XML). Darin werden eine Reihe wichtiger Aspekte zur Beschreibung und Vernetzung von Objekten in der XML definiert, die deren konsistenten Austausch ermöglichen sollen. Zur Beschreibung der Struktur der Informationen in XML-Dateien wird ein weiterer Standard der OMG, die Meta-Object Facility (MOF), verwendet, welcher eine formale Sprache zur Spezifikation von Metamodellen bereitstellt (siehe auch Kapitel 2.8.2). Durch die Verwendung des MOF-Standards wird sichergestellt, dass Softwarewerkzeuge in der Lage sind, die Informationen in XML-Dateien so zu schreiben, dass sie von anderen Softwarewerkzeugen korrekt gelesen werden. Diese mit MOF beschriebenen XML-Schemata erlauben auch gleichzeitig eine gewisse Verifikation von XML-Dateien hinsichtlich der Korrektheit ihrer enthaltenen Informationen. Beispielsweise werden bestimmte Inhalte wie „Schnittstelle“ und „Flussgröße“ definiert, die immer zu modellieren sind. Diese können dann dahingehend geprüft werden, ob an zwei verbundenen Schnittstellen die passenden Flussgrößen übermittelt werden. Das Ziel des XMI-Standards ist also die Sicherstellung korrekter Kommunikation verschiedenartiger Modelle[244].

2.8.8 Standard for the Exchange of Product Model Data (STEP)

Der Standard for the Exchange of Product Model Data (STEP) ist eine Normenreihe, die im Rahmen der ISO für den Austausch von Produktdaten, deren Speicherung und Archivierung sowie die Produktdatentransformation entwickelt wurde und wird.

---

[243] Harold (2002)
[244] OMG XMI (2011)

Der STEP wird formal in der ISO 10303 spezifiziert und stellt verschiedene Applikationsprotokolle für die verschiedenen Anwendungsbereiche zur Verfügung, um die Art der beispielsweise per XML-Dateien transferierten Daten zu klassieren. Die bekanntesten und am weitesten verbreiteten Protokolle sind:

- AP 203 (Configuration Controlled Design)[245]
- AP 212 (Electrotechnical Design and Installation)
- AP 214 (Core data for automotive mechanical design processes)[246]
- AP 218 (Ship structures)
- AP 233 (Systems engineering data representation)[247]
- AP 236 (Furniture product data and project data)

Ziel der ISO 10303 ist es, eine eindeutige, rechnerinterpretierbare Darstellung aller Daten des Produktlebenszyklus eines Produkts zu ermöglichen[248]. Auch wenn sich STEP damit als das passende Datenformat zum Einsatz im Produktlebenszyklusmanagement (PLM) erweist, darf dieser nicht als fertiges Datenformat verstanden werden, in dem sämtliche Datendefinitionen für alle Arten von Produkten vollständig und anwendungsgerecht hinterlegt sind[249]. Es ist vielmehr ein Methodenkatalog und Baukasten, mit dessen Hilfe anwendungsspezifische Produktdatenmodelle beschrieben werden können. Insgesamt deckt STEP ein sehr breites Spektrum ab und ist in einer Reihe von separat entwickelten und veröffentlichten Dokumenten, den sogenannten Parts, organisiert, die neben den Applikationsprotokollen auch Methoden, Konstrukte und Ressourcen und weitere Bestandteile der Standardfamilie spezifizieren[250].

Die Applikationsprotokolle helfen, die Vielfalt der verschiedenartigen in der Produktentstehung generierten Daten zu strukturieren und somit eine Basis für die IT-seitige Kopplung von rechnerbasierten Modellen zu ermöglichen. Diese Möglichkeit wurde auch im Rahmen dieser Arbeit bei der Kopplung von SysML und CAD untersucht und prototypisch eingesetzt[251].

---

[245] ISO-10303-203 (2011)
[246] ISO-10303-214 (2010)
[247] ISO-10303-233 (2012)
[248] Dyla (2002)
[249] Jaros (2007)
[250] Anderl und Trippner (2000)
[251] Siehe Kapitel 6.5.2

2.9 Modellbildungsansätze für die Entwicklung multidisziplinärer Systeme

Die zahlreichen verschiedenen Sprachen, Werkzeuge und Technologien zur Modellbildung, der fachdisziplinübergreifenden Modellierung multidisziplinärer Systeme und der Kopplung von Modellen erfordern geeignete Anleitungen zur aufgaben- und zielgerechten Anwendung. Diese sogenannten Vorgehensmodelle beschreiben auf verschiedenen Granularitätsstufen, welche Handlungsschritte bzw. Arbeitsschritte Anwender durchzuführen haben, um ihr Ziel zu erreichen. Dabei beschränken sich nicht alle Vorgehensmodelle auf die Anwendung konkreter Werkzeuge, sondern beschreiben meist durch Produktentwickler durchzuführende Handlungen. LINDEMANN unterteilt Vorgehensmodelle anhand ihrer Logik in Kategorien, beginnend mit einer feingranularen *Mikrologik*, denen Vorgehensmodelle mit elementaren Handlungsschritten zugeordnet werden (z.B: das TOTE[252]-Modell). Darüber liegen Vorgehensmodelle für die *Problemlösung*, die operative Arbeitsschritte beschreiben oder ganze Arbeitsabschnitte in Phasen einteilen. Auf der abstraktesten Ebene der *Makrologik* finden sich globale Vorgehensmodelle für gesamte Entwicklungsprozesse oder Projekte, die beispielsweise durch Meilensteine untergliedert werden (vgl. Bild 2-36).

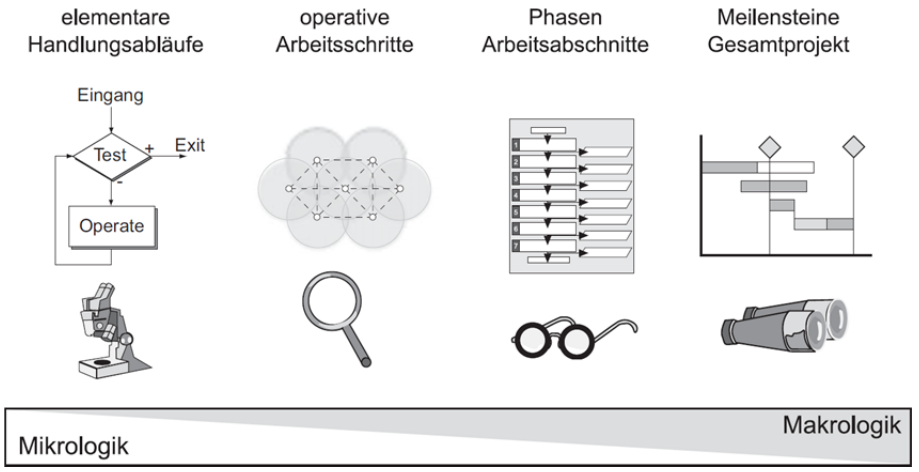


Bild 2-36: Klassierung der Handlungen im Produktentwicklungsprozesses[253]

Die in den nachfolgenden Kapiteln vorgestellte Auswahl etablierter Vorgehensmodelle werden unter dem Fokus ihrer Eignung für die modellbasierte Systementwicklung betrachtet. Vorgehensmodelle der Mikrologik beschreiben vor allem iterative und rekursive Denkprozesse, Beispiele sind neben dem TOTE-Modell

[252] TOTE steht für Test-Operate-Test-Exit
[253] aus Lindemann (2009)

auch der VVR-Zyklus (Vergleich-Veränderung-Rückmeldung) oder der DPS-Zyklus (Discursive Problem Solving). Diese Konzepte finden durchaus Anwendung bei der dekomponierenden Modellierung technischer Systeme, begegnen jedoch nicht der übergeordneten Herausforderung einer sinnvollen Vorgehenssystematik in der Modellierung multidisziplinärer Systeme. Daher werden in den folgenden Kapiteln zunächst Problemlösungszyklen vorgestellt, die wiederum innerhalb der Aktivitäten übergeordneter Produktentstehungsprozessmodelle Anwendung finden können.

2.9.1 Der Problemlösungszyklus der Systemtechnik

Der Problemlösungszyklus der Systemtechnik nach DAENZER UND HUBER besteht aus sechs Schritten, die zur Vereinfachung in die drei Arbeitsabschnitte I) Zielsuche, II) Lösungssuche und III) Lösungsauswahl zusammengefasst werden können (siehe Bild 2-37)[254].

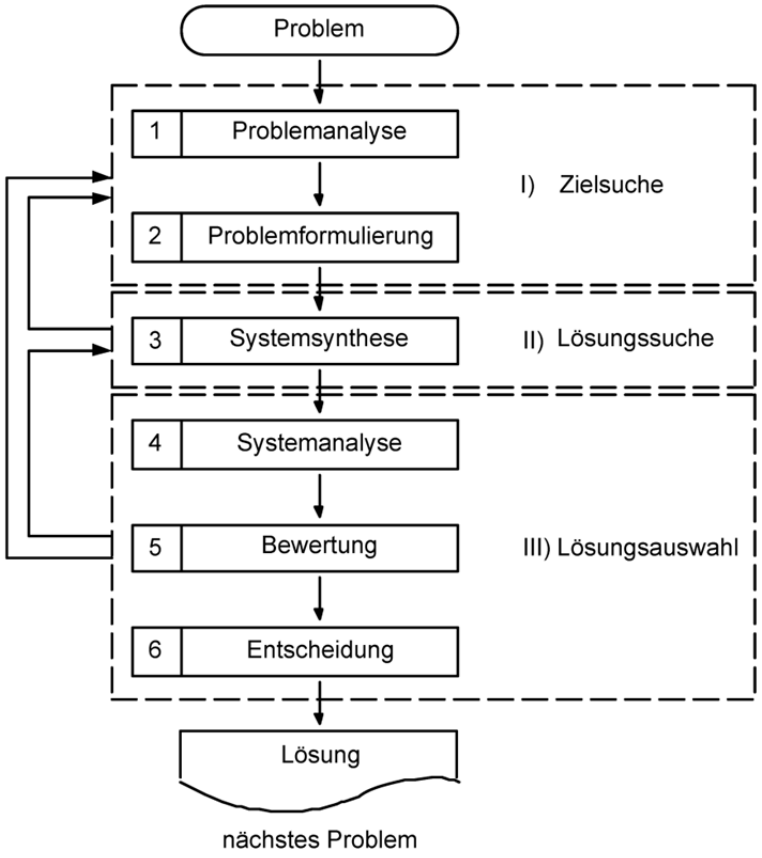


Bild 2-37: Der Problemlösungszyklus der Systemtechnik[255]

Dieser Zyklus kann als eine Abfolge sequentiell durchgeführter Analyse- (*Test*) und Syntheseschritte (*Operate*) gemäß dem TOTE-Schema mit konkreter

[254] Daenzer und Huber (2004)
[255] Daenzer und Huber (2004)

Aufgabenstellung betrachtet werden. Dabei beruht der Problemlösungszyklus auf dem Prinzip der Schaffung einer Lösungsvielfalt mit anschließender Auswahl. Daraus leitet EHRLENSPIEL seinen Vorgehenszyklus (VZ) ab[256].

2.9.2 SIMILAR Prozess

BAHILL UND GISSING stellen den SIMILAR-Prozess als eine Abstraktion menschlichen Denkens und Vorgehens bei der Problemlösung vor[257], der u.a. von RAMOS ET AL. aufgegriffen und auf moderne Systeme angewendet wurde[258]. Das Akronym steht für: **S**tate the problem, **I**nvestigate alternatives, **M**odel the system, **I**ntegrate, **L**aunch the system, **A**ssess performance und **R**e-evaluate. Diese allgemeinen Arbeitsschritte lassen sich auf nahezu jede konkrete Aufgabenstellung anwenden, nach Meinung von RAMOS ET AL. auch zur Beschreibung einer einfachen Vorgehensmethodik für Model-Based Systems Engineering[259].

2.9.3 VDI 2221: Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte

Die Richtlinie VDI2221 beschreibt eine durch den Ausschuss „methodisches Konstruieren“ des Vereins Deutscher Ingenieure entwickelte Methodik zur Entwicklung technischer Systeme. Sie behandelt hierzu allgemeingültige, branchenunabhängige Grundlagen methodischen Entwickelns und Konstruierens und definiert zweckmäßige Arbeitsabschnitte und –ergebnisse als Leitlinie für ein Vorgehen in der Praxis.

[256] Ehrlenspiel (2009)
[257] Bahill und Gissing (1998)
[258] Ramos et al. (2010)
[259] Ramos et al. (2012)

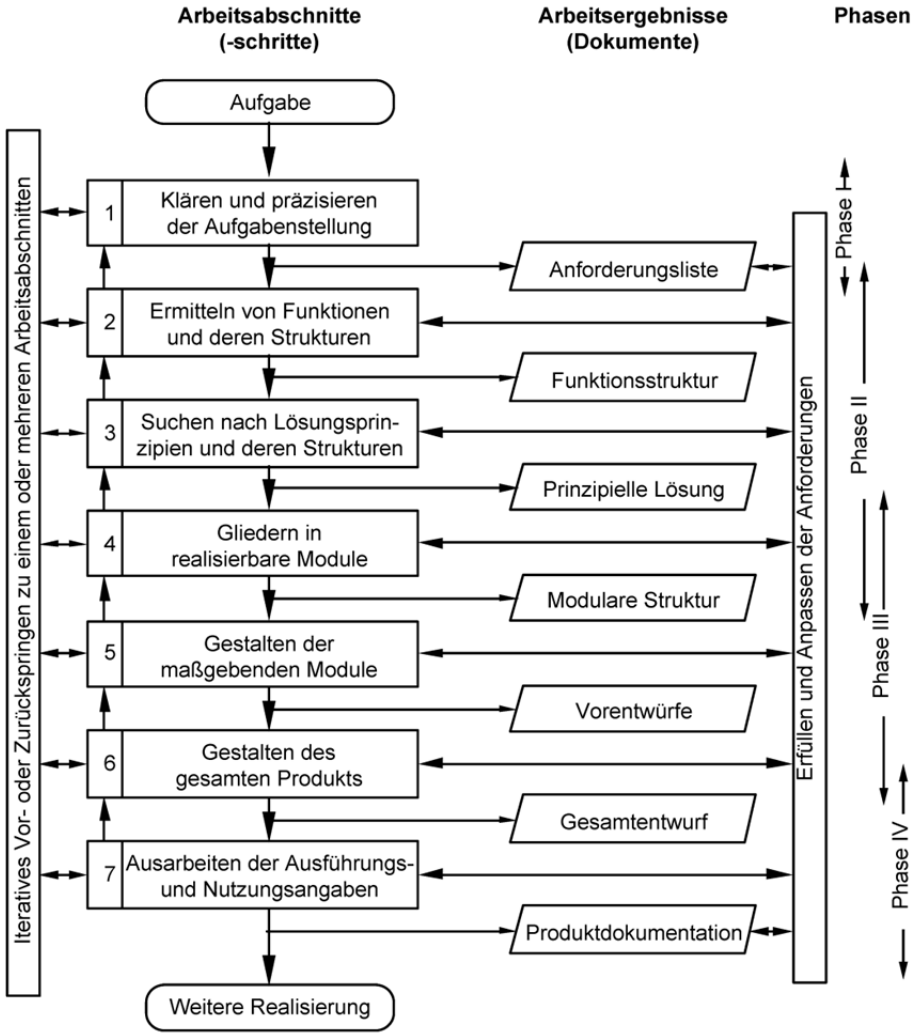


Bild 2-38: Generelles Vorgehen beim Entwickeln und Konstruieren (VDI 2221)[260]

Das Vorgehensmodell umfasst 7 Arbeitsschritte, zwischen denen iteratives Vor- und Zurückspringen im Sinne einer höheren Flexibilität des Modells möglich ist. Die als Ergebnis des ersten Arbeitsschrittes entstehende Anforderungsliste wird kontinuierlich und parallel zu den weiteren Schritten erfüllt und angepasst.

2.9.4 VDI 2206: V-Modell

Eine weitere Richtlinie des Vereins Deutscher Ingenieure beschreibt den Makrozyklus einer Entwicklungsmethodik für mechatronische Systeme, wobei Mechatronik als das Zusammenspiel der Fachbereiche Mechanik, Elektrik und Elektronik sowie Software definiert ist. Die grafische Darstellung der Entwicklungsmethodik wurde an das V-Modell der Softwaretechnik angelehnt und beschreibt die Entwicklungsschritte eines mechatronischen Systems, ausgehend von den Anforderungen über den Systementwurf, gefolgt vom Domänenspezifischen

[260] VDI-2221 (1993)

Entwurf über die Systemintegration mit Eigenschaftsabsicherung (Verifikation und Validierung) bis zum fertigen Produkt.

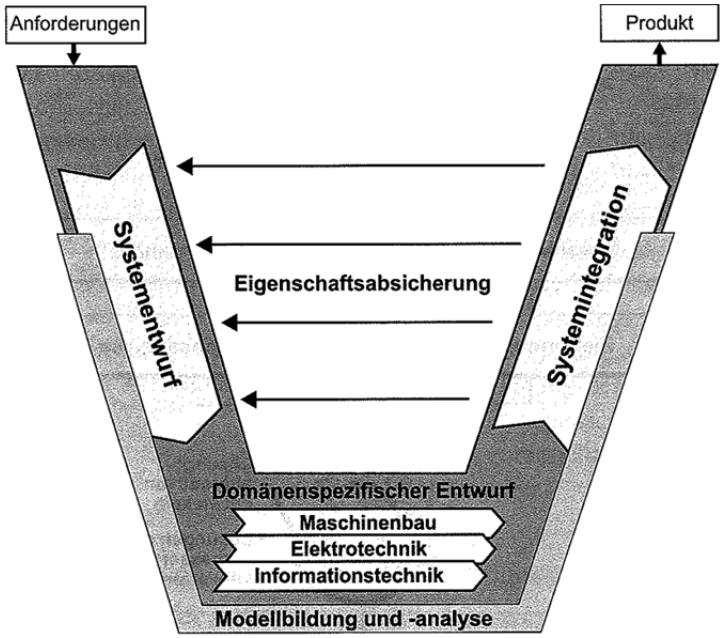


Bild 2-39: Das V-Modell als Makrozyklus nach VDI 2206[261]

Auch wenn das hier gezeigte V-Modell durchaus eine kontinuierliche Eigenschaftsabsicherung vorsieht, weist es doch insbesondere im Domänenspezifischen Entwurf einen sehr sequentiellen Charakter auf. Beispielsweise ist nicht vorgesehen, die als Ausgangspunkt der Entwicklung verwendeten Anforderungen in irgendeiner Form zu ergänzen oder anzupassen. Ebenso wird keine Rückführung der Ergebnisse des Domänenspezifischen Entwurfs in den Systementwurf durchgeführt. Dennoch hat sich das V-Modell in der industriellen Entwicklung mechatronischer Produkte als Leitmodell durchgesetzt und wurde in verschiedenen wissenschaftlichen Ansätzen aufgegriffen und dort meist nur minimal ergänzt. So wurde das V-Modell im Projekt ISYPROM um einige Prozessschritte ergänzt, um ein optimiertes Innovationsmanagement durch Forschung und Entwicklung, eine PLM Integration, eine modellbasierte, domänenübergreifende Verknüpfung sowie die integrierte Produkt- und Produktionssystementwicklung zu berücksichtigen. Hierzu werden zwischen den Modellen (z.B. Anforderungen, Funktionen oder Produktstruktur) Tracelinks explizit im Softwareprototyp ISYFMU modelliert, in den die Modellinformationen aus den proprietären Expertentools über standardisierte Schnittstellen importiert werden. Die Methode „Eco Tracing“ soll den notwendigen Modellierungsaufwand im Sinne einer

[261] VDI-2206 (2004)

industriellen Anwenderfreundlichkeit drastisch reduzieren. Unter anderem sollen damit Änderungsanfragen automatisiert, die Durchführung einer FMEA unterstützt und die Auswirkungen von Änderungen abgeschätzt werden können[262].

Einen weiteren Ansatz auf Basis des V-Modells stellen EIGNER ET AL. vor[263]. Sie integrieren den AFLP[264]-Ansatz als Methodik für modellbasierte Systementwicklung, welche in Kapitel 2.10.4 kurz vorgestellt wird. Die Anwendung der Methodik mit dem Werkzeug SysML wurde bereits in Kapitel 2.6.2.6 vorgestellt.

2.9.5 IPE-Methodik

EHRLENSPIEL beschreibt eine umfassende Methodik zur integrierten Produktentwicklung (IPE-Methodik), die neben Definitionen[265], Klassierungen und organisatorischen Methoden auch Vorgehensweisen zur Produktentwicklung und Produktmodellbildung umfasst[266]. Ehrlenspiel greift die drei übergeordneten Systemarten nach ROPOHL[267] auf: das Zielsystem, das Handlungssystem und das Sachsystem (siehe Bild 2-40).

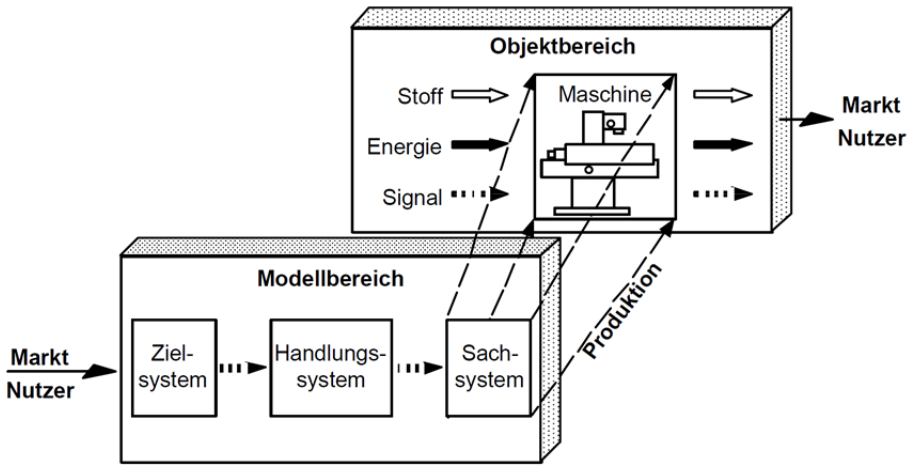


Bild 2-40: Für die Produkterstellung wesentliche Systeme nach ROPOHL[267]

Darin finden sich technische Systeme, die Sachsysteme im oben definierten Sinn darstellen. Diese lassen sich anhand ihrer Eigenschaften weiter klassieren. Als Eigenschaft bezeichnet Ehrlenspiel dabei alles, was durch Beobachtungen, Messergebnisse, allgemein akzeptierte Aussagen usw. von einem Gegenstand festgestellt werden kann. Wichtige kennzeichnende Eigenschaften können dabei zur

---

[262] Damerau et al. (2011)
[263] Eigner et al. (2012a), Eigner et al. (2012b)
[264] Anforderungen, Funktionen, Logische Architektur und Physische Architektur
[265] Siehe bspw. Definition des Systembegriffs in Kapitel 2.1
[266] Ehrlenspiel (2009)
[267] Ropohl (1975), Ropohl (2009)

besseren Hervorhebung mit dem Begriff Merkmal bezeichnet werden. Diese haben eine Bedeutung (Semantik, Qualität) und eine evtl. zahlenmäßige Ausprägung (Quantität). Produktmerkmale, also Merkmale technischer Systeme, werden nach DIN 2330 in den Hauptgruppen Beschaffenheitsmerkmale, Funktionsmerkmale und Relationsmerkmale unterschieden[268]. Diese dienen der Umsetzung von Stoff (Materie), Energie und Information (Signale, Daten), wobei EHRLENSPIEL eine Klassierung technischer Systeme anhand ihrer Hauptumsatzart (Zweck) vorstellt. Neben einer hier nicht näher erläuterten, weiteren Klassierung von Systemen nach ihrer Komplexität[269] folgt eine Klassifikation nach den Modellierungsbegriffen technischer Systeme. Danach erfolgt die Modellierung ausgehend von Anforderungen in den Bereichen Funktion, Physik und Gestalt, die jeweils aufeinander aufbauen (siehe Bild 2-41).

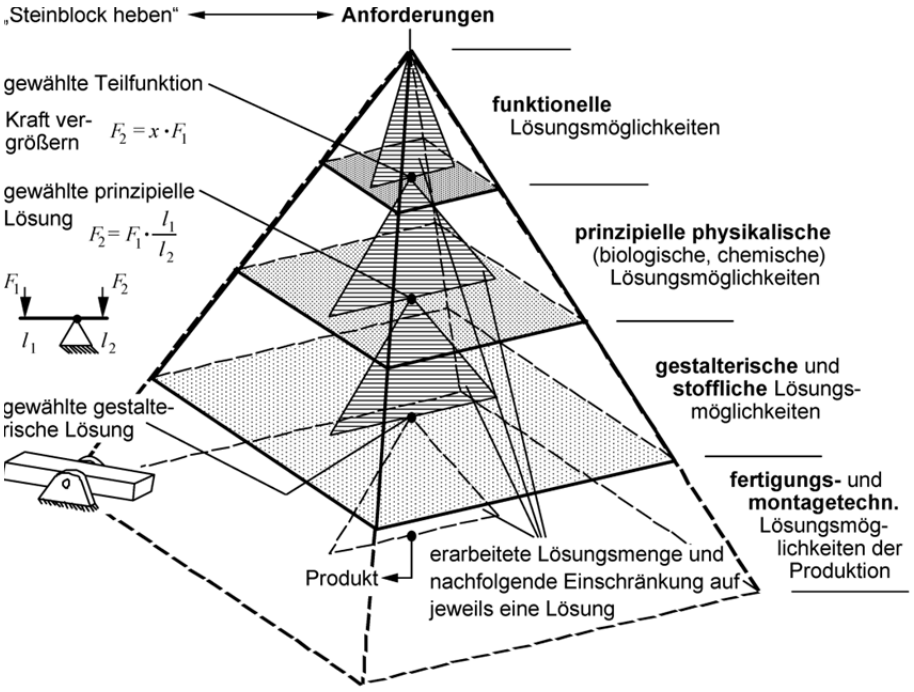


Bild 2-41: Hierarchische Modellierung von technischen Systemen[270]

Auf Basis dieser Definitionen und den Erkenntnissen aus Untersuchungen des menschlichen Handelns entwickelt EHRLENSPIEL seine „Integrierte Produkterstellungsmethodik (IPE-Methodik)“. Sein Gesamtsystem der integrierten Produkterstellung besteht aus mehreren Ziel-, Sach- und Handlungssystemen, wie in Bild 2-42 dargestellt ist.

[268] DIN 2330 (1979)
[269] angelehnt an Koller (1998)
[270] aus Ehrlenspiel (2009)

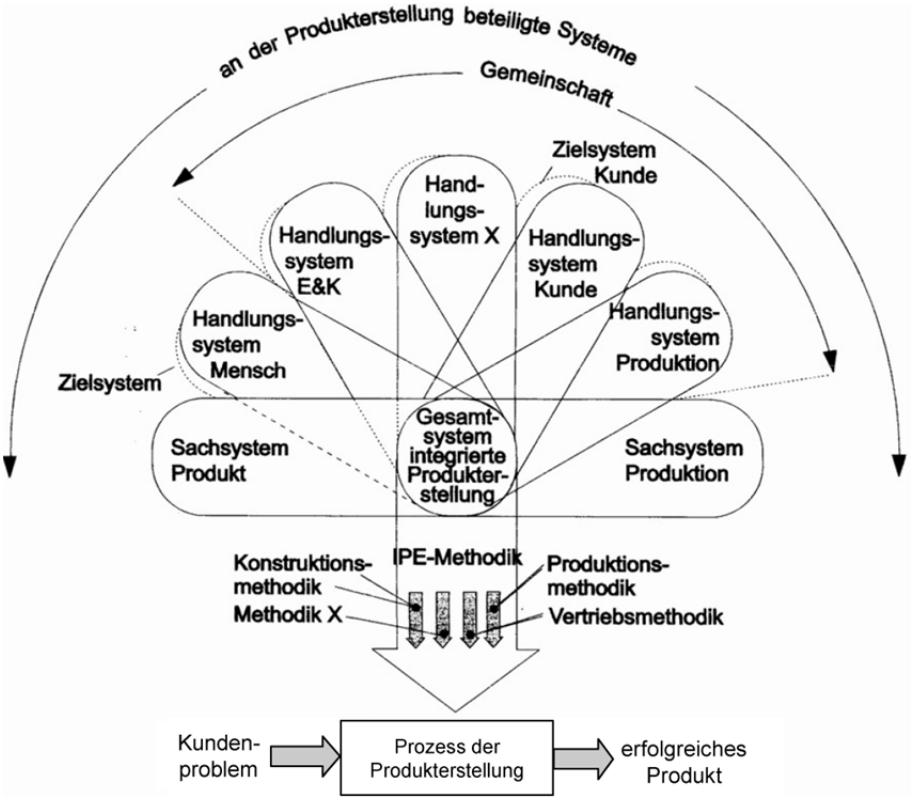


Bild 2-42: Systeme der IPE-Methodik nach EHRLENSPIEL[271]

Die Elemente der Methodik setzen sich aus aufeinander aufbauenden Vorgehensmodellen verschiedener Granularitätsstufen und Ressourcen (Werkzeuge, Produkte, Unternehmen, Menschen etc.) zusammen. Hier wird das Vorgehensmodell näher betrachtet. Darin werden elementare Handlungsabläufe nach dem TOTE-Schema beschrieben, aus denen sich der Vorgehenszyklus (VZ) unter Anwendung von Einzelmethoden aus einem Methodenbaukasten herleitet. Eingerahmt wird die Methodik durch einen Vorgehensplan für das Gesamtprojekt. Bild 2-43 zeigt zwei mögliche, schematische Abläufe der IPE-Methodik, (a) sequentiell und (b) teilparallelisiert über der Zeit.

[271] aus Ehrlenspiel (2009)

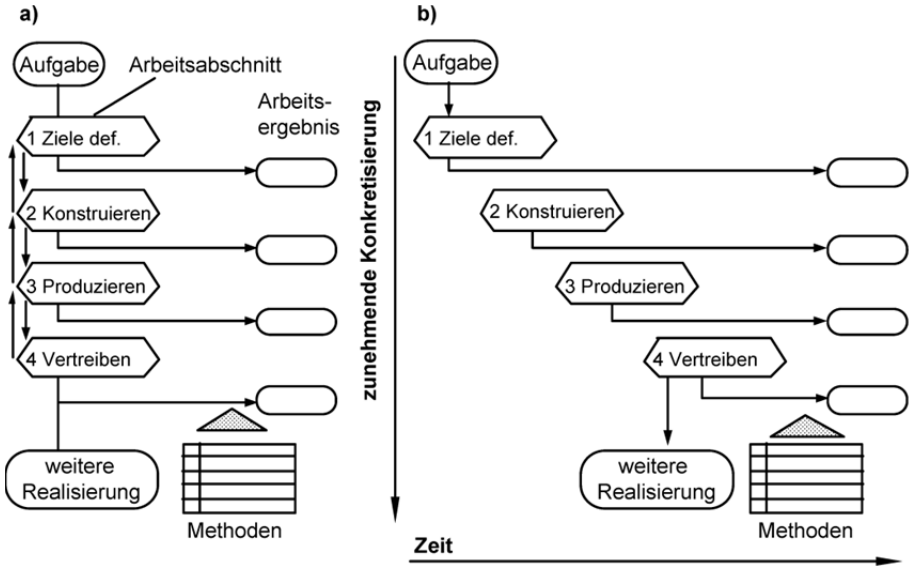


Bild 2-43: Schematische Darstellungen der Methodik[272]

Der Vorgehenszyklus (VZ) ist in jedem Arbeitsabschnitt enthalten, was durch die Form der Kästchen angedeutet wird. EHRLENSPIEL betont die Notwendigkeit einer flexiblen Anwendung der Methodik und nimmt Bezug auf die VDI 2221 (siehe Kapitel 2.9.3) als die Basis zur Strukturierung von Entwicklungsprozessen, erwähnt jedoch auch notwendigen Anpassungsbedarf an heutige Verhältnisse aufgrund ihres sehr prozeduralen Charakters. Als Gründe für die Notwendigkeit, Methoden flexibel einzusetzen, nennt EHRLENSPIEL folgende 6 Gründe:

- die Ergebnisunsicherheit des Entwicklungsprozesses,
- die Problem-/Zielabhängigkeit,
- die Produktabhängigkeit (z.B. Unterschied zwischen mechatronischen und rein mechanischen Produkten),
- die Personenabhängigkeit (Vorbildung; Einfluss der persönlichen Erfahrung),
- die Umsatz- und Zeitabhängigkeit sinnvollen Methodeneinsatzes und
- die Unternehmensabhängigkeit (Groß-, bzw. Kleinunternehmen).

Daher stellt er verschiedene Anwendungsvarianten für seine IPE-Methodik vor, die jedoch immer auf dem zugrundeliegenden, zuvor vorgestellten abstrakten Framework basieren, weshalb an dieser Stelle auf die entsprechende, bereits mehrfach referenzierte Literatur verwiesen sei.

[272] Ehrlenspiel (2009)

2.9.6 Münchner Vorgehensmodell (MVM)

Das Münchener Vorgehensmodell (MVM) bildet die Produktentwicklung als Prozess der Problemlösung ab. Die umfassende Beschäftigung mit der Problemstellung bildet darin einen Schwerpunkt. Der Netzwerkcharakter des Modells unterstützt eine situationsgerechte Navigation durch den Entwicklungsprozess. Die wesentlichen Aspekte des Zwecks des Modells sind (nach LINDEMANN[273]):

- ein Hilfsmittel zur Planung von Entwicklungsprozessen bereitstellen,
- als Orientierungshilfe innerhalb von Prozessen zur Problemlösung dienen und
- Analyse und Reflexion des Vorgehens (auch im Nachhinein).

Zur Erreichung dieser Aspekte unter Wahrung einer weitgehenden Kompatibilität zu bestehenden Vorgehensmodellen baut das MVM grundsätzlich auf den drei Hauptschritten der Problemlösung auf:

- Ziel beziehungsweise Problem klären,
- Lösungsalternativen generieren und
- Entscheidung herbeiführen.

Das MVM besteht aus 7 Verfeinerungen dieser Schritte, die im Gegensatz zu anderen Vorgehensmodellen nicht sequentiell oder iterativ, sondern in Form einer Entwicklungslandkarte mit netzwerkartigem Charakter miteinander gekoppelt sind (siehe Bild 2-44).

[273] Lindemann (2009)

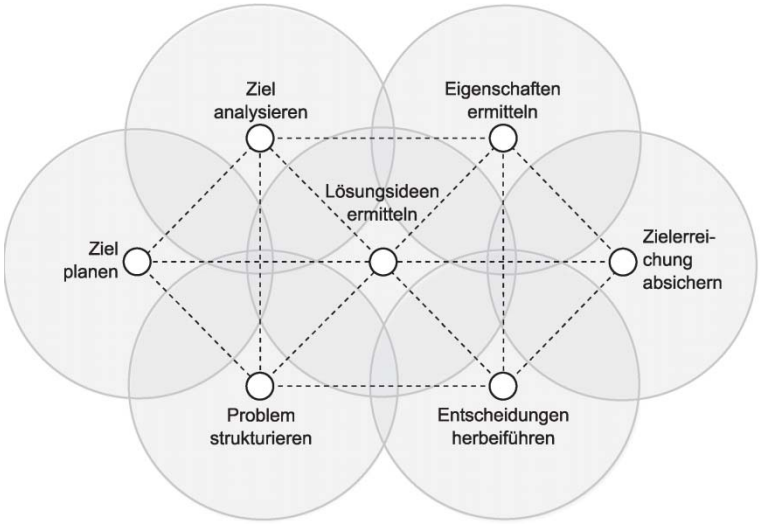


Bild 2-44: Das Münchener Vorgehensmodell (MVM)[274]

Die sich überschneidenden Kreise repräsentieren die teilweise schwierige Abgrenzung der einzelnen Elemente voneinander. Damit dennoch eine gewisse Hilfestellung für ein „Standardvorgehen“ gegeben werden kann, wird empfohlen, die Elemente von links oben nach rechts unten (Ziel planen, Ziel analysieren, Problem strukturieren, Lösungsideen ermitteln, Eigenschaften ermitteln, Entscheidungen herbeiführen, Zielerreichung absichern) zu durchlaufen. Neben diesem Standardvorgehen, das insbesondere „Methodenlaien“ bei der Erstanwendung des MVM unterstützen soll, stellt LINDEMANN[274] auch weitere Beispiele für alternative Nutzungen des Modells vor.

2.9.7 Paderborner 3-Zyklen-Modell der Produktentstehung

Das 3-Zyklen-Modell der Produktentstehung nach GAUSEMEIER ET AL. beschreibt den Prozess von der Produkt- bzw. Geschäftsidee bis zum Serienanlauf mit den drei Hauptaufgabenbereichen der strategischen Produkt- und Technologieplanung, der Produktentwicklung sowie der Produktionsprozess- und Produktionssystem-entwicklung[275] (siehe Bild 2-45).

[274] Lindemann (2009)
[275] Gausemeier et al. (2009b)

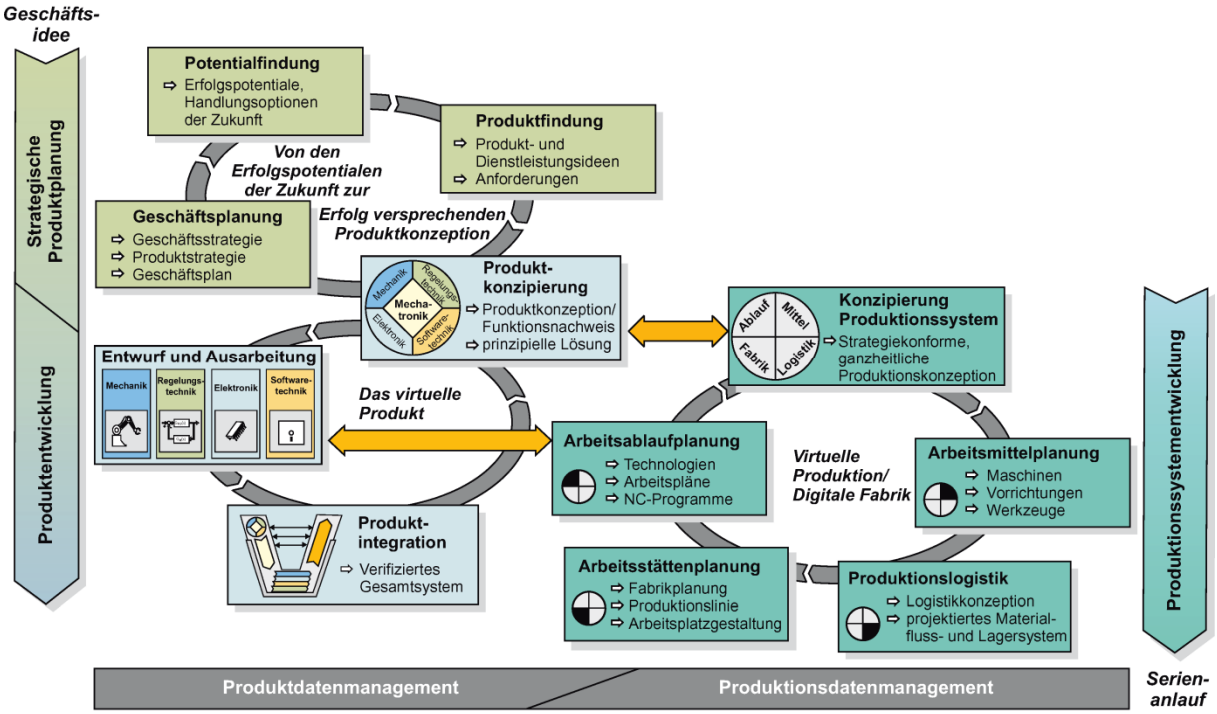


Bild 2-45: 3-Zyklen-Modell der Produktentstehung

Der Produktentstehungsprozess umfasst die Aufgabenbereiche strategische Produktplanung, Produktentwicklung und Produktionssystementwicklung, wobei Letztere auch die Fertigungsplanung bzw. Arbeitsplanung beinhaltet. Nach GAUSEMEIER ET AL. kann der Produktentstehungsprozess nicht als stringente Folge von Phasen und Meilensteinen verstanden werden, sondern als ein Wechselspiel von Aufgaben, die sich allenfalls in drei Zyklen gliedern lassen. Der erste Zyklus dient der methodischen Planung und Vorbereitung der Produktentwicklung, welcher mit der Produktkonzipierung in den zweiten Zyklus, der eigentlichen Produktentwicklung übergeht. Parallel startet der dritte Zyklus zur Produktionssystementwicklung[276].

Wie auch für die Produktentstehung insgesamt, gibt es für den Entwurfsraum von Produkten aufgrund der Komplexität mechatronischer Systeme kein stringentes Phasen-Meilenstein-Vorgehen. Vielmehr wird der Entwurfsraum ähnlich dem MKM nach LINDEMANN[277] in drei Dimensionen aufgespannt (siehe Bild 2-46):

- vom Abstrakten zum Konkreten,
- vom Generellen zum Detail sowie

[276] Gausemeier et al. (2009b)
[277] Vgl. Kapitel 2.4.7

- durch Sichten (Struktur, Verhalten, Gestalt gemäß Y-Modell der Mikroelektronik[278]).

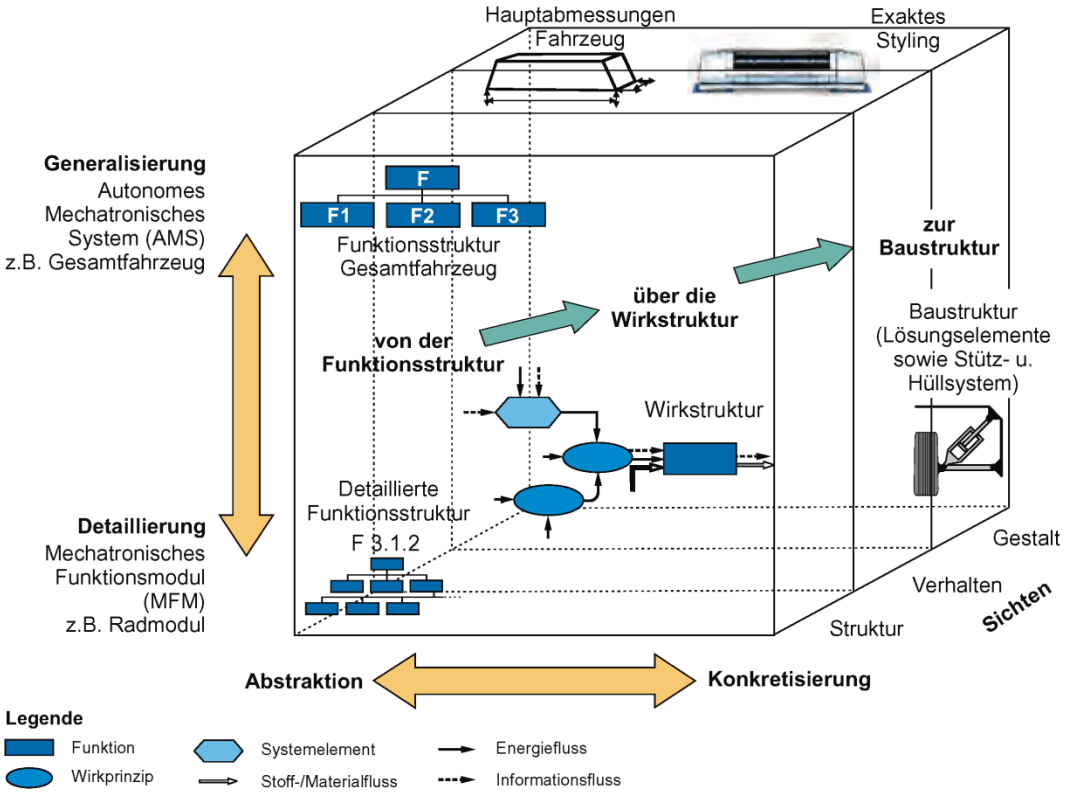


Bild 2-46: Der Entwurfsraum als Grundstruktur des Entwicklungsgeschehens[279]

Das Vorgehen beim Entwurf mechatronischer Systeme wird auch hier als ein Wechselspiel aus Analyse und Syntheseschritten angesehen, wobei jeweils verschiedene Aspekte wie Funktion, Verhalten oder Gestalt betrachtet werden. Dabei sind sowohl Top-Down (Konkretisierung) als auch Bottom-Up (Abstraktion)-Vorgehensweisen möglich. Abhängig von der Entwicklungsaufgabe, den Zielen und den Randbedingungen ergeben sich individuelle Verläufe für die Entwurfsprozesse. Als Werkzeug zur Modellierung der verschiedenen Aspekte eines Produkts schlägt GAUSEMEIER die Spezifikationstechnik CONSENS vor[280].

2.9.8 Das Integrierte Produktentstehungsmodell (iPeM)

Das integrierte Produktentstehungsmodell (iPeM) nach ALBERS ist ein Werkzeug zur Modellbildung von Produktentstehungsprozessen[281]. Es begegnet den Herausforderungen der heutigen, von Zeit- und Kostendruck, hoher Komplexität

[278] vgl. Bleck et al. (1996)
[279] nach Gausemeier et al. (2009b)
[280] Gausemeier et al. (2012), vgl. auch Kapitel 2.6.5
[281] Albers (2010), Albers und Braun (2011a), Meboldt (2008)

durch zahlreiche, stark vernetzte und wechselwirkende Informationen sowie von hoher Dynamik geprägten multidisziplinären Produktentstehung. Ein Ziel des iPeM ist, derzeitige Produktentstehungsprozesse und den darin stattfindenden, intensiven Austausch von Informationen mit einer flexiblen und umfassenden modellbasierten Beschreibung vollständig deskriptiv erfassen zu können. Ein weiteres Ziel ist, durch die gezielte Analyse bestehender Prozesse im Sinne eines kontinuierlichen Verbesserungsprozesses (KVP) die Synthese effizienterer, zukünftiger Prozesse zu ermöglichen und diese zu beschreiben. Zahlreiche Studien belegen, dass insbesondere hier die Schwächen bestehender Prozessmodelle liegen[282]. Ein besonderer Fokus des iPeM liegt daher darauf, einerseits eine hohe Flexibilität und andererseits klare Ordnungsstrukturen für die evidente Unterstützung sowohl von operativ tätigen Produktentwicklern als gleichermaßen auch das Controlling durch das Managements bereitzustellen[283].

Das Metamodell des iPeM basiert auf der allgemeinen Systemtheorie nach ROPOHL[284] und enthält alle Elemente, die notwendig sind, um für individuelle Problemstellungen angepasste Produktentstehungsmodelle daraus abzuleiten[282]. Es beruht auf fünf zentralen Hypothesen, welche die Elemente des Metamodells und ihre Beziehungen untereinander beschreiben[285]:

1. Hypothese: Individualität von Produktentstehungsprozessen

Jeder Produktentstehungsprozess ist einzigartig und individuell. Unterschiedliche Ziele und Randbedingungen sowie unvorhergesehenen Schwierigkeiten werden in subjektiv operierenden und sich verändernden Handlungssystemen behandelt, woraus immer ein einzigartiger Verlauf von Produktentstehungsprozessen resultiert. Dies äußert sich durch zwar ähnliche Elemente (z.B. einzelne Handlungsschritten), jedoch individuelle Beziehungen und Informationsflüsse zwischen ihnen[286].

2. Hypothese: System der Produktentstehung

Auf den Grundlagen der Systemtheorie lässt sich eine Produktentstehung als die Transformation eines (anfangs vagen) Zielsystems in ein konkretes Objektsystem durch ein Handlungssystem beschreiben. Das *Handlungssystem* ist ein sozio-technisches System, das aus strukturierten Aktivitäten, Methoden, Ressourcen und Prozessen aufgebaut ist. Es erstellt und verbindet sowohl das Ziel- als auch das

[282] Meboldt (2008)
[283] Albers und Braun (2011b)
[284] Ropohl (1975), Ropohl (2009)
[285] nach Albers (2010)
[286] Albers et al. (2010b)

Objektsystem, wodurch beide ausschließlich über das Handlungssystem verbunden sind. Das *Zielsystem* beschreibt das gewünschte Produkt und dessen Kontext inklusive aller geplanten Eigenschaften und alle dafür notwendigen Restriktionen, deren Abhängigkeiten und Randbedingungen. Es wird im Verlauf der Produktentstehung kontinuierlich erweitert, angepasst und konkretisiert. Das *Objektsystem* enthält alle Modelle, Dokumente und Artefakte, die als Teillösungen während des Entstehungsprozesses anfallen, einschließlich des resultierenden Produkts selbst. Im Verlauf der Produktentstehung greift das Handlungssystem auf Teile des Objektsystems für Analysen, Verifikation und Validierung zu und leitet daraus neue Ziele und Randbedingungen ab, die wiederum im Zielsystem gespeichert werden. Ziel- und Objektsystem sind vollständig, sobald der angestrebte Zielzustand erreicht ist.

3. Hypothese: Validierung

Die Validierung ist die zentrale Aktivität im Produktentstehungsprozess. Sie führt den Abgleich zwischen Ziel- und Objektsystem durch und erweitert bzw. konkretisiert so das Zielsystem. Daher stellt die Validierung eine eigene Aktivität in der Produktentstehung dar, ist darüber hinaus aber in Form einer Tragweitenanalyse auch Bestandteil jeder weiteren Aktivität der Produktentstehung[287].

4. Hypothese: Zielbeschreibung in der Problemlösung

Die Objekte, die in Produktentstehungsprozessen erstellt werden, müssen hinsichtlich der gewünschten Produktfunktionen, die Teil des Zielsystems sind, beschrieben werden, um die Ziele transparent zu halten. Die Transformation von Zielen in Objekte kann als Problemlösungsprozess betrachtet werden, in dem ein Ist-Zustand in einen gewünschten Soll-Zustand überführt werden soll. Da sowohl der Weg, die Mittel als auch der gewünschte Soll-Zustand teilweise unklar sein können, ist es unabdingbar, Ziele, Vorgehensweisen und (Teil-)Ergebnisse kohärent, also in einer gemeinsamen Sprache zu beschreiben. Nur so können ein dynamischer Projektverlauf mit sich ändernden Parametern frei von Missverständnissen und Spekulationen beschrieben und der Projekterfolg gesichert werden. Hierzu wurde der Contact & Channel – Ansatz (C&C²-A) entwickelt, der in Kapitel 2.4.7 vorgestellt wurde.

[287] Die Tragweitenanalyse ist Teil des Problemlösungsprozesses SPALTEN, der wiederum ein Metamodell für die Mikroaktivitäten innerhalb jeder Makroaktivität darstellt.

5. Hypothese: Beschreibung von Funktionen

Eine technische Funktion benötigt immer mindestens zwei Wirkflächenpaare (WFP) und sie verbindende Leitstützstrukturen (LSS). Ein System, das keine WFP mit seiner Umgebung bildet, erfüllt keine Funktion. Im Umkehrschluss bedeutet dies, dass ein System nur in Interaktion mit seiner Umwelt auch Funktionen ausüben und damit einem Zweck gerecht werden kann. Auch hier hilft der Contact & Channel – Ansatz (C&C²-A), systematisch zu denken, also ein System unter Berücksichtigung seiner Schnittstellen und den Wechselwirkungen mit seiner Umgebung zu entwickeln.

Aus diesen Hypothesen ergibt sich das in Bild 2-47 gezeigte Metamodell des iPeM, bestehend aus dem Systemtriple Zielsystem, Handlungssystem und Objektsystem.

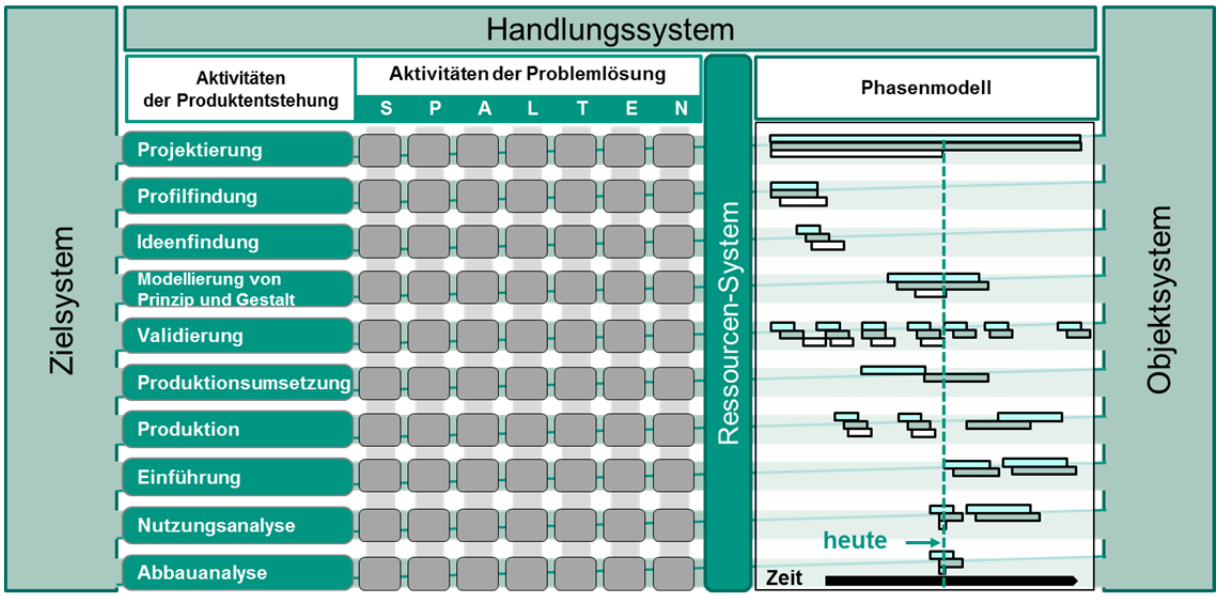


Bild 2-47: Das integrierte Produktentstehungs-Modell (iPeM)[288]

Da das Metamodell insbesondere ein Rahmenwerk zur Beschreibung des Produktentstehungsprozesses ist, unterteilt sich das Handlungssystem weiter in die Aktivitäten der Produktentstehung und die Aktivitäten der Problemlösung. Weiterhin sind das Ressourcen-System und ein Phasenmodell zur grafischen Darstellung des zeitlichen Bezugs der Aktivitäten darin enthalten.

Die Aktivitäten der Produktentstehung orientieren sich an den Lebenszyklusphasen eines Produkts, die matrixförmig dazu angeordneten Aktivitäten der Problemlösung

[288] aus IPEK-PKB (2013), für englische Fassung vgl. Albers und Braun (2011a)

werden anhand des Problemlösungsprozesses SPALTEN untergliedert[289]. Dessen Name ist ein Akronym für die folgenden sieben Problemlösungsschritte:

- Situationsanalyse,
- Problemeingrenzung,
- Alternative Lösungssuche,
- Lösungsauswahl,
- Tragweitenanalyse,
- Entscheiden und Umsetzen sowie
- Nachbereiten und Lernen.

Die Aktivitäten und Problemlösungsschritte sind keinesfalls rein sequentiell abzuarbeiten, vielmehr dient die Matrix den Entwicklern im Prozessverlauf als Orientierung. Jedem der insgesamt 70 Felder der Aktivitätenmatrix lassen sich Tätigkeiten und korrespondierende Entwicklungsmethoden zuordnen. Der geplante und der tatsächliche Verlauf der Aktivitäten eines konkreten Produktentwicklungsprozesses werden im Phasenmodell dargestellt. Das Planungsmodell wird als Implementierungsmodell (graue Balken im Phasenmodell in Bild 2-47) und die Aufzeichnung des realen Verlaufs als Applikationsmodell (weiße Balken) bezeichnet. Auf diese Weise ist es möglich, das Wissen über Prozessverläufe und deren Zusammenhänge mit Ziel-, Objekt- und Ressourcensystem explizit zu speichern und zu kommunizieren. Daraus können beispielsweise Referenzmodelle (türkise Balken im Phasenmodell in Bild 2-47) für nachfolgende Produktentstehungsprozesse oder Teile davon abgeleitet werden, beispielsweise für die Entwicklung der nächsten Produktgeneration oder einer benachbarten Produktbaureihe. Demzufolge dient das Phasenmodell als ein Wissensmanagementwerkzeug zur Steuerung und Optimierung von Produktentstehungsprozessen.

Aktuelle Forschungsarbeiten beschäftigen sich mit der Realisierung einer IT-Implementierungslösung für das iPeM, um beispielsweise Informationszusammenhänge zwischen Produkt und Prozess über semantische Verknüpfungen abbilden zu können oder um dem Entwickler durch eine Übersicht geeigneter Entwicklungsmethoden für die einzelnen Schritte der Aktivitätenmatrix zu unterstützen[290].

---

[289] Albers et al. (2005), Albers und Meboldt (2007)
[290] Albers et al. (2012a), Albers et al. (2012b), Albers et al. (2012f)

2.9.9 Zwischenfazit

Heutzutage etablierte Vorgehensmodelle können die hochdynamischen, multidisziplinären Entwicklungsprozesse komplexer Produkte nicht mehr angemessen beschreiben oder gar eine Hilfestellung für eine problemgerechte, zielführende Vorgehensweise leisten. Dies begründet sich in ihrem meist starren und sequentiell geprägten Aufbau. Dies hat auch die Wissenschaft erkannt und versucht, dem durch flexiblere Modelle entgegenzuwirken, was jedoch häufig in einem zu hohen Abstraktionsgrad und damit einer mangelnden Orientierungshilfe mündet.

Der systemische Gedanke bei der Modellbildung von Systemen wird in den meisten Vorgehensmodellen bereits aufgegriffen. So werden Ansätze für die Modellbildung technischer Systemen über verschiedene Abstraktionsebenen vorgestellt, die sich inhaltlich weitgehend überschneiden und ihre Unterschiede oft vor allem in der Sprache finden.

Das integrierte Produktentstehungsmodell trennt von vornherein Ziele, Handlungen, Ressourcen und entstehende Objekte. Bei der Überführung von Soll zum Ist (im Handlungssystem) wird durch eine Aktivitätenmatrix ein flexibles Modell zur Navigation bereitgestellt, in der Entwickler und Manager basierend auf der realen Differenz zwischen Soll und Ist die jeweils angemessenen Entscheidungen und Handlungen entnehmen können. Gleichzeitig werden sämtliche, in den Handlungen entstehenden Informationen in den entsprechenden Systemen abgelegt. Hierzu bedarf es einer geeigneten Modellbildung von Zielen und Objekten, um diese Informationen auch eindeutig nachvollziehbar speichern und vor allem zur Verfügung stellen zu können. Einen Beitrag hierzu liefert das Denkmodell des Contact & Channel – Ansatzes (C&C²-A), wodurch der Zusammenhang von Funktionen und resultierender Gestalt eines Systems eindeutig beschrieben werden kann. Bisher gibt es für beide Modelle zwar erste prototypische Softwareimplementierungen, jedoch noch keine etablierte und ausgereifte IT-Lösung.

2.10 Systemmodellierungsmethoden

Seit Erscheinen der SysML als fachdisziplinübergreifende Modellierungssprache besteht der Bedarf für eine geeignete Handlungsanleitung für deren zielgerechte Anwendung. Die bekanntesten Methoden werden in den folgenden Kapiteln vorgestellt. ESTEFAN hat im Auftrag der INCOSE eine Studie zu existierenden Modellierungsmethoden durchgeführt, die – abgesehen von der offiziellen INCOSE-Methode OOSEM - meist spezifisch auf die Bedürfnisse der entwickelnden Unternehmen abgestimmt sind, weshalb auf deren Vorstellung in dieser Arbeit verzichtet wird[291].

2.10.1 Object-Oriented Systems Engineering Method (OOSEM)

OOSEM (Object-Oriented Systems Engineering Method) ist eine ursprünglich bei Lockheed Martin entworfene und von der INCOSE unter anderem durch eine geeignete Werkzeugunterstützung weiterentwickelte Methode zur integrativen Anwendung von traditionellem und modellbasiertem Systems Engineering. Dabei handelt es sich um eine szenariogesteuerte Top-down-Entwicklungsmethode unter Anwendung der SysML für Analyse, Spezifikation, Design und Verifikation von Systemen. Wichtige Entwicklungsaktivitäten sind dabei die Analyse der Ziele und Wünsche von Interessensvertretern, die Anforderungsanalyse, der Entwurf logischer Architekturen und die Synthese möglicher physischer Architekturen. Der zugrundeliegende Prozess setzt objektorientierte Konzepte und andere Modellierungstechniken zum Entwurf derart flexibler Systeme ein, die evolvierenden Technologien und dynamisch veränderlichen Anforderungen gerecht werden sollen. Das zentrale Systemmodell stellt ein primäres Ergebnis dieses Prozesses dar, das gleichzeitig die Möglichkeit der Wiederverwendung von Informationen eröffnet[292].

2.10.2 Systems Modeling Methodology (SysMOD) und Functional Architectures for Systems (FAS-Method)

Die von WEILKIENS entwickelte Systems Modeling Method (SysMOD) ist eine pragmatische Methode zur Modellierung komplexer Systeme unter Anwendung der SysML, die unter anderem Rollenbeschreibungen von Projektbeteiligten, Aufgaben, Arbeitsergebnisse und Handlungsanleitungen umfasst[293]. Die Methode wird durch ein SysML-Profil mit entsprechenden Elementen zur Modellierung unterstützt. Darauf aufbauend wurde die integrierbare Methode „Funktionale Architekturen für Systeme“

[291] Estefan (2008)
[292] Friedenthal et al. (2011)
[293] Weilkiens (2008), Korff und Weilkiens (2010)

speziell für den Aspekt der funktionsbasierten Modellierung von Systemarchitekturen entwickelt[294] (vgl. Kapitel 2.6.2.6).

2.10.3 Object Process Methodology (OPM)

DORI[295] und SHARON[296] definieren die Object-Process Methodology (OPM) als formales Paradigma für die ganzheitliche, prozessübergreifende Entwicklung und modellbasierte Repräsentation von Systemen und der Planung entsprechender Prozesse. Die Methode umfasst die Kombination von einfachen, visuellen Diagrammen (Object-Process Diagrams (OPDs)) und gebundene, natürliche Sätze zur Funktions-, Struktur- und Verhaltensbeschreibung von Systemen in einem integrierten Modell. Jedes OPD-Konstrukt wird hierzu durch semantisch äquivalente OPL (Object-Process Language)-Sätze ausgedrückt und umgekehrt. Die OPL ist somit eine zweifach verwendbare Sprache zur Beschreibung von sozio-technischen Systemen, deren zwei zentrale Elemente die Blöcke (Objekte in einem Zustand) und Prozesse (welche Objekte oder deren Zustände transformieren) sind.

Die Prozessplanung wird in einem später als Erweiterung entwickelten Model-Based Project Plan beschrieben (MBPP), dessen Anwendung in der Praxis laut einer Studie aufgrund eines nicht ausreichenden Reifegrades derzeit noch nicht sinnvoll ist[297]. Die Modellierungsmethode wurde unter anderem auch in der INCOSE-Studie von ESTEFAN[298] über die wichtigsten Model-Based Systems Engineering-Methoden untersucht und vorgestellt.

Für die Modellierung mit OPM wurde das Softwarewerkzeug OPCAT[299] entwickelt, in das auch erste Erweiterungen der Modellierungssprache, beispielsweise für Multiagent Systems (MAS), integriert wurden[300]. Der Schwerpunkt der Modellierung mit OPM liegt auf Softwaresystemen, die Anwendung auf Systeme mit mechanischen Elementen ist aufgrund der eingeschränkten Modellierungs- und Visualisierungsmöglichkeiten nur bedingt möglich. Daher stellen GROBSHTEIN UND DORI[301] eine Ergänzung von OPM durch SysML vor, die es ermöglicht, SysML-Diagramme zur Visualisierung von OPM-Systemmodellen zu nutzen. In einem Experiment konnte nachgewiesen werden, dass durch den Standard SysML sowohl

---

[294] Korff et al. (2011), Lamm und Weilkiens (2010)
[295] Dori (2002)
[296] Sharon et al. (2008), Sharon et al. (2009)
[297] Sharon et al. (2011)
[298] Estefan (2008)
[299] http://www.opcat.com/
[300] Sturm et al. (2010)
[301] Grobshtein und Dori (2011)

das Systemverständnis der Teilnehmer als auch die Interoperabilität der Modelle verbessert werden konnte. Hierbei wird betont, dass die gemeinsame Nutzung den Mehrwert erzeugt, und nicht der Ersatz von OPM durch SysML. In einer früheren Studie von REINHARTZ-BERGER UND DORI[302] wurde bereits untersucht, wie sich der Vergleich von OPM mit der UML bei der Modellierung von Softwaresystemen verhält, hier fielen die Ergebnisse hinsichtlich Systemverständnis und Modellierung zugunsten OPM aus. Auch RAMOS ET AL. bekunden der OPM eine deutlich einfachere Handhabung als der SysML aufgrund ihres einfachen Aufbaus und damit eine bessere Eignung für High-Level-Systeme[303].

2.10.4 AFLP-Methode in Anlehnung an VDI 2206

EIGNER ET AL. stellen ein Erweitertes V-Modell für Model Based Systems Engineering vor[304]. Sie integrieren den AFLP[305]-Ansatz als Methodik für eine modellbasierte Systementwicklung, mit der insbesondere die Kopplung von PDM/PLM-Systemen und fachdisziplinspezifischen Werkzeugen an ein multidisziplinäres Systemmodell erreicht werden soll (siehe Bild 2-48).

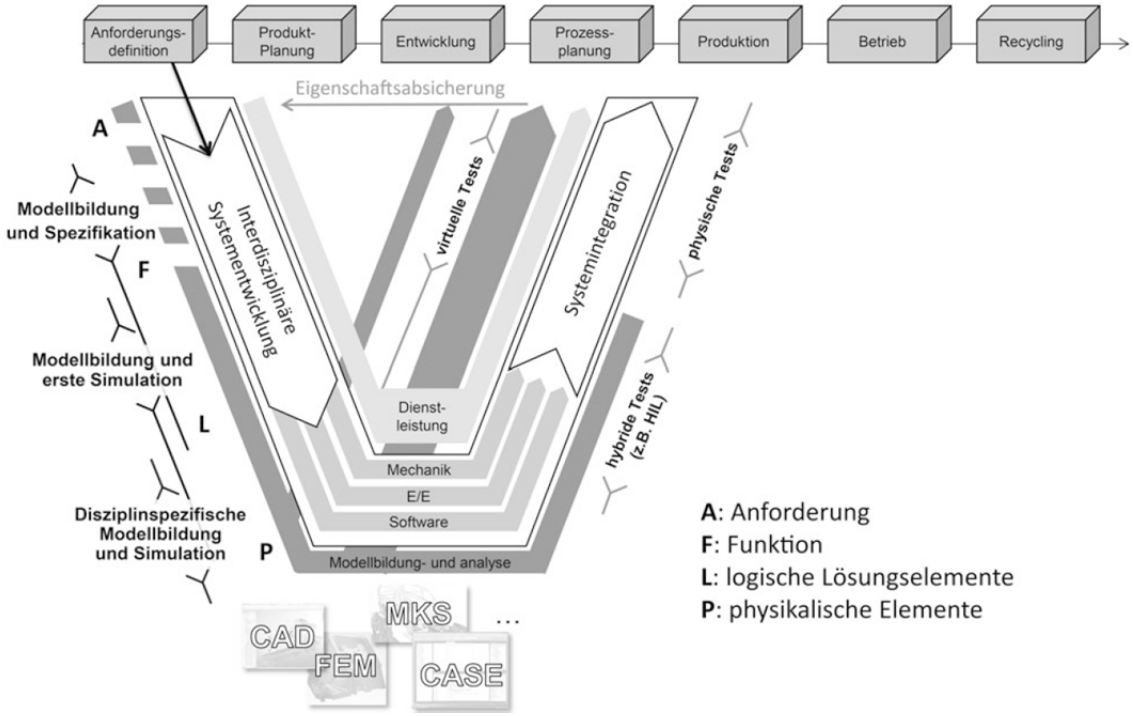


Bild 2-48: „AFLP-Methode“ in Anlehnung an VDI 2206[306]

[302] Reinhartz-Berger und Dori (2005)
[303] Ramos et al. (2012)
[304] Eigner et al. (2012a), Eigner et al. (2012b)
[305] Anforderungen, Funktionen, Logische Architektur und Physische Architektur
[306] VDI-2206 (2004), Eigner et al. (2012b)

Die Anwendung der Modellierungsmethodik mit SysML als Modellierungssprache wurde bereits in Kapitel 2.6.2.6 vorgestellt. Der AFLP-Ansatz bzw. RFLP-Ansatz (englische Variante) wurde auch von Dassault Systèmes in der neuen Generation V6 von CATIA übernommen, wodurch das ursprünglich für 2D- und 3D-CAD-Design entwickelte Werkzeug um die Möglichkeit zur Modellierung von Anforderungen, Funktionen und einer logischen Architektur ergänzt wurde[307]. Der Ansatz, disziplinübergreifende Systemmodelle, Simulationsmodelle, Konstruktionsmodelle und auch eine PDM-Plattform in einem Softwarewerkzeug vereinen zu wollen, erscheint im Sinne des Anwenderbedarfs einer Komplexitätsreduktion wenig zielführend. Darüber hinaus handelt es sich um ein proprietäres Werkzeug, das im Gegensatz zu offenen Modellierungsstandards eine ungewollte Abhängigkeit vom Anbieter der Software hervorruft[308]. Erste Anwendererfahrungen wurden dem Autor der Arbeit u.a. über Arbeitsgruppen der Gesellschaft für Systems Engineering gespiegelt, die einen stark eingeschränkten Funktionsumfang des Systems bei fachdisziplinübergreifender Modellierung im Vergleich zur SysML berichten.

2.10.5 System Core Analyses for Robustness and Safety (SysCARS)

Die Methode SysCARS (System Core Analyses for Robustness and Safety) wurde durch Mitarbeiter der Firma VALEO entwickelt, nachdem diese feststellten, dass bisherige Methoden zur Anwendung der SysML den Bedürfnissen ihres Unternehmens nicht ausreichend gerecht wurden. Die Methode ist speziell auf die Entwicklung und funktionale Absicherung eingebetteter Systeme der Automobilindustrie zugeschnitten[309], weist jedoch sowohl Parallelitäten zu anderen Methoden als auch auf andere technische Systeme übertragbare Aktivitäten auf.

Das Vorgehensschema zur Modellierung in SysML weist die 5 zentralen, aufeinander aufbauenden Aktivitäten auf:

- Definition der Interessensvertreterbedürfnisse,
- Anforderungsanalyse,
- Entwurf der logischen Architektur,
- Design der Physischen Architektur und
- Definition der Komponentenbedürfnisse.

---

[307] EM AG (2012)

[308] Bspw. die individuelle Anbindung von Modellen in Fremdsoftware, die Individualisierung auf bestimmte Branchen oder Fachdisziplinen etc.

[309] Beispielsweise durch die besondere Berücksichtigung der Forderungen der ISO 26262 bezüglich der funktionalen Sicherheit von Straßenfahrzeugen, vgl. ISO-26262 (2012)

Die Methode wurde in internen Pilotprojekten und Forschungskooperationen des Unternehmens evaluiert[310], wobei als offene Punkte für die Zukunft insbesondere die direkte Kopplung von Modellen und die Verbesserung der Anwenderfreundlichkeit kommerzieller Modellierungswerkzeuge benannt wurden[311].

2.10.6 Zwischenfazit

Die vorgestellten Systemmodellierungsmethoden basieren teilweise auf den in Kapitel 2.4 vorgestellten wissenschaftlichen Modellbindungsansätzen, wurden aber deutlich stärker aus der Anwendung heraus entwickelt und vor allem auf die Verwendung einer konkreten Modellierungssprache hin ausgerichtet. Oftmals bleiben auch diese Methoden auf einer sehr generischen Ebene, um nicht zu umfangreich zu werden. Ergänzt um die Erläuterungen der konkreten Diagramme der verwendeten Modellierungssprache mit ihren Entitäten und Relationen bieten vor allem OOSEM, SysMOD und FAS eine zumindest grundlegende Unterstützung für Entwickler bei der Anwendung der Modellierungssprache SysML. Andererseits werden die durchgängige Kopplung der Informationen und eine Beschreibung der Art der Wechselwirkungen nur unzureichend beschrieben. Das größte Defizit der Methoden findet sich jedoch in der mangelnden Eignung von Methode und verwendeter Modellierungssprache für die Beschreibung mechatronischer Systeme, insbesondere der enthaltenen physikalischen Aspekte. Weiterhin bietet keine der Methoden eine angemessene Unterstützung von Managern bei strategischen Entscheidungen, beispielsweise der Auswahl von Lösungskonzepten unter technischen und wirtschaftlichen Randbedingungen. Dies begründet sich insbesondere darauf, dass deren Fokus auf der Modellierung der Produktarchitektur und deren Anforderungen liegt. Prozessrelevante Informationen wie beispielsweise die Auslastung von Ressourcen oder dem Fortschritt bzw. Status von Entwicklungsaktivitäten werden hier nicht berücksichtigt.

[310] Bspw. SAFE (Safe Automotive soFtware architEcture), das Teil des EU-Projekts CESAR war
[311] Andrianarison und Piques (2010), Piques und Andrianarison (2012)

3 Motivation und Zielsetzung

3.1 Motivation

Mit der modellbasierten Systemtechnik etablieren sich Ansätze für eine ganzheitliche, abstrahierte Betrachtung und Beschreibung technischer Systeme, die vor allem die Interaktion eines Produkts mit seiner soziotechnischen Umwelt einbeziehen. So entstanden Begriffe wie Eingebettete Systeme, Intelligente selbstoptimierende Systeme, Systems of Systems oder Cyber-Physical Systems. Sie beschreiben multidisziplinäre Systeme als einen intelligenten Verbund aus Mechanik, Elektrik, Elektronik und Software, die in Interaktion mit weiteren technischen Systemen und dem Menschen ein hochkomplexes System mit zahlreichen Wechselwirkungen und einhergehenden Unsicherheiten sowie vielen Unvorhersehbarkeiten darstellen[312]. Damit gehen sowohl wirtschaftliche Risiken für Unternehmen als auch Sicherheitsrisiken für den Anwender einher, die es zu minimieren gilt.

Der Stand der Forschung zeigt auf, dass in den vergangenen Jahrzehnten verschiedenste Ansätze entwickelt wurden, die sowohl auf generischer Ebene versuchen, Systeme und Wechselwirkungen zu beschreiben als auch Methoden und Handlungsempfehlungen zur Entwicklung von multidisziplinären Systemen anzubieten. Andererseits wurden hochspezialisierte, aus den Fachdisziplinen heraus getriebene Methoden und Werkzeuge entwickelt, was zu einer zunehmenden Schwierigkeit für Entwickler und Manager führt, all diese Ansätze und Technologien in ihrer Arbeit zielführend zum Umgang mit Komplexität und damit zur Minimierung der Unsicherheit und Unvorhersehbarkeit einsetzen zu können.

Die **Motivation** der vorliegenden Arbeit liegt darin, eine bisher nicht umfassend vorhandene[313] Basis für eine durchgängige, fachdisziplinübergreifende Anwenderunterstützung in der Produktentwicklung mittels einer geeigneten Modellierungstechnik zu schaffen. Dies soll durch die Kopplung der Stärken existierender Modellierungssprachen, Technologien und Werkzeuge sowie Konzepte für deren Einbettung in eine übergeordnete Entwicklungsumgebung realisiert werden. Darüber hinaus sollen einhergehende Potentiale sowie fortwährende wissenschaftliche Herausforderungen aufgezeigt werden. Mit dieser Technik soll eine

[312] Lee (2008)

[313] Z. B. stellen Broy et al. (2010) fest, dass sowohl ein semantisches Fundament als auch klarere Modellierungsregeln und entsprechende Toolunterstützung bisher unzureichend sind.

bessere Unterstützung des Anwenders erreicht und dadurch der Einsatz von Modellen in der Produktentstehung nachhaltig gestärkt werden.

Die daraus abgeleitete **Zielsetzung** ist, modellbasierte Systementwicklung in der Produktentstehung durch die Definition einer gemeinsamen Sprache für den Kontext der Modellbildung technischer Systeme, ihre Formalisierung durch Modellbildung, Modellierungsregeln und ein Vorgehensmodell für die Anwendung der entstehenden Modellierungstechnik stärker zu etablieren. Die Aspekte dieser Ziele werden nachfolgend erläutert.

3.2 Zielsetzung

Zunächst werden aus einer Argumentationskette heraus fünf Forschungshypothesen als langfristige Vision bezüglich der modellbasierten Produktentwicklung formuliert. Daraus wird die konkrete Zielsetzung dieser Arbeit abgeleitet.

Heutzutage gibt es zahlreiche, hochspezialisierte Fachdisziplinen für die verschiedensten Forschungsfelder. Durch teilweise gemeinsame Wurzeln wurden Fachbegriffe übernommen und neu interpretiert oder definiert, wodurch eigene Fachsprachen mit teilweise sehr heterogenem Begriffsverständnis entstanden sind. Dies führt zu massiven Kommunikationsproblemen, wenn verschiedene Fachdisziplinen in einem gemeinsamen Entwicklungsprojekt zusammenarbeiten müssen. Daraus leitet sich die erste Forschungshypothese ab:

H1: Die Basis einer erfolgreichen Entwicklung multidisziplinärer technischer Systeme ist eine gemeinsame, kompakte Sprachbasis für den Kontext der Modellbildung technischer Systeme als Kern fachdisziplinübergreifender Kommunikation.

Auf einer solchen Sprachbasis kann ein homogenes System- und Problemverständnis durch die semantische Verknüpfung heterogener Fachbegriffe geschaffen werden. Hierzu dürfen jedoch nicht alle Begriffe vereinheitlicht werden, sondern nur jene, die für die fachdisziplinübergreifende Kommunikation bei der Modellbildung technischer Systeme von zentraler Bedeutung sind. Kommunikation dient dem Austausch von Wissen. Um dieses Wissen explizit und verständlich abzubilden und dabei Redundanzen und Widersprüche zu vermeiden, bedarf es der Nutzung formaler Modelle. Daraus leitet sich die zweite Forschungshypothese ab:

H2: Die nachhaltige Kommunikation fachdisziplinübergreifenden Wissens kann über anwenderfreundliche, multidisziplinäre Modelle erreicht werden.

Dieser Forschungshypothese liegen eine Reihe weiterer Aussagen zugrunde, beispielsweise durch die Erläuterung der Bedeutung von „nachhaltig“ und „anwenderfreundlich“. Unter nachhaltig versteht der Autor, dass

fachdisziplinübergreifend relevante Informationen durch Modelle eindeutig und vollständig beschrieben sowie semantisch verknüpft werden. Somit kann das entstehende Wissen erhalten und im Bedarfsfall abgerufen werden. Dies soll auf anwenderfreundliche Art geschehen, was bedeutet, dass sowohl die Speicherung von generiertem Wissen als auch dessen Abruf auf möglichst einfachem Wege möglich sein müssen. In der Forschungshypothese wird bewusst von Modellen in der Mehrzahl gesprochen, da selbst die Beschränkung auf Wissen, das zwischen Fachdisziplinen zu kommunizieren ist, so umfangreich sein kann, dass man nicht immer jede Art der enthaltenden Information in einem einzigen Modell erfassen kann. Diese Arbeit verfolgt den Ansatz, dass zunächst die relevanten Elemente von Ziel- und Objektsystem für ein konkretes Produkt in einem fachdisziplinübergreifenden Systemmodell beschrieben und vernetzt werden. Darüber hinausgehende Ziele (beispielsweise für folgende Produktgenerationen) wie auch Objekte, die nicht Teil der gewählten Produktlösung sind (z.B. Alternativkonzepte), sowie die Elemente des Handlungssystems werden explizit ausgeschlossen[314]. Die Kopplung des Produktmodells mit weiteren Modellen wie beispielsweise einem Prozessmodell des zugehörigen Handlungssystems wird empfohlen[315], im Rahmen dieser Arbeit jedoch nicht umgesetzt. Die Konzepte des integrierten Produktentstehungsmodells (iPeM, vgl. Kapitel 2.9.8) bieten hierzu ein Rahmenwerk, diese Systeme auf einer übergeordneten Ebene zu orchestrieren. Die Zielsetzung des hier verfolgten Produktmodells ist, die Flexibilität der Modellierungstechnik und damit deren Übertragbarkeit zu maximieren, was in der dritten Forschungshypothese zum Ausdruck kommt:

H3: Ein geeignetes Modell zur fachdisziplinübergreifenden Beschreibung von Ziel- und Objektsystem eines Produkts muss eindeutig und flexibel sein, den Lösungsraum maximieren und gleichzeitig die Handhabung seiner Komplexität durch gezielte Beschränkung auf einen definierten Abstraktionsgrad verbessern, um Unsicherheiten zu minimieren.

Diese Forschungshypothese beinhaltet drei wesentliche Ziele bei der Modellbildung. Einerseits soll die Sprache konkret und eindeutig genug sein, dass Entwickler fachdisziplinübergreifend relevante Aspekte eines technischen Systems eindeutig beschreiben können, ohne dabei Informationen unvollständig oder widersprüchlich

[314] Hiermit werden die im fachdisziplinübergreifenden Modell zu beschreibenden Elemente von Ziel- und Objektsystem bewusst auf ein Maß beschränkt, das nach Kenntnisstand des Autors auf Basis heutiger Technologien im Rahmen dieser Arbeit realisierbar erscheint. So wird hier beispielsweise auf die Entwicklung einer Methode zur Modellierung von Zielen und Objekten für Baukästen / Plattformen oder die Gegenüberstellung von Produktgenerationen verzichtet.

[315] Siehe hierzu Kapitel 8.2.2

abzubilden oder sie gar zu verlieren. Andererseits sollen bei der Modellierung keine technischen Lösungen oder Lösungsprinzipien vorweg genommen werden. Beispielsweise darf bei der qualitativen Beschreibung einer Produktfunktion[316] keine direkte Transformation auf physische Systembausteine erfolgen, da diese auch über Merkmale und Eigenschaften verfügen, die nicht funktionsrelevant sind (und demnach eine aus Gesamtsystemsicht eventuell suboptimale Lösung vorwegnehmen). Darüber hinaus darf die Sprache nicht so umfangreich werden, dass der Modellierungsaufwand den Nutzen überschreitet und ihre Handhabung kompliziert wird, da dies ihre Akzeptanz bei Entwicklern stark beeinträchtigt. Auch soll die Sprache nicht dazu dienen, ein System vollständig zu beschreiben, sondern sich gezielt auf die für den jeweiligen Anwender unbedingt erforderlichen Informationen beschränken. Daraus leitet sich die nächste Forschungshypothese ab:

H4: Die fachdisziplinübergreifende Modellierungssprache soll keine bestehenden Modelle ersetzen, sondern diese miteinander semantisch vernetzen und somit um disziplinübergreifend relevante Abhängigkeiten und Wechselwirkungen ergänzen.

Diese Forschungshypothese setzt voraus, dass Informationen, die bereits in Modellen vorhanden sind, nur dann Teil eines fachdisziplinübergreifenden Produktmodells sein dürfen, wenn sie für die disziplinübergreifende Kommunikation relevant sind, also mindestens zwei Fachdisziplinen betroffen sind. Außerdem ist die fachdisziplinübergreifende Modellierung dann zu beenden, wenn sie innerhalb eines fachdisziplinspezifischen Modells ohne Verlust von Informationen und Wechselwirkungen fortgeführt werden kann. Durch den kontinuierlichen Informationszuwachs durch Konkretisierung der Informationen im Verlauf des Produktentstehungsprozesses müssen entstehende Modelle auch kontinuierlich synchronisiert werden. Dabei ist sicherzustellen, dass der Entstehungsort und -grund neuer Informationen und Abhängigkeiten nachvollziehbar ist, was in den meisten Fällen eine Vernetzung zwischen Ziel- und Objektsystem bedeutet. Diese neu entstandenen Informationen können, wie die Forschungshypothese voraussetzt, nicht in existierenden Modellen abgebildet und dargestellt werden. Daraus ergibt sich die fünfte Forschungshypothese:

[316] Dies meint die Beschreibung logischer Abläufe von Verarbeitungsschritten bzw. der zugehörigen, verarbeiteten Objektflüsse (Stoff, Energie und Information)

H5: Die fachdisziplinübergreifende Modellierungssprache muss Informationen und deren Zusammenhänge anwendergerecht darstellen.

Dies bedeutet, dass jeder Anwender in der Lage sein muss, leicht verständliche Sichten auf das fachdisziplinübergreifende Systemmodell zu erhalten, die alle im Zusammenhang mit der gewünschten Information relevanten Aspekte umfassen und nicht relevante Aspekte explizit ausblenden. Hierbei werden auch Anwender einbezogen, die nicht selbst modellieren, sondern nur Informationen aus dem Systemmodell beziehen, also beispielsweise Diagramme lesen. Diese Anforderung ist aufgrund des äußerst heterogenen Anwendungsspektrums einer fachdisziplinübergreifenden Modellierungssprache meist nicht zu hundert Prozent erfüllbar, jedoch ist es möglich, durch geeignete Diagramme und Filter den Anteil fehlender und zu viel vorhandener Informationen zu minimieren.

Diese Forschungshypothesen stellen eine Vision dar, die in vielen Punkten einzuschränken ist und zu ihrer Realisierung umfassender, über den Rahmen dieser Arbeit hinausgehender Forschung bedarf. Daher lautet die Zielsetzung dieser Arbeit auch, die Potentiale und Herausforderungen derzeitiger Modellbildungsansätze, Modellierungssprachen, Methoden, Vorgehensmodelle und Technologien zu identifizieren und daraus eine Basis einer gemeinsamen Sprache für den Kontext der Modellbildung technischer Systeme sowie eine Modellierungstechnik als Beitrag zur Erreichung der oben beschriebenen Vision vorzuschlagen. Diese Modellierungstechnik soll eine durchgängige Modellierung der fachdisziplinübergreifend relevanten Elemente und Relationen von Zielsystem und Objektsystem **eines** Produkts ermöglichen. Der wissenschaftliche Ansatz zur Erreichung dieses Ziels ist eine Erweiterung der Systems Modeling Language (SysML) um ein Profil zur eindeutigen und differenzierten Beschreibung eines Zielsystems sowie um Konzepte des Contact & Channel – Ansatzes (C&C²-A). Diese sollen insbesondere eine methodische Unterstützung des Anwenders bei der Analyse des Zusammenhangs zwischen Funktionen aus ausführender Gestalt sowie der systematischen Synthese neuer Gestaltlösungen für geforderte Funktionen ermöglichen[317]. Weiterhin werden Modellierungsregeln und ein flexibles Vorgehensmodell für die Anwendung der Modellierungssprache sowie Konzepte für deren Einbettung in eine durchgängige Entwicklungsumgebung vorgestellt. Der Fokus der in dieser Arbeit vorgestellten Modellierungstechnik liegt auf den Anforderungen und Bedürfnissen von Maschinenbauingenieuren und ferner

[317] Eckert et al. (2010) stellen fest, dass Produktmodelle durch die Bereitstellung von Informationen und deren Abhängigkeiten eine Unterstützung des Anwenders sowohl bei der Analyse (deduktives Begründen) als auch der Synthese (kreatives Denken) technischer Systeme bereitstellen

Managern[318], welche in der heutigen, aus der UML und damit der Softwareentwicklung heraus entstandenen SysML, bisher nicht hinreichend berücksichtigt wurden. Die SysML wird in dieser Arbeit eingesetzt und erweitert, da sie einerseits eine hohe Flexibilität aufweist und andererseits Teil einer umfassenden Standardfamilie aus Modellierungssprachen, Datenaustauschformaten und Transformationsbeschreibungen ist. Dadurch erwartet der Autor neben einer höheren Akzeptanz potentieller Anwender gute Implementierungs- und Evaluierungsmöglichkeiten technischer Prototypen. Einschränkend ist zu erwähnen, dass die Möglichkeiten der Erweiterung der SysML insbesondere aufgrund der eingesetzten Modellierungswerkzeuge beschränkt sind, weshalb das resultierende Sprachprofil für die SysML nur einige der identifizierten Konzepte umfasst. Weiterhin ist konkretes Ziel dieser Arbeit, den Mehrwert durch die Nutzung der implementierten Konzepte der Modellierungstechnik an Beispielprojektszenarien zu evaluieren, weiterzuentwickeln und an weiteren Beispielen zu validieren. Parallel soll die Sprachbasis für die Modellbildung technischer Systeme entwickelt werden, bestehend aus Definitionen zentraler Begriffe und ergänzt um grafische Visualisierungen semantischer Zusammenhänge zwischen diesen Begriffen. Daraus soll ein erster Prototyp einer semiformalen Ontologie[319] entstehen, der in nachfolgenden Forschungsarbeiten weiter ausgearbeitet, validiert und verfeinert werden kann.

[318] Manager benötigen insbesondere Informationen über den Verlauf der Aktivitäten des Handlungssystems und der Ressourcen, daher können deren Bedürfnisse durch die hier angestrebte Produktmodellierung in nur sehr eingeschränktem Maße berücksichtigt werden.

[319] Darunter werden hier Grafiken, bestehend aus Begriffen als Entitäten (Objekten) und semantischen Relationen (Beziehungen) verstanden.

4 Forschungsdesign zur Entwicklung einer Modellierungstechnik

In diesem Kapitel wird die Vorgehensweise bei der Entwicklung der angestrebten Modellierungstechnik vorgestellt. Zur Beschreibung der Ziele, der Handlungsschritte und der daraus entstandenen Resultate dieser Arbeit wird das Systemtriple nach ROPOHL zugrunde gelegt. Zunächst wird hierzu das finale Zielsystem dieser Arbeit spezifiziert, bestehend aus den Zielen und den einhergehenden Anforderungen, Herausforderungen und Randbedingungen.

4.1 Das Zielsystem der Modellierungstechnik

Die angestrebte Modellierungstechnik soll den Übergang von einer dokumentenbasierten zur modellbasierten Produktentwicklung durch Unterstützung der Produktentwickler bei der Bildung einer modellbasierten Wissensbasis schaffen. Folgende Ziele ergeben sich für diese Wissensbasis im Produktentwicklungsprozess zur bestmöglichen Unterstützung der Akteure:

- Wissen muss fachdisziplinübergreifend verständlich kommuniziert werden, um ein homogenes Systemverständnis aller Anwender zu erreichen.
- Ziele, Objekte und deren Wechselwirkungen müssen ganzheitlich, durchgängig, redundanzfrei und widerspruchsfrei gespeichert und bereitgestellt werden.
- Die Komplexität der Informationen, begründet aus ihrer Menge, Vielfalt und Unsicherheit[320], muss auf ein anwender- und aufgabengerechtes Maß reduziert werden.
- Der Aufwand zur Speicherung und Bereitstellung der Informationen muss in einem angemessenen Verhältnis zum entstehenden Nutzen (Mehrwert für den Anwender, z.B. durch Zeitersparnis bei der Informationsbeschaffung) stehen.
- Art und Umfang der Anwendung der Modellierungstechnik müssen anwender- und aufgabengerecht beschrieben sein.
- Die Modellierungstechnik muss generalisierbar für ein möglichst breites Spektrum technischer Produkte einsetzbar sein.

[320] Die Unsicherheit von Informationen beruht auf unvollständigem oder fehlendem Wissen, abhängig vom Entwicklungsfortschritt, vgl. Albers et al. (2011c); Lohmeyer (2013)

Aus diesen übergeordneten Zielen ergeben sich konkrete Anforderungen an die fachdisziplinübergreifende Modellierungssprache und die Modellierungsmethode als Hauptbestandteile der Modellierungstechnik, die in den folgenden Kapiteln formuliert werden.

4.1.1 Zielsystem der Modellierungssprache

Das zentrale Werkzeug einer modellbasierten Entwicklungstechnik für multidisziplinäre Systeme ist eine Modellierungssprache. Da eine effiziente und nachhaltige Erzeugung umfangreicher Systemmodelle ein geeignetes Softwarewerkzeug erfordert, ergibt sich daraus die Notwendigkeit einer formalen Spezifikation des Metamodells der Sprache[321]. Grundlage eines formalen Metamodells sind *Entitäten* und *Relationen*, die gemeinsam die abstrakte Syntax („Wortschatz“ und „Grammatik“) und die statische Semantik („Bedeutung von Wörtern und Grammatik“) der Modellierungssprache definieren. *Entitäten* sind Modellartefakte[322] zur Beschreibung bestimmter Objekte wie beispielsweise Anforderungen, Funktionen oder Komponenten. *Relationen* bilden die Abhängigkeiten und Wechselwirkungen zwischen den Entitäten ab (z.B. „Komponente *erfüllt* Anforderung“ oder „Komponente *realisiert* Funktion“). Damit diese Artefakte des Metamodells strukturiert werden können, müssen diese in semantisch eng zusammenhänge Verbünde (Aspekte) wie Struktur und Verhalten gegliedert und daraus resultierend Partialmetamodelle abgeleitet werden. Diese sind nicht unabhängig voneinander, sondern verfügen wiederum über semantische Abhängigkeiten, die jedoch in Qualität und Quantität geringer sind als die Abhängigkeiten der Elemente innerhalb eines Partialmodells[323]. Die Modellbildung dieser Aspekte erfordert wiederum eine gemeinsame Sprache zur Bezeichnung der Modellartefakte der Partialmetamodelle. Neben dem Metamodell sind auch die grafische Notation der Entitäten und Relationen in Form von Symbolen sowie anwendergerechte Sichten auf das Modell zu definieren (die konkrete Syntax), die in

---

[321] Dieser Aussage liegt die Annahme zugrunde, dass Modelle für eine Softwareimplementierung formal definiert sein müssen, um die Anwendung mathematischer und logischer Operationen zu ermöglichen, beispielsweise für die Überprüfung der Konsistenz in der Modellierung. Eine formale Spezifikation umfasst sowohl die Beschreibung einer endlichen Menge an Wörtern und Konkatenationen (Verbindungen, Anm. d. Autors), die in der theoretischen Informatik als Kleenesche Hülle bezeichnet wird (vgl. Hopcroft und Ullmann (1994)) als auch eine Grammatik (vgl. Erk und Priese (2008))

[322] Unter einem Modellartefakt wird eine Entität oder eine Relation verstanden, also ein Element eines Metamodells, das zur Modellierung konkreter Systeme verwendet werden kann

[323] Eine Strukturierung dieser Elemente ließe sich bspw. in Form einer MDM (Multiple Domain Matrix, vgl. Lindemann et al. (2009)) vornehmen, indem eine Gruppierung sehr eng miteinander wechselwirkender Elemente vorgenommen werden kann. Hier haben sich jedoch in zahlreichen anderen Modellbildungsansätzen Partialmodelle etabliert, die auch in der SysML Anwendung finden (bspw. Anforderungs-, Struktur- und Verhaltensmodelle)

Form von Diagrammen realisiert werden. Daraus und aus den übergeordneten Anforderungen der Modellierungstechnik leitet sich das Zielsystem der Modellierungssprache mit folgenden Hauptanforderungen ab:

- Eine konkrete Information[324] muss durch ein eindeutiges Modellartefakt bzw. eine eindeutige Kombination mehrerer Modellartefakte repräsentiert werden.
- Die Bezeichnung der Modellartefakte muss für Entwickler aller Fachdisziplinen verständlich sein und auf einer fachdisziplinübergreifend eindeutigen Begriffsdefinition beruhen.
- Entitäten müssen hierarchisch strukturierbar und miteinander vernetzbar sein, um die Modellierung verschiedener Systemdetaillierungsgrade zu ermöglichen.
- Die Darstellungen von Modellartefakten müssen intuitiv verständlich und in ihrer syntaktischen und semantischen Bedeutung klar voneinander unterscheidbar sein.
- Partialmodelle müssen auf Basis der zu modellierenden Aspekte eines Systems strukturiert und miteinander vernetzt sein.
- Zur Bildung einer konsistenten, multidisziplinären Modellierungssprache müssen die Partialmodelle kohärent zueinander sein.
- Die Sichten auf ein Modell müssen eine Bündelung anwender- und rollenspezifisch relevanter Informationen in leicht verständlicher und übersichtlich dargestellter Form bilden.
- Wechselwirkungen und Abhängigkeiten zwischen Aspekten eines Systems müssen nachvollziehbar und nachverfolgbar modelliert und dargestellt werden.
- Die Modellierungssprache muss flexibel für verschiedene Anwendungsbereiche und Anwender einsetzbar sein, was neben einfacher Erweiterbarkeit und Änderbarkeit auch eine weitgehend freie Gestaltung der Reihenfolge der Modellierung voneinander nicht abhängiger bzw. nicht aufeinander aufbauender Aspekte umfasst.
- Eine technische Vernetzung mit anderen, in der Produktentwicklung eingesetzten Modellen muss möglich sein, um entstehende fachdisziplinübergreifende Modelle sinnvoll in eine durchgängige Entwicklungsumgebung integrieren und eine automatische Kommunikation und Synchronisierung der Modelle zu können.

[324] Eine Information beschreibt ein konkretes Element von Ziel- bzw. Objektsystem oder eine Relation innerhalb dieser bzw. zwischen diesen Systemen.

- Produktvarianten müssen modellierbar und gegenüberstellbar sein.
- Aspekte müssen produktlebenszyklusabhängig modellierbar sein.

Diese Anforderungen stellen die Basis für die Bildung einer objektorientierten, multidisziplinären Modellierungssprache dar. Für die Umsetzung in Form eines Softwaremodellierungswerkzeugs ergeben sich zahlreiche weitere Anforderungen bspw. bezüglich der Werkzeughaptik, Benutzeroberflächendesign, Automatisierung, Analysen, Versionierung, Wiederverwendbarkeit oder Rechteverwaltung, auf die hier nicht weiter eingegangen wird.

Neben den Anforderungen soll die Modellierungssprache für verschiedene Anwendungsfälle mit unterschiedlichem Modellierungszweck einsetzbar sein. Die in der vorliegenden Arbeit zu realisierenden Anwendungsfälle werden nachfolgend definiert:

Anwendungsfälle der Zielsystemmodellierung:

- Modellierung des Systemzwecks bzw. der Systemanwendungsfälle (i.S.v. geforderten Top-Level-Funktionen) mit interagierenden Nachbarsystemen
- Modellierung von Interessensvertreterzielen bezüglich der gewünschten Produktfunktionen
- Modellierung von Randbedingungen
- Modellierung von technischen Anforderungen mit Unterscheidbarkeit verschiedener Anforderungsarten (z.B. funktional und nichtfunktional)
- Modellierung des Übertragungspfades von Kundenzielen und Randbedingungen auf technische Anforderungen
- Modellierung der funktionsrelevanten Charakteristika[325] der Systemumgebung inklusive ihrer Schnittstellen mit dem zu entwickelnden Produkt

Anwendungsfälle der Objektsystemmodellierung:

- Qualitative Modellierung von Funktionen durch logische Abläufe (Bedingungen, Verzweigungen, Entscheidungen etc.) und Objektflüsse (Stoff, Energie und Informationen)
- Modellierung der Systemzustände und Zuordnung der darin bzw. in Zustandsübergängen auszuführenden Funktionen

[325] Unter Charakteristika werden Merkmale, Eigenschaften, Schnittstellen etc. zusammengefasst

- Modellierung dynamisch veränderlicher, funktionaler Systemstrukturen und deren funktionsrelevanter Charakteristika
- Modellierung dynamisch veränderlicher, physischer Systemstrukturen und deren Charakteristika
- Modellierung der Abhängigkeiten zwischen Funktionen und resultierenden, funktionalen Systemstrukturen
- Modellierung des Übertragungspfades von funktionalen auf physische Systemstrukturen
- Modellierung des zeitabhängigen Systemverhaltens unter Einbezug der auszuführenden Funktionen, der einzunehmenden Zustände und der agierenden Komponenten (z.B. für Testfallbeschreibung)
- Modellierung von einzuhaltenden Bedingungen und Abhängigkeiten zwischen Systemmerkmalen

Systemübergreifende Anwendungsfälle:

- Modellierung der erfüllten Anforderungen durch Funktionen oder Komponenten bzw. deren Merkmale oder ihrer aus dem Verhalten resultierenden Eigenschaften
- Zuordnung der auszuführenden Testfälle zur Validierung der Anforderungserfüllung durch die gewählte Systemlösung

Werkzeugauswahl zur Umsetzung:

Basierend auf den geforderten Anwendungsfällen und den zuvor definierten Anforderungen wurde eine Evaluation möglicher Modellierungssprachen auf Basis verfügbarer Literatur zu existierenden Ansätzen durchgeführt. REICHWEIN UND PAREDIS bieten beispielsweise einen Überblick über bestehende MBSE-Sprachen und stellen bereits zuvor durchgeführte Studien zu MBSE-Standards vor[326]. Sie stellten dabei jedoch fest, dass ein direkter Vergleich der Stärken und Schwächen von Modellierungssprachen bisher in keiner Studie durchgeführt wurde. Daher wurden die Stärken und Schwächen existierender Modellierungssprachen als Basis der Entscheidungsfindung bereits direkt im Anschluss an deren Vorstellung in den Unterkapiteln von Kapitel 2.6 beleuchtet. ALT untersuchte und bewertete verschiedene Sprachkandidaten hinsichtlich dedizierter, den hier vorliegenden Anforderungen weitgehend entsprechenden Kriterien und hat daraus die SysML als

[326] Reichwein und Paredis (2011)

die geeignetste Sprache ausgewählt[327]. So stellte er unter anderem fest, dass die SysML viele Konzepte anderer Sprachen in sich vereint.

Für diese Arbeit wurde die SysML neben diesen Kriterien insbesondere wegen ihres mittlerweile großen Verbreitungs- und Bekanntheitsgrades und aufgrund der zwei großen, weltweit agierenden und sie vorantreibenden Communities INCOSE und OMG ausgewählt. Aus diesem Grund ist die Sprache auch Teil einer großen Standardfamilie und bietet daher eine umfangreiche technische Kompatibilität zu anderen Sprachen und Transformationsstandards. Dadurch sind zahlreiche freie und kommerzielle Modellierungswerkzeuge am Markt verfügbar. Aufgrund ihrer objektorientierten Konzeption und expliziter Erweiterungsmechanismen sowie dem Profilmechanismus weist sie zudem eine sehr hohe Flexibilität auf. Darüber hinaus bietet die SysML für viele der genannten Anwendungsfälle und Anforderungen bereits geeignete Modellelemente und Diagramme an.

Da jedoch einige Anwendungsfälle und Anforderungen von der SysML noch nicht zufriedenstellend erfüllt werden, sind einige konzeptionelle Erweiterungen und Anpassungen notwendig. Die Konzepte für die Anpassungen basieren insbesondere auf den Forschungserkenntnissen aus der durchgängigen Modellierung von Zielsystemen sowie der integrierten Modellierung von Funktionen und ausführender Gestalt gemäß dem Contact & Channel – Ansatz (C&C²-A). Daraus leiten sich weitere, konkrete Anforderungen an die zu realisierenden Funktionalitäten der in der vorliegenden Arbeit vorgestellten Modellierungssprache auf Basis der SysML ab:

- Anforderungen müssen in verschiedene Arten differenziert werden, um spezifische Attribute und Relationen zuordnen zu können.
- Anforderungen, die von Interessensvertretern (z.B. Kunden) als Teil des Lastenhefts definiert werden, müssen von im Pflichtenheft durch Entwickler definierten, technischen Anforderungen eindeutig unterscheidbar sein.
- Entitäten des Zielsystems, wie Anwendungsfälle und verschiedene Anforderungsarten, müssen semantisch durch aussagekräftige Relationen verknüpfbar sein, um die Nachvollziehbarkeit der Entstehung von Anforderungen oder Elementen des Objektsystems herzustellen.
- Funktionen müssen vollständig beschrieben werden können, was neben Ein- und Ausgangsflüssen sowie deren Verarbeitung auch funktionsrelevante Merkmale und Eigenschaften umfasst. Diese können beispielsweise

[327] Vgl. Alt (2009), S. 71

Materialparameter, geometrische Abmessungen, Kosten, Verhaltensgrößen, Schnittstellen oder Normkonformitäten umfassen.

- Funktionen müssen Informations-, Stoff- und Energieflüsse oder Kombinationen daraus übertragen können. Der Typ des Objektflusses darf abhängig von der Art des Flusses nur eingeschränkt wählbar sein. Beispielsweise darf ein Informationsfluss keine Energie übertragen.
- Funktionale Strukturen müssen unabhängig von physischen Strukturen modellierbar sein.
- Die Transformation von funktionalen Strukturen zu physischen Strukturen muss durch eine explizite Zuordnung erfolgen. Die physischen Strukturen müssen dabei die Merkmale der funktionalen Strukturen übernehmen, ohne diese verändern zu können (da dies Funktionen ungewollt beeinflussen würde).
- Funktionale und physische Strukturen sind dynamisch veränderlich. Für Schnittstellen muss daher die Systemzustandsabhängigkeit bezüglich ihrer Existenz modellierbar sein.

Die Realisierung dieser Anforderungen an die Modellierungssprache zur Umsetzung der wissenschaftlichen Konzepte ist zentraler Bestandteil der vorliegenden Arbeit. Diese zugrundeliegenden Konzepte zur Erfüllung der Anforderungen und der Realisierung der Anwendungsfälle werden in Kapitel 4.3.2 und 4.3.3 beschrieben. Im Anschluss an die Vorstellung der Sprachbasis in Kapitel 5 wird in den Kapiteln 6 und 7 eine prototypische Implementierung der Konzepte in die Modellierungssprache an Evaluations- und Validierungsbeispielen im Detail vorgestellt und diskutiert.

4.1.2 Zielsystem der Methode zur multidisziplinären Systemmodellierung

Das zweite zentrale Element der Modellierungstechnik ist eine Methode zur Anwendung der Modellierungssprache. Auch hierfür werden weitere, konkretisierte Anforderungen formuliert:

- Ein übergeordnetes Vorgehensrahmenwerk muss eine Übersicht möglicher Handlungsschritte der Anwender beinhalten.
- Entsprechende Anwendungsformen der Modellierungssprache müssen auf Basis von Anwenderaufgaben und deren Rolle im Projektteam definiert werden.
- Ein flexibel auf den Modellierungszweck und die involvierten Anwender und ihre Aufgaben anpassbares Vorgehensmodell muss definiert werden.
- Das Vorgehensmodell zur Modellierung von Ziel- und Objektsystem muss sich an den Aktivitäten der Produktentstehung orientieren.

- Die Methode muss Modellierungsregeln definieren, die klare Vorgaben zur Modellierung bestimmter Informationen festlegen und damit verschiedene Modellierungswege (und –ergebnisse) für gleichartige Informationen vermeiden.

4.2 Handlungssystem der Entwicklung der Modellierungstechnik

Nachfolgend wird das Handlungssystem zur Entwicklung der angestrebten Modellierungstechnik und der Konkretisierung ihres zuvor in der finalen Form eingeführten Zielsystems vorgestellt. Das Handlungssystem gliedert sich in vier wesentliche Abschnitte, beginnend mit der Situationsanalyse und Problemeingrenzung durch Identifikation des Handlungsbedarfs aus verschiedenen Entwicklungsprojekten unter Beteiligung des Autors, Experteninterviews und Workshops[328], Literaturanalysen und Umfragen. Darauf folgen die Beschreibung der Vorgehensweise bei der Entwicklung der Sprachbasis sowie der Modellierungstechnik mit ihren zugrundeliegenden Evaluationsprojekten. Abschließend werden die Ergebnisse der Validierungsaktivitäten am Beispiel einiger weiterer Entwicklungsprojekte vorgestellt.

4.2.1 Identifikation des Handlungsbedarfs

Mobilitätssysteme sind komplexe, stark mit ihrer Umwelt vernetzte mechatronische Systeme. Daher lassen sich daran die Herausforderungen bei der interdisziplinären Zusammenarbeit von Entwicklung und Management sehr gut beobachten. Die umfangreichen, durch Interviews, Gespräche und in verschiedenen Forschungskooperationen mit Industriepartnern gesammelten Erfahrungen bilden die zentrale Motivation durch den dort identifizierten Handlungsbedarf. Eines dieser Projekte beschäftigte sich mit der Entwicklung von Methoden und Werkzeugen zur Herstellung einer Vergleichbarkeit von objektivierten Fahrbarkeitsuntersuchungen in verschiedenen Prüfumgebungen wie der reinen Simulation, dem Rollenprüfstand und der Teststrecke[329]. Unter anderem wurden dabei die Interaktion der Systeme Fahrzeug, Prüfstand und Umwelt hinsichtlich auftretender Schwingungsphänomene und ihrer Einflüsse auf die Messsignalqualität untersucht sowie Systeme und Methoden zur Reduktion der Einflüsse entwickelt[330]. In einem weiteren Kooperationsprojekt wurde eine energieeffiziente Fahrstrategie unter Nutzung eines

[328] Hierzu wurde insbesondere das Netzwerk der Gesellschaft für Systems Engineering e.V. (GfSE) genutzt, in dem zahlreiche Experten im Bereich der modellbasierten Produkt- und Systementwicklung aktiv sind.

[329] Albers et al. (2010c), Albers und Zingel (2010)

[330] Albers et al. (2009b), Albers et al. (2010d)

Accelerator-Force-Feedback-Pedals zur Fahreranleitung entwickelt. Darin wurde ein Steuergerätecode-Konzept erarbeitet, welches ein eingebettetes System innerhalb des Systemverbunds „Fahrzeug“, das wiederum in Interaktion mit seiner Umgebung (Verkehr, Strecke, Verkehrsschilder etc.) und dem Fahrer steht, darstellt[331]. Im Rahmen all dieser Projekte wurde die SysML als Beschreibungssprache zur Modellierung der untersuchten Systeme und ihrer Wechselwirkungen eingesetzt[332]. Darüber hinaus fand die SysML auch in weiteren am IPEK durchgeführten Projekten zur modellbasierten Beschreibung einer Vehicle-in-the-Loop Prüfumgebung zum manöverbasierten Testen Anwendung[333]. Der Entscheidung für den Einsatz der SysML lagen unter anderem die positiven Evaluationsergebnisse hinsichtlich der Nutzung als Beschreibungssprache für Fahrzeuge mit komplexen Architekturen am Beispiel von Hybridfahrzeugen zugrunde, an denen der Autor dieser Arbeit ebenfalls maßgeblich beteiligt war[334].

Trotz der Vorteile der SysML als derzeit am besten geeignete, verfügbare Sprache zur Modellierung multidisziplinärer Systeme wurde ein umfangreicher Handlungsbedarf zu deren Weiterentwicklung identifiziert. Insbesondere die unzureichende technische Eindeutigkeit bei der Modellierung physischer Systemstrukturen einschließlich ihrer Funktionen und Merkmale sowie ihrem resultierenden Verhalten und Eigenschaften sowie ein unzureichend transparenter Übertragungspfad zwischen Zielen und Lösungen wurden festgestellt. Darüber hinaus sind die Syntax und Semantik der Sprache mit ihren Elementen und Relationen sehr stark auf Softwaresysteme ausgelegt, was die Verständlichkeit für Anwender aus nicht IT-affinen Fachbereichen deutlich erschwert. Weiterhin wurde ein großer Weiterentwicklungsbedarf bei der grafischen Notation und den Sichten, also der Darstellung modellierter Informationen in Diagrammen sowie in der Handhabung der Modellierungswerkzeuge[335] identifiziert.

Aus diesen Erkenntnissen ergaben sich konkrete Handlungsmaßnahmen, die zunächst zu einer umfangreichen Literaturrecherche bezüglich Forschungsarbeiten zum Verständnis zentraler Begriffe wie „Funktion“ in der Produktentwicklung führten, die auch in fachdisziplinübergreifenden Modellen Verwendung finden. Hierzu wurden unter anderem von ERDEN ET AL., ECKERT ET AL. und ALINK bereits umfangreiche

[331] Albers et al. (2011e), Albers et al. (2012c)
[332] Zingel (2008)
[333] Albers und Düser (2009), Albers und Düser (2010)
[334] Zingel (2007), Albers und Düser (2007)
[335] Diese Feststellung ist herstellerübergreifend gültig. Keines der mittlerweile zahlreich am kommerziellen Markt sowie offen (open source) erhältlichen Softwarelösungen zur Modellierung von SysML-Modellen weist eine in allen Belangen zufriedenstellende Haptik und Funktionalität auf.

Studien durchgeführt, aus denen hervorgeht, dass das Verständnis des Begriffs Funktion und damit der Umgang mit funktionsbasierten Entwicklungsmethoden in der Produktentwicklung sehr heterogen ist[336]. Darüber hinaus wiesen Pilotprojekte und Studien zur Verwendung und Akzeptanz der SysML nach, dass die Sprache zwar ein großes Potential aufweist, aber noch umfangreiche Weiterentwicklungen für einen erfolgreichen Einsatz als Modellierungswerkzeug in der industriellen Produktentwicklung erforderlich sind, was die zuvor vorgestellten eigenen Erkenntnisse bestätigen[337]. Für ausführliche Informationen zu relevanten Studien sei hier auf ALBERS UND ZINGEL verwiesen[338].

Da die existierenden Studien sich konkret mit der Anwendung der SysML beschäftigen und keine Hinterfragung der Syntax der Sprache vornahmen, wurde eine Online-Umfrage im Rahmen der vorliegenden Forschungsarbeit durchgeführt, die zur Klärung folgender zwei Fragestellungen beitragen sollte:

- Wie ist das Verständnis der zentralen Begriffe „Funktion“, „Verhalten“ und „Wirkkette“ unter Systemingenieuren?
- In welchem Umfang wird die SysML eingesetzt, wie ist die Wahrnehmung des Mehrwerts für die tägliche Forschungs- bzw. Entwicklungsarbeit und wo gibt es Weiterentwicklungspotential?

Die Umfrage wurde bewusst nur an Personen aus Forschung und Industrie versandt, die sich bereits mit Systems Engineering beschäftigen, da Systemingenieure den Kern zur Homogenisierung von Begriffsverständnissen und damit der Bildung einer gemeinsamen Sprache darstellen sollten. Die freien Definitionen der Teilnehmer zeigten, dass das Begriffsverständnis auch unter Systemingenieuren sehr heterogen ist, was zu Problemen in der Kommunikation führen kann. Weiterhin wurde der SysML von den meisten Teilnehmern zwar bereits ein erkennbarer Mehrwert zugesprochen, jedoch auch Weiterentwicklungsbedarf aufgezeigt. Die Ergebnisse der Umfrage werden ausführlich in ALBERS UND ZINGEL[338] diskutiert, deren Erkenntnisse in die Basisdefinition einer gemeinsamen Sprache für die Modellbildung technischer Systeme einflossen.

4.2.2 Entwicklung der Sprachbasis

Bereits die ersten Projekte, in deren Rahmen die SysML als Modellierungssprache zum Einsatz kam, zeigten die Notwendigkeit einer einheitlichen Sprachbasis auf.

---

[336] Erden et al. (2008), Eckert et al. (2011), Alink (2010)
[337] Z.B. Cloutier und Bone (2010), Friedenthal (2009), Karban et al. (2009)
[338] Albers und Zingel (2013a)

Insbesondere die aus der Softwarewelt entstammenden Bezeichnungen der Sprachkonstrukte[339] der SysML bereiteten einigen Anwendern Schwierigkeiten, deren Bedeutung zu verstehen und sie zielführend zur Modellierung einzusetzen. Oftmals war nicht klar, welchen Aspekt oder welche Information man mit welchen Konstrukten modelliert. Hier macht die Spezifikation der SysML auch relativ wenige Vorgaben, vielmehr wird empfohlen, projektspezifische Vereinbarungen zur Modellierungstechnik zu treffen. Daher ist das primäre Ziel der Entwicklung einer Sprachbasis für den Kontext der Modellbildung technischer Systeme, semantische Brücken zwischen der Begriffswelt der Maschinenbauingenieure, der Elektrotechniker/Elektroniker und der Softwareentwickler zu bilden. Weiterhin werden semantische Zusammenhänge zwischen den Entitäten und Relationen der Modellierungssprache und den Aspekten der funktionszentrierten Modellierung definiert[340]. Nach der Identifikation zentraler und häufig heterogen aufgefasster Begriffe in Anwenderinterviews wurden grafische Darstellungen der Syntax und semantischer Zusammenhänge erarbeitet und eingeführt, beispielsweise im Rahmen einer sowohl bei Industriepartnern als auch intern durchgeführten SysML-Schulung[341]. Diese wurden anschließend in wissenschaftlichen Diskussionen sowie in Projekten evaluiert und verfeinert[342]. Eine Grafik mit semantischen Definitionen im Kontext des Funktionsbegriffs als Teil der Sprachbasis wurde in einer Online-Umfrage hinsichtlich der Nachvollziehbarkeit dargestellter Zusammenhänge evaluiert und anschließend auf Basis der gewonnen Erkenntnisse weiterentwickelt[343].

Anschließend wurden die Definitionen in Forschungsgesprächen vorgestellt und diskutiert. Auf deren Basis und unter Einbezug weiterer, existierender kontextspezifischer Metamodelle am IPEK[344] wurde in Workshops ein Entwurf einer semiformalen Ontologie erzeugt, die in einem weiteren Forschungsgespräch vorgestellt wurde. Die Erkenntnisse flossen in den heutigen Stand der Sprachbasis ein, die in Form der resultierenden Ontologiediagramme sowie weiteren grafischen Darstellungen und den resultierenden Begriffsdefinitionen in Kapitel 5 vorgestellt wird.

---

[339] Überbegriff für ein Element einer Modellierungssprache, das entweder eine Entität, eine Relation oder ein Diagramm sein kann

[340] Hierbei wurde eine Vorgehensweise in Anlehnung an die Heidelberger Strukturlegetechnik (vgl. Scheele und Groeben (1988)) gewählt, aus einzelnen Definitionselementen eine semantische Struktur zu erzeugen.

[341] Zingel (2012)*

[342] Albers et al. (2011d), Zingel et al. (2012)

[343] Albers und Zingel (2013a)

[344] Ein Beispiel ist das integrierte Metamodell von Prozess und Produkt, welche die Basis der laufenden Entwicklung einer Sharepoint-Plattform zur integrierten Modellierung von produkt- und prozessrelevanten Informationen ist, vgl. Albers et al. (2012a), Albers et al. (2012b)

4.2.3 Entwicklung der Modellierungstechnik

Die Sprachbasis bildet die Grundlage für den nächsten Schritt, der Formalisierung ihrer Elemente und Relationen für eine prototypische Werkzeugimplementierung auf Basis der SysML. Die Erweiterung wurde in mehreren Schritten durchgeführt. Zunächst wurden erste Konzepte des C&C²-A durch ein Profil in SysML implementiert[345]. Dieses Profil wurde in verschiedenen Entwicklungsprojekten eingesetzt und hinsichtlich seiner Anwendbarkeit auf verschiedene technische Systeme evaluiert.

In einem dieser Projekte wurde am Beispiel einer Ölversorgungsbohrung in einer Kurbelwelle untersucht, wie sich geforderte Kundenfunktionen auf realisierte Produktfunktionen in SysML abbilden lassen und wie sehr eine fachdisziplinübergreifende Modellierung ins technische Detail gehen kann und sollte. Dabei wurde insbesondere der Modellierungsaufwand dem Nutzen durch ein solches Modell gegenübergestellt. Sowohl die zu modellierenden Aspekte als auch Anwenderfeedbacks wurden in diesem Projekt durch Interviews mit Fachexperten des Unternehmens zum modellierten System erfasst[346].

Im Rahmen einer Unterbeauftragung aus dem EU-Projekt CESAR wurde die Modellierungstechnik zur Modellierung des Anwendungsfalls „Rekuperation“ für einen Hybridantriebsstrang eingesetzt und weiterentwickelt[347]. Die Zielsetzung dieser Unterbeauftragung war, eine Durchgängigkeit von Anforderungen auf die Produktarchitektur und Testfälle auf Basis eines Systemmodells in SysML herzustellen.

Im Rahmen eines ZIM-Kooperationsprojekts mit einem Softwareanbieter[348] wurden Werkzeugimplementierungsmöglichkeiten für den Contact & Channel - Ansatz erforscht. Hier wurde ein Prototyp entwickelt und evaluiert, mit dem das Elektrik/Elektronik-Architekturentwicklungswerkzeug des Unternehmens um die Modellierung grundlegender physischer Systemelemente und -merkmale auf deren Ebene der logischen Architektur erweitert wurde. Insbesondere in diesem Projekt konnten besonders viele Erkenntnisse bezüglich der Heterogenität der fachdisziplinspezifischen Sprachunterschiede gewonnen werden. Weiterhin wurde im

---

[345] Albers und Zingel (2011)

[346] Maletz und Zingel (2011)*

[347] Mehr Informationen zum EU-Projekt CESAR (Cost-efficient methods and processes for safety relevant embedded systems) finden sich auf http://www.cesarproject.eu

[348] Aquintos GmbH, Karlsruhe

ein Prototyp einer CAD-Schnittstelle für die E/E-Software des Unternehmens entwickelt[349].

In mehreren studentischen Arbeiten wurden Beiträge zur Entwicklung der Modellierungstechnik geleistet. Unter anderem wurden eine Basismethode zur Implementierung erster Konzepte des C&C²-A in einem SysML-Modellierungswerkzeug, zwei Beiträge zur methodischen Nutzung einer SysML-Simulink-Schnittstelle, eine prototypische Schnittstelle zwischen SysML und CAD sowie ein Prototyp eines Software-Plug-Ins mit Automatismen für die Modellierungstechnik entwickelt[350].

All diese Forschungs- und Entwicklungsprojekte trugen zur kontinuierlichen Weiterentwicklung und Verfeinerung der Modellierungstechnik und ihres Zielsystems bis zum in der vorliegenden Arbeit vorgestellten Stand bei.

4.2.4 Validierung der Modellierungstechnik

Anschließend an die Entwicklungs- und Evaluationsaktivitäten wurde die Modellierungstechnik anhand der Modellierung von Gesamtfahrzeugentwicklungsumgebungen validiert. Das Ziel des zugrundeliegenden Forschungsprojekts „Funktionale Lenkung mechatronischer Produkte (FLmP)“ ist die Bereitstellung einer funktionsbasierten Produktportfoliobeschreibung als Informationsbasis zur Entscheidungsfindung und –begründung im Portfoliomanagement. Hierzu wurde ein Methodenrahmenwerk erarbeitet, das eine modellbasierte Anwenderunterstützung bei der Entscheidungsfindung durch die Kopplung verschiedener Technologien bereitstellt[351].

Daran angelehnt wurden Konzepte für anwenderfreundliche Visualisierungen von SysML-Diagrammen entwickelt, um die Akzeptanz bei den angestrebten industriellen Anwendern zu erhöhen. Hierzu wurden die Konzepte am Beispiel verschiedener Sichten auf die Architektur einer Gesamtfahrzeugentwicklungsumgebung und deren Zielsystem in einem SysML-Modell umgesetzt und den Anwendern vorgestellt. Aufgrund des positiven Feedbacks wurden die Visualisierungskonzepte auch im Rahmen des Validierungsprojekts eingesetzt.

In einem weiteren Projekt wurden die Anwenderfunktionen des Mensch-Maschine-Interfaces eines Automatisierungssystems für Rollenprüfstände mit SysML beschrieben. Die exportierten Diagramme wurden für die Beschreibung zu

---

[349] Bull (2012)

[350] Winter (2010), Bopp (2010), Gloderer (2010), Kruppok (2012), Weigel (2012)

[351] Weiterführende Informationen finden sich in Denger et al. (2012), Denger et al. (2013) und in Kapitel 7

integrierender Funktionalitäten des existierenden Produktes in die zu entwickelnde Plattformlösung für Prüfstände jeder Art als Produktnachfolge des heutigen Automatisierungssystems herangezogen. Hier konnten wichtige Erkenntnisse zum Nutzen der SysML bei der funktionsbasierten Modellierung in Form direkten Feedbacks der Plattformentwickler gewonnen werden.

Weiterhin wurde die Modellierungsmethodik in einer weiteren Kooperation in einem anderen technischen Kontext eingesetzt und evaluiert[352]. Auch hier wurde der Nutzen erkannt und daraus weiterführende wissenschaftliche Untersuchungen angestoßen.

4.3 Objektsystem der Modellierungstechnik

Die Ergebnisse der Entwicklungs- und Validierungsaktivitäten sind eine Basisdefinition einer gemeinsamen Sprache für die Modellbildung technischer Systeme und eine darauf basierende Modellierungstechnik. Diese umfasst neben einer Sprache auf Basis der SysML auch entsprechende Modellierungsregeln und ein flexibles Vorgehensmodell. Eine Übersicht über die Ergebnisse wird im Folgenden gegeben, die Details werden in den anschließenden Hauptkapiteln ausführlich erläutert.

4.3.1 Basisdefinitionen einer gemeinsamen Sprache für die Modellbildung technischer Systeme

Die Basisdefinition umfasst Definitionen zentraler Begriffe im Kontext der Modellbildung technischer Systeme, unterstützt durch Grafiken zur Veranschaulichung semantischer Zusammenhänge zwischen den Begriffen. Außerdem wurde ein Entwurf einer semiformalen Ontologie aus Objekten und Relationen erarbeitet. Diese Ergebnisse stellen eine Grundlage für weiterführende Forschungsarbeiten dar und sind daher nicht als endgültig zu verstehen. Ziele der Basisdefinition im Kontext dieser Arbeit sind eine Verbesserung des Verständnisses für die in SysML modellierten Aspekte multidisziplinärer technischer Systeme sowie eine Erleichterung der Anwendung der SysML durch Personen mit IT-fremder Vorbildung. Die Sprachbasis wird in Kapitel 5 im Detail vorgestellt.

4.3.2 Durchgängige Zielsystemmodellierung

Die Modellierungstechnik fokussiert sich auf zwei Hauptaspekte bezüglich der Modellierung multidisziplinärer Systeme. Der erste Aspekt ist die durchgängige Zielsystemmodellierung mit dem Ziel einer besseren Differenzierung seiner Elemente

[352] Martini (2012)

und deren semantischer Zusammenhänge. Diese umfasst folgende Teilaspekte in Form einer Erweiterung bzw. Modifikation der SysML inklusive entsprechender Modellierungsregeln:

- Aktivitäten zur Verfeinerung von Anwendungsfällen als Teil des Zielsystems werden als *Target Function** (Ziel-Funktion) bezeichnet, um sie von den Funktionen der realisierten Systemarchitektur unterscheiden zu können.
- Test Cases (Testfälle) sind spezialisierte Anwendungsfälle
- Es wird zwischen fünf Anforderungsarten unterschieden: *Stakeholder Objective** (Interessensvertreterziel), *Boundary Condition** (Randbedingung), *Property RQ** (Nichtfunktionale Anforderung), *Continuous Function RQ** (Kontinuierlich-funktionale Anforderung) und *Discrete Function RQ** (Diskret-funktionale Anforderung)

Für diese drei Erweiterungen wurden Modellierungsregeln entwickelt, die nach heutigem Stand zwar noch nicht vollständig automatisiert durch ein Modellierungswerkzeug überprüft werden, jedoch ist diese Softwarefunktionalität technisch durch einen Werkzeuganbieter problemlos auf Basis des erweiterten Metamodells der SysML möglich. Die Detailergebnisse zum aktuellen Stand werden in Kapitel 6.2 vorgestellt.

4.3.3 Funktionsbasierte Objektsystemmodellierung

Der zweite zentrale Aspekt der Modellierungstechnik befasst sich mit der funktionsbasierten Objektsystemmodellierung auf Basis der Konzepte des Contact & Channel – Ansatzes (C&C²-A). Hier wurden folgende Erweiterungen vorgenommen:

- Zwei Arten von Funktionen: *Continuous Function** (Kontinuierliche Funktion) und *Discrete Function** (Diskrete Funktion)
- Drei Klassen von Objektflüssen (Material-, Energie- und Informationsfluss)
- Zwei Arten von Strukturen: die funktionale Struktur (entsprechend der Wirkstruktur bzw. Tragstruktur des C&C²-A) zur expliziten Modellierung der funktionsrelevanten Merkmale sowie die physische Struktur, die eine Konkretisierung der funktionalen Struktur darstellt.
- Verschiedene Klassen von Schnittstellen (u.a. für Material, Energie und Informationsübertragung sowie Mischformen)
- Mechanismen zur Modellierung dynamischer, in Abhängigkeit von Systemzuständen veränderlicher Strukturen
- Verschiedene Automatismen im Modellierungswerkzeug sowie Methoden zur Modellstrukturierung (Ordnerstruktur des Modellbaums)

Auch hierzu wurden entsprechende Modellierungsregeln entwickelt, die teilweise bereits in technischen Demonstratoren umgesetzt wurden. Sämtliche weitere Regeln lassen sich – wie auch im Fall der Zielsystemmodellierungsregeln – problemlos auf Basis der vorgenommenen Spracherweiterungen durch einen Werkzeuganbieter implementieren. Die hier genannten Aspekte werden in Kapitel 6.3 im Detail vorgestellt.

4.3.4 Vorgehensmodell für die Modellierungstechnik

Um eine zielkonforme und rollengerechte Anwendung der Modellierungssprache zu erlauben, wurde ein flexibles Vorgehensmodell entwickelt. Dieses ist im eigentlichen Sinne vielmehr ein Metamodell, um daraus konkrete Vorgehensmodelle für definierte Modellierungszwecke bzw. Anwendungsfälle der Modellierungstechnik abzuleiten. Das Ziel dieses Metamodells ist analog zum Konzept des iPeM die Möglichkeit zur Erzeugung von Referenzvorgehensmodellen als „Best Practice“-Vorgabe zukünftiger Entwicklungsprozesse sowie die Möglichkeit, ein im Vorhinein für das konkrete Projekt definiertes Implementierungsmodell mit dem tatsächlich im Applikationsmodell erfassten Ablauf im Rahmen der Projektnachbereitung abzugleichen. Das Vorgehensmodell wird in Kapitel 6.1 eingeführt.

4.3.5 Konzept zur Integration der Modellierungssprache in eine Entwicklungsumgebung

Neben Modellierungssprache und Vorgehensmodell wurde im Rahmen dieser Arbeit auch ein Konzept als Teil der Modellierungstechnik erarbeitet, wie die Modellierungssprache zukünftig in eine durchgängige, IT-gestützte Entwicklungsumgebung eingebettet werden kann. Erste Untersuchungen und Prototypen zur Modellkopplung von SysML und CAD sowie SysML und Simulink zeigen die technische Machbarkeit auf Basis bestehender Technologien auf. Darüber hinaus wird ein Konzept für ein Framework vorgestellt, das zur funktionsbasierten Lenkung mechatronischer Produkte in einem Produktportfoliosteuerungsprozess eingesetzt werden soll. Diese Ansätze und Konzepte werden in Kapitel 6.5 vorgestellt und diskutiert.

5 Basisdefinition einer gemeinsamen Sprache im Kontext der Modellbildung technischer Systeme

Grundlage jeder zielführenden Zusammenarbeit von Menschen ist, dass diese miteinander kommunizieren können, also die gleiche Sprache sprechen[353]. Dabei stellt weniger das reine Vokabular als vielmehr deren einheitliches und konsistentes Verständnis ihrer Bedeutung eine zentrale Herausforderung dar[354]. Dies gilt neben der „internen“ Kommunikation im Projektteam bzw. dem Entwicklungsunternehmen verstärkt für die „externe“ Kommunikation mit Kunden und Zulieferern. „Die Kommunikation mit den Kunden kann sich mitunter sehr schwierig gestalten, da sie oft eine andere „Sprache“ sprechen. Kunden denken primär in Anforderungen und Anwendungen (Problemsicht), Entwickler und Konstrukteure dahingegen oft in Komponenten und Spezifikationen (Lösungssicht).“[355]

Aus zahlreichen Ansätzen und Methoden zur strukturierten Beschreibung technischer Systeme und damit der Schaffung einer gemeinsamen Sprache zur Modellierung hat sich die Systems Modeling Language (SysML) als zunehmend verbreitetes Werkzeug der Wahl hervorgetan[356]. Sie ist wie alle Sprachen darauf angewiesen, dass Anwender ihre Syntax und Semantik verstehen, um zielführend eingesetzt werden zu können. Projekterfahrungen in der Anwendung der SysML in der Form, wie sie durch die OMG in der Spezifikation beschrieben und in heutiger Modellierungssoftware umgesetzt ist, haben jedoch gezeigt, dass insbesondere Anwender mit IT-fremder Vorbildung Schwierigkeiten haben, sowohl deren Konzepte für die Modellierung als auch die grafische Notation intuitiv in ihrer täglichen Arbeit einzusetzen. Dies trifft beispielsweise gehäuft auf Maschinenbauingenieure, Wirtschaftsingenieure und Manager mit betriebswirtschaftlicher Ausbildung zu.

Die in Kapitel 2.5 vorgestellten und in den oben genannten Fachrichtungen meist nicht gelehrten Grundkonzepte der Objektorientierung sind eine Grundvoraussetzung zum Verständnis der Logik einer objektorientierten Modellierungssprache wie der SysML, die nur durch Schulungen, Trainings und Coaching vermittelt werden können. Auf der anderen Seite gibt die Spezifikation der SysML eine Terminologie

---

[353] Ropohl (2009), S. 50 erläutert am Beispiel eines Computers, „[…] dass die geläufige Umgangssprache nicht dazu verhelfen kann, neue Technik angemessen zu verstehen; […]“

[354] Hubka und Eder (1992), S. 89 erkennen, dass häufig eine falsche Überzeugung herrscht, dass mehrere Begriffe „klar“ seien.

[355] Ponn und Lindemann (2011), S.12

[356] Vgl. Kapitel 2.6.2 sowie den Abschnitt „Werkzeugauswahl zur Umsetzung“ in Kapitel 4.1.1

vor, die aus der Softwaremodellierung mit der UML entstammt und aufgrund einiger von der eigenen Fachdisziplin abweichenden Definitionen gängiger Begriffe eine zusätzliche Hürde darstellt. Darüber hinaus sind die Diagrammarten und damit die Sichten auf das SysML-Modell nicht hinreichend auf die Bedürfnisse aller möglicherweise in Entwicklungsprojekten involvierten Menschen ausgelegt.

Diese beiden Defizite der heutigen SysML-Spezifikation können durch den mitgelieferten Profilmechanismus[357] teilweise überwunden werden. Leider ergibt sich hieraus ein Widerspruch, denn die Spezifikation erlaubt zwar, nur einen Teil der möglichen Artefakte zur Modellierung zu verwenden, jedoch ist ein Hinzufügen neuer Artefakte nicht spezifikationskonform und somit streng genommen keine SysML-Modellierung mehr. Allerdings ist genau diese Erweiterung in vielen Modellierungswerkzeugen möglich. Im Gegensatz dazu ist meist nicht vorgesehen, in der SysML verwendbare, aber nicht benötigte Artefakte einfach auszuschließen oder deren Anwendungsumfang einzuschränken.

Vor diesem Hintergrund wurde zunächst eine von der verwendeten Modellierungssprache SysML losgelöste Basisdefinition einer gemeinsamen Sprache im Kontext der Modellbildung technischer Systeme vorgenommen. Dabei soll die Sprache insbesondere ein einheitliches Basisverständnis zentraler Begriffe in der Breite, also über die Fachdisziplinen hinweg schaffen, dabei jedoch möglichst schlank bleiben und dementsprechend nicht jeden fachspezifischen Kontext in der Tiefe voll abdecken. Die wissenschaftliche Grundlage dieser Sprache liefern insbesondere der Contact & Channel – Ansatz nach ALBERS, die Systemtheorie der Technik nach ROPOHL sowie die Fachliteratur zur Entwicklung und Konstruktion technischer Systeme bspw. nach PONN UND LINDEMANN, PAHL ET AL, EHRLENSPIEL, HUBKA UND EDER, WEBER etc.. Dabei wurden Konzepte aus den teilweise geringfügig divergierenden Ansätzen in die resultierenden Begriffsdefinitionen mit der Terminologie der objektorientierten Modellierung integriert.

Zusammenfassend hat die gemeinsame Sprache das Ziel, die Basis für eine eindeutige und unmissverständliche fachdisziplinübergreifende Kommunikation zwischen den Akteuren im Produktentstehungsprozess zu bilden[358]. Dadurch wird

[357] Vgl. Kapitel 2.6.1.3

[358] Diese Zielsetzung formuliert auch Ropohl für seine Allgemeine Systemtheorie, auf der zahlreiche Konzepte der hier vorgestellten Sprache basieren: „Die Bedeutung der Allgemeinen Systemtheorie besteht darin, eine einheitliche formale Sprache für die geordnete Beschreibung verschiedenartiger Erfahrungsbereiche anzubieten und auf diese Weise deren Ähnlichkeiten, Überschneidungen und Verknüpfungen aufzudecken und zu präzisieren.“ (Ropohl (2009))

der ersten Forschungshypothese der Zielsetzung dieser Arbeit Rechnung getragen[359].

Zur Veranschaulichung werden Begriffsdefinitionen durch grafische Darstellungen und Beispiele inhärenter Bestandteile sowie semantischer Zusammenhänge mit anderen Begriffen der Sprachbasis unterstützt. Diese Methode der Sprachbeschreibung unter Einführung und Anwendung einer an die objektorientierte Modellierung angelehnte, konkrete Syntax unterstützt die anschließende Formalisierung durch ein Metamodell. Die Konstrukte der Sprache, bestehend aus Begriffen und semantischen Beziehungen zwischen den Begriffen, werden in den folgenden Unterkapiteln eingeführt.

5.1 Sprachkonstrukte der Zielsystembeschreibung

Nach ALBERS ist das Zielsystem wie folgt definiert: „Ein Zielsystem beinhaltet alle expliziten Ziele eines zu entwickelnden Produktes, einschließlich derer Abhängigkeiten und Randbedingungen, innerhalb eines definierten Interessenbereichs (d.h. innerhalb eines System-of-Interest) zu einem bestimmten Zeitpunkt“[360]. Daraus lassen sich für das (bei Projekten meist im Lasten- bzw. Pflichtenheft zu definierende) *Initiale Zielsystem*[361] folgende, festzulegende Aspekte ableiten:

- *Ziele der Interessensvertreter* (Stakeholder), beispielsweise des Kunden oder der Unternehmensstrategie.
- Einzuhaltende *Randbedingungen*, bspw. aus Infrastruktur, Gesetzgebung oder Normen und Standards.
- Den zu erfüllenden *Zweck* des zu entwickelnden Systems bzw. dessen geforderte *Anwendungsfälle* in der Interaktion mit seiner Systemumgebung. Hierzu zählen auch Anwendungsfälle, die speziell zur Absicherung der Funktionssicherheit und der Eigenschaftsabsicherung des Systems durchgeführt werden (z.B. Dauerbelastungstests, Crashtests etc.).
- Die für das zu entwickelnde System relevanten Eigenschaften, Merkmale und Schnittstellen der relevanten *Systemumgebung* (des System of Interest)[362].

---

[359] Siehe Kapitel 3.2

[360] Aus Albers et al. (2012d)

[361] Vgl. Lohmeyer (2013), Definition „Initiales Zielsystem“: Ein initiales Zielsystem beinhaltet die ersten grundlegenden Ziele. Es wird zu Beginn des Produktentstehungsprozesses in der Aktivität Projektierung erstellt. Das initiale Zielsystem entspricht meist dem Lasten-/Pflichtenheft bzw. der Produktspezifikation.

[362] Vgl. Albers et al. (2012d)

Die Aspekte eines initialen Zielsystems betrachten das zu entwickelnde System als „Black Box“[363], was bedeutet, dass sie sich immer auf das zu entwickelnde System als Ganzes beziehen. Diese Maßnahme vermeidet die Vorwegnahme bspw. der Existenz bestimmter technischer Teilsysteme und damit Einschränkungen des Lösungsraums. Die Aspekte des initialen Zielsystems werden im Verlauf des Produktentstehungsprozesses fortwährend erweitert und konkretisiert. Dabei ist die erste Aktivität der Produktentwickler, die teilweise vagen, möglicherweise widersprüchlichen oder technisch so nicht realisierbaren Vorgaben in Abstimmung mit den Interessensvertretern in ein Pflichtenheft und eine Anforderungsliste zu überführen, die kontinuierlich im Verlauf der Produktentstehung erweitert und aktualisiert werden[364]. Hierzu werden zwei weitere Aspekte des Zielsystems eingeführt:

- *Technische Anforderungen* beschreiben konkrete Anforderungen an Funktionen oder Strukturelemente. Sie werden vom Entwickler aus Aspekten des initialen Zielsystems oder aus Elementen der entstehenden Systemarchitektur abgeleitet. Letztere ergeben sich folglich aus technischen Teillösungen und sind dementsprechend mit den Elementen des Objektsystems per Ableitungsbeziehung verknüpft.
- *Ziel-Funktionen* beschreiben zu entwickelnde Aspekte technischer Funktionen aus Anwendersicht und detaillieren damit Anwendungsfälle weiter aus[365].

Bild 5-1 zeigt eine grafische Darstellung der Aspekte des Zielsystems, welche schematisch der Abgrenzung von Lastenheft, Pflichtenheft und Anforderungsliste nach KRUSE[366] untergeordnet wurden.

[363] Unter einer „Black Box“ wird in Anlehnung an Ropohl (1975), Ehrlenspiel (2009) etc. die Betrachtung eines Systems als Einheit, die über Schnittstellen mit anderen Konstrukten in Verbindung steht bzw. interagiert und deren innerer Aufbau zum momentanen Betrachtungszeitpunkt nicht bekannt ist.

[364] Lohmeyer (2013) beschreibt die Aspekte des Zielsystems, sowie den Prozess der Entwicklung und Konkretisierung des Zielsystems detailliert, worauf hier nicht näher eingegangen wird.

[365] Angelehnt an den teleologischen Funktionsbegriff nach Ropohl (2009) bzw. die „intentional Description“ nach Vermass (2010), vgl. Kapitel 2.1

[366] Vgl. Kruse (1996)

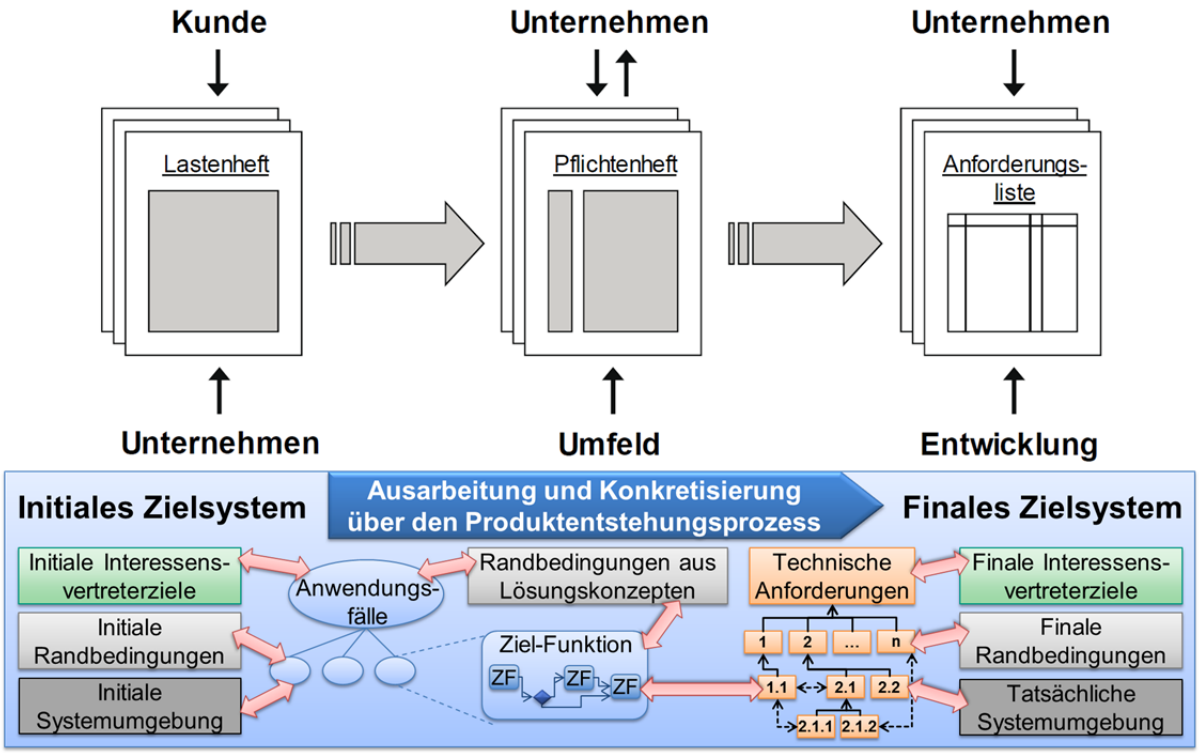


Bild 5-1: Abgrenzung von Lastenheft, Pflichtenheft und Anforderungsliste nach KRUSE[367] mit Einordnung der Aspekte des Zielsystems

Die initialen *Interessensvertreterziele*, die initialen *Randbedingungen* sowie die initiale *Systemumgebung* sind Bestandteile des initialen Zielsystems und folglich Ausgangspunkt jedes Produktentstehungsprozesses.

Interessensvertreterziele beschreiben hierbei aus Sicht des jeweiligen Interessensvertreters (z.B. des Kunden), was das zu entwickelnde System in welcher Form leisten soll, ohne konkret auf technische Lösungskonzepte oder zu realisierende Subsysteme einzugehen. Diese dürfen ausschließlich in Absprache mit dem betreffenden Interessensvertreter geändert werden.

Randbedingungen ergeben sich aus Gegebenheiten, die außerhalb des Designraums des zu entwickelnden Systems liegen und daher einzuhalten und nicht veränderbar bzw. beeinflussbar sind. Änderungen dürfen ausschließlich von der festlegenden Instanz, beispielsweise ein Normungsinstitut, der Gesetzgebung oder der für die Infrastruktur verantwortlichen Einrichtung vorgenommen und an das entwickelnde Unternehmen kommuniziert werden.

Die *Systemumgebung* grenzt den Designraum des zu entwickelnden Systems gegenüber seiner Umgebung ab und stellt über Konnektoren eine Verbindung zu den interagierenden bzw. wechselwirkenden Nachbarsystemen (der Infrastruktur des zu

[367] Kruse (1996)

entwickelnden Systems) her. Diese Konnektoren beinhalten basierend auf dem derzeit aktuellen Interessensbereich die für die Entwicklung relevanten Informationen der jeweiligen Nachbarsysteme[368]. Diese Verbindung hat den Vorteil, dass eventuell zu einem späteren Zeitpunkt und bedingt durch den Entwicklungsfortschritt relevant werdende Informationen zugänglich gemacht und in das Zielsystem, bspw. in Form von Randbedingungen oder Schnittstellen, aufgenommen werden können.

Anwendungsfälle repräsentieren den Zweck des Systems, da sie primär dessen Aufgaben beschreiben. Zusätzlich werden auch Anwendungsfälle in Form von *Testfällen* definiert, die speziell der Funktions- oder Eigenschaftsabsicherung von (Teil-)Systemen bzw. zweckgebundenen Modellen von Teilsystemen dienen (z.B. Extrembelastungsfälle wie Crashtests). Die Möglichkeit der Konkretisierung von Anwendungsfällen durch *Ziel-Funktionen* hilft, die oftmals komplexen Aufgaben detaillierter zu spezifizieren. Dieses Konzept ist an die Detaillierung von Anwendungsfällen durch Aktivitäten in SysML angelehnt, da es sich als sehr nützlich erwiesen hat, die unformalen, textuellen Beschreibungen von Anwendungsfällen bzw. Ziel-Funktionen auf diese Weise exakter abbilden zu können.

Im weiteren Verlauf der Produktentwicklung wird das Zielsystem durch aus ersten technischen Konzepten resultierende weitere Randbedingungen sowie insbesondere *Technische Anforderungen* konkretisiert, aktualisiert und ergänzt. Letztere leiten sich aus den übergeordneten Zielen oder Randbedingungen ab und können direkt an erste, im Objektsystem entstehende Lösungskonzepte gestellt werden. Hierzu werden drei verschiedene Arten technischer Anforderungen eingeführt, die später durch spezifische Elemente des Objektsystems erfüllt werden (siehe Bild 5-2).

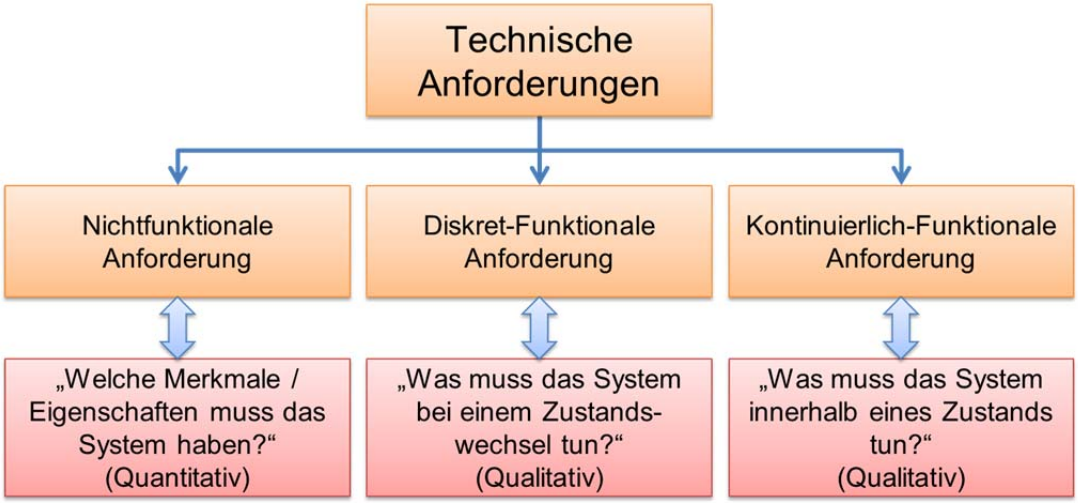


Bild 5-2: Technische Anforderungsarten

[368] Vgl. Albers et al. (2012d), Figure 4

Nichtfunktionale Anforderungen fordern Merkmale oder Eigenschaften der Systemstruktur wie beispielsweise Kosten, Zuliefererdaten, Materialparameter (z.B. Steifigkeit, Dichte, Leitfähigkeit), Schnittstellen (z.B. Flansch, Steckanschluss, Datenschnittstelle) oder geometrische Merkmale (z.B. Länge, Durchmesser, Lage). *Diskret-Funktionale Anforderungen* fordern Funktionen zur Zustandsänderung des Systems (z.B. Einschalten, Kupplung schließen, Datenpaket komprimieren). Diese haben einen definierten Anfangszustand und einen definierten Endzustand und damit eine endliche Dauer (diese sog. *Diskreten Funktionen* enden mit Abschluss des Zustandswechsels). Im Gegensatz dazu fordern *Kontinuierlich-Funktionale Anforderungen* Funktionen, die kontinuierlich innerhalb eines Zustandes ablaufen, bis sie von extern beendet werden (z.B. Signal messen, Drehmoment übertragen, Strom leiten)[369]. Aufgrund ihrer Eigenschaft, dass diskrete Funktionen die Handlungen eines (Teil-)Systems beim Zustandswechsel beschreiben und *Kontinuierliche Funktionen* innerhalb eines Zustandes ablaufen, werden sie durch unterschiedliche Elemente beschrieben[370]. Demnach können Diskret-Funktionale Anforderungen auch ausschließlich durch die entsprechenden Diskreten Funktionen erfüllt werden sowie analog Kontinuierlich-Funktionale Anforderungen durch Kontinuierliche Funktionen. Diese Unterscheidung ermöglicht folglich auch die Implementierung spezieller Regeln zur Handhabung dieser Elemente in einem Systemmodell.

Für alle Elemente des Zielsystems gilt, dass sie nicht statisch zu betrachten sind, sondern sich dynamisch mit dem Fortschritt der Produktentstehung ändern und daher zu pflegen sind. Unterschiede in deren Handhabung sind insbesondere in der Abstimmung im Falle einer Änderung zu sehen. So sind *Randbedingungen* nur passiv anzupassen, beispielsweise bei der Veröffentlichung einer neuen Norm oder veränderten Infrastrukturen des Systems. *Ziele von Interessensvertretern* und *Anwendungsfälle* dürfen nur in Absprache mit den betroffenen Interessensvertretern geändert werden[371]. Die daraus abgeleiteten technischen Anforderungen, die im Gegensatz zu den zuvor genannten Elementen das zu entwickelnde System nicht als „Black Box“ betrachten, können jedoch durch das Entwicklungsteam verändert werden. Hier sind lediglich projektteaminterne Abstimmungen erforderlich.

---

[369] Dieser Ansatz basiert auf der Unterscheidung von Funktionen in „Zustandsänderung“ und „Zustandserhaltung“ nach Ropohl (1975) und Ropohl (2009).

[370] Vgl. Kapitel 5.2.6

[371] Interessensvertreter formulieren teilweise auch technische Anforderungen. Hier ist darauf zu achten, dass diese auf ihre Richtigkeit und Existenz eines Bezugselements (z.B. Funktion oder Komponenten) geprüft werden müssen. Ein häufig in Projekten beobachteter Fehler ist, Technische Anforderungen aus Vorgängergenerationen mitzuführen, die bezüglich des aktuellen Lösungskonzepts längst obsolet geworden sind.

Eine zentrale Zielsetzung der Modellierung des Zielsystems ist die „Traceability“, also die Rückverfolgbarkeit von konkreten, technischen Anforderungen auf ihre Ursache mit dem Ziel der Argumentierbarkeit bzw. Begründbarkeit ihrer Existenz[372]. Diese kann neben definierten Zielen oder Randbedingungen (Traceability innerhalb des Zielsystems) auch in einer zuvor vollzogenen Auswahl eines technischen Lösungsprinzips (bspw. entwickelte Ist-Funktionen für definierte Ziel-Funktionen) liegen (Traceability zwischen Zielsystem und Objektsystem). Auch hier ist stets für eine durchgängige Nachvollziehbarkeit durch entsprechende Relationen zu sorgen. Die Umsetzung der hier vorgestellten Elemente und Relationen des Zielsystems werden in Kapitel 6.2, die Vernetzung von Ziel- und Objektsystem in Kapitel 6.4 vorgestellt.

5.2 Sprachkonstrukte der Objektsystembeschreibung

Das Objektsystem umfasst nach ALBERS: „alle Dokumente und Artefakte, die als Teillösungen während des Entstehungsprozesses anfallen“[373]. Wie die Modellbildungsansätze im Stand der Forschung übereinstimmend aufzeigen[374], sind die zentralen Aspekte zur Beschreibung eines technischen Systems insbesondere Funktionen, Wirkprinzip, Verhalten und Struktur (Gestalt). Damit werden automatisch zentrale Begriffe der Ingenieurssprache eingeführt, die auf die Syntax der Modellbildung semantisch abzubilden sind. Anschließend ist zu klären, wie die oben genannten Aspekte in einem formalen Modell strukturiert abgebildet werden können.

5.2.1 Technische Funktion

Der erste und gleichzeitig wichtigste Begriff ist die *Technische Funktion*, da diese den Zweck eines Systems begründet. Dabei gibt es verschiedene Sichten auf die Funktion eines technischen Systems: einerseits die beabsichtigten Funktionen eines Anwenders und andererseits die im System realisierten Funktionen[375]. Dies wurde in der Entwicklung der Sprache durch die Differenzierung in *Ziel-Funktion* und *Ist-Funktion* berücksichtigt. Die Einordnung der Ziel-Funktion als Element des Zielsystems wurde im vorangegangenen Kapitel bereits vorgestellt. Anschließend wird die Ist-Funktion als Element des Objektsystems weiter in *Diskrete Funktion* und *Kontinuierliche Funktion* differenziert. Zuvor erfolgt jedoch eine semantische Kopplung der Ingenieursbegriffswelt mit jener der funktionszentrierten Modellierung in SysML, unter anderem auch für Begriffe, die bereits in den Definitionen 4, 4.1 und

[372] Vgl. Muschik (2011), Kapitel 3.1.5.1 (The Role of Objectives for Decision Making)
[373] Albers und Braun (2011b)
[374] Vgl. Kapitel 2.9
[375] Vgl. Kapitel 2.1, eine ausführlichere Erläuterung folgt in Kapitel 5.2.5

4.2[375] verwendet wurden. Hierbei liegt der Fokus auf der Einführung von relevanten Elementen und Relationen zur umfassenden Beschreibung einer technischen Ist-Funktion.

Funktionen sollten die Leitgrößen bei der Entwicklung eines technischen Systems darstellen. Dies erfordert jedoch vor allem für Maschinenbauingenieure ein grundlegendes Umdenken weg von Bauteilen hin zu Funktionen. Bisher werden *Ursachen*, *Merkmale, Eigenschaften* und *Effekte* physischen Komponenten zugeordnet, die eine *Wirkung* hervorrufen. Diese Sprache wird nun in Bild 5-3 semantisch auf Begriffe der funktionszentrierten Modellbildung abgebildet.

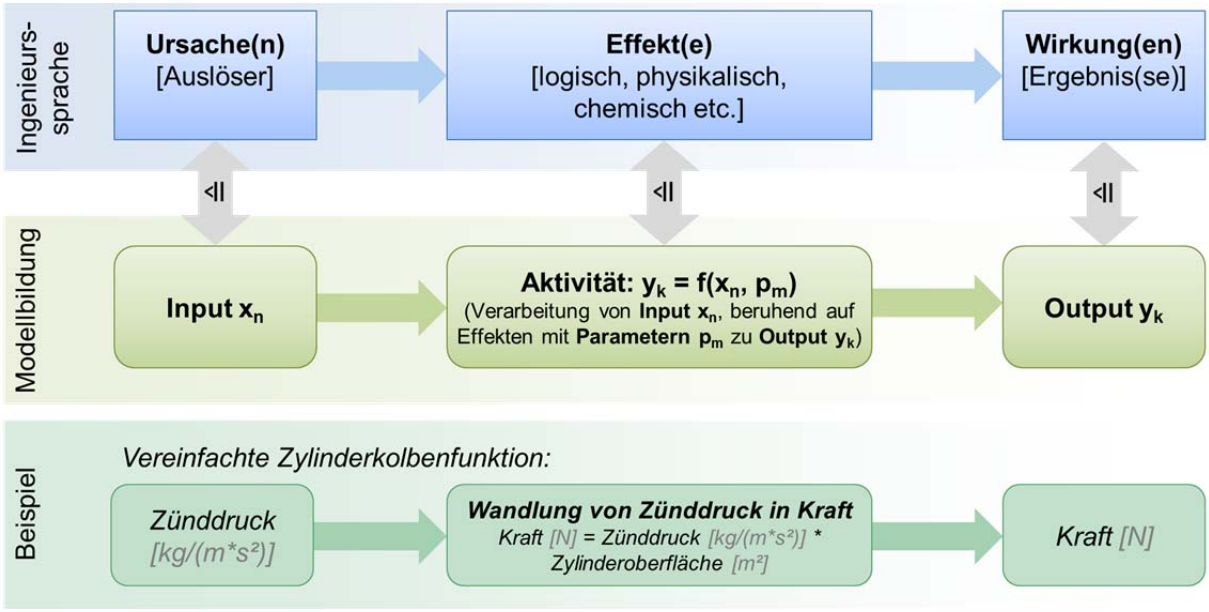


Bild 5-3: Semantische Kopplung des Funktionsverständnisses in der Ingenieurssprache auf die funktionszentrierte Modellbildung

Diese Darstellung beinhaltet drei zentrale Aussagen:

- Eingangsobjektflüsse von Stoff, Energie oder Information (Inputs) lösen die Ausführung einer Funktion aus, wobei diese Inputs sowohl diskrete Trigger (Ereignisse) als auch kontinuierliche Flüsse sein können.
- Die Inputs werden durch Aktivitäten zu Outputs verarbeitet. Dabei beruht diese Verarbeitung auf Effekten, die wiederum abhängig vom Detaillierungsgrad der Betrachtung durch Gleichungen oder Gleichungssysteme unter Nutzung von Parametern, die in SysML Merkmale und Eigenschaften repräsentieren, quantitativ beschrieben werden können.
- Die Ausgangsobjektflüsse von Stoff, Energie oder Information (Outputs) entsprechen der messbaren Wirkung einer Funktion, insofern diese an der Systemgrenze vorliegen. Outputs, die innerhalb des Designraums auftreten, können gleichzeitig wieder Inputs weiterer Funktionen sein.

Mit diesen Elementen kann eine Funktion grundsätzlich beschrieben werden. Jedoch ist diese Darstellung sehr stark abstrahiert und ein Bezug zur letztlich ausführenden Systemstruktur (bzw. Gestalt) wurde, abgesehen von den Parametern, bisher nicht hergestellt. Zur expliziten Modellierung des Funktions-Gestalt-Zusammenhangs als zwingende Voraussetzung für ein ganzheitliches Systemverständnis wurde der Contact & Channel-Ansatz (C&C²-A) zugrunde gelegt. Der besondere Vorteil des C&C²-A im vorliegenden Kontext einer besseren Unterstützung von Maschinenbauingenieuren ist, dass man ihn auf in der Konstruktion geläufige Visualisierungen technischer Systeme als Grundlage der Modellbildung anwenden kann[376].

5.2.2 Funktions-Gestalt-Zusammenhang in Wirknetzen und Wirkstrukturen

Die geeignete Ebene zur Kopplung einer rein qualitativen Funktionsbeschreibung im Sinne des EVA-Prinzips der Datenverarbeitung[377] mit der Gestalt bzw. der Baustruktur eines technischen Systems sind *Wirknetze* und die *Wirkstruktur* als Konzept des Contact & Channel-Ansatzes. Diese stellen den Zusammenhang zwischen der Verarbeitungsaktivität von Inputs zu Outputs durch Effekte mit den ausführenden Strukturelementen und deren funktionsrelevanten Merkmalen und Eigenschaften her. Bild 5-4 zeigt in abstrakter Form die Elemente der qualitativen Funktionsbeschreibung (Grün) sowie die funktionsrelevanten Strukturelemente[378] (Rot). In Grau sind die für die betrachtete Funktion nicht relevanten Strukturelemente dargestellt.

[376] Vgl. Kapitel 2.4.9

[377] EVA steht für Eingabe-Verarbeitung-Ausgabe (engl. IPO Input-Processing-Output), vgl. Lintermann et al. (2008), S. 16. Diese Ebene entspricht weitgehend der Funktionsebene des MKM nach Ponn und Lindemann (2011) bzw. der Funktionsbeschreibung nach Pahl et al. (2006).

[378] Dies kann entweder ein einzelnes Wirknetz sein oder auch – insofern es keine weiteren Wirknetze gibt, die Wirkstruktur der Funktion.

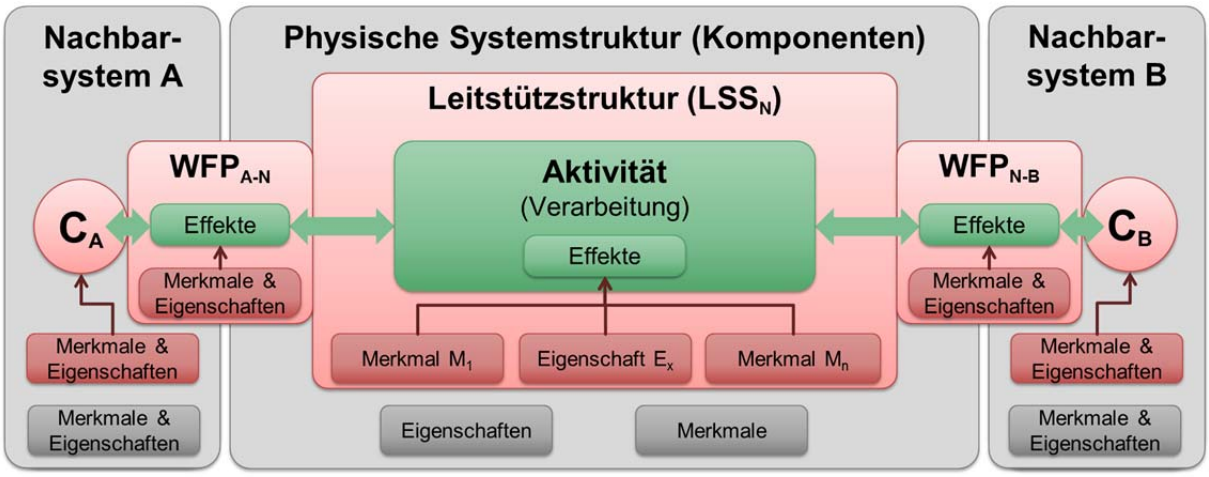


Bild 5-4: Elemente zur Modellierung von Funktion und Struktur

Die Grafik konkretisiert die semantischen Zusammenhänge der zuvor in Bild 5-3 eingeführten Begriffe. Hierzu werden Elemente des Contact & Channel-Ansatzes zur Modellbildung verwendet:

- Die beiden Connectoren beinhalten die funktionsrelevanten Eigenschaften und Merkmale der interagierenden Nachbarsysteme sowie eine Wirkfläche, die in der Grafik als Teil der Wirkflächenpaare implizit dargestellt ist. Darüber hinaus verfügen die Nachbarsysteme über weitere Merkmale und Eigenschaften, die für die betrachtete Funktion nicht relevant und sind.
- Die Wirkflächenpaare umfassen funktionsrelevante Merkmale und Eigenschaften der Oberflächen, über welche die Systeme interagieren (z.B. Form, Rauigkeit oder Oberflächenhärtung). Darüber hinaus finden im Wirkflächenpaar Effekte statt (z.B. Reibung), die funktionsbeeinflussend sind und die übertragenen Energie-, Stoff- oder Informationsflüsse beeinflussen (z.B. die übertragene Energie durch Wärmeverluste reduzieren).
- Die Leitstützstruktur als Teil der betrachteten physischen Systemstruktur beinhaltet funktionsrelevante Merkmale und Eigenschaften des Volumenelements (z.B. Dichte, Temperatur). Auch in ihr finden Effekte statt, die funktionsbeeinflussend sind und die übertragenen Energie-, Stoff- oder Informationsflüsse beeinflussen. Darüber hinaus verfügt die physische Systemstruktur über weitere Merkmale und Eigenschaften sowie Wirkflächen, welche die betrachtete Funktion nicht beeinflussen.

5.2.3 Effekte, Eigenschaften und Merkmale

In der Summe sind die naturwissenschaftlichen *Effekte* für die Art und Weise der Wandlung von Stoff, Energie und Information verantwortlich. Diese werden in Literatur zur systematischen Gestaltung technischer Produkte oft auch als Wirkprinzipien bezeichnet[379]. Bei der Systembeschreibung auf einem geringen Detaillierungsgrad können diese Effekte meist nur qualitativ beschrieben und bspw. durch Kennfelder oder Input-Output-Verhältnisse charakterisiert werden. Werden Systeme und deren Funktionen mit einem hohen Detaillierungsgrad beschrieben, können Effekte meist durch Gleichungen oder Gleichungssysteme beschrieben werden. Darin beschreiben Operatoren die Art der Wandlung, Konstanten (skalar, Vektoren oder Matrizen) sind funktionsrelevante Merkmale. Die zu ermittelnden Größen entsprechen den Outputs, die Variablen im Gleichungssystem den Inputs. Das Beispiel eines sehr einfachen Feder-Dämpfer-Systems in Form einer Zug-Druck-Belasteten Stabfeder soll dies veranschaulichen.

Grundlage: Feder-Dämpfer-Differentialgleichung

$$m \cdot \ddot{x}(t) + D \cdot \dot{x}(t) + C \cdot x(t) = 0 \tag{5.1}$$

Die Dämpferkonstante D ist ein funktionsrelevantes *Merkmal*, das direkt vom Entwickler durch die Materialauswahl der Stabfeder festgelegt werden kann. Abhängig von der Ausprägung der Anregung des Feder-Dämpfer-Systems durch die dynamische Kraft $F = m \cdot \ddot{x}(t)$ ist es einerseits möglich, dass eine rein elastische Schwingung vorliegt (D=0). Andererseits könnte jedoch im Fall eines viskosen Stabmaterials auch der Effekt einer inneren Reibung vorliegen, die eine Dämpfung in Abhängigkeit des Anregungskraftverlaufs hervorruft.

Die Federkonstante C hingegen ist eine funktionsrelevante *Eigenschaft*, da sie nicht direkt vom Entwickler festgelegt werden kann, sondern nur über Merkmale beeinflusst werden kann. Die Federkonstante einer Zug-Druck-belasteten Stabfeder wird durch Gleichung (5.2) berechnet:

$$C = \frac{A \cdot E}{l} \tag{5.2}$$

Darin stecken nun die Merkmale A (Stabfläche), E (Elastizitätsmodul des Stabmaterials) sowie l (Länge des Stabes). Unter Annahme eines konstanten Elastizitätsmoduls ist C ein Merkmal. Im Falle einer Temperaturabhängigkeit wäre es eine Eigenschaft, da C dann von einer Inputgröße abhängig wäre. Daraus leitet sich auch die Unterscheidung in direkt in der Synthese durch Entwickler bestimmbare

[379] Z.B. in Pahl et al. (2006), Ponn und Lindemann (2011)

Merkmale und in der Analyse ermittelbarer *Eigenschaften* ab, die an den CPM-Ansatz nach WEBER angelehnt ist[380]. Eigenschaften werden in der vorliegenden Arbeit nach zwei Unterkategorien unterschieden:

- *Verhaltenseigenschaften*, die direkt messbare Outputs (z.B. Beschleunigungen, Vibrationen, Geräusche) oder eine bspw. durch Gewichtung und/oder Skalenzuordnung in Relation gesetzte Menge von Outputs umfassen. Letztere sind Outputs, die hinsichtlich gesetzter Ziele durch eine menschliche Instanz interpretiert bzw. bewertet wurden (bspw. Skalenbewertungssysteme[381]).
- *Struktureigenschaften*, welche gemessene oder berechnete statische Strukturkennwerte umfassen (z.B. Gewicht, Kosten, Montagegerechtheit etc.).

Funktionen, Wirkstrukturen und resultierende physische Strukturen können mit diesen Elementen über verschiedene Abstraktionsstufen und Detaillierungsebenen hinweg beschrieben und einander zugeordnet werden[382]. Insbesondere ermöglicht die explizite Trennung von funktionsrelevanten Strukturmerkmalen/-eigenschaften und Wirkstrukturen gegenüber physischen Strukturen eine Maximierung des Lösungsraums und fördert damit die Entwicklung innovativer Systeme. Im Rahmen dieser Arbeit wurde ein Konzept erarbeitet, diese Ebenen bei der systematischen Modellierung mit SysML als Werkzeug zu nutzen[383]. Anschließend werden die logischen Zusammenhänge sowie die semantischen Begriffszusammenhänge vorgestellt, welche die Grundlage der Methodik auf werkzeugneutraler Ebene darstellt.

5.2.4 Testfälle und resultierendes Systemverhalten

Wird ein technisches System hinsichtlich seiner Funktion analysiert, erfolgt dies über *Testfälle*. Diese stellen konkrete Ausprägungen von entweder vorgesehenen Anwendungsfällen durch Anwender (die gewünschten Funktionen) oder auch Extremsituationen (z.B. Belastungstests wie Crash etc.) dar, die später der Verifikation der geforderten Systemattribute dienen (z.B. Performance, Haltbarkeit, Sicherheit etc.). Testfälle beschreiben, wie (Sub-)Systeme hinsichtlich ihrer Anforderungserfüllung validiert werden. Dies geschieht durch die Beaufschlagung von konkreten Inputs unter dedizierten Randbedingungen (z.B. Zuständen von Subsystemen), die eine konkrete Abfolge von Funktionen, also ein *Wirknetz*

---

[380] Weber (2013), vgl. auch Kapitel 2.4.7

[381] z.B. die ATZ-Skala, vgl. Aigner (1982) oder das Bewertungssystem zur Flugzeugbeurteilung nach Cooper und Harper (1969)

[382] Gemäß dem fraktalen Charakter des Contact&Channel–Ansatzes, vgl. Albers und Sadowski (2013)

[383] Die Implementierung der Konzepte in SysML wird in Kapitel 6 vorgestellt.

auslösen[384]. Diese äußert sich in einem über die Zeit beobachtbaren bzw. messbaren *Verhalten*[385] (siehe Bild 5-5).

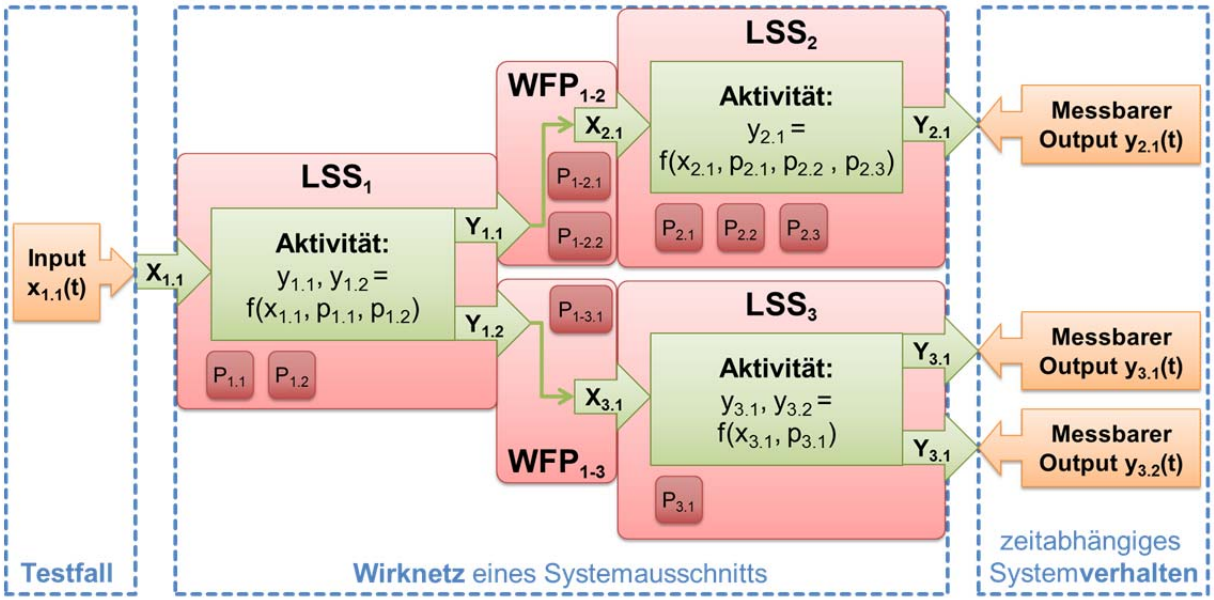


Bild 5-5: Semantische Zusammenhänge zwischen Testfall, Wirknetz und Verhalten

Die Grafik zeigt ein einfaches, generisches Beispiel für drei Leitstützstrukturen, die miteinander über WFP in einer Wirkstruktur verbunden sind. Die LSS besitzen Merkmale und Eigenschaften, die hier in Form von Parametern (p_n) aufgeführt sind, und üben darauf und auf den Inputs basierend Aktivitäten aus. Auch in den WFP kann eine Transformation geschehen, was hier jedoch nicht der Fall ist. Wird durch einen *Testfall* nun der Eingang von LSS_1 mit einem konkreten Eingangsobjektflussverlauf $x_{1.1}(t)$ beaufschlagt, so werden durch die in der Aktivität erzeugten Ausgangsobjektflussverläufe $y_{1.1}(t)$ und $y_{1.2}(t)$ zwei weitere Aktivitäten in den LSSen 2 und 3 ausgeführt und damit ein *Wirknetz* aus Funktionen gebildet. Deren Ausgänge wiederum liegen an der Systemgrenze, wo sich ein messbares System*verhalten* äußert.

Ein erstes, konkretes Beispiel unterschiedlicher Wirknetze in einem Hybridantriebsstrang in Abhängigkeit der Ausprägung von Inputs sowie der Subsystemzustände wurde bereits in Kapitel 2.4.9 mit Bild 2-16 gezeigt. Die Hauptfunktion „Antriebsenergie bereitstellen“ des Hybridantriebsstrangs muss unter dem Gesichtspunkt der Maximierung der Energieeffizienz differenzierter betrachtet werden. So lautet eine Nebenfunktion „Verzögerungsenergie rückgewinnen und

[384] Zur logischen Bedeutung eines Wirknetzes vgl. Kapitel 2.4.9

[385] Messbar sind die Wirkungen (Outputs) der einzelnen, in dem Wirknetz ausgelösten Funktionen, beispielsweise durch Schwingungen, Geräusche, Kräfte, Daten etc.

speichern“. Eine weitere Nebenfunktion ist „Verschiebung des Lastpunktes des Verbrennungsmotors in einen effizienteren Betriebsbereich“. Diese Funktionen werden jeweils durch bestimmte Zustände der Teilsysteme realisiert. Die Herausforderung ist jedoch, sowohl diese auch als Betriebszustand[386] bezeichneten Nebenfunktionen im Systemkontext, also unter möglichst allen denkbaren Interaktionen mit Nachbarsystemen (z.B. Streckensteigung, Gegenwind, Temperatur) sowie Zuständen der Subsysteme (z.B. Ladezustand der HV-Batterie, Motorbetriebsmodus) zu testen. Darüber hinaus sind auch die Zustandsübergänge hinsichtlich ihrer Zielkonformität (z.B. NVH-Verhalten oder Performanceaspekte wie das Ansprechverhalten) zu validieren. Dies kann aus Zeit- und Kostengründen nie vollständig für das gesamte Betriebsspektrum erfolgen, weshalb hier gezielt entweder charakteristische, repräsentative Betriebsbereiche oder Extremfälle (z.B. besonders hohe Belastung) getestet werden. Um möglichst realitätsnah[387] zu bleiben, kommt in der Automobilbranche immer häufiger manöverbasiertes Testen zu Einsatz[388]. Beispiele hierfür sind Einzelmanöver oder Testzyklen (auch Testsequenzen genannt), die sich aus einer Sequenz von Einzelmanövern zusammensetzen[389]. Einzelmanöver sind bspw. Volllastbeschleunigung, Tip-In[390] oder Schaltvorgänge. Testzyklen sind bspw. der Neue Europäische Fahrzyklus (NEFZ) oder die Federal Test Procedure (FTP75).

5.2.5 Ziel-Funktionen und Ist-Funktionen

Das Ziel bei der Verifikation und Validierung von Anforderungen durch Testfälle ist, so viele Wirknetze wie möglich mit so wenigen Testfällen wie möglich auszulösen, um das Systemverhalten sowie seine Merkmale und Eigenschaften in seiner gesamten Breite möglichst effizient zu validieren. Dabei soll einerseits die Ausprägung beabsichtigter (durch Entwickler vorgedachter) *Ziel-Funktionen* hinsichtlich ihrer Anforderungserfüllung verifiziert werden. Andererseits sollen aber insbesondere auch weitere, nicht beabsichtigte (nicht vorgedachte) *Ist-Funktionen* samt Ausprägung und Auswirkung auf die interagierenden Nachbarsysteme und

---

[386] Hier: Rekuperation und Lastpunktanhebung

[387] Damit ist gemeint, dass die Tests derart ablaufen sollten, wie sie auch in der späteren Nutzung des Systems auftreten. So ist es bspw. unwahrscheinlich, dass ein Verbrennungsmotor über Stunden am Stück unter Volllast läuft. Häufige Lastwechsel hingegen treten oft auf.

[388] Vgl. Beidl et al. (2013)

[389] Durch diese Sequenzen werden bewusst verschiedene Abfolgen von Systemzuständen und deren Übergänge angestoßen und validiert.

[390] Ein Beispiel für sehr umfangreiche Fahrbarkeitsuntersuchungen des positiven Lastwechselmanövers (Tip-In) wird in Albers et al. (2010c) vorgestellt. Dort lag der Fokus auf der Verbesserung der Reproduzierbarkeit und Vergleichbarkeit der getesteten Tip-In-Manöver in verschiedenen charakteristischen Betriebsbereichen.

deren Funktionen ermittelt und analysiert werden. Dieser Sachverhalt wird nachfolgend am Beispiel eines Zweigang-Getriebes erläutert (vgl. Bild 5-6).

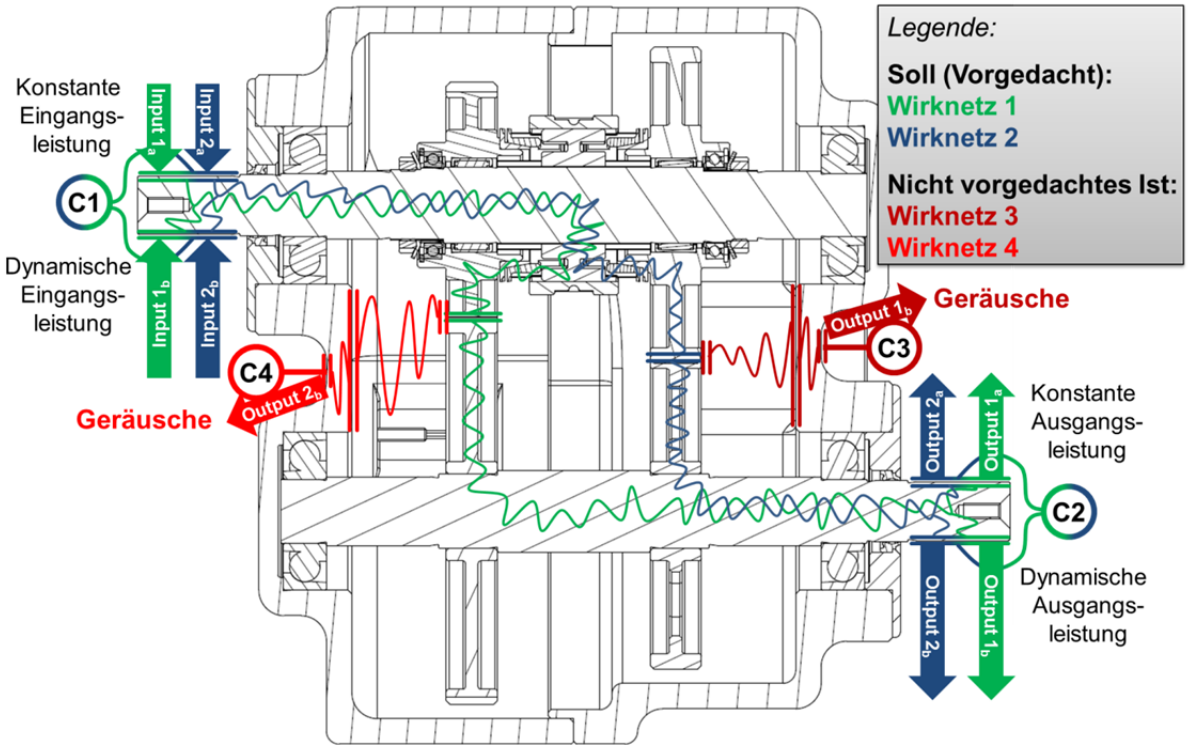


Bild 5-6: C&C²-Modell in Abhängigkeit der Inputs und des Systemzustandes auftretender vorgedachter und nicht vorgedachter Wirknetze eines Zweigang-Getriebes (CAD-Schnitt)

Beaufschlagt man an der Eingangswelle eine konstante Leistung, so kann dies einmal für jeden Gang geschehen, um die Wandlung von Drehmoment und Drehzahl am Ausgang und eventuelle Verluste zu validieren. Hierbei treten die beiden vorgedachten Wirknetze (grün für Gang 1, blau für Gang 2) auf. Wird jedoch die Ausprägung der Eingangsleistung geändert, also beispielsweise von konstant zu alternierend verändert, so können bei einem Getriebe aufgrund von Spiel oder geometrischen Toleranzen der Bauteile ungewollte (also nicht vorgedachte) weitere Wirknetze auftreten, welche mittel- und hochfrequente Geräusche übertragen, die bspw. an den Zahnflanken im Wirkkontakt[391] oder anderen Bauteilen entstehen und sich über die Luft und das Gehäuse ausbreiten (vereinfacht in Rot dargestellt). Beispiele möglicher, entstehender Geräusche in Abhängigkeit der beaufschlagten Eingangsleistung sind Heulen/Pfeifen (engl. whine), Klappern/Rasseln (engl. rattle) oder hochfrequente Lastwechselanschlaggeräusche (engl. clonk)[392].

[391] Ein Wirkkontakt ist der Teil eines Wirkflächenpaares, in dem aktuell die Wechselwirkungen stattfinden, vgl. Kapitel 2.4.9

[392] Vgl. Naunheimer et al. (2007), S. 266

Ein weiteres Ziel bei der Testfallbeschreibung ist, den Zusammenhang zwischen den zu testenden Funktionen, den getesteten Strukturelementen von virtuellen oder realen (Teil-)Systemmodellen und den zu verifizierenden Anforderungen herzustellen und zu beschreiben. Ein Aspekt ist hierbei die bereits mehrfach angesprochene Berücksichtigung von (Teil-)Systemzuständen und der in diesen Zuständen bzw. in Zustandsübergängen ausgeführten Funktionen. Diese Informationen sind zentraler Bestandteil der Systemverhaltensbeschreibung und zwingende Voraussetzung für eine zielführende Testfalldefinition. Daher werden die Begriffe *Zustand* und *Transition* (Zustandsübergang) als Teil der Basissprache im folgenden Abschnitt in ihrer Bedeutung charakterisiert und semantisch mit den anderen Aspekten der Beschreibung technischer Systeme verknüpft.

5.2.6 Zustände und Transitionen

Wie an den Beispielen bereits hergeleitet, prägen sich Wirknetze einerseits abhängig von Inputs und ihren Änderungen über die Zeit an der Systemgrenze aus, andererseits jedoch auch vom Systemzustand bzw. Zuständen der Subsysteme. Dabei werden Zustände neben konkreten Werten[393] der Merkmale und Eigenschaften von (Teil-)Systemen auch durch deren Wirkflächenpaare charakterisiert[394]. Da Technische Systeme dynamisch veränderliche Strukturen aufweisen können, sind nicht in jedem Systemzustand alle für eine Funktion erforderlichen WFP vorhanden und das Wirknetz nimmt eine andere Form an. Beispielsweise existiert das Wirkflächenpaar zwischen zwei Kupplungsplatten nur im Kupplungszustand „geschlossen“, nicht jedoch im Zustand „offen“. Folglich kann sich ein Wirknetz bei der Übertragung von Drehmoment und Drehzahl nur dann ausbilden, wenn die Kupplung im Zustand „geschlossen“ ist. Eine derartige Strukturdynamik gibt es auch in der Softwarewelt. Beispielsweise nutzt eine Messdatenkonditionierungssoftware abhängig von der Ausprägung der Rohdaten sowie den Einstellungen des Anwenders verschiedene Algorithmen zur Datenaufbereitung (z.B. Glättung, Beseitigung von Ausreißern etc.). Auch kann eine Software eventuell nicht ausgeführt werden, wenn ein Port (eine Softwareschnittstelle und damit ein Softwarewirkflächenpaar) nicht verfügbar ist, weil er bspw. von einer anderen Software genutzt wird. Diese virtuellen Wirkflächenpaare können analog wie reale Wirkflächenpaare behandelt werden.

ROPOHL klassiert Ist-Funktionen als Teil des Objektsystems in 5 Arten. Die wichtigste Unterscheidung nach Zustands-Attributen trifft er bezüglich ihrer Ausführung

---

[393] Abweichend zu Ponn und Lindemann (2011) wird hier nicht die Unterscheidung zwischen Merkmal und Eigenschaft darin getroffen, ob ein konkreter Wert vorliegt oder nicht.

[394] Vgl. Matthiesen (2002)

innerhalb bestimmter (Teil-)Systemzustände bzw. zur Durchführung von Zustandsübergängen (vgl. Bild 5-7)[395].

	Output-Attribute Y, RY, TY	Zustands-Attribut Z
Input-Attribute X, RX, TX	WANDLUNG $Y \neq X$ (qualitativ/quantitativ)	ZUSTANDSVERÄNDERUNG $X \neq const$ $Z \neq const$
	TRANSPORT $Y = X$ $RY \neq RX$ $TY \neq TX$	ZUSTANDSERHALTUNG $X \neq const$ $Z = const$
	SPEICHERUNG $Y = X$ $RY = RX$ $TY \neq TX$	X Inputs Y Outputs Z Zustände RX, RY Raumkoordinaten TX, TY Zeitkoordinaten const unverändert

Bild 5-7: Funktionsklassen der Sachsysteme nach ROPOHL[396]

Diese Klassierung wurde auch in der Basissprache aufgrund ihrer klar unterscheidbaren, spezifischen Charakteristika übernommen:

- *Diskrete Funktion*, die explizit die Änderung eines (Teil-)Systemzustands ausführt, sobald eine Änderung des Inputs eintritt sowie
- *Kontinuierliche Funktion*, die innerhalb eines (Teil-)Systemzustands abläuft und diesen gegen störende Inputs erhält

Weiterhin ist ein Vorteil dieser Klassierung, dass diese auch in objektorientierten Modellierungssprachen wie der SysML bereits spezifisch modelliert werden kann (siehe abstraktes Modellierungsbeispiel im Zustandsdiagramm in Bild 5-8).

[395] Vgl. Ropohl (1975), S. 40ff bzw. Ropohl (2009), S. 123ff
[396] Vgl. Ropohl (1975), S. 41 bzw. Ropohl (2009), S. 125

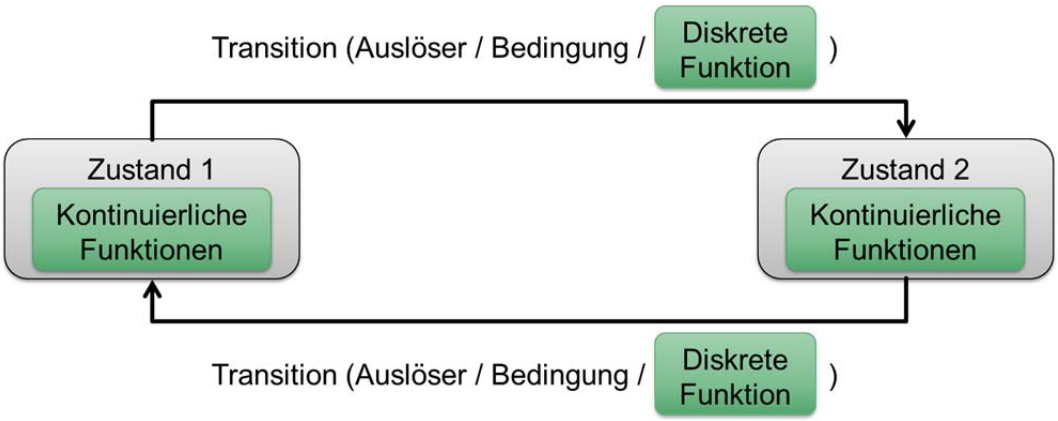


Bild 5-8: Systemzustände und Funktionsarten

Der Zustandswechsel eines technischen (Teil-)Systems heißt *Transition* und wird durch eine spezifische Änderung des Inputs ausgelöst. *Diskrete Funktionen* zur Durchführung des Zustandswechsels werden beispielsweise durch ein diskretes Ereignis oder in Form eines diskreten Inputs[397] (z.B. „Schalthebel wird betätigt“ oder „Schaltfläche ‚Speichern‘ wird betätigt“) ausgelöst. Optional bzw. alternativ müssen definierte Bedingungen zur Auslösung erfüllt sein (z.B. „System ist eingeschaltet“ oder „Eingangsdrehmoment = 0 Nm“). Beides lässt sich ebenfalls in UML/SysML in Zustandsdiagrammen modellieren, wie auch in Bild 5-8 zu sehen ist.

Auf eine weitere, beispielsweise nach Raum- oder Zeitattributen basierende Übergliederung wird hier verzichtet, um den Umfang der gewollt schlanken Basissprache nicht zu sprengen. Zusammenfassend werden die zentralen Begriffe der Basissprache im Kontext der Modellbildung technischer Systeme entsprechend der zuvor beschriebenen Merkmale definiert.

5.2.7 Begriffsdefinitionen

Die nachfolgenden Definitionen basieren auf der Systemtheorie der Technik[398] sowie dem Contact & Channel-Ansatz (C&C²-A)[399] und wurden durch ALBERS und sein Forscherteam des IPEK ausgearbeitet:

[397] Ein solcher zeitdiskreter Input wird insbesondere in der IT häufig als Trigger bezeichnet
[398] Vgl. Kapitel 2.1, insbesondere die bereits dort eingeführten Begriffsdefinitionen
[399] Vgl. Kapitel 2.4.9, insbesondere die bereits dort eingeführten Begriffsdefinitionen

Definition 4.2.1: Kontinuierliche Funktion

Die *Kontinuierliche Funktion* ist eine Ist-Funktion, die innerhalb eines Systemzustandes ausgeführt wird. Sie dient der Zustandserhaltung und muss durch bestimmte Ausprägungen von Inputs beendet werden.

Definition 4.2.2: Diskrete Funktion

Die *Diskrete Funktion* ist eine Ist-Funktion, die in der Transition zwischen zwei Systemzuständen ausgeführt wird. Sie wird durch eine charakteristische Änderung von Inputs ausgelöst und endet mit Abschluss der Transition der betreffenden Zustandsattribute (Merkmale, Eigenschaften, WFP).

Definition 5: Zustand

Ein *Zustand* repräsentiert eine konkrete Ausprägung der Struktur eines technischen Systems oder eines Subsystems mit einer bestimmten Menge von Werte- oder Wertebereichskombinationen ihrer Attribute.

Definition 6: Transition

Eine *Transition* beschreibt einen Zustandswechsel und verändert die Anordnung von WFP und/oder die Eigenschaften von LSS in der Struktur eines technischen Systems oder Teilsystems.

Definition 7: Ursache

Eine *Ursache* löst Funktionen in Form charakteristischer Inputs aus.

Definition 8: Effekt

Ein *Effekt* beschreibt eine naturwissenschaftliche Gesetzmäßigkeit zur Bestimmung des Verhältnisses zwischen Inputs und Outputs in WFP und LSS unter Einbezug ihrer relevanten Merkmale und Eigenschaften.

Definition 9: Wirkung

Eine *Wirkung* beschreibt die aus einer Funktion resultierenden Outputs, die innerhalb eines technischen Systems gleichzeitig Ursachen (Inputs) mit Auslösung weiterer Funktionen sein können. An der Systemgrenze sind Wirkungen messbar.

Definition 9: Wechselwirkung

Eine *Wechselwirkung* ist eine bidirektionale Interaktion zweier Funktionen über ein WFP, die gleichzeitig Ursache (Input) und Wirkung (Output) beider beteiligter Funktionen darstellt.

Definition 10: Wirknetz

Ein *Wirknetz* beschreibt die Propagation (Wandlung) der beaufschlagten Inputs durch eine Wirkstruktur im Moment der Ausführung technischer Funktionen unter definierten, konkreten Konditionen (Inputverläufe über der Zeit sowie Zustandsgrößen der beteiligten LSSen) bis hin zu den resultierenden Outputs.

Definition 11: Wirkstruktur

Die *Wirkstruktur* beschreibt das Gefüge (Aufbau und Anordnung) der LSS, WFP und C eines technischen Systems sowie deren funktionsrelevanten Merkmale und Eigenschaften. Sie ist die Summe aller möglichen Wirknetze einer Funktion.

Definition 12: Verhalten

Das *Verhalten* ist die Reaktion eines Technischen Systems auf konkrete Inputs, die in Wirknetzen gewandelt werden. Die resultierenden Outputs sind an den Systemgrenzen als Wirkungen messbar und ermöglichen eine Interpretation (Validierung) des Systemverhaltens.

Definition 13: Physische Struktur (Baustruktur)

Die *Physische Struktur* (Baustruktur) enthält die Charakteristika der Wirkstruktur und konkretisiert die LSS und WFP durch die Ausarbeitung zu resultierenden physischen Komponenten und Schnittstellen. Diese umfassen darüber hinaus Reststrukturen und nicht funktionsrelevante Merkmale und Eigenschaften.

Definition 14: Gestalt

Die *Gestalt* ist die quantitative Beschreibung der Summe aller Merkmale und Struktureigenschaften der physischen Struktur eines technischen Systems.

Definition 15: Merkmal

Ein *Merkmal* ist ein Attribut eines Strukturelements eines technischen Systems (z.B. Datenformat, Interfaceart, Form, Lage, Stoff) und wird durch den Entwickler festgelegt.

Definition 16: Eigenschaft

Eine *Eigenschaft* ist ein nicht unmittelbar vom Entwickler beeinflussbares Attribut eines Technischen Systems.

Definition 16.1: Verhaltenseigenschaften

Verhaltenseigenschaften charakterisieren direkt gemessene Outputs eines Wirknetzes (z.B. NVH-Größen, Leistung) oder durch Menschen hinsichtlich der Zielerreichung interpretierte Outputs (z.B. Umwelteigenschaften, Sicherheit) eines technischen Systems.

Definition 16.2: Struktureigenschaften

Struktureigenschaften charakterisieren gemessene oder berechnete statische Strukturkennwerte (z.B. Gewicht, Kosten, Montagegerechtheit, Ästhetik).

Definition 17: Anwendungsfall

Ein *Anwendungsfall* beschreibt zeitlich zusammenhängende und zielgerichtete logische Abfolge von Ziel-Funktionen eines technischen Systems.

Definition 18: Testfall

Ein *Testfall* beschreibt eine konkrete Ausprägung eines Anwendungsfalls. Er beschreibt eine Sequenz von Beaufschlagungen konkreter Inputverläufe über der Zeit, wodurch die Ausführung von Ist-Funktionen verursacht wird. Dadurch bilden sich Wirknetze aus. Testfälle validieren den Zweck eines technischen Systems durch Messung und Interpretation der Outputs von Ist-Funktionen bzw. verifizieren Merkmale seiner Struktur.

Die hier vorgestellte Sprachbasis ist das Resultat der bisher durchgeführten Forschungsarbeiten. Eine Forschungsaktivität darunter war die Durchführung einer Online-Umfrage zum Begriffsverständnis unter Systemingenieuren mit einer anschließenden Evaluation einer Vorläuferversion von Bild 5-3. Die Ergebnisse werden im folgenden Kapitel vorgestellt.

5.3 Evaluation der Sprachbasis

Erste Konzepte der Darstellung semantischer Zusammenhänge zentraler Begriffe zur Bildung einer Sprachbasis basierten auf Projekterfahrungen unter Beteiligung des Autors sowie den Erkenntnissen aus der Forschung am Contact & Channel – Ansatz und Literaturrecherchen[400]. Neben dem Einbezug von direkten Feedbacks im Zuge der Projektarbeiten unter Anwendung der Modellierungstechnik wurde im Rahmen einer Online-Umfrage das Begriffsverständnis von „Funktion“, „Verhalten“ und „Wirkkette“[401] abgefragt, indem die Teilnehmer um ihre eigene Definition der Begriffe gebeten wurden. Erst im Anschluss daran wurde eine Grafik präsentiert, in der semantische Zusammenhänge von in der Ingenieurssprache gängigen Begriffen mit Aspekten der Modellbildung einer Funktion dargestellt waren, und um die Bewertung der Nachvollziehbarkeit einzelner semantischer Aussagen gebeten.

[400] vgl. Albers und Zingel (2011), Albers et al. (2011d), Zingel et al. (2012)

[401] Dies war die damalige Bezeichnung von Wirknetz. Der Begriff wurde durch ALBERS (vgl. Albers und Sadowski (2013)) durch weiterführende Forschung unter Beteiligung des Autors dieser Arbeit neu geprägt.

Die insgesamt 50 Datensätze eines ausgewählten, Systems Engineering-affinen deutschen Teilnehmerkreises aus Industrie (27 Datensätze) und Forschung (23 Datensätze) setzten sich aus Angehörigen verschiedenster Fachdisziplinen zusammen, wie in Bild 5-9 dargestellt ist. Hierbei war eine Mehrfachauswahl zur Berücksichtigung multidisziplinärer Ausbildungen möglich.

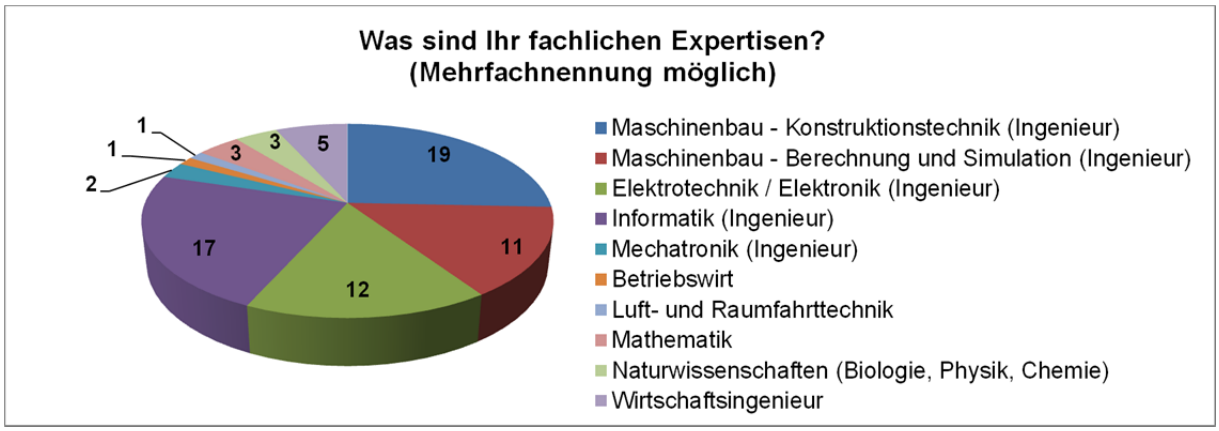


Bild 5-9: Fachliche Expertisen der Umfrageteilnehmer

Zur Beantwortung der erfragten eigenen Begriffsdefinitionen von „Funktion“, „Verhalten“ und „Wirkkette“ hatten die Teilnehmer die Möglichkeit, ihre Definitionen als Freitext einzugeben. Sie konnten darüber hinaus ihre Definition in mehrere Aspekte untergliedern bzw. mehrere Definitionen in eigenen Textfeldern angeben. Das Ergebnis waren 78 Aussagen zur Definition von „Funktion“, 70 Aussagen zu „Verhalten“ und 58 Aussagen zu „Wirkkette“. Die Aussagen zeigten zwar einige Übereinstimmungen, jedoch konnte insgesamt ein hochgradig heterogenes Verständnis aller drei Begriffe identifiziert werden, was sich auch mit den Erkenntnissen von anderen Studien deckt[402]. Die Aussagen, ihre Gemeinsamkeiten und Widersprüche werden ausführlich durch ALBERS UND ZINGEL diskutiert[403].

Bild 5-10 zeigt die in der Umfrage gezeigte Grafik, auf deren Basis die Aussagen zu bewerten waren.

[402] Z.B. Eckert et al. (2011)
[403] Albers und Zingel (2013a)

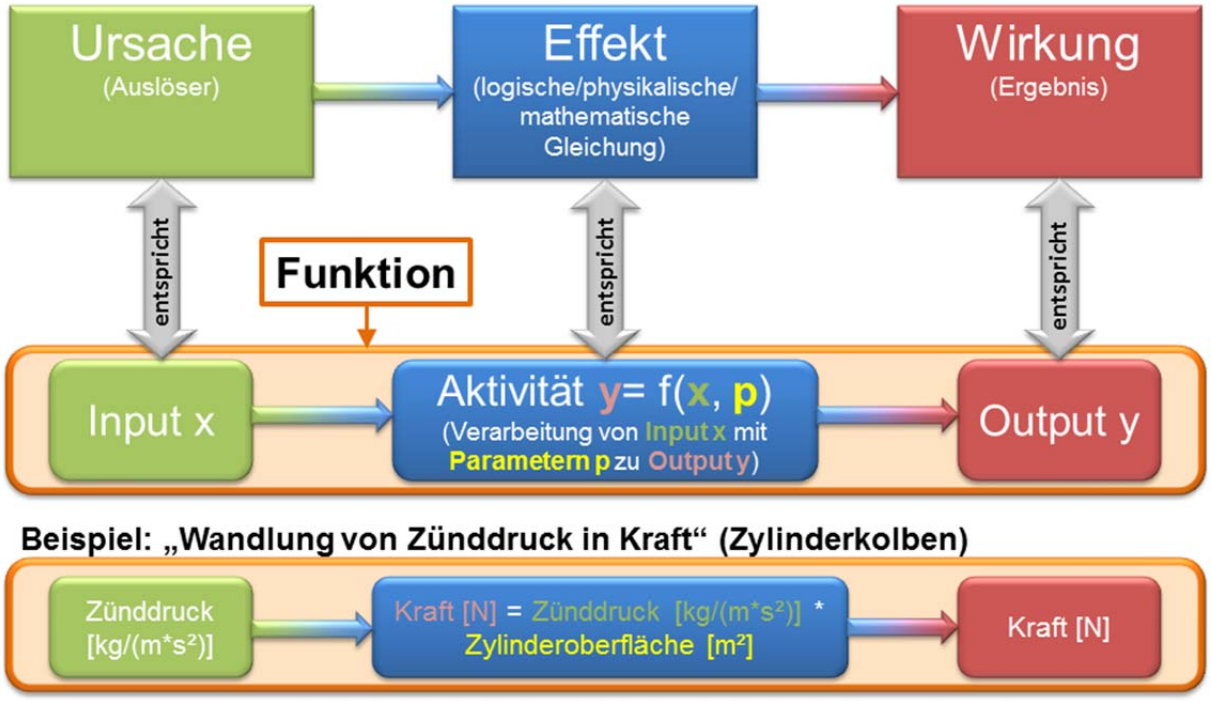


Bild 5-10: Grafik aus der Umfrage als Basis der zu bewertenden Aussagen

Tabelle 2 zeigt die Ergebnisse der hinsichtlich ihrer Nachvollziehbarkeit zu bewertenden Statements zu Aspekten der Modellbildung des Funktionsbegriffs.

Tabelle 2: Bewertungen der Nachvollziehbarkeit der Statements zu Aspekten der Modellbildung des Funktionsbegriffs

zu bewertende Aussagen der Umfrage:	Lässt sich gut anwenden	Lässt sich nur bedingt anwenden	Lässt sich nicht anwenden	Keine Wertung
Eine Ursache ist ein Input und gleichzeitig der Auslöser einer Verarbeitungsaktivität (Grüne Elemente in der Grafik).	68%	12%	8%	12%
Ein Effekt beruht auf mathematischen, physikalischen oder logischen Prinzipien und kann in Gleichungen beschrieben werden (Blaue Elemente in der Grafik).	50%	38%	4%	8%
Eine Wirkung ist das Ergebnis, also der Output der Verarbeitungsaktivität (Rote Elemente in der Grafik).	76%	12%	4%	8%
Eine Verarbeitung verwendet Parameter (Eigenschaften) der ausführenden Komponente (z. B. Zylinderoberfläche, gelb in der Grafik).	62%	28%	0%	10%
Die Beschreibung einer Funktion umfasst Input -> Verarbeitung -> Output, sowie verwendete Parameter der ausführenden Komponente.	62%	22%	4%	12%

Wie der Tabelle zu entnehmen ist, fanden bereits die aus der in der Umfrage gezeigten Grafik abgeleiteten Aussagen mehrheitliche Zustimmung. Auffällig war die Aussage zwei, der nur 50% voll zustimmten. Die Teilnehmer hatten die Möglichkeit, im Falle einer nicht vollen Zustimmung zusätzlich Anmerkungen zur Begründung abzugeben. Hier wurde mehrfach angemerkt, dass eine mathematische

Beschreibung eines Effekts insbesondere auf einem höheren Abstraktionsgrad nur schwer abzuleiten ist. Für die Modellierungstechnik wurde diese Erkenntnis in der Hinsicht berücksichtigt, als dass die Modellierung von Effekten – wenn sie denn der zu modellierenden Funktion zugrunde liegen - sowohl in WFP als auch LSS auch rein textuell geschehen kann. Dies entspricht der Anwendung des Grey Box-Prinzips (nicht alle Informationen müssen vollständig beschrieben werden), da eine mögliche Ausdetaillierung direkt im fachdisziplinübergreifenden Modell ebenso gegeben ist wie die Ausdetaillierung in fachdisziplinspezifischen Modellen. Weiterhin wurde zu Aussage 4 angemerkt, dass nicht nur Merkmale und Eigenschaften der Komponente (des Volumenelements), sondern auch der Oberfläche in die Verarbeitung einfließen. Dies wurde durch die Einbindung von Parametern der WFP berücksichtigt. Nahezu alle weiteren Anmerkungen bezogen sich auf Verständnisschwierigkeiten, die durch zusätzliche textuelle Erläuterungen bzw. grafische Darstellungen, wie im vorangegangenen Kapitel gezeigt, aufgelöst werden können.

5.4 Konzept einer semiformalen Ontologie der Modellbildung technischer Systeme

Basierend auf der zuvor erarbeiteten Sprachbasis sowie weiterer Forschungsarbeiten am IPEK (z.B. ALBERS ET. AL[404]) wurde ein erstes Konzept einer semiformalen Ontologie entwickelt und grafisch aufbereitet. Das Ziel dieser Ontologie ist ein Zwischenschritt zur Herstellung einer Rechnerlesbarkeit und damit einer Rechnerimplementierbarkeit. Letztlich wird dadurch die rechnerbasierte Abfrage von Instanzen der Sprachbasis (also Modellen konkreter Systeme)[405] ermöglicht. Gleichzeitig ist diese semiformale Darstellung der Begriffe und ihrer Abhängigkeiten eine Grundlage für die Erweiterung der SysML (durch Profilbildung[406]) als Bestandteil der Modellierungstechnik.

Zunächst wurden die zuvor definierten Begriffe der wissenschaftlichen Grundlagen und Modellbildungsansätze[407] als Entitäten erfasst und durch semantische Beziehungen (Relationen) miteinander verknüpft. Dabei kommen Prädikationen als Sonderform einer Relation zum Einsatz, also eine Zuordnung einer spezielleren Form einer Entität zu einer Gruppierung in einer allgemeineren Form (bspw. „Eine *Ist-Funktion* **ist eine** *Technische Funktion*“). Diese Beziehung wird in der Informatik als Spezialisierung bezeichnet und durch einen dreieckigen, geschlossenen Hohlpfeil

[404] Vgl. bspw. Albers et al. (2009a), Fig. 3; Albers et al. (2012a), Fig. 1

[405] Diese Technik wird im Forschungsprojekt „Funktionale Lenkung mechatronischer Produkte“ genutzt, vgl. Kapitel 6.5.3

[406] Vgl. Kapitel 2.6.1.3

[407] Vgl. Kapitel 2.1, Kapitel 2.4.9, Kapitel 5.1 sowie Kapitel 5.2.7

dargestellt (⊣▷). Dies bedeutet, dass alle spezielleren Formen des Begriffs über die gleichen Attribute und Relationen verfügen wie auch der allgemeinere Begriff. Alle anderen Relationen werden durch offene Pfeile dargestellt (➔), wobei bewusst auf die Darstellung der stets vorhandenen Passivform aus Platzgründen verzichtet wurde (z.B.: „malt“ und die inverse Relation „wird gemalt von“)[408]. Einige Beziehungen weisen Kardinalitäten auf. Diese geben an, wie viele Zielinstanzen für eine Quellinstanz von Begriffen logisch Sinn machen[409].

Bei der Bildung der semiformalen, grafischen Darstellung der Ontologie stellten sich zahlreiche Herausforderungen, von denen nur die wichtigsten aufgeführt sind:

- Beziehungen zwischen Elementen können auf mehreren Abstraktionsebenen bestehen. Beispielsweise erfüllt ein technisches System Anforderungen. Eine nichtfunktionale Anforderung ist eine konkrete Anforderungsart, die durch Merkmale und Eigenschaften erfüllt wird, die wiederum Attribute des technischen Systems sind.
- Relationen können nicht nur innerhalb einer, sondern auch über mehrere Abstraktionsebenen hinweg existieren. Dies erschwert die Gliederung der Darstellung in mehrere Teildarstellungen zur Steigerung der Übersichtlichkeit. In den nachfolgend gezeigten Darstellungen wurden daher einige Entitäten und Relationen mehrfach in verschiedenen Darstellungen verwendet.
- Teilweise treten semantisch redundante Relation in der Art und Weise auf, als dass mehrere Relationen auf indirektem Weg die gleiche Aussage treffen wie eine direkte Relation. Beispielsweise äußert eine Struktur ein Verhalten (direkte Relation), jedoch wird das Verhalten eigentlich durch die Summe der Outputs von Ist-Funktionen ausgelöst, die wiederum durch die Wirkstruktur als funktionsrelevanter Anteil der Gesamtstruktur des technischen Systems ausgeübt werden (gleichbedeutend indirekte Relation).

Der in Bild 5-11 dargestellte erste Ausschnitt der Ontologie zeigt verschiedene Elemente und ihre Abhängigkeiten auf sehr abstrakter Ebene. Die Orange dargestellten Elemente sind Teil des Zielsystems. Die Begriffe „Technisches System“ und „Technische Funktion“ als Kernbegriffe sind weiß hinterlegt mit schwarzem Rand dargestellt, da sie weder Ziel- noch Objektsystem klar zugeordnet werden können. Hierzu wurden Spezialisierungen der Begriffe verwendet, die wiederum den Systemen zugeordnet werden konnten. Verhaltenselemente des Objektsystems sind

[408] Diese Umkehrbeziehung wird inverse Relation genannt und dient der Rückverfolgbarkeit in beide Richtungen. Z.B. von „Bild → gemalt von → Maler“ oder von „Maler → malt → Bild“

[409] Beispielsweise

in violett, Strukturelemente in grün dargestellt. Grau dargestellte Entitäten sind Elemente der Systemumgebung (in der folgenden Abbildung nicht enthalten). Blau dargestellte Elemente und Relationen bilden Schnittstellen zwischen Ziel- und Objektsystem.

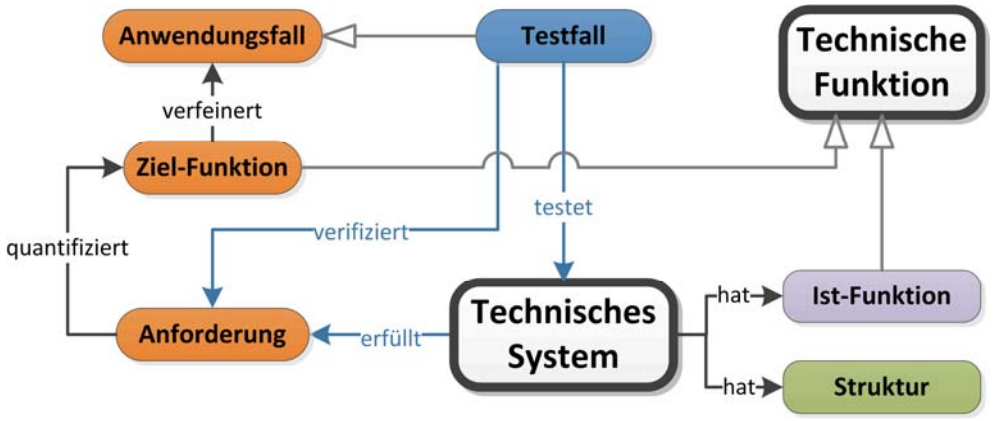


Bild 5-11: Verknüpfung von Ziel- und Objektsystemelementen und Rolle von Testfällen

Hervorzuheben ist hier insbesondere der Testfall, der einerseits eine konkrete Form eines Anwendungsfalls[410] (Zielsystemelement) darstellt, andererseits aber Informationen zur Struktur und den Ist-Funktionen des technischen Systems enthält. Die weiteren Darstellungen der Ontologie finden sich in Anhang 2: Ontologiedarstellungen.

5.5 Diskussion und Zwischenfazit

Die Sprachbasis im Kontext der Modellbildung technischer Systeme ist das konsolidierte Ergebnis aus Literaturrecherchen sowie umfassenden Forschungsarbeiten am IPEK. Sie ist insbesondere maßgeblich durch die Systemtheorie der Technik nach ROPOHL[411] sowie dem Contact & Channel-Ansatz nach ALBERS[412] geprägt, wurde jedoch auch durch andere Forschungsarbeiten im Bereich der systemischen Produktentwicklung, beispielsweise PONN UND LINDEMANN[413], EHRLENSPIEL[414], PAHL ET AL.[415] oder WEBER[416] beeinflusst.

Die in Kapitel 5.2.7 eingeführten Begriffsdefinitionen sind dabei teilweise das Ergebnis sehr aktueller Forschungserkenntnisse und demnach noch in der

[410] vgl. Kapitel 5.2.4
[411] Ropohl (1975), Ropohl (2009)
[412] Z.B. Albers und Sadowski (2013)
[413] Ponn und Lindemann (2011)
[414] Ehrlenspiel (2006)
[415] Pahl et al. (2006)
[416] Weber (2013)

praktischen Anwendung zu evaluieren bzw. zu schärfen. Gleiches gilt für das in Kapitel 5.4 vorgestellte Konzept einer semiformalen Ontologie. Daher stellen diese Ergebnisse vor allem eine Basis für weitere Forschungsarbeiten dar.

Aufgrund technischer Beschränkungen der auf dem Markt erhältlichen Modellierungswerkzeuge konnten im Rahmen dieser Arbeit nicht alle Konzepte der Sprachbasis durchgängig als SysML-Profil implementiert werden. Entsprechende Einschränkungen werden in den folgenden Kapiteln bei der Vorstellung der Modellierungstechnik adressiert.

6 Eine neue Technik zur Modellierung multidisziplinärer Systeme

Eine universell einsetzbare Technik zur Modellierung multidisziplinärer technischer Systeme in Produktentstehungsprozessen birgt die zentrale Herausforderung, Umfang und Konkretisierungsgrad möglichst anwendergerecht auszubalancieren. Wird die Modellierungssprache konkreter, steigt der Umfang ihres „Vokabulars“, um allen Fachdisziplinen gleichermaßen ausreichend konkrete Artefakte zur Modellierung ihrer spezifischen Systeme bereitzustellen. Beschränkt sich die Modellierungssprache auf einen schlanken Umfang, sind ihre Artefakte generisch und damit nur schwer für Anwender spezifischer Fachdisziplinen anwendbar. Es muss also ein Kompromiss aus Abstraktion und Umfang der Modellierungssprache gebildet werden. Da dies – wie die SysML beweist[417] - allein kaum zu bewältigen ist, müssen weitere Stellgrößen zur Steigerung der praktischen Anwendbarkeit identifiziert werden.

6.1 Ein flexibles Anwendungsframework der Modellierungstechnik

Die wichtigste Stellgröße zur Verbesserung der Anwendbarkeit einer in ihrer Ganzheit komplizierten Technik ist der Anwender selbst. Unterscheidet man Anwender nach ihrer fachlichen Expertise und ihrer Rolle im Projekt, kann der Umfang der Möglichkeiten der Modellierungstechnik auf ein anwenderspezifisch notwendiges Minimalmaß reduziert werden. Dies kann auf verschiedene Arten und Weisen erfolgen. Einerseits können auch die Sichten auf das modellierte System auf die für das Fachgebiet des Anwenders relevanten Aspekte begrenzt werden. Dies kann sowohl durch die Einschränkung auf Teilsysteme (z.B. sieht der Kupplungsentwickler nur das Kupplungssystem und seine Schnittstellen, der Rest des Fahrzeugs ist nicht sichtbar) als auch durch die Einschränkung auf bestimmte Funktionen der Modellierung geschehen. Beispielsweise ist für Person A nur das Modellieren von Leistungselektronik erlaubt, Datenleitungen, Komponentenplatzierungen etc. können jedoch nicht modifiziert werden.

Andererseits können die Zugriffsrechte eines Anwenders auf ein Modell gemäß seinem Verantwortungsbereich beschränkt werden (siehe Bild 6-1).

[417] Die SysML ist eine bereits recht umfangreiche Sprache mit relativ hohem Lernaufwand und dennoch wird sie oft als zu generisch für die Modellierung realer Systeme angesehen

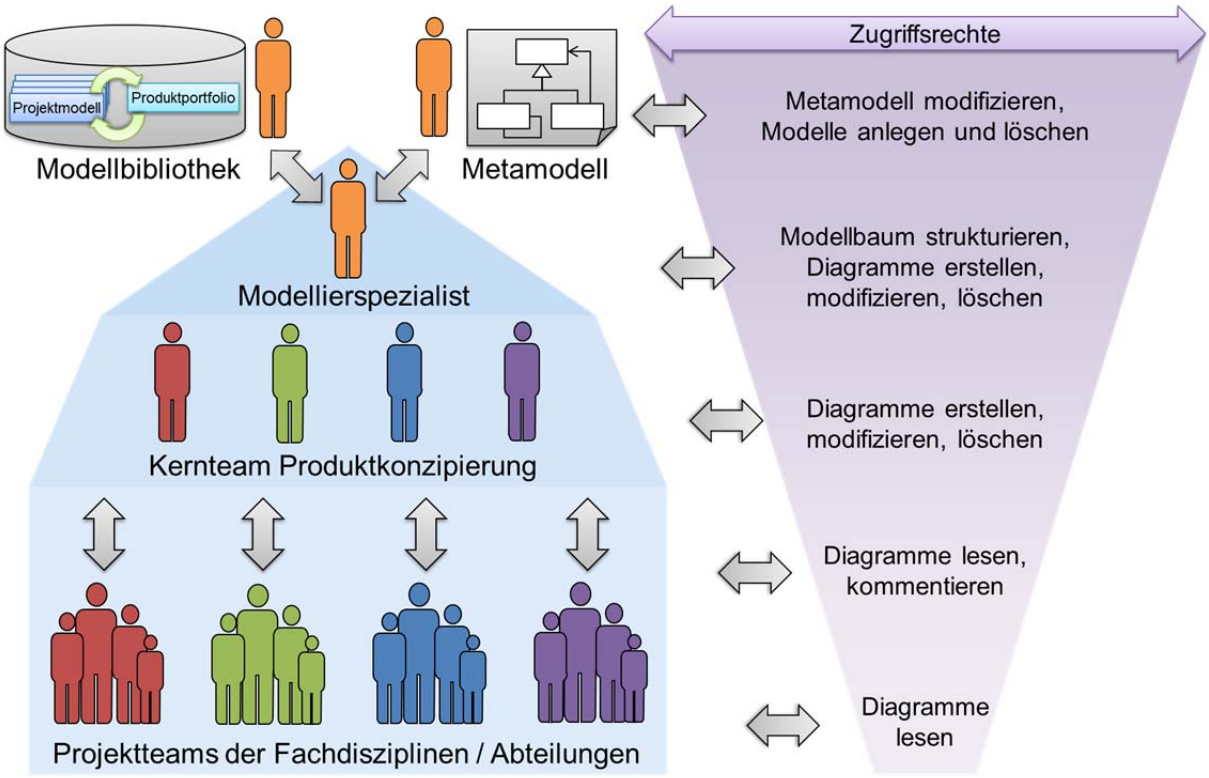


Bild 6-1: Strukturierung der Anwendung der Modellierungstechnik nach Zugriffsrechten

In dieser Zugriffsstruktur darf ein Projektmitarbeiter lediglich seine Informationen per Leserecht aus dem Modell beziehen und ggf. Kommentare einfügen, während bspw. sein Teamleiter als Mitglied des Kernteams auch Aspekte seines Fachbereichs erstellen, modifizieren oder löschen darf. Ein Modellierspezialist ist verantwortlich für das Systemmodell in einem konkreten Projekt, darf dieses strukturieren und hat Vollzugriff auf das gesamte Modell. Darüber hinaus gibt es noch einen Verantwortlichen für das Metamodell, das einheitlich für das gesamte Unternehmen sein sollte. Seine Rolle ist, unternehmensspezifische Erweiterungen und Vereinbarungen zu Modellierungselementen und Relationen durch Profile in die Modellierungssprache einzupflegen. Weiterhin gibt es eine Modellbibliothek, in der einerseits Funktions- und Bauteilbibliotheken abliegen sowie alle konkreten Produkt- oder Projektmodelle, die möglicherweise auch in ein Produktportfolio migriert werden können. Diese ist durch mindestens eine verantwortliche Person zu pflegen.

Der Vorteil der oben gezeigten Zugriffsstruktur ist, dass nicht jeder Mitarbeiter den vollen Funktionsumfang der Modellierungstechnik erlernen muss, sondern nur einige wenige Kernteammitglieder, die wiederum durch einen zentralen Spezialisten unterstützt werden. Übergreifende Fragen nach der Strukturierung von Modellen und Produkten in Portfolios sowie zur Modellierungssprache selbst werden zentral im Spezialistengremium diskutiert, die sich wiederum ihre Spezialkenntnisse in der Anwendung mit fachlichem Hintergrund, der Modellbildung und der Modellverwaltung aufteilen.

Die Definition eines allgemein gültigen, starren Vorgehensmodells für die Anwendung einer universell für sämtliche Arten technischer Systeme einsetzbare Modellierungstechnik ist nicht möglich. Die iterativ durchführbaren Aktivitäten bei der Anwendung der Modellierungstechnik sind schematisch in Bild 6-2 dargestellt.

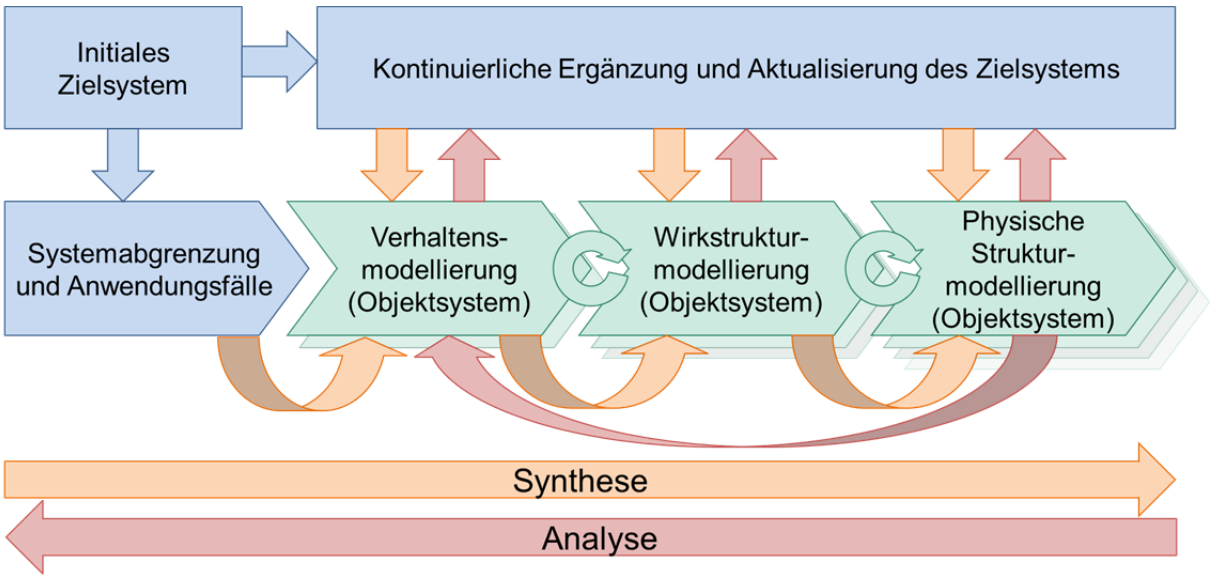


Bild 6-2: Schema der Modellierungsaktivitäten[418]

Produktentwicklung ist nie rein sequentiell, sondern ein dynamischer Prozess mit Iterationen und Rekursionen. Dem wird durch die Darstellung der logischen Aktivitätsabfolge im Synthese- und im Analyseprozess Rechnung getragen. Die ist Modellierungstechnik fraktal aufgebaut, was bedeutet, dass zu jedem Zeitpunkt zwischen Synthese und Analyse gewechselt werden kann. Die entsprechend orange (Synthese) und rot (Analyse) gefärbten Pfeile stellen dabei die jeweils möglichen Folgeaktivitäten dar.

Ausgehend vom initialen Zielsystem werden im **Syntheseprozess** neben den Systemgrenzen mit interagierenden Nachbarsystemmodellen[419] und deren Schnittstellen (Systemabgrenzung) die Anwendungsfälle des zu entwickelnden technischen Systems definiert. Anschließend wird das Systemverhalten durch Aktivitäts- und Zustandsdiagramme modelliert. Die entspricht einer qualitativen Funktionsmodellierung über Abläufe der Wandlungsschritte von Objektflüssen. Daraus leitet sich die Wirkstruktur mit den entsprechenden Leitstützstrukturen und Wirkflächenpaaren ab[420]. Diese kann durch funktionsrelevante Eigenschaften und Merkmale ergänzt werden, um Funktionen auch quantitativ parametrieren und

[418] Weiterentwicklung der Darstellung aus Albers und Zingel (2013b)
[419] Diese entsprechen den Connectoren des Contact & Channel-Ansatzes, vgl. 2.4.9
[420] Die Vernetzung mit Connectoren wurde bereits durch die Systemabgrenzung modelliert

verifizieren zu können (bspw. durch Simulation und Tests). Daraus werden wiederum die resultierenden physischen Strukturen modelliert, wobei Konstruktionsmethoden und –heuristiken wie Trennung der Funktionen oder räumliche Integration von Funktionen zum Einsatz kommen[421]. Hierbei ist es innerhalb jeder Aktivität zur Entwicklung des Objektsystems (grün) möglich, den Detaillierungsgrad zu erhöhen oder zu verringern (analog zum fraktalen Charakter des zugrundeliegenden Contact & Channel-Ansatzes). Parallel zur Modellierung des Objektsystems wird das Zielsystem aktualisiert und konkretisiert.

Im Analyseprozess werden die Aktivitäten grundsätzlich rückwärts durchlaufen, was bedeutet, dass man zu Beginn der Entwicklung einer neuen Produktgeneration[422] zunächst die existierende physische Struktur der Vorgängergeneration modelliert. Aus technischen Gründen ist es unter Verwendung objektorientierter Modelle nicht möglich, daraus direkt die Wirkstruktur zu extrahieren[423]. Daher wird nun analysiert, welche Objektflüsse im System verarbeitet werden und welche Zustände die Teilsysteme einnehmen können. Die Analyseergebnisse werden dann in Form von Aktivitäts- und Zustandsdiagrammen modelliert, was wiederum den Ausgangspunkt für die Synthese einer neuen Produktgeneration darstellt. Gleichzeitig wird auch hier das Zielsystem aktualisiert und ergänzt. Der Wechsel zwischen Synthese und Analyse kann auch innerhalb der Entwicklung vollzogen werden, beispielsweise in der Tragweitenanalyse modellierter technischen Teillösungen.

Aus der konkreten Modellierung eines technischen Systems mit definiertem Modellierungszweck kann ein konkretes Vorgehensmodell abgeleitet werden. Bild 6-3 zeigt hierzu ein fraktales Modellierungsaktivitätsmodell mit einem beispielhaften Sequenzmodell und den (stark vereinfachten) Wechselwirkungen zwischen diesen.

---

[421] Für eine Liste konkreter Beispiele vgl. Albers und Sadowski (2013), S. 13

[422] ALBERS geht davon aus, dass mehr als 90% aller Innovationsprojekte in der Produktentstehung auf bekannten Lösungsprinzipien beruhen, also eine *evolutionäre Produktgenerationsentwicklung* darstellen, vgl. Albers (2011)

[423] Der Übergang von Wirkstrukturen zu physischen Strukturen in der Modellierungstechnik nutzt das Konzept der Vererbung, das nicht umkehrbar ist.

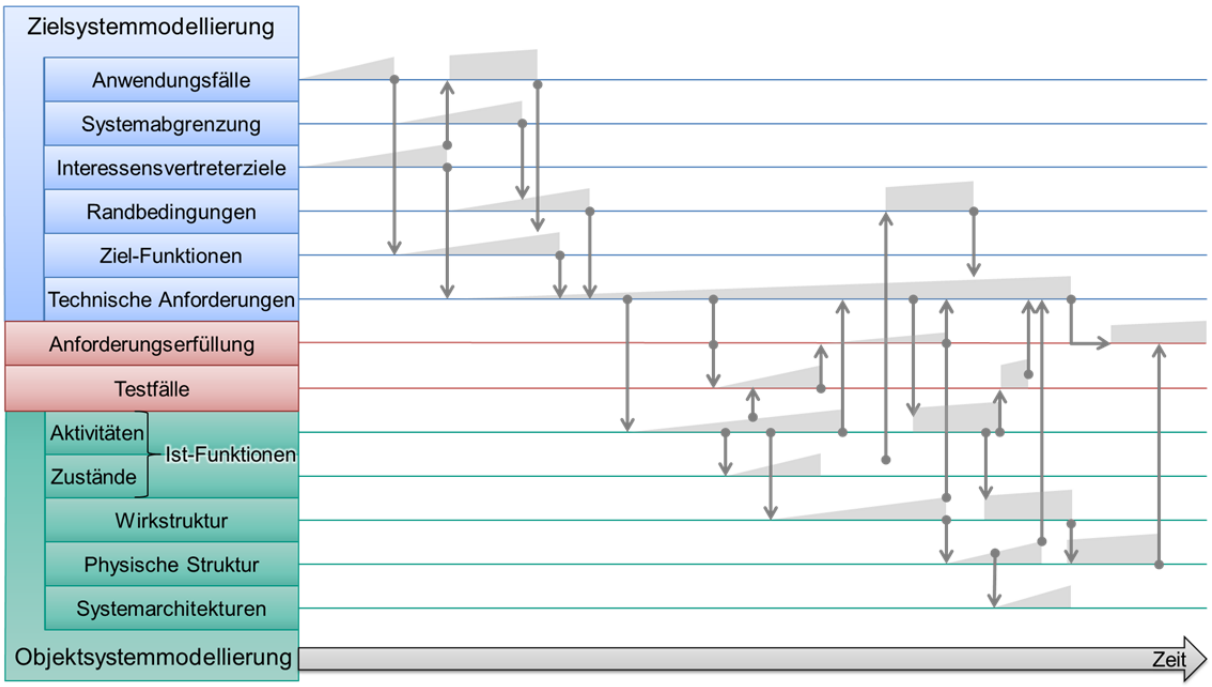


Bild 6-3: Rahmenwerk möglicher Modellierungsaktivitäten mit beispielhaftem Sequenzmodell

Ähnlich wie beim integrierten Produktentstehungsmodell iPeM nach ALBERS kann ein Sequenzmodell für durchzuführende Modellierungsaktivitäten in Form eines *Referenzmodells* für einen definierten Modellierungszweck aus den links gezeigten Aktivitäten abgeleitet werden, das als Vorgabe für eine Reihenfolge der Modellierung aller relevanten Aspekte des Modellierungszwecks dient. Daraus wiederum kann für ein konkretes Projekt ein weiter auf die zu Projektbeginn bekannten, spezifischen Bedingungen (z.B. konkrete Projektteams) angepasstes *Implementierungsmodell* abgeleitet werden, das als konkreter Handlungsleitfaden für das Projekt dient.

Die Dreiecke in Bild 6-3 repräsentieren den wachsenden Informationsgehalt jedes Aspekts über die Zeit, die Pfeile zwischen den Aspekten repräsentieren die modellierbaren Wechselwirkungen zwischen den Aspekten. Aspekte des Zielsystems sind in blau, Aspekte des Objektsystems in grün dargestellt. Die rot dargestellte Vernetzung der beiden Systeme wird durch Testfälle zur Anforderungsverifikation sowie durch die Vernetzung von Anforderungen mit Elementen des Objektsystems durch Erfüllungsbeziehungen realisiert. Die Summe aller Dreiecke und Pfeile steht für ein Sequenzmodell eines konkreten Projekts bzw. einer Projektart. Das gezeigte Sequenzmodell stellt jedoch eine starke Vereinfachung der in der Realität oftmals iterativen und rekursiven Modellierungsaktivitäten dar.

Die Diamantdarstellung in Bild 6-4 zeigt eine qualitative Darstellung des Informationsgehalts der Aspekte eines typischen Neuentwicklungs-Produktmodells auf Basis der Modellierungsaktivitäten.

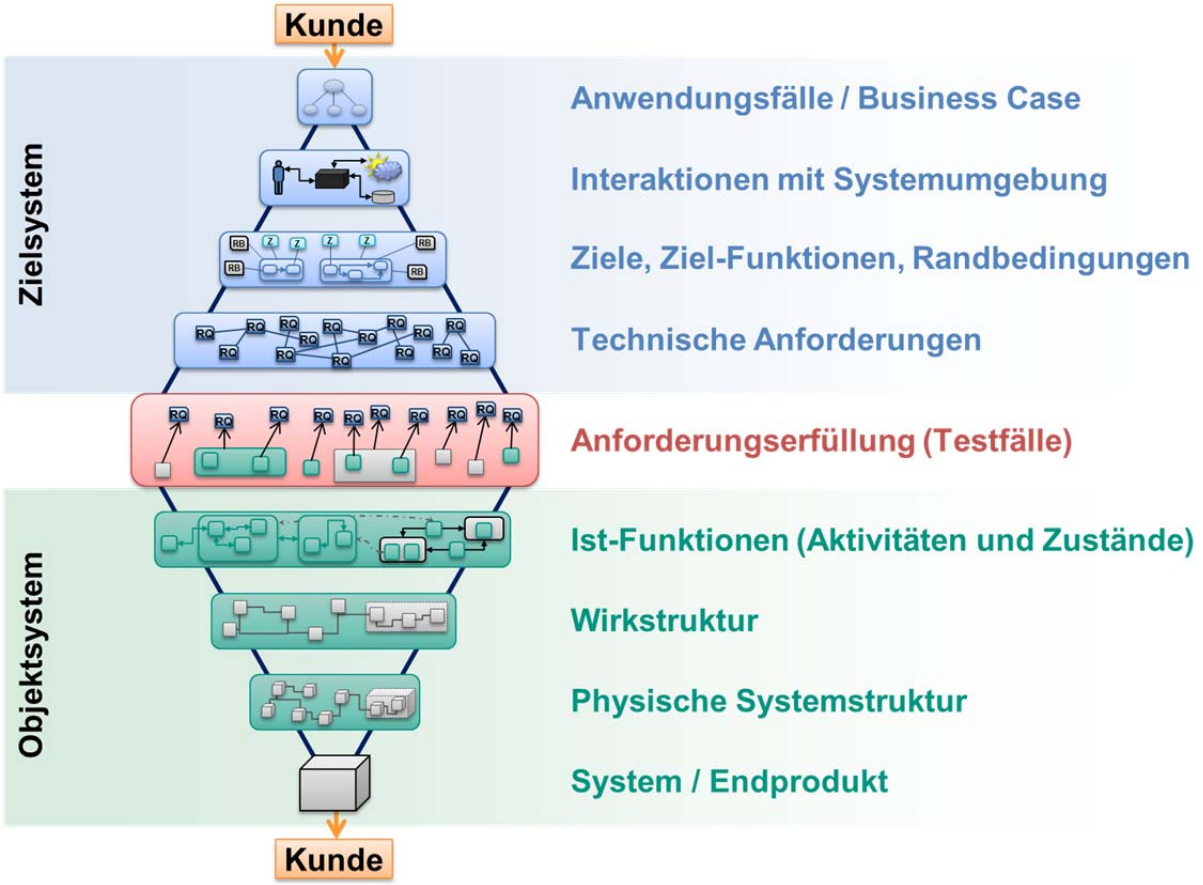


Bild 6-4: Modellierungsdiamant[424]

Sollen sämtliche fachdisziplinübergreifend relevanten Informationen erstmals in einem Produktmodell erfasst und vernetzt werden, entsteht ein Top-Down-Modell[425] von Ziel- und Objektsystem. Das Zielsystem besteht neben Anwendungsfällen, Systeminteraktionen mit Nachbarsystemen, Interessensvertreterzielen, Randbedingungen und Ziel-Funktionen insbesondere aus technischen Anforderungen, die aus den vorangegangenen Modellierungsaspekten abgeleitet wurden.

Das Objektsystem wird für eine Neuentwicklung ebenfalls Top-Down innerhalb der Aspekte modelliert, was bedeutet, dass zunächst die Hauptfunktionen mit Gesamtsystemzuständen eines neu zu entwickelnden Produkts modelliert und anschließend iterativ in ihre Sub- und Nebenfunktionen mit Teilsystemzuständen dekomponiert werden. Aus den Funktionen werden dann Wirkstrukturen abgeleitet.

[424] In Anlehnung an Denger et al. (2012) und Denger et al. (2013)

[425] Top-Down steht für die Modellierung vom Groben ins Feine bzw. die Verfeinerung innerhalb der jeweiligen Aspekte sowie die Verfeinerung von Aspekten durch weitere Aspekte. Beispielsweise verfeinern Randbedingungen, welche Schnittstellen der Infrastruktur mit dem technischen System wechselwirken.

Die Verfeinerung von Funktionen und entsprechenden Wirkstrukturen erfolgt wiederum iterativ, also aus jeder Detaillierungsebene der modellierten Ist-Funktionen wird eine entsprechende Wirkstruktur abgeleitet. Anschließend werden die Wirkstrukturen in eine resultierende physische Struktur überführt. Aus der Realisierung der LSS und WFP durch physische Komponenten können sich jedoch neue technische Anforderungen ergeben, die iterativ den bestehenden Anforderungen hinzuzufügen und mit ihrer Ursache, der technischen Teilsystemlösung (die verantwortliche Komponente der physischen Struktur) per Ableitungsbeziehung zu vernetzen ist. Diese neuen Anforderungen sind dann wiederum durch entsprechende Lösungen zu erfüllen.

Ein Beispiel für eine solche Iteration in der Modellierung ist die Realisierung der Funktion „Drehmoment übertragen und wandeln“ durch Wellen und Zahnräder in der physischen Struktur. Wird nun eine weitere Anforderung berücksichtigt, die eine gewisse Mindestlebensdauer des Systems fordert, kann sich aus dem genannten Lösungsprinzip ergeben, dass eine Schmierung der Zahnflanken erforderlich wird. Dies wird zunächst als funktionale Anforderung definiert und anschließend wiederum durch entsprechende Funktionen, eine Wirkstruktur und letztlich eine physische Struktur realisiert.

Folglich bedeutet dies, das der in Bild 6-4 gezeigte Modellierungsdiamant nicht sequentiell von oben nach unten durchlaufen wird, sondern zwischen den Ebenen respektive den Aspekten iterativ hin- und hergesprungen werden kann, um die im Zuge der Produktentwicklung entstehenden Informationen in den jeweiligen Aspekten hinzuzufügen. Auf diese Weise werden Zielsystem und zugehöriges Objektsystem kontinuierlich aktualisiert und konkretisiert.

In der Modellierung von Softwaresystemen können neben Iterationen auch Rekursionen auftreten, also der Aufruf einer Funktion von sich selbst mit veränderten Eingangsgrößen. Für vertiefende Informationen zu rekursiven Funktionen bzw. Algorithmen in der Informatik sei hier jedoch auf weiterführende Fachliteratur verwiesen[426].

Wird die Modellierungstechnik in einem konkreten Projekt angewendet, treten aufgrund der Dynamik realer Entwicklungsprozesse meist Abweichungen zu dem geplanten Vorgehen auf. Daher wird analog zum Konzept des Sequenzmodells des iPeM auch hier ein *Applikationsmodell* zur Beschreibung der real durchgeführten Modellierungsaktivitäten innerhalb eines konkreten Projekts eingeführt. Dieses Modell dient dem Soll-Ist-Abgleich der durchgeführten Modellierungsaktivitäten im

[426] bspw. Klima und Selberherr (2010), Kap. 18; Müller und Weichert (2011), Kap. 6

Sinne des Nachbereiten und Lernens und kann im Falle des Gewinns übertragbarer Erkenntnisse in ein Referenzmodell für ähnlich geartete, zukünftige Projekte überführt werden.

Die folgenden Kapitel stellen die Artefakte der Modellierungssprache zur Durchführung der im Vorgehensmodell eingeführten Modellierungsaktivitäten im Detail anhand einfacher Beispiele aus den durchgeführten Evaluationsprojekten vor. Konstrukte gemäß der SysML-Spezifikation sind zur klareren Unterscheidung in ihrer englischen Originalbezeichnung und *kursiv* hervorgehoben. Die im Rahmen der vorliegenden Arbeit implementierten ergänzenden Konstrukte sind darüber hinaus blau hervorgehoben und mit einem *Stern** gekennzeichnet.

6.2 Durchgängige Zielsystemmodellierung

Im Rahmen der iPeM-Aktivitäten Projektierung und Profilfindung[427] wird nach Möglichkeit gemeinsam mit dem Kunden das initiale Zielsystem definiert. Dies besteht aus Anwendungsfällen und Ziel-Funktionen, Systeminteraktionen mit Nachbarsystemen, Interessensvertreterzielen, Randbedingungen und bekannten Systemschnittstellen zur Umwelt[428]. Hierbei wird das zu entwickelnde technische System, wie bereits im Kapitel 5.1 eingeführt, als Black Box betrachtet. Das Ergebnis entspricht der modellbasierten Abbildung des Lastenhefts. Anschließend erfolgt die „Übersetzung“ der o.g. Aspekte in technische Anforderungen durch das entwickelnde Unternehmen. Das initiale Zielsystem wird im Verlauf des Produktentstehungsprozesses kontinuierlich aktualisiert und konkretisiert. Die folgenden Unterkapitel stellen die Modellierung der genannten Aspekte an Beispielen vor und beschreiben den generierten Informationszuwachs des Modells und die Bedeutung der zu verwendenden Konstrukte[429] der zugrundeliegenden Modellierungssprache auf Basis der SysML. Die entsprechenden Modellierungsregeln sind nach Aspekten gegliedert in Anhang 1: Modellierungsregeln zu finden.

6.2.1 Modellierung der Anwendungsfälle

Anwendungsfälle beschreiben in erster Linie die gröbste Ebene der Ziel-Funktionen eines technischen Systems und definieren damit gleichzeitig seinen Zweck durch die Beantwortung der Frage, **was** das System in Interaktion mit seiner Systemumgebung

[427] Makroaktivitäten des iPeM, vgl. bspw. Albers (2010)

[428] Bei externen Kunden ist es durchaus üblich, dass das initiale Zielsystem in Form eines Lastenhefts allein durch ihn definiert wird. In diesem Falle wären das Lastenheft retrospektiv im Modell abzubilden und auftretende Unstimmigkeiten bzw. offene Punkte mit dem Kunden abzustimmen.

[429] Entitäten, Relationen und Diagrammarten

tun soll. Weiterhin können Anwendungsfälle auch qualitativ beschreiben, welche Tests mit einem System durchgeführt werden, die keinen normalen Nutzungsfall betreffen wie beispielsweise Crashtests, Belastungstests oder Dauertests[430]. Bei der Anwendungsfallmodellierung ist zu beachten, dass keinerlei technische Lösungen vorwegzunehmen sind – es wird also explizit **nicht** beschrieben, **wie** das System seine Hauptfunktionen ausüben soll.

Anwendungsfälle werden in erster Linie für die Lebenszyklusphase der Nutzung in einem Anwendungsfalldiagramm definiert. Abhängig vom Modellierungszweck können darüber hinaus weitere Sichten erforderlicher Lebenszyklusphasen wie Entwicklung, Montage, Wartung oder Demontage/Entsorgung in Form weiterer Anwendungsfalldiagramme definiert werden, die in folgenden Aspekten ebenfalls zu berücksichtigen und in die Entwicklung einzubeziehen wären. Im Rahmen der vorliegenden Arbeit wird nur die Nutzungsphase weiter betrachtet.

Bild 6-5 zeigt ein Anwendungsfalldiagramm für die Nutzung eines Hybridfahrzeuges, beschränkt auf dessen Hauptfunktionen mit Strukturierungsbeispielen.

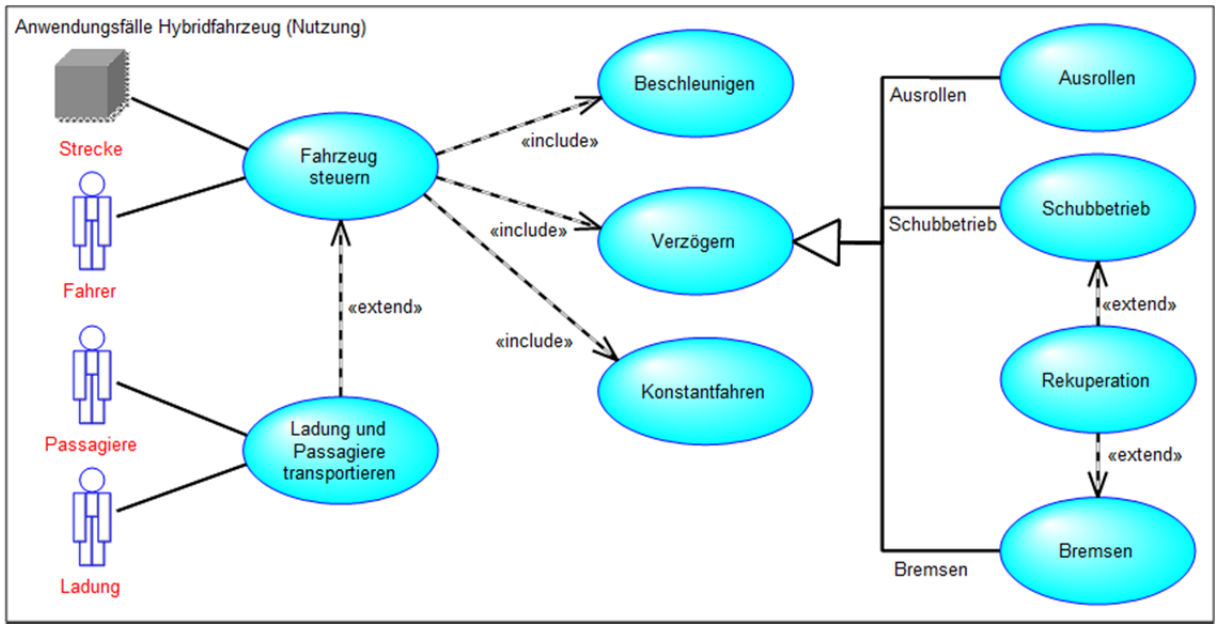


Bild 6-5: Anwendungsfalldiagramm

Viele Hauptfunktionen technischer Systeme stehen in Relation zueinander und sind oftmals nicht als gleichwertig zu betrachten. So auch im Fall der gezeigten Anwendungsfälle eines Hybridfahrzeugs. Das Modell des zugrundeliegenden Evaluationsprojekts hatte u.a. den Zweck, die Funktion „Rekuperation“ eines Hybridfahrzeugs im Detail abzubilden. Entsprechend wurden die Hauptfunktionen bis

[430] Vgl. Kapitel 5.1 und Kapitel 5.2.4

zur Ebene der Rekuperation dekomponiert. Vereinfachend wurden weitere, interagierende Nachbarsysteme wie andere Verkehrsteilnehmer, das Wetter/Klima oder Kommunikationssysteme ausgeblendet. Die Anwendungsfälle selbst wurden mittels der Erweiterungsbeziehung (*extend*), der Enthält-Beziehung (*include*) sowie der Spezialisierungs-/Generalisierungsbeziehung (*generalization*) strukturiert[431]. Ergänzend zur Standard-SysML wurde der Stereotyp *Technical Neighbor System** eingeführt, der technische und nicht-technische Akteure[432] u.a. optisch in Form eines grauen Würfels voneinander abgrenzt. Anwendungsfälle können – wie jedes SysML Artefakt – durch textuelle Beschreibungen oder Verlinkungen auf Sekundärdokumente ergänzende Informationen erhalten.

Da Anwendungsfalldiagramme eine rein hierarchische Strukturierung erlauben, besteht in SysML die Möglichkeit, bereits bekannte bzw. geforderte interne Abläufe von Anwendungsfällen mittels Aktivitäten zu beschreiben. Da Aktivitäten jedoch auch zur Beschreibung von realisierten Funktionen zum Einsatz kommen, wurde diese Entität durch den Stereotyp *Target Function** spezialisiert, der eine gewünschte Verarbeitungsaktivität einer Ziel-Funktion repräsentiert.

Bild 6-6 zeigt die Verwendung von *Target Functions** am Beispiel „Rekuperation im Schubbetrieb“ (rechts) und den zugehörigen Modellbaum (links).

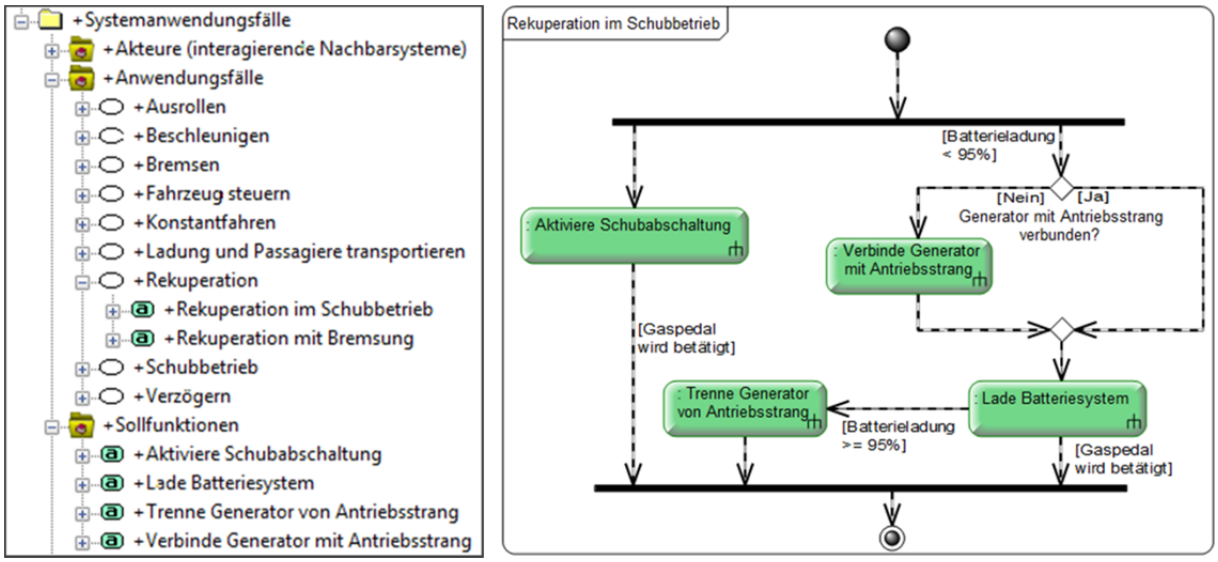


Bild 6-6: Modellbaumausschnitt und Ziel-Funktionen im Aktivitätsdiagramm

Das gezeigte *Activity Diagram* zeigt *Target Functions** (grün) und ihre logische Abfolge mit Entscheidungsknoten (Rauten) und Bedingungen (in eckigen Klammern).

---

[431] Diese Beziehungen sind Teil der Standard-SysML. Auf die Erläuterung wird daher an dieser Stelle verzichtet. Informationen zur Standardspezifikation der SysML finden sich in OMG SysML (2012).

[432] Der zugrundeliegende *Actor* ist eine Metaklasse der UML, die ein Mensch-Symbol verwendet

6.2.2 Modellierung der Kundenziele

Anwendungsfälle eines technischen Systems werden durch Interessensvertreter meist um konkrete Zielvorgaben ergänzt, die quantitative Angaben zu geforderten Funktionen in Form von geforderten Eigenschaften, aber auch nichtfunktionalen Merkmalen des Systems beinhalten.

Diese Ziele werden im *Requirement Diagram* durch *Stakeholder Objectives** modelliert, die eine spezialisierte Anforderung (SysML-Stereotyp *Requirement*) darstellen. Ihr wesentliches Merkmal ist, dass sie nur in Absprache mit dem definierenden Auftraggeber geändert werden dürfen und keinesfalls selbstständig durch den Auftragnehmer. Da Auftraggeber sowohl externe Kunden als auch interne Instanzen des Unternehmens sein können, besteht die Möglichkeit, den Typ des Ziels durch das Attribut *Objective_Type** zu definieren. Ziele beziehen sich immer auf das Gesamtsystem als Black Box, es werden folglich keine Teilsysteme explizit adressiert. Da Ziele eine Anforderungsart darstellen, sollten diese, ebenso wie alle anderen nachfolgend vorgestellten Anforderungsarten, gewisse Merkmale aufweisen, die „gute“ Anforderungen ausmachen. Hierzu zählen insbesondere Erreichbarkeit, Verifizierbarkeit, Eindeutigkeit oder Vollständigkeit[433].

Bild 6-7 zeigt einige Ziele für die Entwicklung eines Hybridantriebsstrangs.

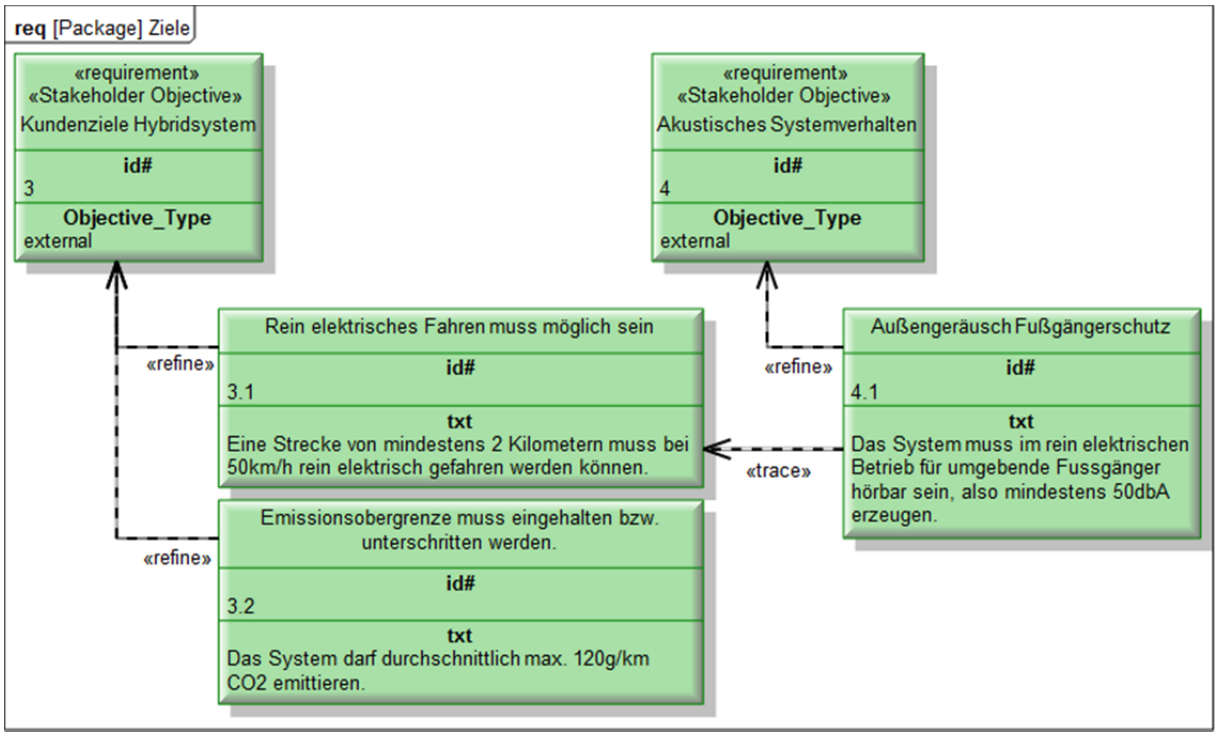


Bild 6-7: Ziele im Anforderungsdiagramm

[433] Das Department of Defense formuliert bspw. Merkmale guter Anforderungen in DoD (2001), S. 36

Die Abbildung zeigt beispielhaft, dass verschiedene Attribute von Zielen je nach Informationsbedarf des Anwenders ein- und ausgeblendet werden können. So wird der *Objective_Type** nur für die Hauptanforderungen gezeigt, dafür ihr Text im Gegensatz zu den Subanforderungen ausgeblendet. Ein Anforderungsmodell ist nicht in einer reinen Baumstruktur aufgebaut, auch wenn eine überwiegend hierarchische Zuordnung häufig angewendet wird (Anforderungsbaum). Wie die *trace*-Beziehung in Bild 6-7 exemplarisch aufzeigt, erzeugen die zahlreichen Querbeziehungen vielmehr ein Anforderungsnetzwerk.

6.2.3 Modellierung der Systemumgebung

Als Teil der Anwendungsfallmodellierung wurden bereits qualitativ die Interaktionen zwischen dem zu entwickelnden System und Akteuren (*Actors*) modelliert. Jedoch wurden darin keine Informationen zur Art der meist bekannten Interaktionen und Wechselwirkungen festgehalten. Dies geschieht in der Modellierung des Systems in seiner Umgebung und den zugehörigen Schnittstellen in Blockdiagrammen. Hierzu wird ein abstraktes Supersystem definiert (die Systemumgebung) und dessen interne Struktur mit dem zu entwickelnden Teilsystem in seinem Kontext im *Internal Block Diagram* modelliert. Als Teil der Modellierungstechnik wurden zur klareren Abgrenzung verschiedene Schnittstellentypen (z.B. *WS Material*, WS Energy**, *WS Information**) als Spezialisierung des Stereotyps *Port* eingeführt. Diese Schnittstellen sind Wirkflächen, die über das SysML-Element *Connector*[434] zu Wirkflächenpaaren verbunden werden können. Darüber hinaus können relevante Eigenschaften und Merkmale der beteiligten Systeme durch Parameter, in SysML *Block Property (Value)*, abgebildet werden. Demnach entsprechen die Blöcke der Nachbarsysteme mit ihren Ports (Wirkflächen) dem C&C²-A-Modellelement „Connector“. Die konkrete Syntax weicht hier jedoch von jeder des C&C²-A aus technischen Gründen ab. Bild 6-8 zeigt einen beispielhaften Ausschnitt der Systemumgebung eines Hybridfahrzeugs.

[434] Der SysML-Stereotyp „Connector“ ist ein Verbinderelement für Ports und darf nicht mit dem C&C²-A-Element Connector verwechselt werden (für dessen Definition siehe Kapitel 2.4.9)

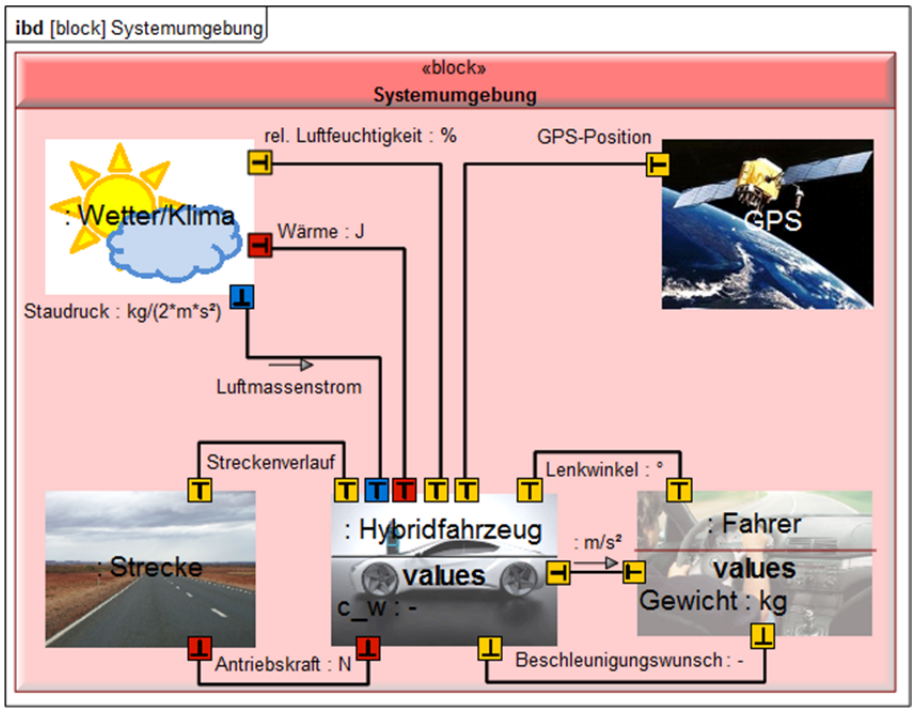


Bild 6-8: Systemumgebung im Internen Blockdiagramm

Wie in der Abbildung zu sehen, besteht die Möglichkeit, Bilder und Piktogramme zur leichteren Lesbarkeit von Diagrammen einzubinden. Dies ist eine Funktion, die durch Modellierungswerkzeuge in stark unterschiedlichem Umfang zur Verfügung gestellt wird. Die Verwendung solcher visueller Hilfsmittel kann die Akzeptanz abstrakter Modellierungssprachen bei Anwendern deutlich erhöhen. Im Falle zahlreicher Interaktionen und Wechselwirkungen des zu entwickelnden Systems mit seiner Umgebung besteht die Möglichkeit, mehrere Sichten auf diesen Aspekt zu erzeugen, indem mehrere Interne Blockdiagramme desselben Blocks erzeugt werden. Beispielsweise kann analog der Vorgehensweise bei der Erstellung von Contact & Channel-Modellen je ein Diagramm pro Funktion und der darin interagierenden Nachbarsysteme (Connectors) erzeugt werden.

Ein kleiner Automatismus, der dem Modellierungswerkzeug im Rahmen dieser Arbeit hinzugefügt wurde, ist das Klonen von Diagrammen inklusive seinem Layout. Dies erleichtert die Arbeit des Anwenders massiv, da er so nicht gezwungen ist, wiederzuverwendende Elemente erneut anzuordnen, sondern direkt die Änderungen im Vergleich zur bestehenden Sicht modifizieren kann.

6.2.4 Modellierung und Ableitung von Randbedingungen

Aus bereits modellierten Aspekten sowie darüber hinaus aus gesetzlichen Vorgaben oder Normen lassen sich projekt- bzw. systemrelevante Randbedingungen ableiten, die bei der Entwicklung einzuhalten sind. Diese werden mittels *Boundary Conditions**, einer spezialisierten SysML-*Requirement*, modelliert. Bild 6-9 zeigt ein in der erweiterten SysML modelliertes Beispiel für Randbedingungen.

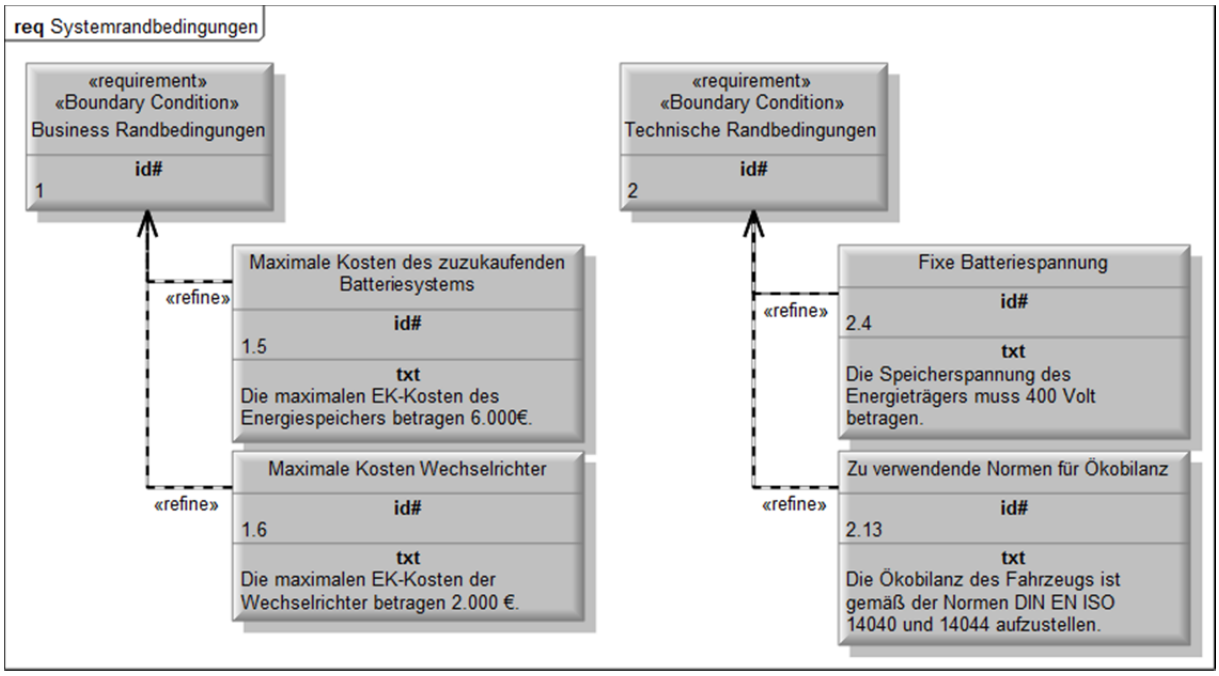


Bild 6-9: Randbedingungen im Anforderungsdiagramm

Wie im gezeigten Diagramm zu sehen ist, können sich Randbedingungen auch auf konkrete Subsysteme beziehen, die beispielsweise zugekauft werden und damit nicht unmittelbarer Teil des zu entwickelnden Systems sind, sondern mit diesem zu vernetzen sind. Hier können auch gegebenenfalls vorhandene und in der Entwicklung zu berücksichtigende Schnittstellen spezifiziert werden. Randbedingungen dürfen nur in Absprache mit der definierenden Instanz, also beispielsweise dem Kunden, modifiziert werden. Im Falle von Normen oder gesetzlichen Bestimmungen sind diese bei Veröffentlichung zu aktualisieren.

6.2.5 Ableitung von initialen technischen Anforderungen

Die in den vorangegangenen Kapiteln modellierten Aspekte sind allesamt Teil des Lastenhefts und somit retrospektiv nachmodellierte Elemente des Zielsystems, die ihren Ursprung außerhalb des Entwicklungsteams haben. Oftmals werden dabei Informationen generiert, die nicht zwingend direkt verwendbar sind, weil offene Punkte, ungeklärte Fragestellungen oder gar Widersprüche verbleiben. Daher sollte die Gelegenheit genutzt werden, die Aspekte gemeinsam mit dem Auftraggeber zu besprechen und ggf. zu aktualisieren. In der Praxis werden teilweise Anforderungen definiert, die sich bereits auf technische Lösungen beziehen, die der Auftraggeber auf Basis seines teilweise unvollständigen Wissens über das existierende Produktportfolio des auftragnehmenden Unternehmens extrahiert und in die Anforderungen aufgenommen hat. Hier kann der Fall eintreten, dass eigentlich eine andere Systemkonfiguration als die vom Auftraggeber vorweggenommene für die geplante Nutzung des zu entwickelnden Produkts besser geeignet wäre. Um dieses mögliche Problem zu umgehen, müssen Anwendungsfälle und Anforderungen des

Auftraggebers explizit für das System als Black Box ohne Vorwegnahme einer bestimmten Produktauswahl für eine mögliche Systemkonfiguration definiert werden. Konkret bedeutet das, sich auf Anwendungsfälle, Ziele und Randbedingungen zu beschränken, ohne sich bereits auf konkrete Systemkomponenten, beispielsweise aus bereits vorhandenen Produkten des Unternehmens oder aus Produktkatalogen zu berufen.

Der nächste logische Schritt in der Modellierung ist die Ableitung initialer technischer Anforderungen als Ausgangspunkt der Systemkonfiguration. Hierzu wurden drei Arten technischer Anforderungen in Form von spezialisierenden Stereotypen der SysML-*Requirement* definiert:

- *Continuous Functional RQ**: Dies ist eine funktionale Anforderung, die durch eine kontinuierliche Funktion innerhalb eines (Teil-)Systemzustandes zu erfüllen ist.
- *Discrete Functional RQ**: Dies ist eine funktionale Anforderung, die durch eine diskrete Funktion zur Durchführung eines Zustandswechsels eines (Teil-) Systems zu erfüllen ist.
- *Property RQ**: Dies ist eine nichtfunktionale Anforderung, die durch Eigenschaften oder Merkmale von Systemkomponenten zu erfüllen ist.

Die funktionalen Anforderungen fordern allgemein gesprochen etwas, was das zu entwickelnde System **tun** soll, wohingegen die nichtfunktionale Anforderung etwas fordert, was das zu entwickelnde System **haben** oder **sein** soll. Auch wenn letzterer Anforderungstyp nichtfunktional ist, können hierbei Eigenschaften von Funktionen des Systems in einer bestimmten Ausprägung eingefordert werden (z.B. „Beschleunigung in weniger als 7 Sekunden von 0 auf 100 km/h“). Auch Merkmale sind innerhalb der Wirkstruktur durch in Wirkflächenpaaren und Leitstützstrukturen auftretende naturwissenschaftliche Effekte funktionsbeeinflussend. Beispiele hierfür sind Materialparameter wie Steifigkeit, Dichte oder Leitfähigkeit oder geometrische Merkmale wie Längen, Durchmesser oder Dicken. Beispiele für tatsächlich nicht zwingend unmittelbar funktionsrelevante Eigenschaften sind Kosten bzw. Merkmale die Ästhetik (z.B. Farbe). Ein Beispiel für verschiedene abgeleitete technische Anforderungen zeigt Bild 6-10.

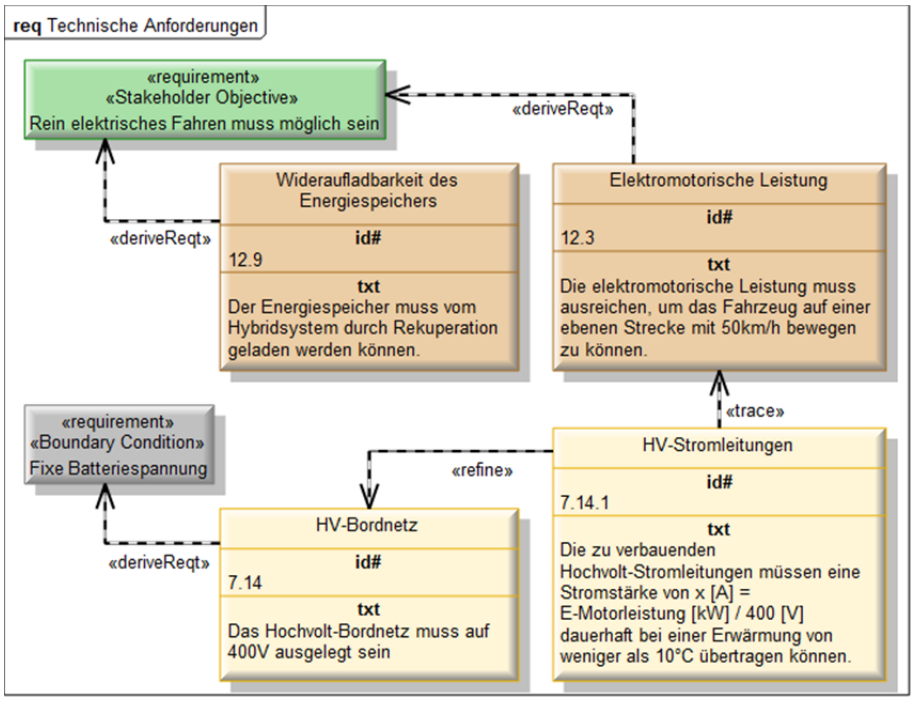


Bild 6-10: Abgeleitete, initiale technische Anforderungen im Anforderungsdiagramm

Wie bereits dem relativ einfachen Beispiel des gezeigten Diagramms als Teil des Zielsystems zu entnehmen ist, können Anforderungen nie in einer reinen Baumstruktur einer einzelnen Anforderung untergeordnet werden. Vielmehr entsteht ein komplexes Netzwerk aus Anforderungen mit zahlreichen, verschiedenartigen Wechselwirkungen, wie beispielsweise die textuell formulierte Gleichung in Anforderung #7.14.1 verdeutlicht.

Zur Modellierung parametrischer Zusammenhänge stellt die SysML einen eigenen Diagrammtyp bereit, das *Constraint Diagram* (Zusicherungsdiagramm). Da es jedoch einerseits auf die Modellierung parametrischer Zusammenhänge von Komponenten ausgelegt ist und andererseits nur qualitative Zusammenhänge ohne mathematische oder logische Operatoren abbilden kann, wird hier auf die Vorstellung verzichtet. Auf dieser Ebene der Detaillierung wird häufig auf leistungsfähigere Werkzeuge zur logischen oder mathematischen Modellierung wie beispielsweise Modelica oder Matlab/Simulink zurückgegriffen. Am Markt erhältliche Modellierungswerkzeuge bieten bereits heute vorkonfigurierte Schnittstellen zu Simulationswerkzeugen an, wodurch Informationen zwischen SysML und anderen Modellen ausgetauscht werden können[435].

[435] Vgl. Kapitel 6.5

6.2.6 Kontinuierliche Aktualisierung des Zielsystems

Technische Anforderungen werden zunächst aus existierenden Aspekten abgeleitet und im weiteren Verlauf des Produktentwicklungsprozesses kontinuierlich aktualisiert. So leiten sich beispielsweise zahlreiche weitere, technische Anforderungen aus gewählten Lösungskonzepten bei der Dekomposition von Ist-Funktionen und ausführenden Wirkstrukturen bzw. physischen Strukturen ab. Diese werden im *Requirement Diagram* modelliert. Da neue Relationen nicht durch Anwender des verwendeten Modellierungswerkzeugs allein implementierbar sind, wurde für diese Ableitungsbeziehung die allgemeine *trace*-Beziehung verwendet (siehe Bild 6-11). Die SysML bietet zwar eine „derive“-Beziehung, jedoch ist diese nur zwischen Anforderungen und nicht in Verbindung mit anderen Elementen nutzbar.

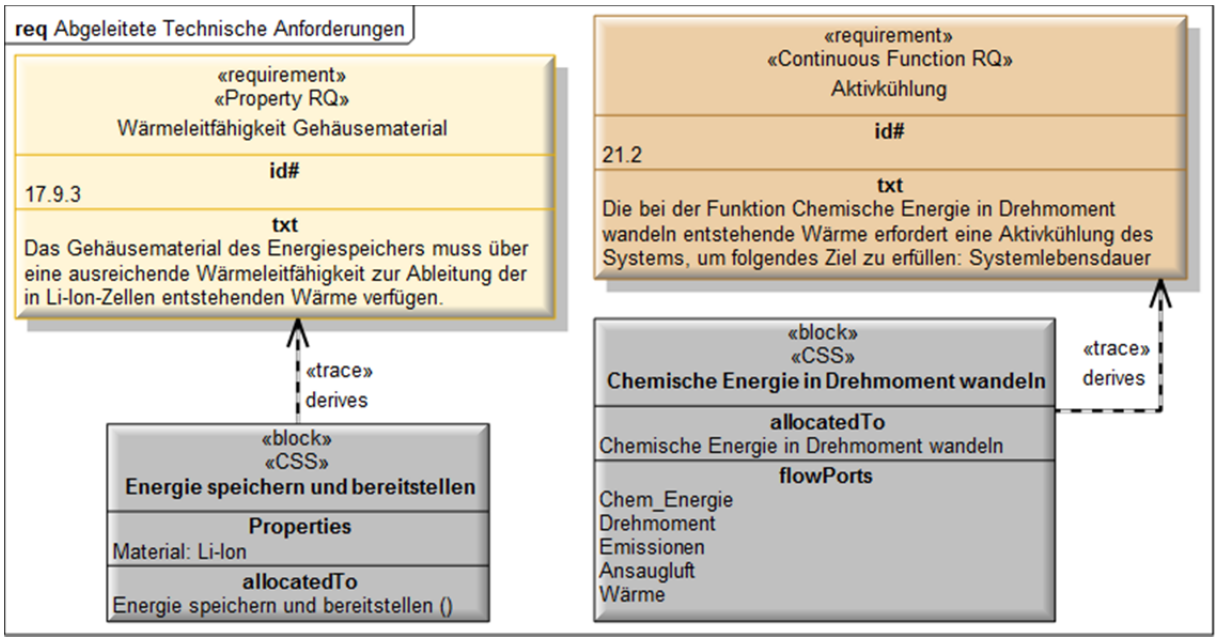


Bild 6-11: Abgeleitete Technische Anforderungen im Anforderungsdiagramm

Das Beispiel zeigt eine nichtfunktionale und eine funktionale Anforderung, die aus zwei Leitstützstrukturen abgeleitet wurden. Diese wurden erzeugt, um zur Realisierung der im Bereich *allocatedTo* dargestellten, allokierten Funktionen im Systemverbund beizutragen.

Durch diese Modellierungsmethode werden Anforderungen durchgehend nachverfolgbar untereinander und mit Elementen anderer Aspekte verknüpft. So kann beispielsweise der Übertragungspfad durchgängig von Zielen über technische Anforderungen, erfüllende Funktionen, ausführende Wirkstrukturelemente, abgeleitete weitere Technische Anforderungen, wiederum erfüllende Wirkstrukturen und schließlich physische Strukturen nachvollzogen werden. Selbstverständlich ist dies nur ein Beispiel eines möglichen Übertragungspfades, der in einem realen

System noch deutlich länger werden und sich auch mehrfach verzweigen kann. Die Handhabung genau dieser Komplexität der durchgängigen Nachvollziehbarkeit bzw. Begründbarkeit der Existenz von Anforderungen und entsprechenden Lösungen ist ein zentraler Mehrwert einer modellbasierten Beschreibung.

6.3 Funktionsbasierte Objektsystemmodellierung

Das fachdisziplinübergreifende Modell eines technischen Systems beschreibt die Konzepte und Prinzipien zur Realisierung der geforderten Funktionen, Eigenschaften und Merkmale unter den gegebenen Randbedingungen. Das Ziel gemäß der Forschungshypothese 2 ist, ein derartiges Modell als Kommunikationsgrundlage der fachdisziplinübergreifenden Zusammenarbeit zu etablieren. Dies umfasst neben der für Anwender aller Fachdisziplinen nachvollziehbaren Repräsentation von Informationen in Sichten (Diagrammen) auch die Schaffung einer technischen Basis zur automatisierten, rechnergesteuerten Synchronisation von Informationen zwischen dem zentralen Systemmodell und weiteren rechnerbasierten Modellen der Produktentstehung (allgemein: CAx-Modellen, z.B. CASE, MKS, CAD, CAE, CAO…)[436].

Das im Rahmen dieser Arbeit entwickelte Plug-In für das Modellierungswerkzeug MagicDraw, das neben dem Erweiterungsprofil auch Automatismen zur Erleichterung der Modellierung sowie Modellierungsregeln umfasst, dient vor allem der Synthese neuer technischer Systeme unter Anwendung der Konzepte des Contact & Channel – Ansatzes (C&C²-A)[437]. Dennoch ist auch die retrospektive Modellierung im Rahmen der Analyse technischer Systeme unter Anwendung der hier vorgestellten Modellierungstechnik gleichermaßen möglich.

In den folgenden Kapiteln werden in gleicher Weise wie zuvor die einzelnen zu modellierenden Aspekte kurz vorgestellt. Aufgrund der besseren Erweiterbarkeit werden neben Ausschnitten aus dem hauptsächlich verwendeten Modellierungswerkzeug Artisan Studio der Firma atego auch Beispiele aus dem Modellierungswerkzeug MagicDraw der Firma NoMagic gezeigt.

6.3.1 Modellierung der Aktivitäten von Funktionen

Die SysML verfügt über kein Element mit der Bezeichnung „Function“. Das Modellelement, das der Beschreibung von Funktionen am nächsten kommt, ist die

---

[436] Gemäß der Forschungshypothese 4. Vgl. auch Kapitel 4.1.1

[437] Ohmer (2008) beschreibt in seiner Dissertation Handlungsrahmen und Vorgehensmuster zur werkzeugneutralen Verwendung des C&C²-A bei der Synthese technischer Systeme, die als weiterführende Literatur empfohlen wird.

Metaklasse *Activity*, die aus der Softwaremodellierungssprache UML übernommen wurde. Folglich beschränkt sich der Beschreibungsumfang auf das EVA[438]-Prinzip der Informatik. Im zugehörigen *Activity Diagram*, das bereits zur Modellierung von Ziel-Funktionen (*Target Functions**) als Teil des Zielsystems zum Einsatz kam, können sowohl logische Abläufe als auch Objektflüsse abgebildet werden. Dieser Modellierungsschritt ist der lösungsneutrale Anteil der Funktionsmodellierung, der jedoch nach dem hier zugrundeliegenden Funktionsverständnis eine unvollständige Funktionsbeschreibung darstellt, da der zwingend erforderliche Zusammenhang zur ausführenden Wirkstruktur bisher nicht berücksichtigt wird (vgl. Kapitel 2.1 sowie Kapitel 5.2). Dieses zentrale Defizit der SysML kann teilweise aufgewogen werden, indem die folgenden Aspekte (insbesondere der Wirkstruktur) ebenfalls modelliert werden. Da die beispielhaft in Bild 6-12 gezeigte bisherige Modellierung von logischen Abläufen und Objektflüssen von *Activities* nur unvollständige, aber keine widersprüchlichen oder gar falschen Informationen abbildet, kann dieser „Workaround“ bis zu einer nachhaltigen Weiterentwicklung der SysML angewendet werden. Zur besseren Übersicht wurden logische Abläufe und Objektflüsse für das gezeigte Beispiel „Rekuperation im Schubbetrieb“ auf zwei Diagramme verteilt.

[438] EVA steht für Eingabe – Verarbeitung – Ausgabe, ein Prinzip der Datenverarbeitung

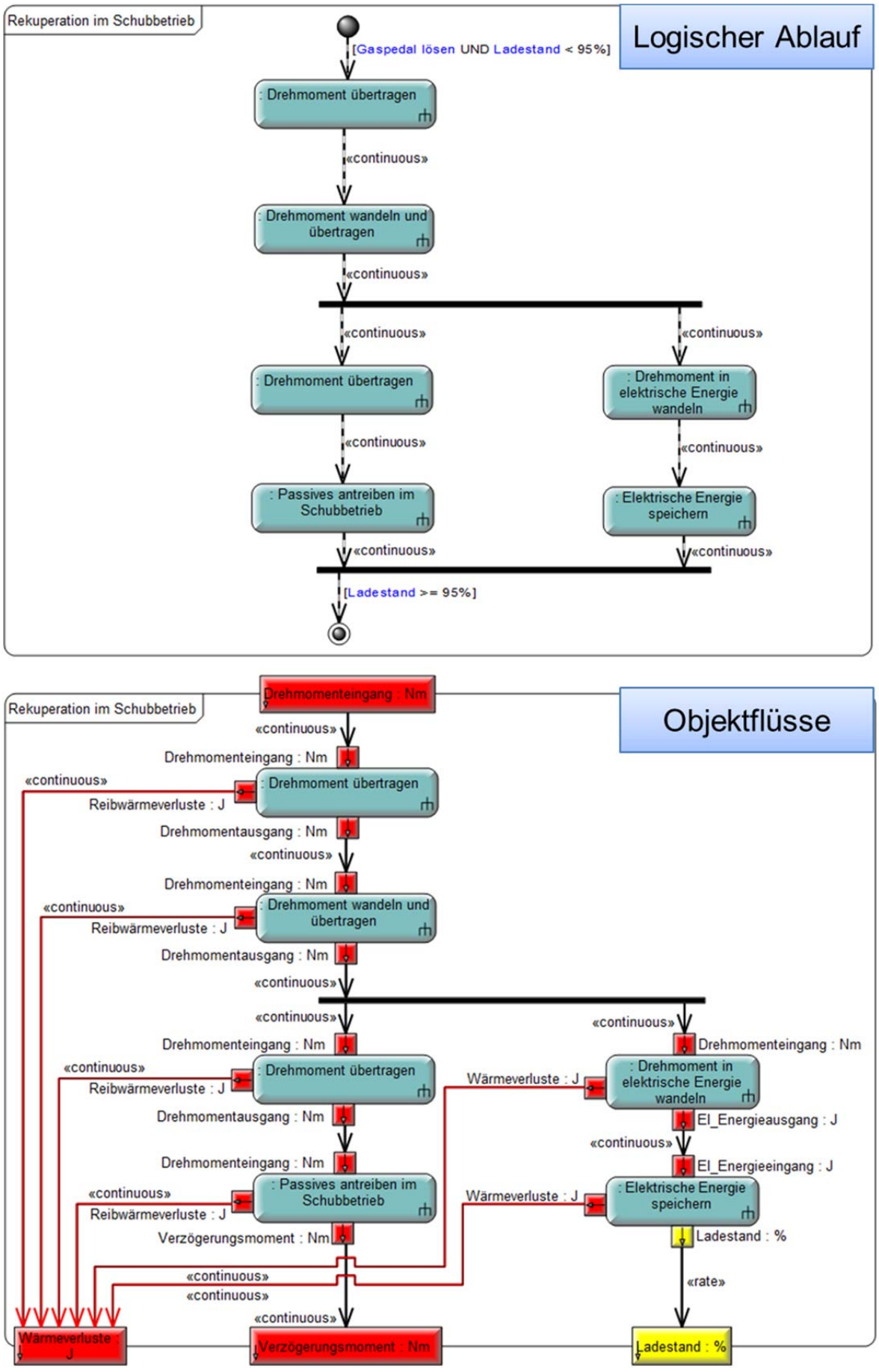


Bild 6-12: Aktivitätsmodellierung im Aktivitätsdiagramm

Activities basieren in der SysML auf Klassen und Rollen[439], weshalb die Klassen (*Activities* selbst) in Bibliotheken abgelegt und in verschiedenen Kontexten als Rolle in Form einer *Call Behavior Action* verwendet werden können. Die *Call Behavior Action* kennt – im Gegensatz zu ihrer Klasse, der *Activity* – neben ihren internen Abläufen auch die Verknüpfung ihrer Schnittstellen mit benachbarten *Activities*, die in *Activity Diagrams* per *Control Flows* bzw. *Object Flows* modelliert werden. Die Bildung von Rollen aus der Klasse einer Funktion (*Activity*) hat den Vorteil, dass die Möglichkeit besteht, Basisfunktionen in Form von elementaren *Activities* in Bibliotheken anzulegen und wiederverwenden zu können[440]. Dieses Prinzip funktioniert jedoch nicht für Funktionen höheren Abstraktionsgrades, da deren Teilfunktionen auf unterschiedliche Weisen technisch realisiert werden können. Beispielsweise haben sowohl Wankelmotor als auch Hubkolbenmotor die Hauptfunktion, chemische Energie in Drehmoment zu wandeln, lösen dies jedoch durch teilweise völlig unterschiedliche Subfunktionen.

Bilden komplexe Funktionen mehrere Wirknetze mit individuellem internen Ablauf aus, können diese durch eigene *Activity Diagrams* mit den entsprechenden Rollen (*Call Behavior Actions*) modelliert werden. Die Summe aller Diagramme bildet die Summe der Aktivitäten der Wirkstruktur einer Funktion.

Activities können sowohl diskret als auch kontinuierlich ausgeführt werden. Dabei wird die Unterscheidung darin getroffen, ob die *Activity* durch eine interne oder externe Ursache (z.B. Trigger, Überschreitung eines Grenzwertes durch einen Input) beendet wird. Dieser Sachverhalt wird am Beispiel der Funktionen eines Sensors veranschaulicht (Bild 6-13).

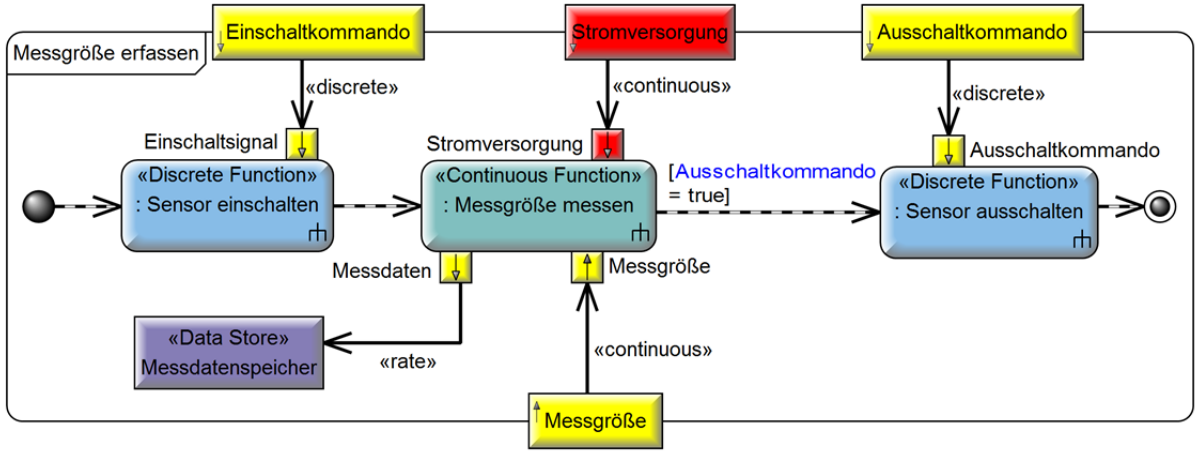


Bild 6-13: Unterscheidung diskreter und kontinuierlicher Funktionen (Aktivitäten)

[439] Vgl. Kapitel 2.5.1
[440] So geschehen bspw. in Kruse et al. (2012)

Die in der Abbildung modellierte diskrete Funktion (erweiterter Stereotyp *Discrete Function**) „Sensor einschalten“ wird durch den diskreten Input „Einschaltkommando“ aktiviert. Sobald der Einschaltvorgang abgeschlossen ist, wird die Funktion beendet und zu der kontinuierlichen Funktion (erweiterter Stereotyp *Continuous Function**) „Messgröße messen“ übergegangen. Diese misst solange eine Messgröße und bezieht aus dem kontinuierlichen Input „Stromversorgung“ seine Energie, bis sie von extern über ein „Ausschaltkommando“ beendet wird. Diese Bedingung für die Beendigung einer kontinuierlichen Funktion ist hier als *Guard Condition* des durch einen gestrichelten Pfeil dargestellten Kontrollflusses (*Control Flow*) modelliert. Diese repräsentieren den logischen Ablauf der Funktionen.

Folglich werden kontinuierliche Funktionen solange ausgeführt, bis eine nachfolgende diskrete Funktion ausgelöst wird. Diskrete Funktionen beenden sich selbst nach Vollendung ihrer Verarbeitungsschritte und benötigen dafür keinen externen Auslöser. Ein weiteres Unterscheidungsmerkmal der beiden Funktionsarten sind Systemzustände. Eine diskrete Funktion wird ausschließlich zur Beschreibung des Wechsels von Zuständen (*Transition*) verwendet, während kontinuierliche Funktionen immer innerhalb eines Systemzustandes ausgeführt werden. Daher ist es zwingend zur Modellierung von Funktionen erforderlich, auch die Systemzustände und deren Übergänge zu modellieren, was im folgenden Kapitel vorgestellt wird.

6.3.2 Modellierung von Systemzuständen

Technische Systeme verfügen immer über mehrere Zustände. Die kleinste mögliche Menge ist hier die Unterscheidung zwischen „Funktionserfüllung“ und „keine Funktionserfüllung“ (z.B.: „ein“ und „aus“ oder für eine Axialkolbenpumpe: Förderbetrieb“ und „Stillstand“)[441]. Zustände sind daher ein zentrales Konstrukt der Verhaltensbeschreibung technischer Systeme und damit deren Funktionen. Zur Zustandsmodellierung bietet die SysML einen eigenen Diagrammtyp an: das *State Diagram*. Bild 6-14 zeigt ein Beispiel auf Basis der in Bild 6-13 modellierten Funktionen eines einfachen Sensors. Neben der Modellierung der Zustände und Zustandsübergänge wurden auch die diskreten und kontinuierlichen Funktionen des Sensors zugeordnet.

[441] Vgl. Matthiesen und Ruckpaul (2012)

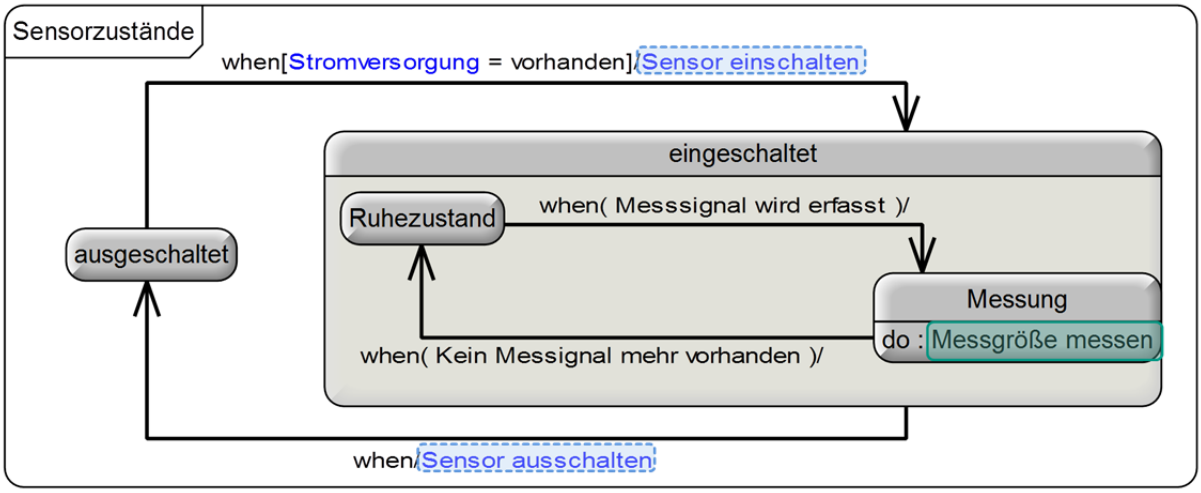


Bild 6-14: Zustände im *State Diagram*

Die Zuordnung der Funktionen erfolgt im Modellierwerkzeug per Drag & Drop. Die zugeordneten diskreten Funktionen des Sensorbeispiels sind gestrichelt, die kontinuierliche Funktion mit einer durchgezogenen Umrandung hervorgehoben. Durch diese Modellierungsmethode ist es möglich, eine Überprüfungsregel zu implementieren, die sicherstellt, dass kontinuierliche Funktionen nur innerhalb eines Zustandes (*States*) und diskrete Funktionen nur in Zustandsübergängen (*State Transitions*) abgelegt werden. Zustände können, wie in der Abbildung zu sehen, auch verschachtelt modelliert werden, um bspw. Teilsystemzustände oder gar parallel eingenommene Zustände verschiedener Teilsysteme zu modellieren. Die zugeordneten Funktionen werden als reiner Text im Diagramm dargestellt.

Der Contact & Channel – Ansatz berücksichtigt ebenfalls Systemzustände, da sie zur Beschreibung und Abgrenzung dynamisch veränderlicher Strukturen beitragen[442]. Konkret bedeutet dies, dass abhängig vom Systemzustand auch die Anordnung der Elemente seiner Wirkstruktur zur Funktionserfüllung und damit letztlich auch der physischen Struktur ändern kann. Dabei werden entweder neue Wirkflächenpaare gebildet oder bestehende aufgelöst. Hierzu ist es erforderlich, Wirkflächenpaaren zwischen Leitstützstrukturen (bzw. den resultierenden Komponenten), die nur in bestimmten Zuständen existieren, eine entsprechende Eigenschaft zuzuordnen. So können Systemstrukturen auf die Möglichkeit zur Ausübung einer Funktion in Abhängigkeit von ihrem Zustand überprüft werden. Konkret bedeutet dies, dass für einen Objektfluss, der in einem *Activity Diagram* als Teil einer Funktion beschrieben wurde, auch die entsprechende Schnittstelle (in SysML: der *Connector* zwischen zwei *Ports*) im zugewiesenen Zustand vorhanden ist. Beispielsweise kann eine

[442] Vgl. bspw. Albers et al. (2008c), Matthiesen und Ruckpaul (2012), Albers und Sadowski (2013)

Kupplung nur dann Drehmoment übertragen, wenn sie sich auch im Zustand „geschlossen“ befindet. Die Modellierung dieser zustandsabhängigen Existenz von Schnittstellen (Wirkflächenpaaren) wird im Rahmen der Modellierung der Wirkstruktur in Kapitel 6.3.4 mittels einer Erweiterung der SysML vorgenommen. Das folgende Kapitel stellt einen strukturierenden Zwischenschritt vor der Ableitung von Wirkstrukturen aus den modellierten Aktivitäten und Zuständen vor.

6.3.3 Gruppierung von funktionalen Einheiten

Bisher wurden nur die Abläufe von Funktionen in Activity Diagrams modelliert, jedoch bisher keine funktionsrelevanten Merkmale und Eigenschaften berücksichtigt. Dies erfolgt in der Wirkstrukturmodellierung, für welche der nun vorgestellte Modellierungsschritt eine Vorarbeit in Form der Strukturierung in funktionale Einheiten darstellt. Dieser Schritt ist im Grunde optional, da prinzipiell einfach für jede *Activity* und ihre *PIN*'s (die Schnittstellen für Objektflüsse) direkt eine entsprechende Leitstützstruktur mit Wirkflächen angelegt werden könnte. Durch die nachfolgend gezeigte Vorstrukturierung erhält der Modellierer jedoch die Freiheit, Teilfunktionen zu gruppieren, um die Betrachtungstiefe flexibler handhaben zu können. Diese Maßnahme ist beispielsweise sinnvoll, wenn mehrere Softwarefunktionen zusammengefasst werden sollen oder Strukturen nur auf einem geringeren Detaillierungsgrad in Form von Baugruppen abgebildet werden müssen.

Diese Vorstrukturierung erfolgt durch die Bildung von funktionalen Gruppen mittels *Activity Partitions* in *Activity Diagrams*, wie sie auch in der FAS-Methode nach LAMM UND WEILKIENS erfolgt[443]. Bild 6-15 zeigt diese Gruppierung in funktionale Einheiten am Beispiel der Funktion „Rekuperation im Schubbetrieb“ durch eine Zuordnung von *Activity Partitions*. Da im gezeigten Beispiel zu erwarten ist, dass sämtliche Teilfunktionen später in unterschiedlichen Komponenten realisiert werden, wurde für jede *Activity* eine eigene *Activity Partition* angelegt.

[443] Vgl. Lamm und Weilkiens (2010)

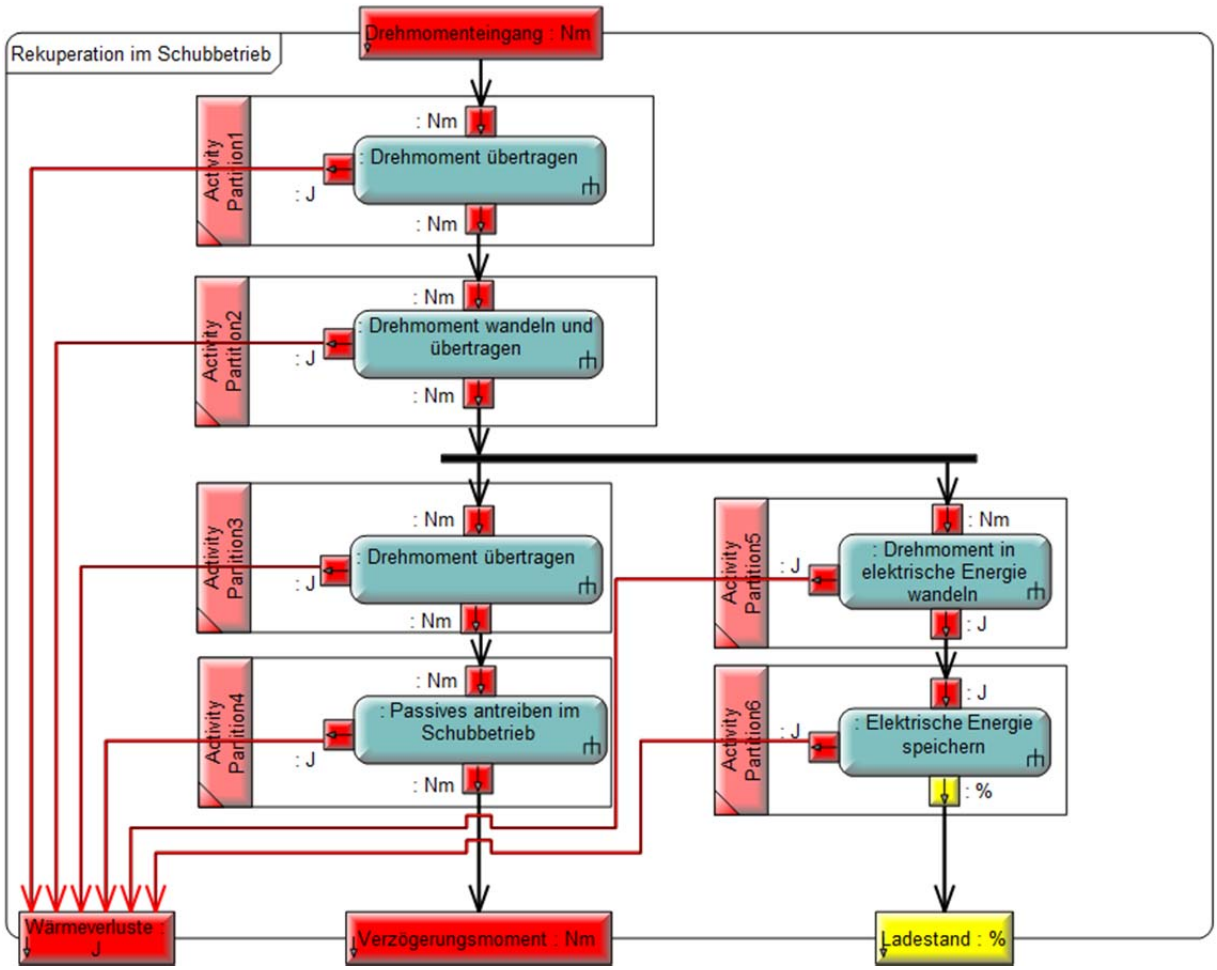


Bild 6-15: Funktionale Einheiten im *Activity Diagram*

Nachdem die Gruppierung in funktionale Einheiten vorgenommen wurde, verfügt das Modell automatisch über sämtliche relevanten Informationen, um Leitstützstrukturen (LSS) und Wirkflächen (WF) anzulegen. Da sich im Verlauf der Arbeit herausstellte, dass die Implementierung eines entsprechenden Automatismus für diesen Modellierungsschritt sich als nicht ohne Support des Toolherstellers machbar erwies, wurde dies in einem anderen, weit verbreiteten Modellierungswerkzeug mit Java-Schnittstelle realisiert, dem Werkzeug MagicDraw der Firma NoMagic. Aufgrund der sehr offenen Schnittstelle[444] findet dieses Werkzeug sehr hohe Verbreitung in der Forschungscommunity der INCOSE. Ein erster Prototyp dieser kontinuierlich in der Weiterentwicklung befindlichen Implementierung wurde von ALBERS UND ZINGEL vorgestellt[445].

Im nächsten Kapitel wird die Wirkstruktur auf Basis der nun vorgenommenen funktionalen Partitionierung abgeleitet und ausmodelliert.

[444] Application programming interface
[445] Albers und Zingel (2013b)

6.3.4 Ableitung und Ausmodellierung der Wirkstruktur

Die Gruppierung der *Activities* in *Activity Partitions* wird dazu verwendet, LSSen (in SysML *CSS**, steht für *Channel and Support Structure*) und WF (in SysML *WS**, steht für *Working Surface*) automatisiert abzuleiten.

Ein als *CSS** stereotypisierter *Block* kann in Abweichung zur Leitstützstruktur gemäß der Definition im C&C²-A mehr als 2 Wirkflächen erhalten, wodurch es sich streng genommen nicht um eine einzelne LSS, sondern um eine Gruppe von LSS handelt. Diese explizite Abweichung erst ermöglicht die Gruppierung und Dekomposition von Wirkstrukturen über mehrere Ebenen, da gerade Wirkelemente geringeren Detaillierungsgrades meist über zahlreiche Wirkflächen mit Nachbarsystemen interagieren. Eine stringente Implementierung von LSS **mit genau 2 WF** in SysML wäre ein enormer technischer Aufwand, der im Rahmen dieser Arbeit nicht möglich war. Der besseren Lesbarkeit geschuldet wird der stereotypisierte Block fortwährend dennoch als Leitstützstruktur bezeichnet.

Daraus können dann mit geringem Aufwand Wirkstrukturen durch Vernetzung der generierten Elemente modelliert werden. Hierzu werden die Blockdiagramme der SysML verwendet. Bild 6-16 zeigt die Realisierung des obigen Beispiels der Rekuperation im Schubbetrieb in einem als *iwsd** (Internes Wirkstrukturdiagramm) stereotypisierten *Internal Block Diagram*.

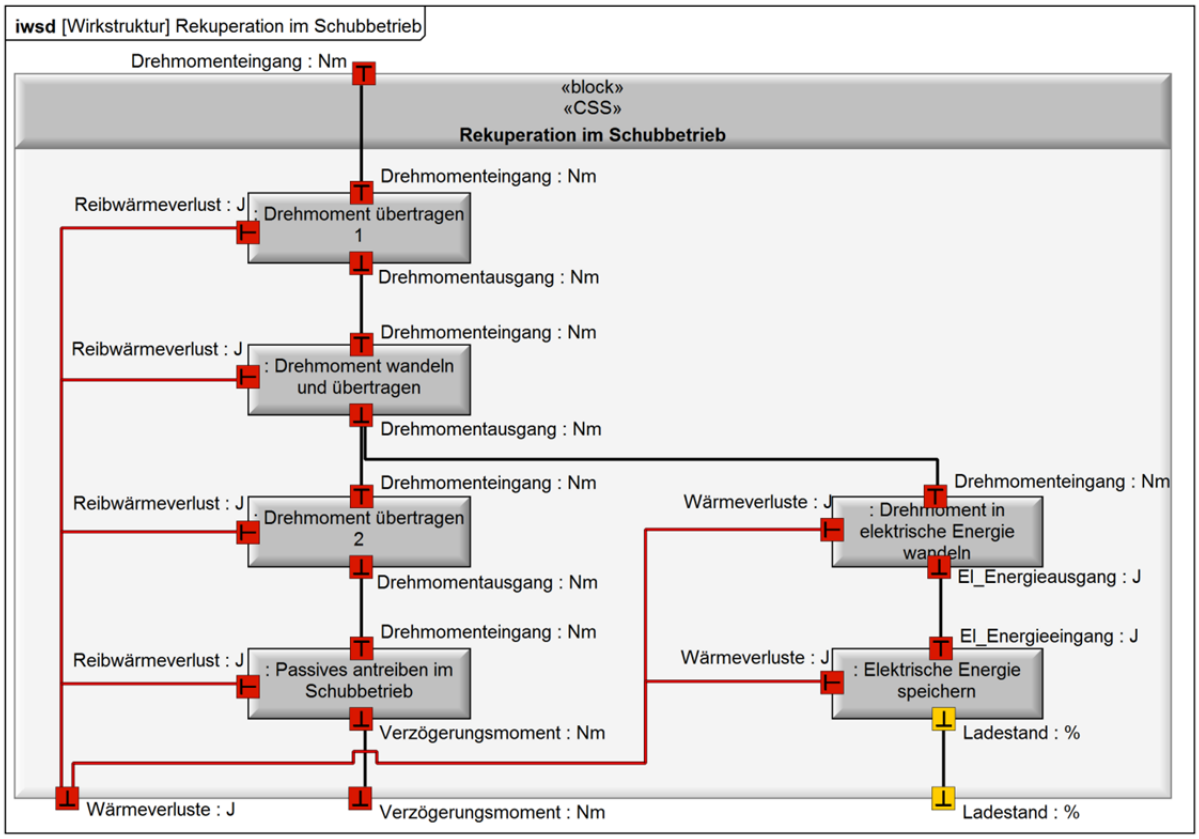


Bild 6-16: Wirkstruktur im *Internal Block Diagram*

Das Diagramm zeigt die Wirkstruktur der Funktion „Rekuperation im Schubbetrieb“. Betrachtet man einen Hybridantriebsstrang insgesamt, würde „Rekuperation“ nur eines von mehreren Wirknetzen der Hauptfunktion „Antriebsenergie bereitstellen“ darstellen (vgl. Kapitel 2.4.9, darin Bild 2-16)[446].

Im vorliegenden Evaluationsprojekt war der Zweck der retrospektiven Modellierung die Analyse der Rekuperationsfunktion für ein existierendes Hybridfahrzeug. Daher wurde bspw. auch die Anwendungsfallstruktur (vgl. Bild 6-5) von vornherein anders strukturiert. Weiterhin waren aus diesem Grund auch die realen Komponenten und damit die entsprechenden, an der Funktion beteiligten Wirkelemente (LSS und WFP) bereits bekannt. Die in Bild 6-16 gezeigte Wirkstruktur der Funktion „Rekuperation im Schubbetrieb“ besteht aus den LSS der daran partizipierenden Subfunktionen, ihren WF und deren Verbindungen durch *Connectors*, wodurch WFP gebildet werden. Zur besseren Übersicht wurden die *Connectors* der Wärmeverluste – wie auch im zuvor gezeigten *Activity Diagram* deren *Object Flows* – rot eingefärbt, um sie vom Drehmomentfluss optisch abzuheben. Diesen LSS können nun weitere funktionsrelevante Merkmale und Eigenschaften durch Parameter (in SysML: *Block Property (Value)*) zugewiesen werden.

Eine weitere fundamentale Erweiterung der SysML neben der Einführung von Elementen zur Modellierung von Wirkstrukturen, bestehend aus LSS und WFP, ist die Erweiterung der Eigenschaften von Wirkflächenpaaren hinsichtlich ihrer Dynamik. Die SysML ist von Hause aus nur in der Lage, statische Strukturen zu beschreiben und darzustellen. Im Rahmen dieser Arbeit wurde hierzu als Teil des bereits erwähnten Plug-In’s eine Erweiterung des SysML-Elements *Connector* durch den Stereotyp *CCAConnector** vorgenommen. Er verfügt über das Attribut *ExistsFor**, durch das er Zustände zugewiesen bekommen kann, für den das durch ihn modellierte WFP überhaupt existieren soll. Dieser Aspekt ist von zentraler Bedeutung für die Modellierung dynamischer Systeme, da erst sie eine Verifikation des Modells hinsichtlich der Existenz der für die Ausführung einer bestimmten Funktion erforderlichen WFP’s ermöglicht. Befindet sich beispielsweise ein System in einem bestimmten Zustand (z.B. ein Getriebe in „Gang 1 geschaltet“), so kann mit dieser Erweiterung geprüft werden, ob die Systemstruktur für diesen Zustand in der Lage ist, die entsprechenden Kraftflüsse im Getriebe zu übertragen, die eine konkrete

[446] Die SysML-Implementierung der Konzepte des C&C²-A sieht keine explizite Unterscheidung der Diagramme in Wirknetze und Wirkstrukturen vor, jedoch besteht die Möglichkeit, Diagramme zur Darstellung von Ausschnitten der gesamten Wirkstruktur zu gebrauchen. Daher kann ein Diagramm je ein Wirknetz einer Funktion in einer bestimmten Ausprägung darstellen. Die Verknüpfung, welcher Testfall jedoch genau diese Wirkstruktur auslöst, ist derzeit noch nicht möglich.

Funktion fordert. Da diese Erweiterung ebenfalls im Werkzeug „MagicDraw“ implementiert wurde, zeigt Bild 6-17 ein Beispieldiagramm der Wirkstruktur eines Zweiganggetriebes für beide Gänge inklusive der (farblich hervorgehobenen) Zuweisung der Zustände, in denen die Schnittstellen existieren. Das Getriebe repräsentiert die interne Wirkstruktur von „Drehmoment übertragen und wandeln“ aus Bild 6-16. Zur besseren Übersicht wurde ein existierendes Getriebe mit nur zwei Gängen gewählt und wie auch der Hybridantriebsstrang des übergeordneten Beispiels retrospektiv modelliert. Daher waren auch alle LSSen bekannt, was die Beschreibung mehrerer Iterationen bei der Entwicklung der finalen Struktur vermeidet, die ansonsten den Rahmen dieser Arbeit sprengen würden.

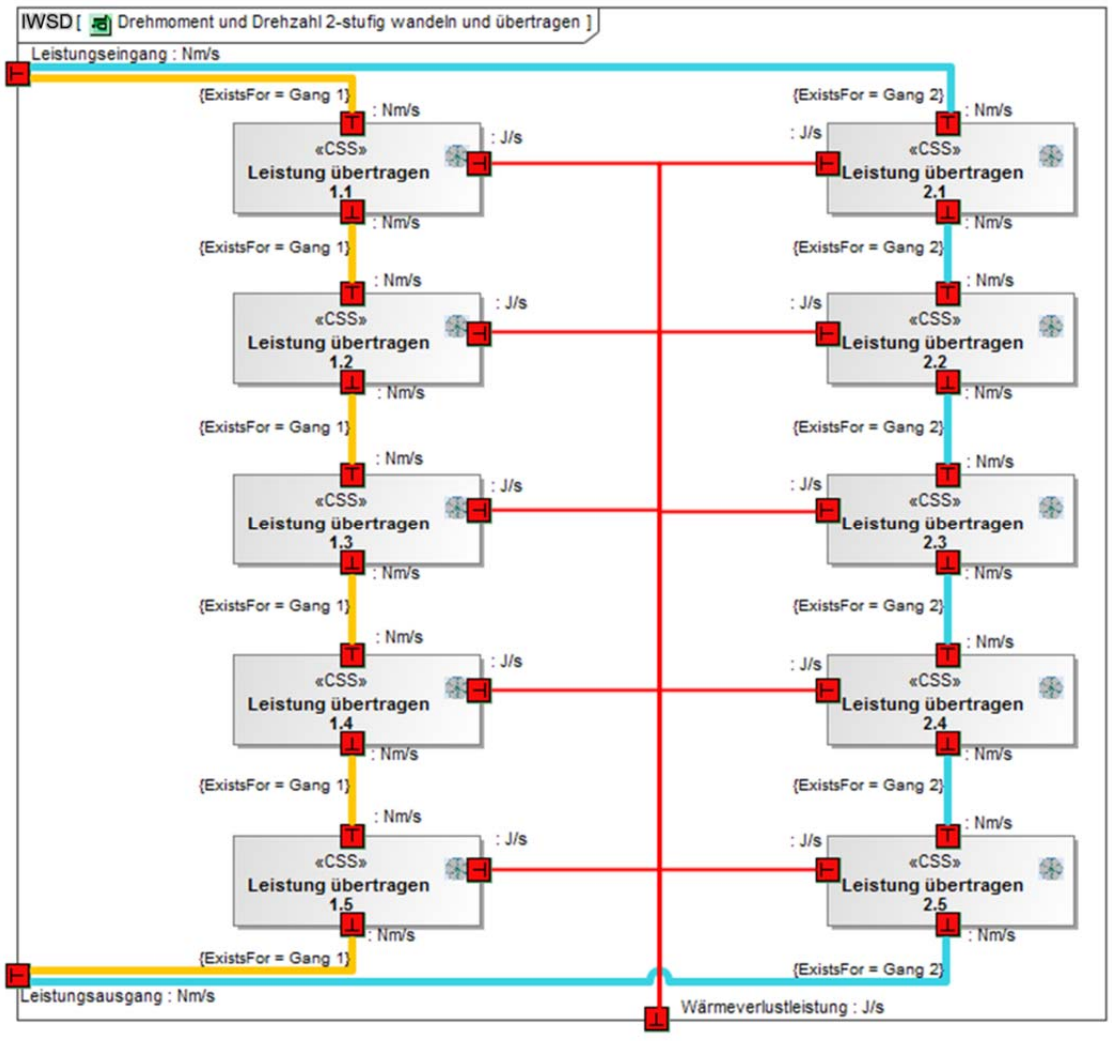


Bild 6-17: Zustandsbasierte Wirkstruktur im *Internal Block Diagram*

Die gezeigten Leitstützstrukturen (grau) übertragen die Leistung über Wirkflächenpaare (rot). Die Funktion verfügt aufgrund der beiden Gänge über zwei Wirknetze mit je fünf aktiven LSS, da hier von den später zuzuweisenden Komponenten „Eingangswelle“, „Schaltmuffe“, „Zahnrad 1“, „Zahnrad 2“ und „Ausgangwelle“ ausgegangen wird. Hier wurden beide Wirknetze der Funktion

gemeinsam in einem Diagramm, also folglich die gesamte Wirkstruktur dargestellt. Die Leitstützstrukturen der gezeigten Wirkstruktur werden durch Konstruktionsheuristiken wie räumliche Integration von Funktionen auf die resultierenden physischen Komponenten überführt, was in Kapitel 6.3.5 vorgestellt und erläutert wird.

Erst die Kombination der beiden Diagramme (*Activity Diagram* und *Internal Wirkstructure Diagram* bzw. *IWSD**) und deren Vernetzung im Modell ist in der Lage, den Funktions-Gestalt-Zusammenhang nach dem Verständnis des Contact & Channel – Ansatzes in SysML zu beschreiben. Dieser ergibt sich aus der Beschreibung der Objektflüsse aus Stoff, Energie und Information und deren Verarbeitung einerseits (*Activity Diagram*) und der Beschreibung der ausführenden Wirkstruktur aus LSS und WFP mit funktionsrelevanten Merkmalen und Eigenschaften andererseits (*IWSD**). Die Modellierung von naturwissenschaftlichen Effekten ist insofern möglich, als dass zumindest die parametrischen Zusammenhänge in Zusicherungsdiagrammen abgebildet werden können (vgl. Kapitel 6.3.6). Die Darstellung der Abhängigkeiten in SysML unterliegt jedoch Einschränkungen. So ist es beispielsweise nur möglich, die Zuweisung von LSS auf *Activities* in einer Darstellung zu zeigen (siehe graue Flächen in Bild 6-18: zugewiesene *CSS**), die Schnittstellen (WF) und Objektflüsse sind nicht gemeinsam darstellbar. Hier wird derzeit am IPEK an einer intuitiveren Darstellung geforscht.

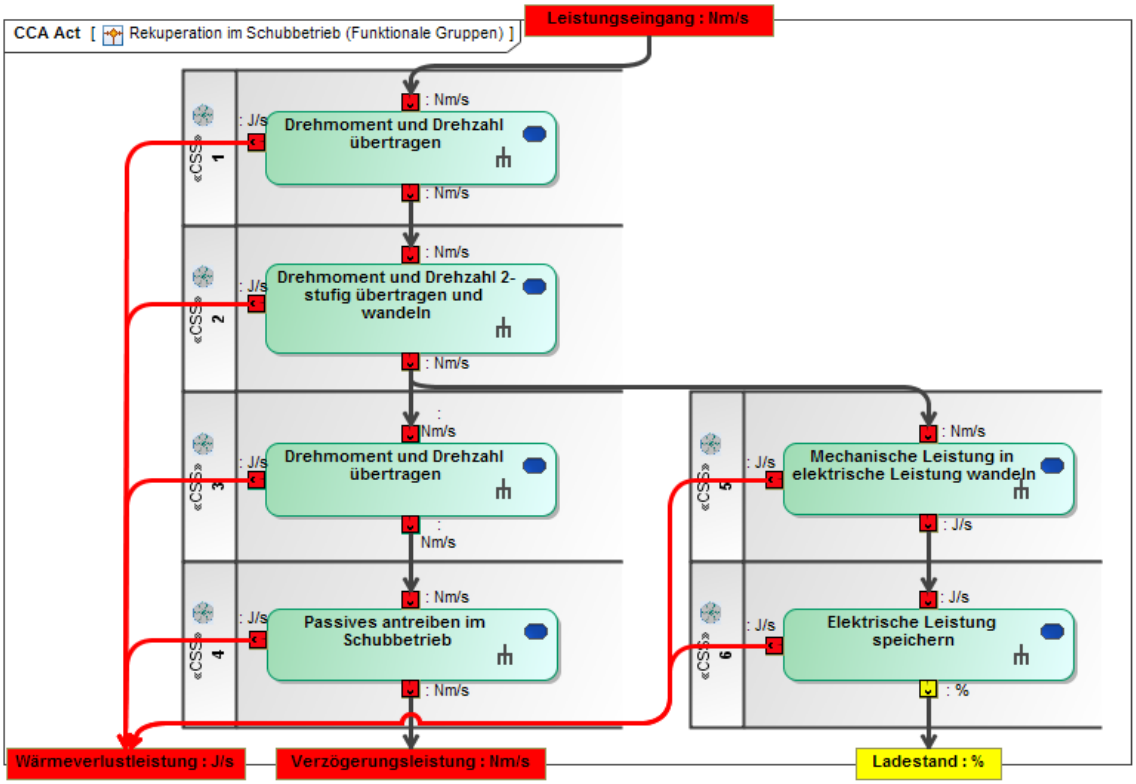


Bild 6-18: Funktionale Systemarchitektur im *Activity Diagram*

Die hier gezeigte Zuweisung der *CSS** zu den *Activity Partitions* wurde wie auch deren Erzeugung durch ein Plug-In automatisiert. Dieses ist auch in der Lage, Wirkstrukturen über mehrere Hierarchieebenen hinweg automatisch aus *Activity Diagrams* abzuleiten[447]. Folglich können gleichzeitig mehrere Detaillierungsebenen von *Activities* modelliert und deren ausführende Wirkstrukturen generiert werden.

6.3.5 Ableitung der physischen Struktur aus der Wirkstruktur

Die Wirkstruktur ist ein Werkzeug zur expliziten Modellierung von funktionsrelevanten Strukturelementen sowie ihrer Merkmale und Eigenschaften. Die resultierende physische Struktur ergibt sich jedoch zudem aus Randbedingungen wie z.B. Fertigungs- oder Montagerestriktionen. Erst diese Trennung ermöglicht die Wahrung eines wesentlichen Freiheitsgrads in der Konstruktion von Produkten: die Anwendung von Konstruktionsheuristiken wie der funktionalen Integration oder der Trennung der Funktionen (Modularisierung). Hierzu ist es erforderlich, den Entwicklern der jeweiligen Fachabteilung jene Strukturelemente an die auszukonstruierenden Systemkomponenten zu übermitteln, die für die Ausübung ihrer zugewiesenen Funktionen erforderlich sind. Daraufhin erfolgt die eigentliche Konstruktionsarbeit durch die Ausgestaltung der physischen Struktur bis hin zur finalen Gestalt. Hierbei kommen fachspezifische Werkzeuge wie CAD-Modelle zum Einsatz. Die bereits in SysML modellierten Informationen können durch Synchronisation von SysML und CAD übermittelt werden, was zum heutigen Zeitpunkt jedoch technisch nur in ersten Protoptypen realisiert wurde (vgl. Kapitel 6.5.2).

Das Mapping von Wirkstrukturelementen auf physische Komponenten erfolgt in der Modellierung mit SysML folgendermaßen: *Physical Components** ***erben*** alle funktionsrelevanten Merkmale, Eigenschaften sowie Wirkflächen der ihnen zugewiesenen Wirkstrukturelemente. Diese Funktionalität wurde im Rahmen dieser Arbeit als Teil der Erweiterung der SysML implementiert. Eine Matrix zur Gegenüberstellung von LSSen und Komponenten erleichtert die Zuordnung durch einfaches „anhaken“ (siehe Bild 6-19).

[447] Vgl. Albers und Zingel (2013b)

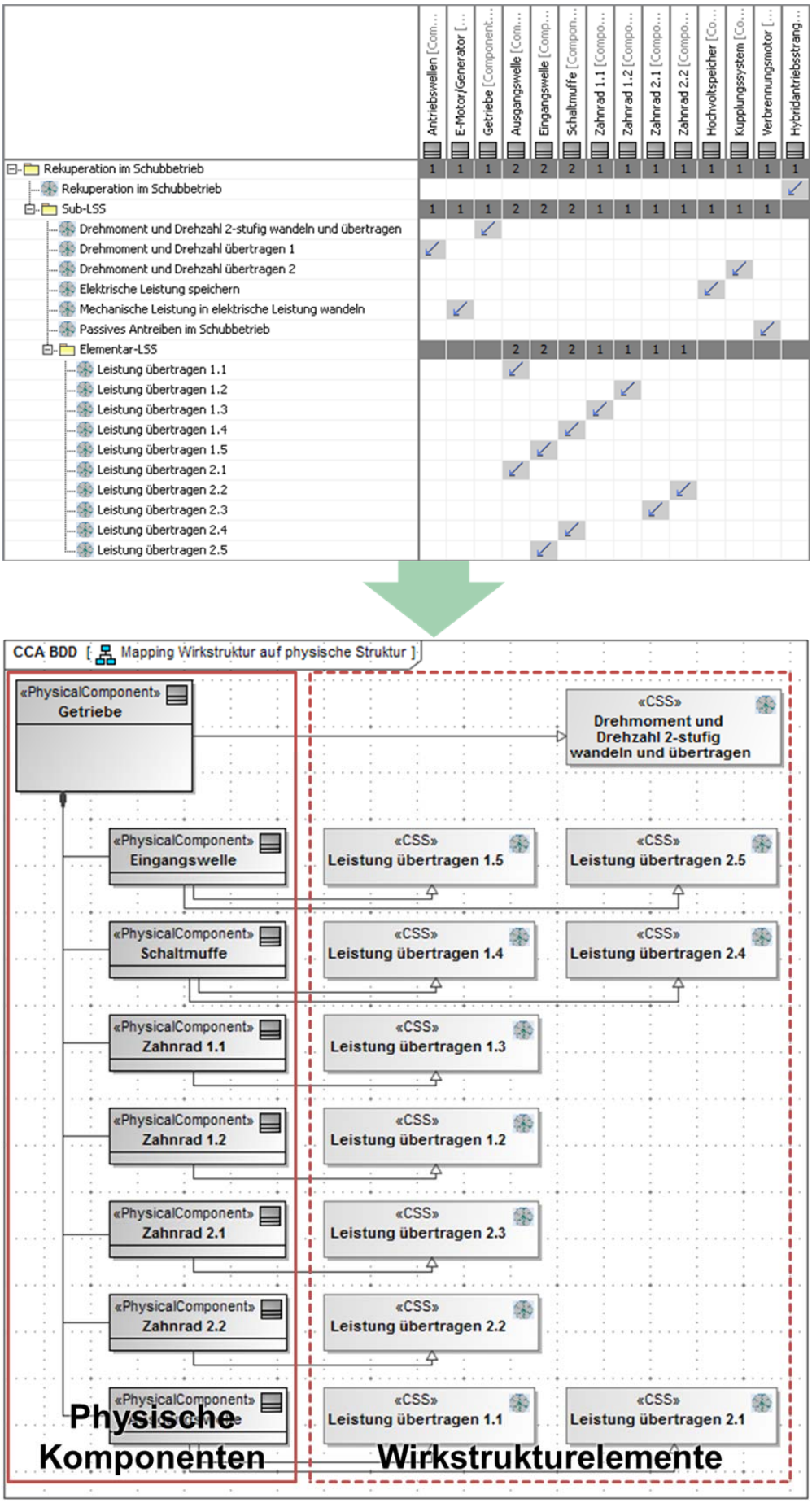


Bild 6-19: Vererbung der Wirkstrukturelemente auf physische Strukturen

Die Zeilen der Matrix in Bild 6-19 (oben) stellen die LSSen (*CSS**) dar, die Spalten die physischen Komponenten (*PhysicalComponent**). Die Pfeile stehen für die Zuordnung von LSSen (deren Wirkflächen, Merkmale und Eigenschaften sind darin enthalten, hier jedoch nicht dargestellt) auf Komponenten und stehen folglich operativ für eine Vererbungsbeziehung von LSS zu Komponente. Wie in der Matrix zu sehen, wurden den beiden Wellen und der Schaltmuffe zwei LSSen zugewiesen. Aus diesem Modellierungsschritt ergibt sich nun die physische Struktur. Das hierfür ausmodellierte als *IPSD** (Internal Physical Structure Diagram) stereotypisierte *Internal Block Diagram* ist in Bild 6-20 zu sehen.

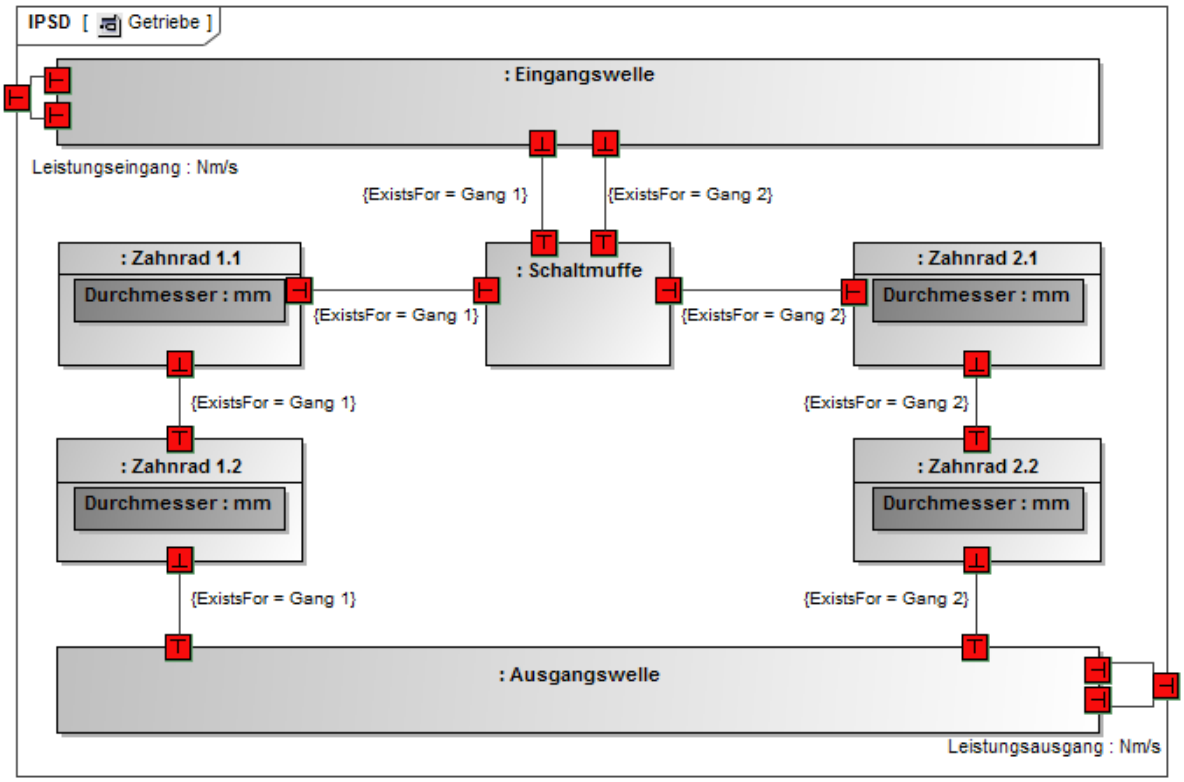


Bild 6-20: Physische Struktur im *Internal Block Diagram*

In der Abbildung werden beispielhaft die für die Funktion „Wandlung von Drehmoment und Drehzahl“ relevanten Merkmale (Radien der Zahnräder) sowie die zustandsabhängigen WFP jeweils mit ihrem Attribut *ExistsFor** angezeigt.

Die Strukturdiagramme sind die von Anwendern der SysML am häufigsten eingesetzten[448], weshalb hier noch ergänzend hinzugefügt wird, dass die Diagramme durch den zusätzlichen Einsatz von Bildern und Grafiken optisch noch verständlicher für Diskussionen der Systemkonzepten aufbereitet werden können. Ein Beispiel zeigt

[448] Wie beispielsweise die Umfrage von Albers und Zingel (2013a) belegte.

Bild 6-21, worin eine CAD-Schnittdarstellung des Getriebes als Hintergrundbild und eine leichte Anpassung der Abmessungen der Modellelemente sowie die farbliche Hervorhebung der WFP bereits einen großen Effekt bewirken.

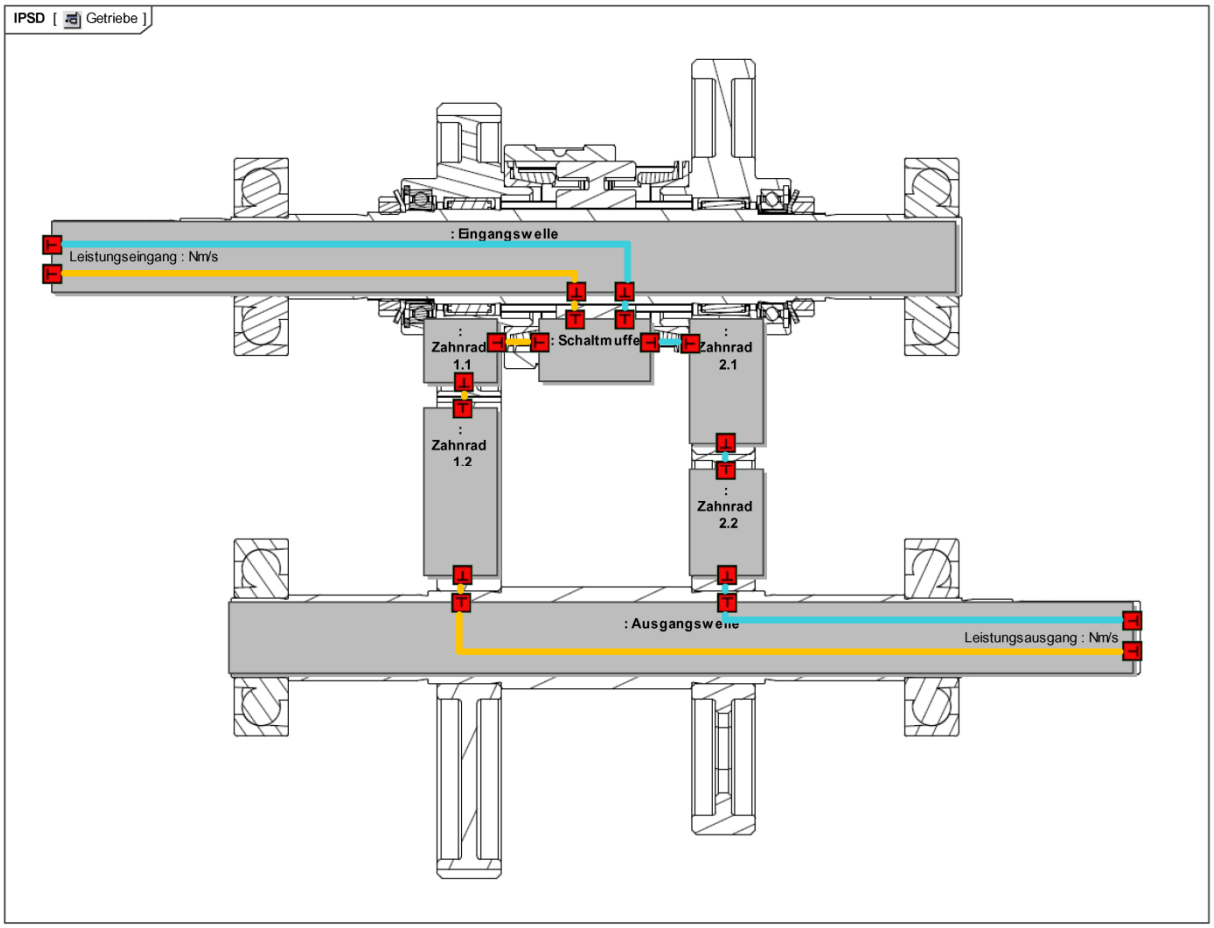


Bild 6-21: Optisch aufbereitete physische Struktur im *Internal Block Diagram*

In diesem Diagramm wurde ein besonderes Augenmerk darauf gelegt, den Kraftfluss der beiden Wirknetze (je eines für Gang 1 und Gang 2) der Funktion „Wandlung von Drehmoment und Drehzahl“ farblich analog Bild 6-17 hervorzuheben. Dafür wurden verschiedene Elemente wie Parameter und Merkmale für diese Sicht ausgeblendet. Die Vision in Bezug auf die Modellierung von Wirkstrukturen und physischen Strukturen ist, in Zukunft auch eine direkt gestaltbehaftete Darstellung in einem Systemmodell aus CAD-Modellen von Vorgängergenerationen heraus generieren und die physische Struktur auf die Wirkstruktur abstrahieren zu können. Dies ist Bestandteil derzeitiger Forschung am IPEK.

6.3.6 Modellierung von Zusicherungen

Die Modellierung mancher Aspekte wie beispielsweise die Beschreibung von Effekten (vgl. Kapitel 6.3.4) erfordert die Beschreibung und Darstellung von Bedingungen bzw. Abhängigkeiten. Hierzu bietet die SysML ein Element zur Modellierung von parametrischen Abhängigkeiten: die Zusicherung (*Constraint*).

Ähnlich wie Blöcke können auch *Constraints* als Klasse angelegt werden und dann in Strukturdiagrammen wie dem *Internal Block Diagram* oder auch im *Constraint Diagram* in einem konkreten Kontext instanziiert werden. Somit ist es möglich, parametrische Zusammenhänge von funktionsrelevanten Merkmalen bzw. Eigenschaften (hier: *Block Properties (Values)*) von *CSS** oder *WSP** zu modellieren. Dies wurde im vorliegenden Beispiel für die Funktion „Wandlung von Drehmoment und Drehzahl“ getan, um die Übersetzungsverhältnisse der beiden Wirknetze für Gang 1 und Gang 2 im *IWSD** abzubilden (siehe Bild 6-22).

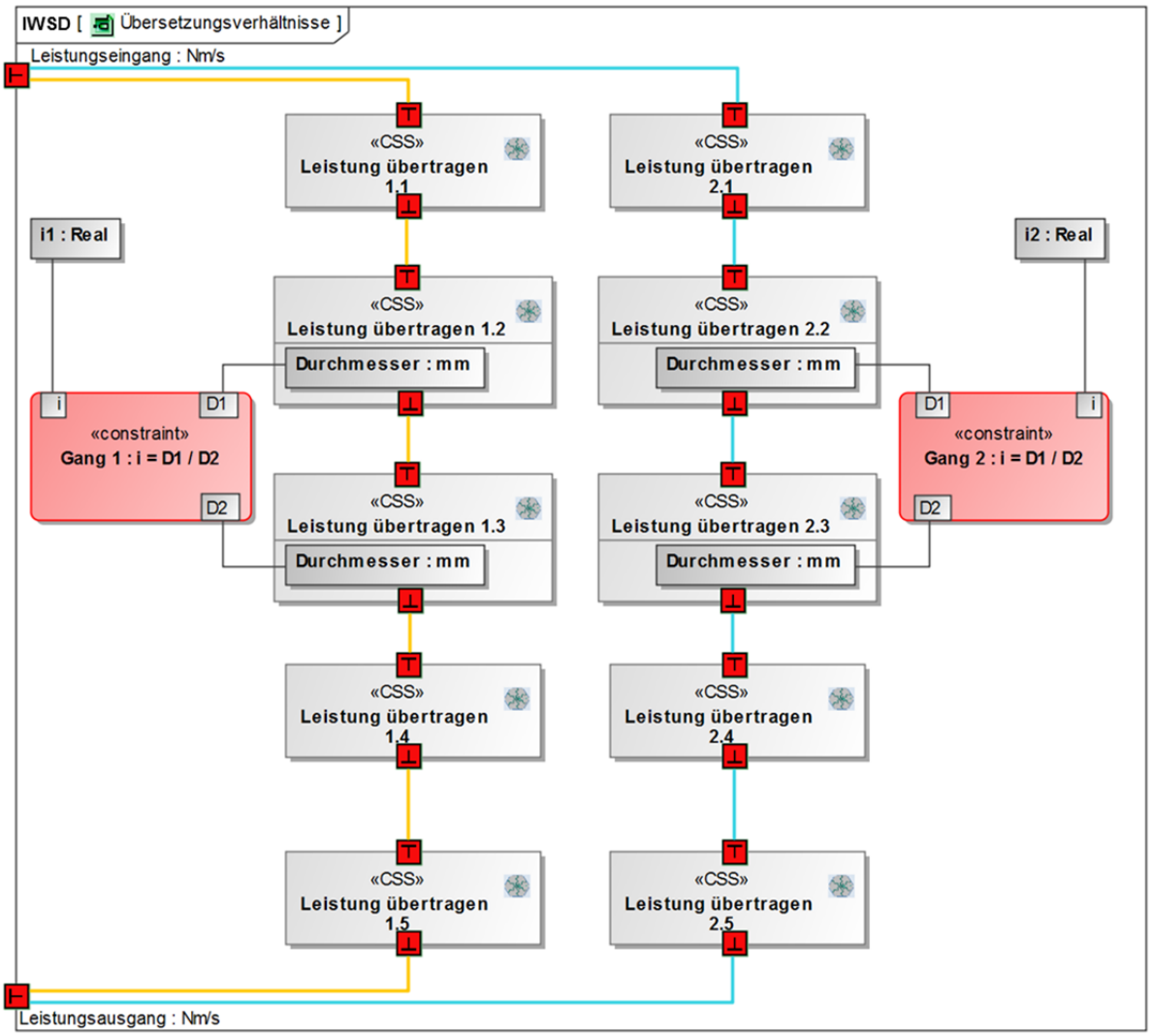


Bild 6-22: Parametrische Zusammenhänge im *Internal Wirkstructure Diagram*

Die *Constraints* verbinden die Parameter der LSSen miteinander und beschreiben deren Zusammenhang. Die Namen der Parameter der *Constraints* können aufgrund ihrer universellen Einsatzbarkeit für Übersetzungsverhältnisse jeder Art von jenen der in Relation gesetzten Parameter der LSSen abweichen, was die Bedeutung jedoch nicht beeinträchtigt. Durch diese Maßnahme wurde im Modell abgebildet, dass sich die Funktion „Wandlung von Drehmoment und Drehzahl“ erst aus der

Interaktion der LSSen „Kraft übertragen 3“ und „Kraft übertragen 4“ im Wirknetz von Gang 1 bzw. „Kraft übertragen 8“ und „Kraft übertragen 9“ im Wirknetz von Gang 2 ergibt. Dies begründet gleichzeitig, welche Bedeutung Effekten bei der konstruktiven Realisierung von Funktion zugutekommt und wie diese mit den funktionsrelevanten Merkmalen und Eigenschaften der ausführenden Strukturelemente korrelieren. Andernfalls wäre im vorliegenden Beispiel die Abbildung der technischen Realisierung der Wandlung selbst verloren gegangen.

Das Konzept zur parametrischen Modellierung von Zusicherungen in SysML ist insbesondere zur Beschreibung von Effekten sicherlich sinnvoll, jedoch muss an dieser Stelle auch angemerkt werden, dass es noch einiger Weiterentwicklungen bedarf. Die Modellierung ist derzeit noch rein qualitativ, da das Modell beispielsweise keine mathematischen Operationen kennt (diese werden rein textuell eingefügt). Da Simulationswerkzeuge wie Matlab/Simulink oder Modelica deutlich leistungsfähiger sind, wurden bereits verschiedene, weiterführende Konzepte vorgestellt, die von der direkten Kopplung von SysML-Modellierungswerkzeugen mit Simulationswerkzeugen bis hin zur Erweiterung der SysML-Werkzeuge um Simulations-Plugins oder um erweiternde Sprachprofile wie ModelicaML reichen[449].

6.4 Modellierung der Vernetzung von Zielsystem und Objektsystem

In den vorangegangenen Kapiteln wurden die Aspekte von Ziel- und Objektsystem vorgestellt, die in SysML modellierbar sind. Ein zentraler Mehrwert in einem fachdisziplinübergreifenden Systemmodell liegt jedoch auch darin, diese beiden Systeme miteinander semantisch koppeln und somit einen systemübergreifenden Übertragungspfad herstellen zu können. Dies geschieht über die Kopplung von Anforderungen aus dem Zielsystem und deren erfüllender Systemarchitektur als Teil des Objektsystems sowie über die Modellierung von Testfallsequenzen[450].

6.4.1 Vernetzung von Anforderungen und Systemarchitektur

Anforderungsmanagementwerkzeuge sind in der Lage, Anforderungen zu beschreiben und zu strukturieren. Jedoch besteht darin keine Möglichkeit, deren Erfüllung oder auch ihre Entstehungsursache modellbasiert nachzuvollziehen. Hier bietet die SysML mittels entsprechender Relationen die Möglichkeit, Anforderungen mit Elementen der Systemarchitektur, also beispielsweise Aktivitäten, Zuständen,

[449] Vgl. u.a. Johnson et al. (2008), Schamai et al. (2009), Paredis et al. (2010), Zingel et al. (2012), Gloderer (2010), Kruppok (2012)

[450] Vgl. Kapitel 5.2.4 sowie Kapitel 6.4.2

Leitstützstrukturen oder physischen Komponenten zu vernetzen. Insbesondere die Nachvollziehbarkeit ihrer Erfüllung durch das entwickelte System ist hierbei von zentraler Bedeutung sowohl für Entwickler als auch das Projektmanagement. Bild 6-23 zeigt beispielhaft eine kleine, exportierte Excel-Matrix mit Anforderungen (Zeilen) und den erfüllenden Elementen (Spalten), die auf dem rechts daneben gezeigten *Requirement Diagram* basiert und durch das Modellierungswerkzeug automatisch erzeugt wurde.

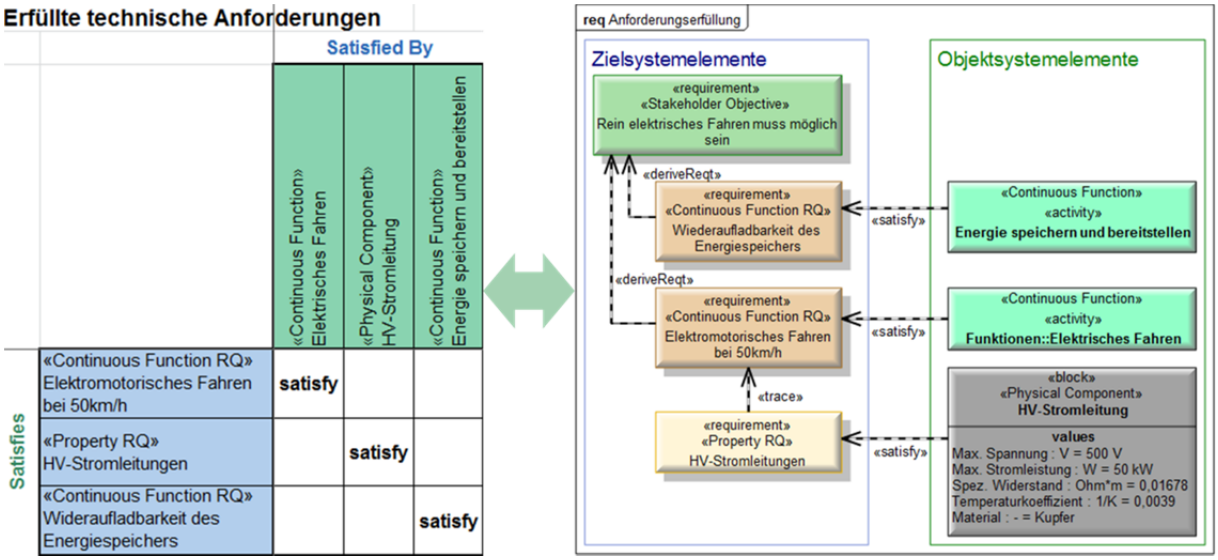


Bild 6-23: Anforderungserfüllungsmatrix (links) und *Requirement Diagram* (rechts)

Neben den gezeigten Erfüllungsbeziehungen (*satisfy*) können auch weitere Beziehungen zwischen SysML-Elementen wie beispielsweise die Ableitungsbeziehung (*derive*) von neuen Technischen Anforderungen aus gewählten Teillösungen in solchen Matrizen bzw. Tabellen exportiert werden. Gleichermaßen kann der Übertragungspfad zwischen Elementen des Ziel- und des Objektsystems auch direkt in SysML-Modellen auf verschiedene Weisen nachvollzogen werden. Beispielsweise ist dem rechts in Bild 6-23 gezeigten Diagramm zu entnehmen, dass die *Physical Component** „HV-Stromleitung“ die *Property RQ** „HV-Stromleitungen“ erfüllt. Die hängt unter anderem mit der *Continuous Function RQ** „Elektromotorisches Fahren bei 50 km/h“ zusammen, die sich wiederum aus dem *Stakeholder Objective** „Rein elektrisches Fahren muss möglich sein“ ableitet. Dieser mehrstufige Übertragungspfad wird von Entwicklern teilweise als Wirkkette

bezeichnet[451], jedoch besteht hier kein Zusammenhang zum Begriff *Wirknetz* als Konzept des Contact & Channel-Ansatzes, der auch Teil der Sprachbasis ist[452].

Die im Rahmen dieser Arbeit implementierten Erweiterungen der SysML bieten gleich zwei Vorteile: einerseits ist die Nachvollziehbarkeit durch die Differenzierung der Anforderungsarten wesentlich eindeutiger, andererseits können somit auch eindeutigere Regeln zur Modellierung definiert werden. Beispielsweise ist es nur möglich, *Continuous Functional RQ** durch *Continuous Functions** zu erfüllen und eben nicht durch andere Elemente wie *Discrete Functions**, *CSS** oder *Physical Components**[453].

6.4.2 Ableitung von Testfällen und Modellierung von Testfallsequenzen

Ein weiteres Konzept der SysML sieht vor, aus definierten allgemeinen Anwendungsfällen[454] konkrete Testfälle abzuleiten, die der Verifikation von Anforderungen dienen. Hierzu wird einerseits die *verify*-Beziehung eingesetzt, um Testfall und zu verifizierende Anforderungen modellbasiert zu verknüpfen. Andererseits kann der Ablauf eines Testfalls in Form einer Testfallsequenz mit beteiligten Systemelementen mittels eines *Sequence Diagrams* explizit modelliert werden. Bild 6-24 zeigt am Beispiel von „Rekuperation im Schubbetrieb“, durch welchen Testfall die entsprechende *Continuous Function RQ** verifiziert wird.

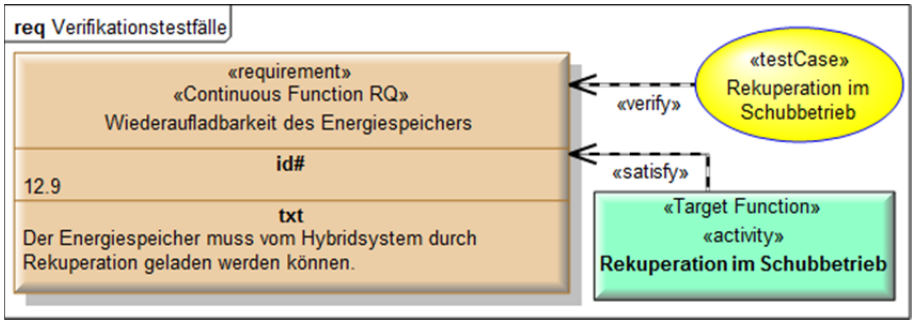


Bild 6-24: Anforderungsverifikation durch Testfall im *Requirement Diagram*

Der Ablauf des Testfalls kann weiter durch ein *Sequence Diagram* ausmodelliert werden, wie Bild 6-25 für das Beispiel „Rekuperation im Schubbetrieb“ zeigt.

---

[451] Diese Interpretation wurde in der Umfrage zur Definition von „Wirkkette“ mehrfach genannt, siehe auch: Albers und Zingel (2013a)

[452] Vgl. Kapitel 2.4.9 sowie Kapitel 5.2.2

[453] Vgl. Kapitel 11.10 (Modellierungsregeln)

[454] Vgl. Kapitel 6.2.1

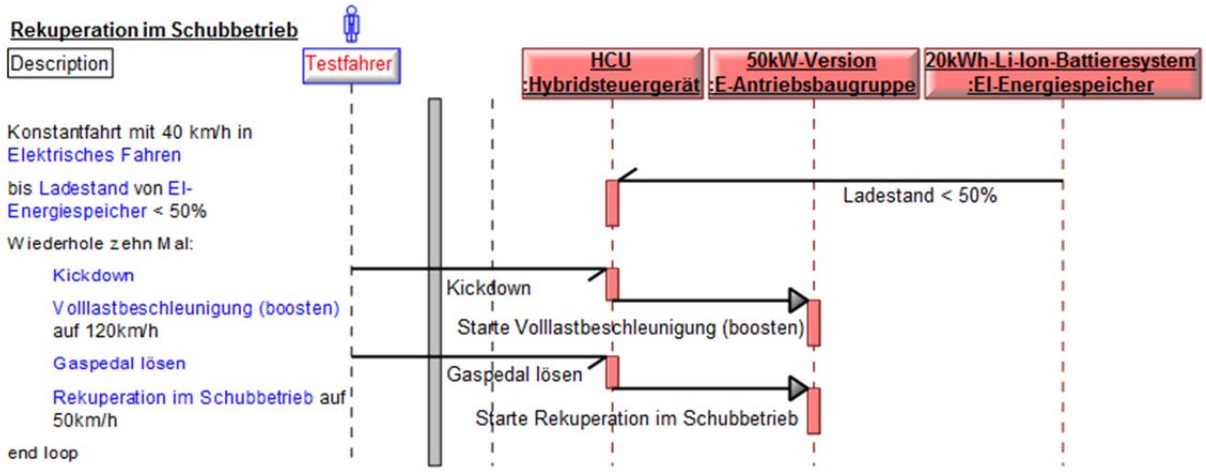


Bild 6-25: Testsequenz im *Sequence Diagram*

Das Diagramm zeigt links den Ablauf, der auch Wiederholungsschleifen oder parallele Vorgänge abbilden kann. Involvierte Parameter (*Values*) wie „Ladestand“ oder Ist-Funktionen (*Activities*) wie beispielsweise „Elektrisches Fahren“ können zwar aktiv im Text verlinkt werden, jedoch ist das Modell nicht in der Lage, daraus zu interpretieren, dass diese Funktionen hier ausgeführt werden sollen. Oben im Diagramm sind die im Test agierenden Akteure (hier: der Testfahrer) und Instanzen von Systemkomponenten (hier: u.a. „HCU“ vom Typ „Hybridsteuergerät“) zu sehen. Die eigentlichen Interaktionen im Testverlauf werden durch Ereignisse (*Events*) und Operationen (*Operations*) zwischen den gestrichelten Lebenslinien modelliert.

Die Sequenzmodellierung wurde durch die SysML unverändert von der UML übernommen. Daher ist der Mehrwert dieses Diagrammtyps vor allem beim Test von Softwaresystemen gegeben. Testsequenzen mechatronischer Systeme können nur bedingt modelliert werden, da hier nur Ereignisse und Operationen verwendet werden können, sich aber keine konkreten Inputs (*Object Flows*) oder quantitative Werte für Merkmale und Eigenschaften vorgeben lassen.

Möglichkeiten und Grenzen der Testfallmodellierung in SysML wurden u.a. durch ALT sowie DÜSER untersucht, worauf für weitere Details an dieser Stelle verwiesen sei[455].

6.5 Konzept zur Integration der Modellierungssprache in eine Entwicklungsumgebung

Die vorgestellte Modellierungstechnik verwendet eine erweiterte Form der SysML als Sprache. Damit entsteht ein zentrales, übergeordnetes Systemmodell, das jedoch nicht isoliert und parallel zu den weiteren Arbeiten der Entwickler entstehen, sondern

[455] Alt (2009), Düser (2010), Alt (2012)

vielmehr auch IT-seitig als Kommunikationszentrale und als vernetzendes Element weiterer Modelle dienen sollte. Auf eine umfassende Erläuterung einer möglichen IT-seitigen Implementierung eines SysML-Modells in die Entwicklungssoftware-landschaft wird hier aufgrund der zahlreichen technischen Möglichkeiten und Randbedingungen verzichtet. Dennoch werden nachfolgend einige Konzepte aufgezeigt und prototypische Vorarbeiten zu Schnittstellenimplementierungen zur Demonstration der technischen Machbarkeit vorgestellt. Bild 6-26 zeigt auf Basis des 3-Ebenen-Konzepts für Produktmodelle nach EIGNER einige im Produktentstehungsprozess eingesetzte Arten von Modellen und exemplarische Schnittstellen[456].

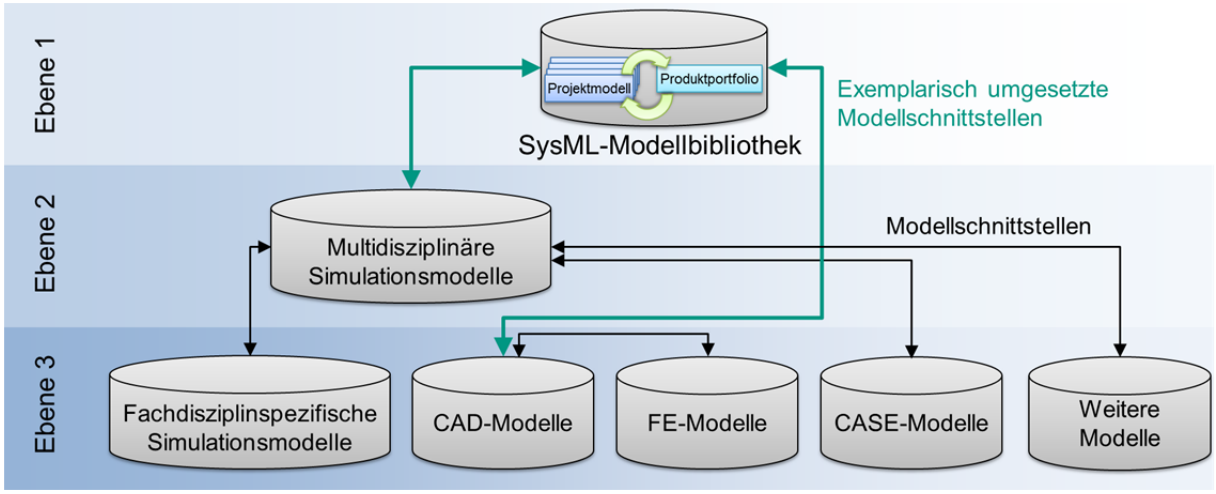


Bild 6-26: Modelle der Produktentstehung und mögliche Schnittstellen

Die Untergliederung in 3 Ebenen erfolgte in Anlehnung an EIGNER anhand ihres Zwecks:

Ebene 1: Modellbildung und Spezifikation
(qualitativ beschreibende, multidisziplinäre Modelle, z.B. SysML)

Ebene 2: Modellbildung und Konzeptsimulation
(quantitative, multidisziplinäre Simulationsmodelle, z.B. Matlab/Simulink, Modelica)

Ebene 3: Disziplinspezifische Modellbildung und Simulation
(ausgestaltende Modelle, z.B. CAD-, CASE- oder CAE-Modelle)

Diese Untergliederung soll verdeutlichen, dass die Kopplung von Modellen nicht zwingend immer über ein zentrales SysML-Modell erfolgen muss, was den Nachteil hätte, dass einerseits bestehende und etablierte Schnittstellen (z.B: das STEP-format zum Austausch von CAD- und FE-Modellen) nicht genutzt würden und

[456] Vgl. Eigner et al. (2012a), Eigner et al. (2012b), siehe auch Kapitel 2.6.2.6

andererseits die auszutauschende Informationsmenge und damit der Umfang des zentralen SysML-Modells explodieren würde. Dies wiederum würde dessen Handhabung umständlich machen und damit seine Anwenderakzeptanz massiv mindern. Die in der Grafik gezeigten Pfeile stehen für Schnittstellen zwischen Modellen, die jeweils durch entsprechende Modelltransformationen zu implementieren sind. Das Ziel dieser Modelltransformationen ist die direkte IT-gestützte Synchronisation von in mehreren Modellen verwendeten Daten. Beispielsweise wird die übergeordnete Systemarchitektur mit ihren Komponenten und Schnittstellen sowohl in SysML als auch möglicherweise in Simulationswerkzeugen abgebildet, wenn auch jeweils mit unterschiedlichen Zielsetzungen und in variierendem Detaillierungsgrad ihrer Merkmale. Weiterhin sind zur Beschreibung der Systemfunktionen in SysML funktionsrelevante Merkmale erforderlich, die beispielsweise auch geometrische und folglich mit CAD-Modellen zu synchronisierende Informationen beinhalten. All diese an unterschiedlichen Stellen bzw. in verschiedenen Modellen entstehenden Informationen sollten langfristig automatisch synchronisiert werden, um Widersprüche und Fehler ebenso zu vermeiden, wie den Verlust relevanter Informationen zu riskieren. Die verschiedenen Modelle werden in unterschiedlichen Aktivitäten der Produktentstehung eingesetzt, weshalb Art, Zeitpunkt und Umfang der zu synchronisierenden Informationen variieren können. Bild 6-27 visualisiert die Zusammenhänge zwischen verschiedenen, in der Produktentwicklung eingesetzten Modellen an einigen Beispielen grafisch.

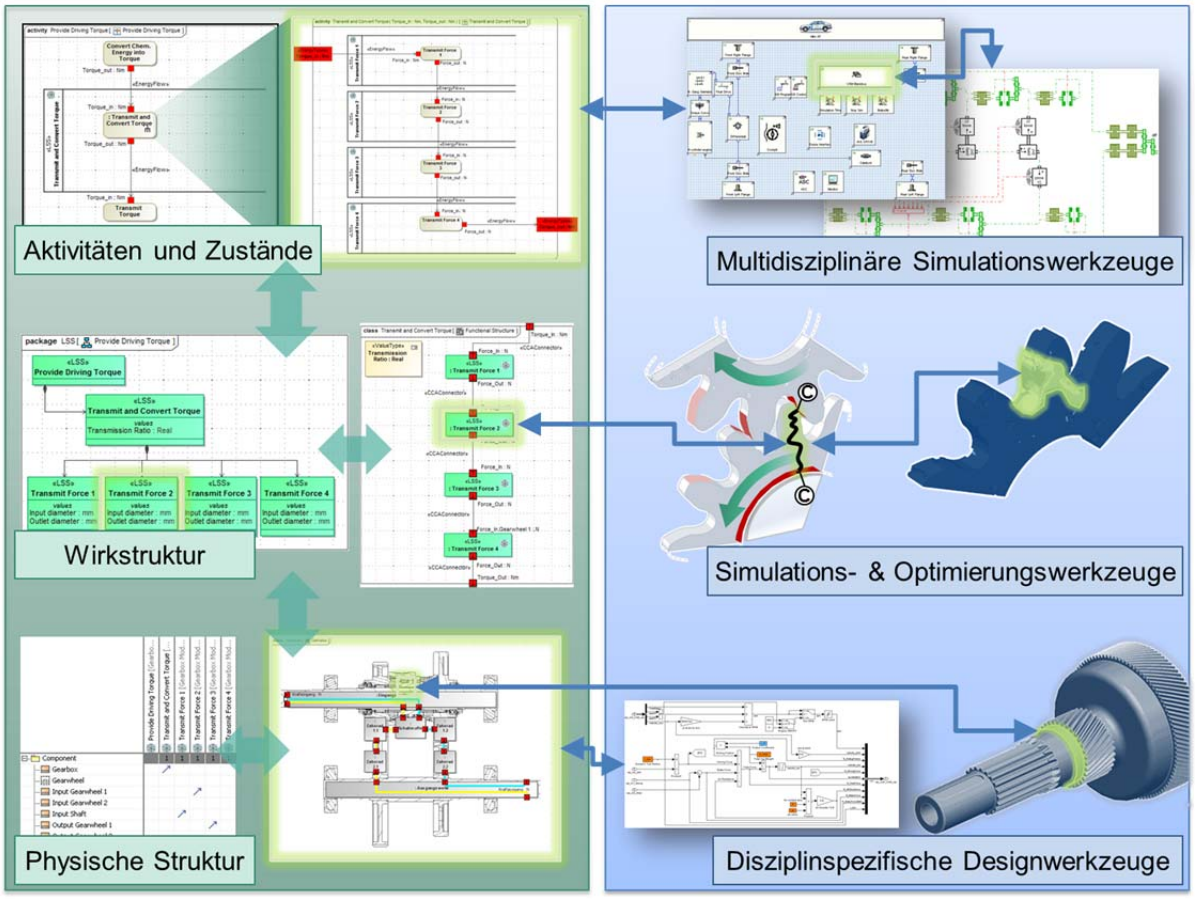


Bild 6-27: Synchronisation von Informationen zwischen Modellen[457]

Die Synchronisation von Modellen kann in den meisten Fällen nicht direkt erfolgen, da verschiedene Modelle auch über verschiedene Elemente (Entitäten und Relationen) verfügen, die Informationen in unterschiedlicher Form und in unterschiedlichen Zusammenhängen und Detaillierungsgraden abbilden. Hierzu ist eine Modelltransformation erforderlich, deren technisches Funktionsprinzip mit den Ebenen der Sprach- und Regelspezifikation in Bild 6-28 schematisch dargestellt ist.

[457] Vgl. auch Albers und Zingel (2013b)

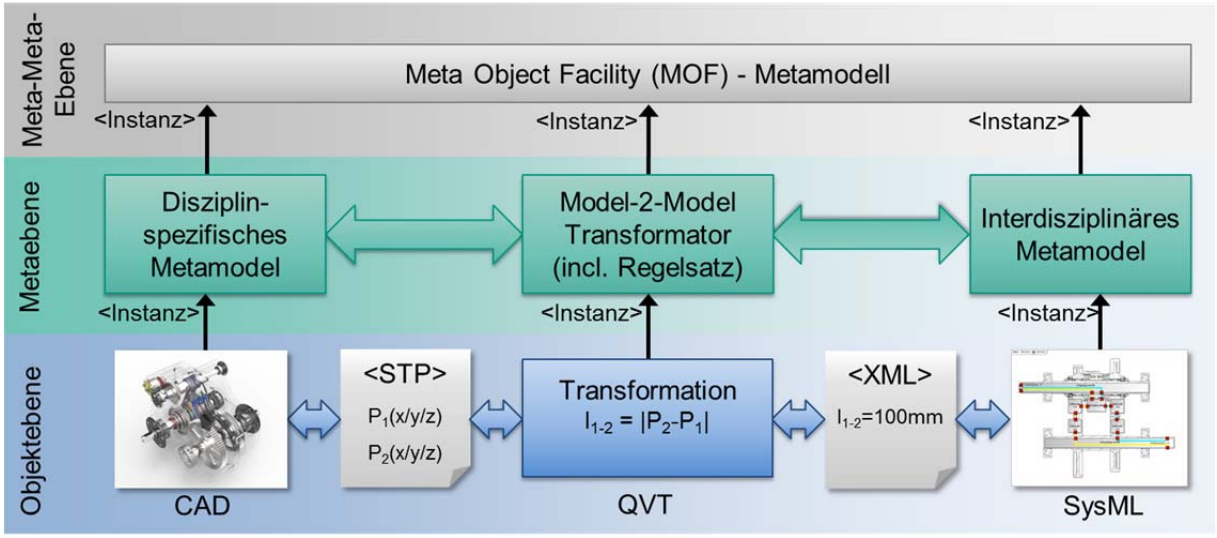


Bild 6-28: Schematische Ebenen der Spezifikation von Modelltransformationen

Der konsistente und eindeutige Austausch von Informationen zwischen verschiedenen Modellen erfordert eine gemeinsame technische Basis, über welche die Kommunikation erfolgen kann. Dies ist im gezeigten Beispiel die Meta-Sprache MOF[458]. Sie bildet unter anderem die Basis für die Beschreibung der Sprache SysML (das interdisziplinäre Metamodell). Modelle, die bisher nicht durch die MOF beschrieben wurden, müssen bei der Integration eines solchen Modells darin spezifiziert werden. Die Modelltransformation selbst beschreibt Regeln, wie die Informationen zwischen den Modellen zusammenhängen und kann unter Verwendung der Transformationsbeschreibungssprache QVT erfolgen[459].

Da zahlreiche verschiedenartige Modelle in der Produktentwicklung zum Einsatz kommen und der Schwerpunkt dieser Arbeit auf der besseren Unterstützung von Konstrukteuren liegt, wurden zwei Schnittstellen zwischen Modellen, die häufig von Maschinenbauingenieuren eingesetzt werden, prototypisch realisiert. Diese werden nachfolgend kurz vorgestellt.

6.5.1 Synchronisation von SysML und Simulink

Eine häufig auftretende Kritik an der SysML ist das Fehlen einer Ausführbarkeit seiner Modelle. Dies widerspräche jedoch dem übergeordneten, beschreibenden Charakter der Sprache, weil es sie viel zu umfangreich und kompliziert in der Handhabung werden ließe. Aus diesem Grund bieten Modellierungswerkzeuge bereits Schnittstellen an, mit denen sich Informationen aus SysML mit jenen in Simulationswerkzeugen austauschen lassen.

---

[458] Meta-Object Facility, vgl. Kapitel 2.8.2
[459] Query View Transformation, vgl. Kapitel 2.8.5

Simulink ist neben Modelica eine etablierte Entwicklungsumgebung und Programmiersprache zur Visualisierung, numerischen Berechnung und Programmierung mathematischer Ausdrücke[460]. Derartige Werkzeuge werden häufig bereits zur Simulation von Systemkonzepten unter Annahme von Vereinfachungen oder der Verwendung von Kennfeldern von physikalisch noch nicht beschreibbaren Subsystemen eingesetzt. Spezialisiertere Simulationsmodelle für die Automobilbranche sind beispielsweise IPG CarMaker, AVL CRUISE oder AVL VSM zur Antriebsstrang- oder der Fahrdynamiksimulation. Die meisten dieser Simulationswerkzeuge haben gemeinsame technische Wurzeln, die entweder auf der kausalen Modellierung (z.B. beruhen CarMaker und VSM auf Simulink) oder einer akausalen Modellierungstechnik (z.B. Modelica) beruhen. Kausal bedeutet hier, dass die Ports der funktionalen Blöcke immer gerichtet sind und damit keine Simulation einer umgekehrten Flussrichtung möglich ist. Dies war auch bei der prototypischen Applikation der Schnittstelle SysML-Simulink am Beispiel eines Hybridantriebsstrangs zu beachten, die Bild 6-29 schematisch darstellt.

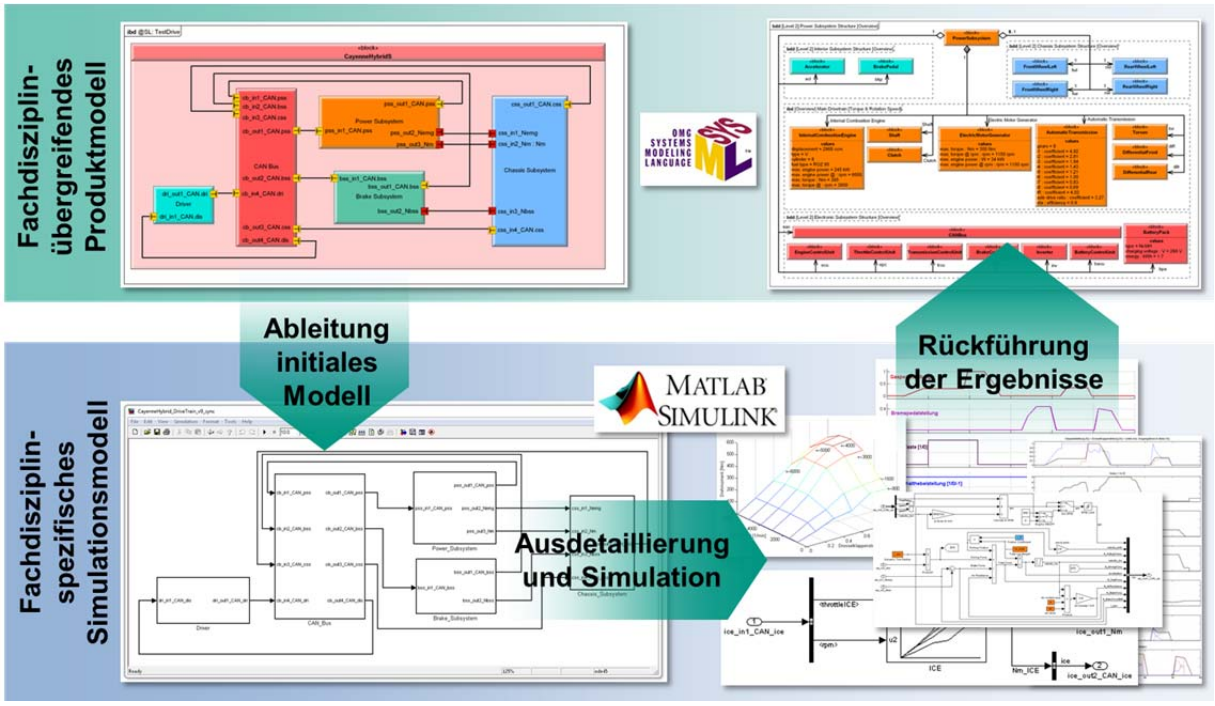


Bild 6-29: Funktionsweise der SysML-Simulink-Schnittstelle

Die SysML ist in der Lage, auch akausale Modelle, also bidirektionale Flussrichtungen abzubilden, was beispielsweise bei der Entwicklung der ModelicaML angewendet wurde[461]. Dies bedeutet gleichermaßen, dass für die Kopplung von

[460] Vgl. Kapitel 2.6.9
[461] Vgl. Kapitel 2.6.10

SysML und Simulink entsprechende Diagramme unter Einhaltung unidirektionaler Schnittstellen zu erzeugen sind. Die bereits im Modellierungswerkzeug Artisan Studio implementierte SysML-Simulink Schnittstelle wurde im Rahmen dieser Forschungsarbeit durch die studentischen Arbeiten von GLODERER und KRUPPOK eingehend hinsichtlich ihres Funktionsumfangs und der Nutzbarkeit in der Entwicklung von Automobilsystemen untersucht[462]. Mit ihr können *Internal Block Diagrams* und in eingeschränktem Umfang auch *Constraint Diagrams* der SysML mit Simulink synchronisiert werden. Es werden automatisch sowohl *Functional Blocks* als auch *Ports* mit Datentypen in Simulink erzeugt, die anschließend physikalisch ausmodelliert und simuliert werden können. Simulationsergebnisse lassen sich nach SysML zurückübermitteln (vgl. Bild 6-29). Weitere Diagrammarten der SysML wie beispielsweise Verhaltensdiagramme können jedoch nicht berücksichtigt werden. Auch Testfälle können derzeit noch nicht automatisiert übermittelt und simuliert werden. Dennoch ist somit die Möglichkeit gegeben, Fehler und Doppelarbeiten durch händisches Nachmodellieren bereits in SysML vorhandener Informationen in Simulink zu vermeiden, was auch einen Mehrwert dieser Schnittstelle begründet[463]. Aufgrund ihres Einsatzes zur Simulation der Simulation des Verhaltens von Systemkonzepten sollten ausschließlich Wirkstrukturdiagramme (*IWSD**), wie sie in Kapitel 6.3.4 vorgestellt wurden, synchronisiert werden.

6.5.2 Synchronisation von SysML und CAD

In Wirkstrukturen werden funktionsrelevante Merkmale, Eigenschaften und Schnittstellen abgebildet, die auch für weitere Modelle relevant sind. Diese können beispielsweise aus Simulationen resultierende geometrische Merkmale wie Dimensionierungsergebnisse oder auch verwendete Materialparameter umfassen, die für CAD-Modelle relevant sind. Eine Schnittstelle zwischen SysML und CAD wird jedoch nicht von Hause aus von SysML-Modellierungswerkzeugen angeboten. Daher wurde eine prototypische Schnittstelle im Rahmen der studentischen Arbeit von BOPP[464] zwischen Artisan Studio und dem CAD-Werkzeug ProEngineer implementiert. Hierbei wurden Parameter und Oberflächen (Wirkflächen) eines CAD-Modells durch ein Java-Programm ausgelesen und in einer XML-Datei abgelegt. Diese wiederum konnte von Artisan Studio eingelesen und Blöcken (also auch *CSS**en oder *Physical Components**) als Wirkfläche (*Port*) Parameter zugeordnet werden, wie Bild 6-30 am Beispiel einer Welle darstellt.

---

[462] Gloderer (2010), Kruppok (2012)
[463] Zingel et al. (2012)
[464] Bopp (2010)

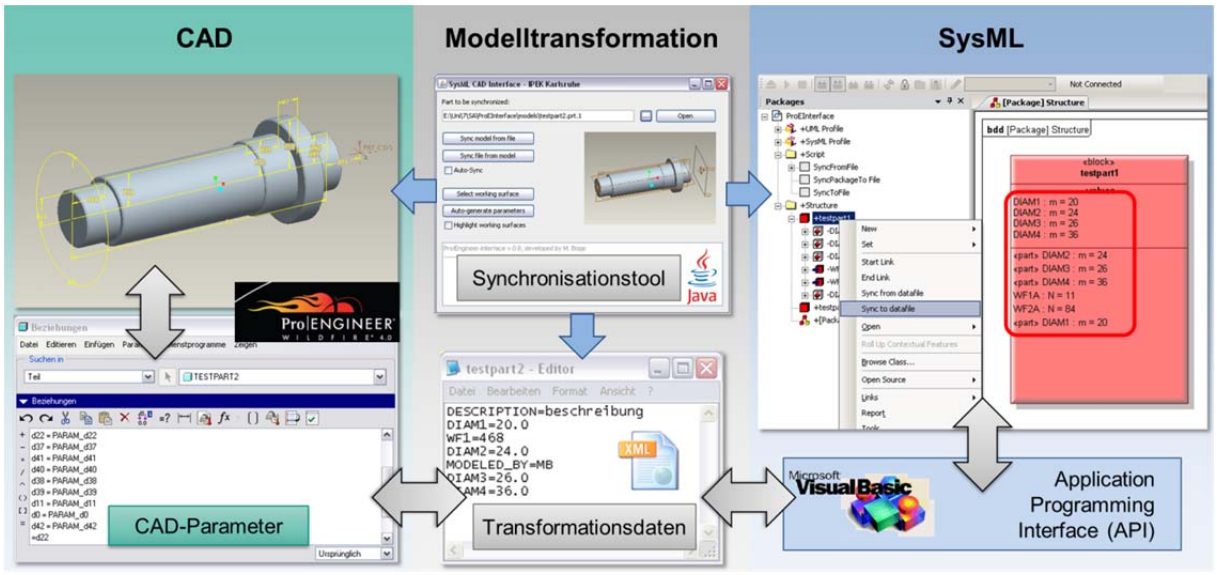


Bild 6-30: Modellsynchronisation zwischen SysML und CAD

Die Modellsynchronisation ist hier technisch relativ aufwändig, da ein Java-Softwaretool[465] das CAD-Programm ProEngineer von außen steuert (z.B. automatisch startet und Daten ausliest) und diese in eine XML-Datei ablegt. Ein weiterer Softwarecode, der unter Verwendung der Application Programming Interface (API) von Artisan Studio in Visual Basic programmiert wurde, liest diese Daten in das SysML-Modell ein. Je nach Art der Daten werden sie entweder Ports zugeordnet, die somit eine Oberfläche eines CAD-Modells in SysML repräsentieren oder sie werden *Values* zugeordnet und speichern damit geometrische Informationen von Strukturelementen.

In diesem Prototyp werden keine Transformationen vorgenommen, sondern nur eine direkte Übermittlung von Daten. Weiterhin basiert die Auswahl der Daten auch nicht auf existierenden Standards wie der STEP-Familie und ihren Applikationsprotokollen.

Eine weitere studentische Arbeit untersuchte die Möglichkeiten der Kopplung von CAD (Catia V5) und PREEvision, einer proprietären E/E-Architekturmodellierungssoftware der Firma aquintos (heute Teil der Vector Informatik GmbH)[466]. Technologisch ist das Metamodell von PREEvision dem der SysML sehr ähnlich, da beide auf MOF basieren. Daher ist auch eine weitgehende Übertragbarkeit gewährleistet. Im Rahmen dieser Arbeit wurden bei der technischen Einrichtung der Schnittstelle auch erste Transformationsregeln prototypisch implementiert. Darüber hinaus wurde das Format der zur Datenübermittlung

[465] Java wurde hier gewählt, da ProEngineer über eine Java-Schnittstelle verfügt. Somit ist die entstandene Software auch ausschließlich für dieses CAD-Werkzeug nutzbar.

[466] Bull (2012)

eingesetzten XML-Datei auf Basis der STEP-Applikationsprotokolle AP 203 und AP 214 definiert[467]. Bild 6-31 zeigt den technischen Aufbau der Kopplung der beiden Modelle.

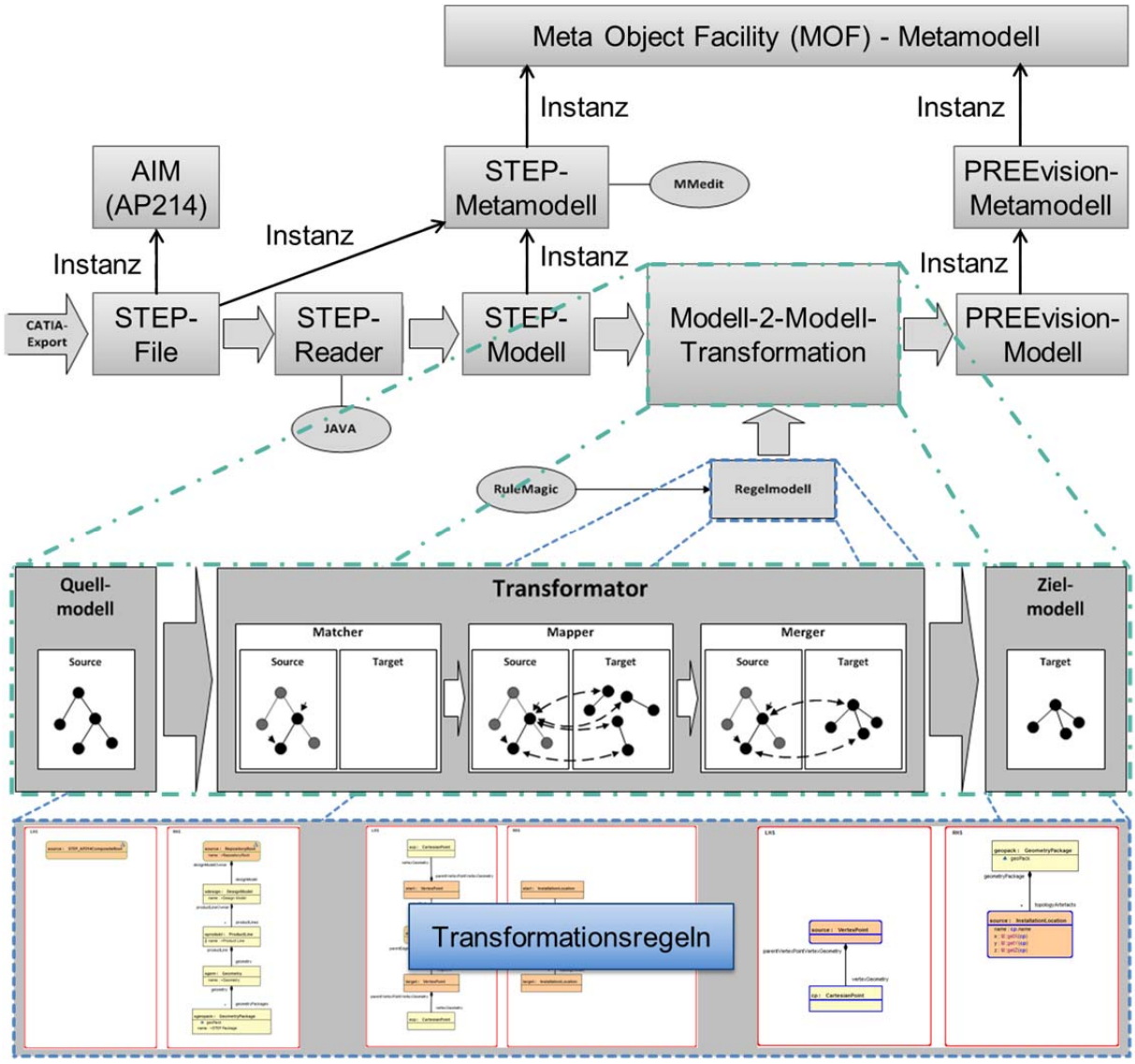


Bild 6-31: Kopplung von PREEvision und CATIA über STEP-Standard

Mit einer prototypische Implementierung sämtlicher in der Grafik dargestellten Transformationsschritte konnte die technische Machbarkeit und damit die Verfügbarkeit der informationstechnologischen Grundlagen nachgewiesen werden. Auch ABULAWI beschäftigt sich in ihrer Dissertation eingehend mit der Komplexität von CAD-Modellen und deren Handhabung durch den MDA-Ansatz[468] in Form einer

[467] ISO-10303-203 (2011), ISO-10303-214 (2010)
[468] MDA = Model-driven Architecture, vgl. Kapitel 2.8.1

Kopplung von CAD- mit SysML-Modellen, worauf an dieser Stelle für weiterführende Informationen verwiesen sei[469].

6.5.3 Kopplung von SysML und Ontologien zur funktionsorientierten Lenkung im Produktportfolioprozess

Die SysML ist eine Sprache zur fachdisziplinübergreifenden Modellierung von Produkten. Sie ist hierbei in der Lage, sowohl Elemente des Zielsystems als auch des Objektsystems zu beschreiben. Jedoch haben Projekterfahrungen gezeigt, dass die verfügbaren Diagramme nicht hinreichend für die Visualisierung von Management-relevanten Daten geeignet sind. Beispielsweise sind die Visualisierung von Übersichten bezüglich Kosten, Entwicklungsfortschritt oder Ressourcenverteilung nicht möglich, da insbesondere die häufig für diese Aspekte eingesetzten Torten-, Balken-, Linien- oder Netzdiagramme nicht zur Verfügung stehen. Darüber hinaus bietet die SysML derzeit noch keine ausgereiften Mechanismen und Konstrukte zur Gegenüberstellung von Produktvarianten oder gar Produktportfolios, die insbesondere als Arbeits- und Entscheidungsunterstützung im Multiprojektmanagement und der strategischen Unternehmensführung dienen[470].

Dennoch enthalten Produktmodelle zahlreiche Informationen, die in die oben genannten Managementaufgaben einzubeziehen sind. Hierzu wird im Rahmen des fortlaufenden Forschungsprojekts „Funktionale Lenkung mechatronischer Produkte“ unter Mitwirkung des Autors dieser Arbeit ein Methodenrahmenwerk entwickelt, das insbesondere die Kopplung von SysML mit Ontologien[471] und Methoden des strukturellen Komplexitätsmanagements[472] vorsieht[473]. Darin werden die in verschiedenen SysML-Modellen hinterlegten Aspekte von Ziel- und Objektsystemen von Produkten mit den Aspekten aus Modellen der zugehörigen Handlungssysteme laufender und geplanter Entwicklungsprozesse semantisch verknüpft und anwender- bzw. rollengerecht aufbereitet werden.

Der Lenkungsprozess des Produktportfolios wird unabhängig von den Produktentstehungsprozessen durchlaufen und startet, sobald ein interner Kunde (z.B. eine benachbarte Fachabteilung) oder externer Kunde (z.B. ein Endverbraucher) eine Anfrage für eine Systemlösung stellt, die aus dem vorhandenen Produktportfolio konfiguriert werden muss. Der schematische Ablauf dieses Portfolioprozesses ist in Bild 6-32 dargestellt.

---

[469] Abulawi (2012)
[470] DIN 69901 (2009), Kajikawa (2008), Schuh und Klappert (2011)
[471] Vgl. Kapitel 2.7
[472] Lindemann et al. (2009)
[473] Denger et al. (2012), Denger et al. (2013)

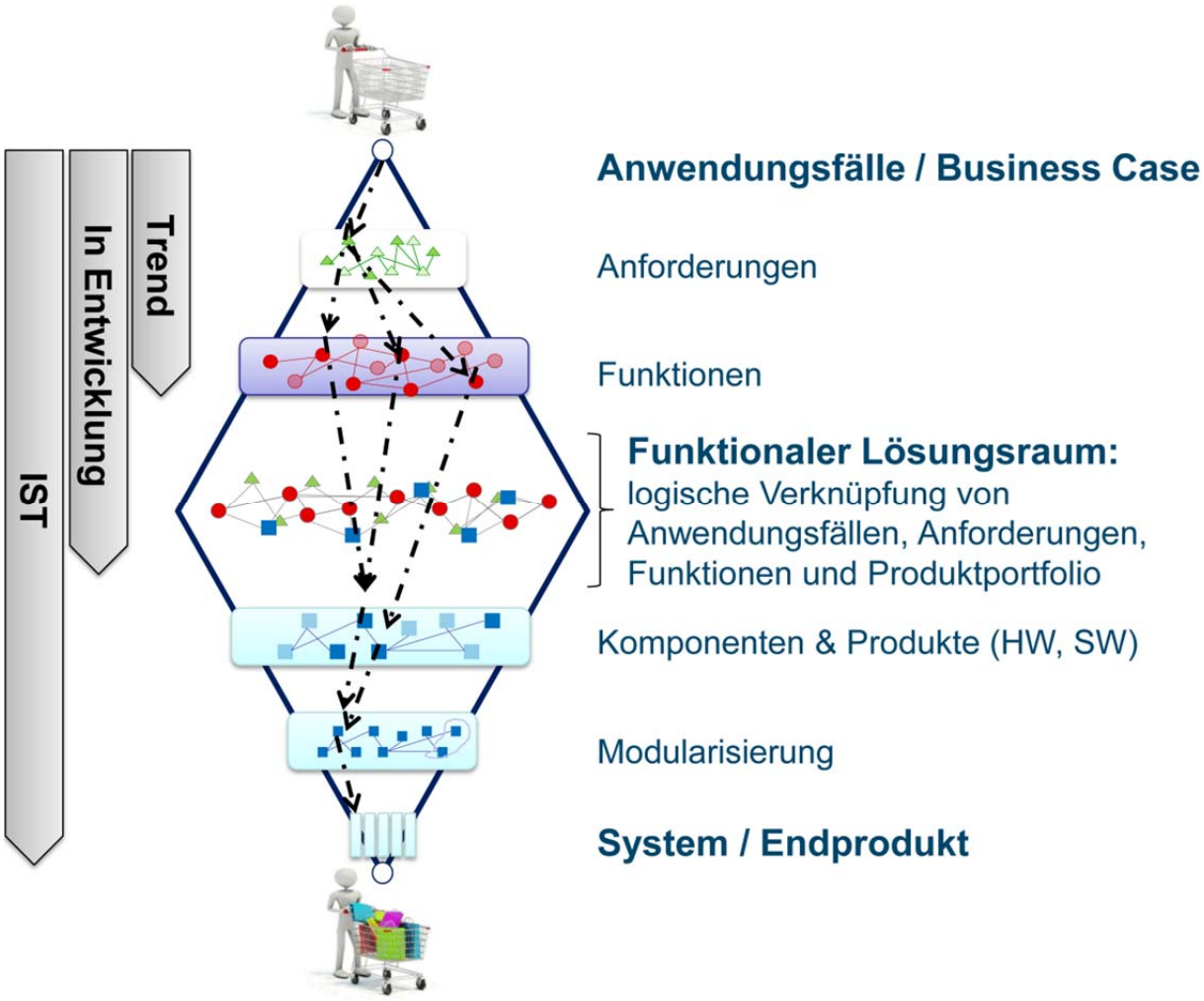


Bild 6-32: Funktionsbasierter Produktportfolio-Lenkungsprozess[474]

Ausgehend von den Anwendungsfällen und Anforderungen des Kunden werden zu realisierende Funktionen des geforderten Produkts bzw. der Systemlösung abgeleitet. Diese werden im funktionalen Lösungsraum mit den bereits verfügbaren oder in derzeit in Entwicklung befindlichen Funktionen des Produktportfolios abgeglichen. Dem liegt eine funktionale Beschreibung der Einzelprodukte oder Produktkomponenten auf Basis der hier vorgestellten Modellierungstechnik zugrunde. Sollten Teile der geforderten Funktionen noch nicht verfügbar sein, kann auf Basis der Informationen in den Modellen des Produktportfolios die Herbeiführung einer Entscheidung unterstützt werden, ob ein Produktentwicklungsprozess zur Realisierung der geforderten Funktionen das Produktportfolio sinnvoll ergänzt. Beispielsweise wird hier entschieden, wie viele weitere Kunden die geforderte, aber

[474] Aus den Vortragsfolien zu Denger und Fritz (2013)

derzeit nicht im Portfolio angebotene Funktionalität nutzen könnten und wie gut sich diese in das bestehende Portfolio bzw. die Unternehmensstrategie eingliedert. Abhängig von diesen Untersuchungen, zu denen u.a. auch die Systemmodelle herangezogen werden, kann eine Entscheidung zur Entwicklung einer Individualentwicklung oder einer universeller einsetzbaren Produktlösung mit möglicherweise weiterem Funktionsumfang getroffen werden. Zudem kann eine Einschätzung von Aufwand und Risiko bei dieser Neuentwicklung erfolgen. Darüber hinaus kann man durch die semantische Kopplung der Portfolioinformationen mit Informationen aus den laufenden Entwicklungsprozessen eine weitere Grundlage zur Entscheidungsunterstützung im Portfoliolenkungsprozess bereitstellen und die für den jeweiligen Verantwortlichen relevanten Informationen extrahieren und visualisieren (siehe Bild 6-33).

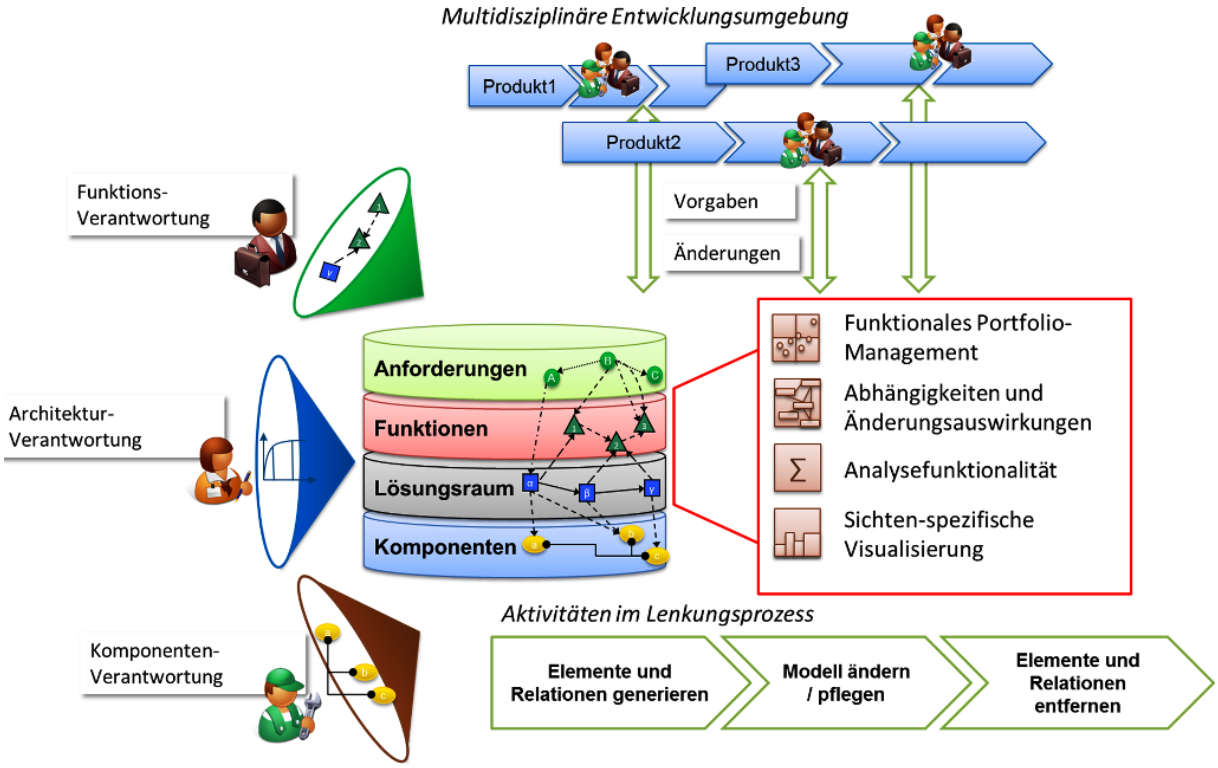


Bild 6-33: Framework der funktionsorientierten Lenkung mechatronischer Produkte[475]

Zur Realisierung dieser Vision sieht das Framework die Bildung einer Wissensdomäne vor, die durch Abfragen und Filter nach gewünschten Informationen durchsucht werden kann und diese anwendergerecht repräsentiert. Dadurch sollen die Informationen aus den Produkt- und Prozessmodellen extrahiert, transformiert und in einer Wissensdomäne abgelegt werden können. Deren Informationen und Abhängigkeiten werden in einer Datenbank abgelegt, aus der die

[475] Aus den Vortragsfolien zu Denger und Fritz (2013)

anwenderorientierten Sichten per Abfragen individuell ausgeleitet werden können. Bild 6-34 zeigt das Konzept der technischen Struktur eines Demonstrators, der im Rahmen des o.g. Forschungsprojekts realisiert wird.

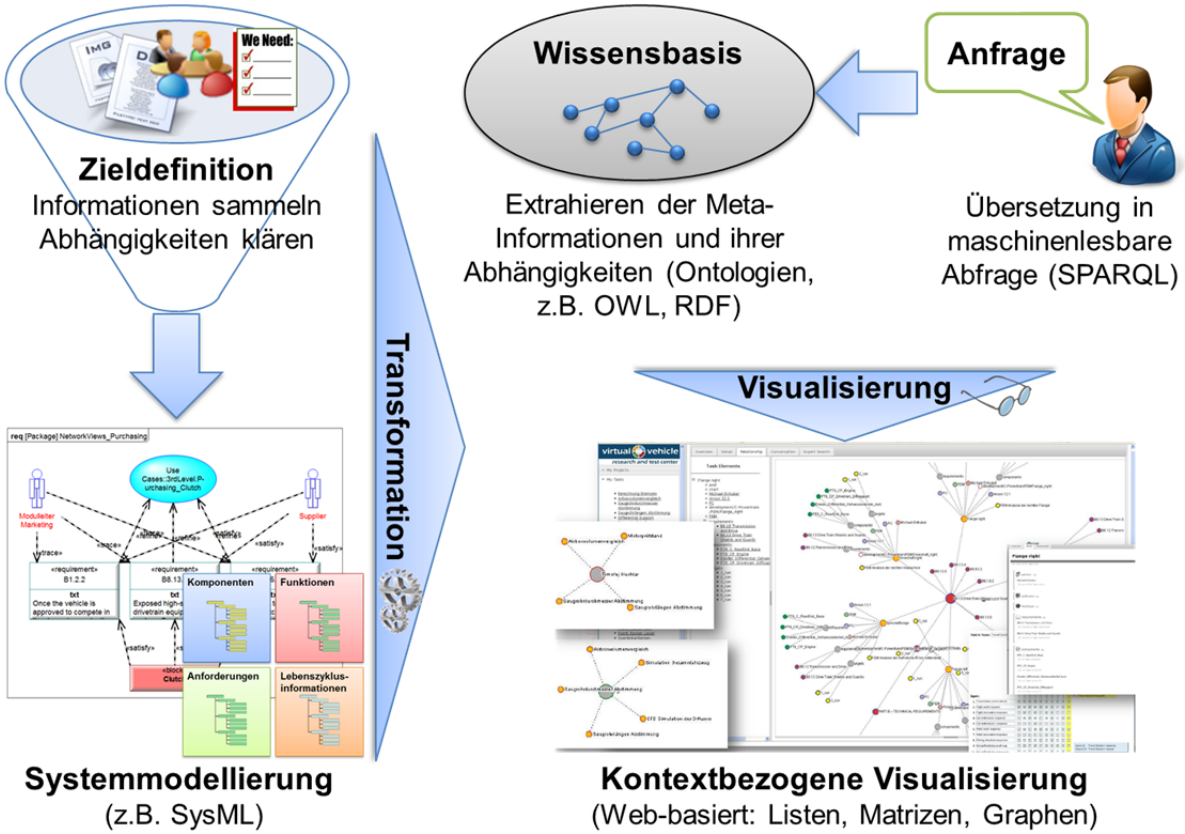


Bild 6-34: Konzept der technischen Struktur des Demonstrators zur funktionsbasierten Portfoliolenkung[476]

Der Kern des Demonstrators ist eine Wissensbasis aus mittels einer Ontologie wie OWL semantisch vernetzter Modelle, die neben Prozessmodellen vorwiegend aus Systemmodellen bestehen, welche mit der in dieser Arbeit vorgestellten Modellierungstechnik erzeugt wurden (links unten in der Grafik). Die kontextbezogene Visualisierung erfolgt dann mittels webbasierter Werkzeuge, die auf die Darstellung von Triple-Store-Abfragen (z.B. mit SPARQL) spezialisiert sind und daraus anwendergerechte Grafiken erzeugen.

Die Validierung der Kopplung von SysML mit Ontologien erfolgt durch die exemplarische Beschreibung zweier Industrieanwendungsfälle. Der Anwendungsfall der E/E-Architekturentwicklung inklusive der Kabelstränge wird funktional mit dem Ziel modelliert, ein modulares Portfolio mit minimaler innerer Komplexität und maximaler äußerer Variantenvielfalt zu generieren. Der zweite Anwendungsfall

[476] Leicht modifizierte Grafik aus Denger und Fritz (2013), Denger et al. (2013)

betrifft ein breites Produktspektrum im Bereich Prüfstandssysteme und bietet neben Einzelprodukten wie Messsystemen, Automatisierungssystemen, Simulationsmodellen und Prüfständen auch kundenspezifische Systemlösungen wie Gesamtfahrzeugentwicklungsumgebungen an. Hier besteht die besondere Herausforderung, Einzelprodukte eigenständig verkaufen und diese gleichzeitig auch als Teil von Systemlösungen anbieten zu können. Insbesondere die Prüfstandssysteme stehen im Rahmen dieser Forschungsarbeit im Vordergrund, die gleichzeitig ein Validierungsbeispiel der Modellierungstechnik in Kapitel 7 darstellen, wo diese auch näher vorgestellt werden.

6.5.4 Konzept für ein Rahmenwerk durchgängiger IT-basierter Modellkopplung

Aus den exemplarisch umgesetzten Modellschnittstellen und dem Konzept eines Frameworks zur funktionsbasierten Lenkung von mechatronischen Produktportfolios wurde im Rahmen dieser Arbeit ein Konzept für ein Rahmenwerk einer durchgängigen, IT-gestützten Kopplung von in der Produktentstehung eingesetzten Modellen abgeleitet. Die Umsetzung eines solchen Rahmenwerks erfordert weitere Forschungsarbeiten bezüglich der Art und Menge der zu synchronisierenden Informationen sowie umfangreiche informationstechnische bzw. softwareseitige Implementierungsarbeiten und wird dementsprechend nur schematisch in Bild 6-35 dargestellt.

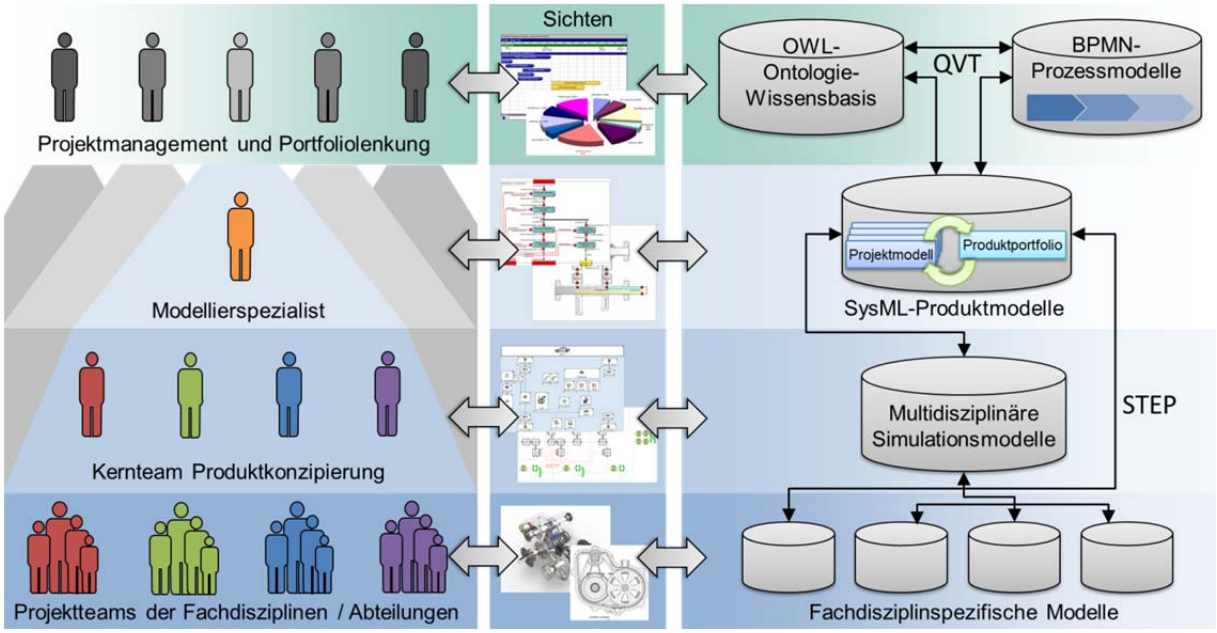


Bild 6-35: Konzept für ein Rahmenwerk zur durchgängigen, IT-seitigen Modellkopplung

Das Konzept schlägt vor, die Ansätze zur Kopplung der in der Produktentstehung eingesetzten fachdisziplinspezifischen und fachdisziplinübergreifenden Modelle (vgl. Bild 6-26) in ein übergeordnetes Rahmenwerk einzugliedern. Somit können

Prozessmodelle (bspw. in BPMN modelliert[477]) über Ontologien semantisch gekoppelt werden, um auch management-gerechte Informationen ausleiten und darstellen zu können. Weiterhin werden auf der linken Seite schematisch die Ebenen der Modellnutzung durch die Anwender[478], also Entwickler und Manager, im Rahmen von Produktentwicklungsprozessen und Portfoliolenkungsprozessen dargestellt.

Ein derartiges Rahmenwerk kann durch die Anwendung existierender Beschreibungsstandards wie beispielsweise den Modellierungssprachen OWL[479], SysML und BPMN sowie dem Transformationsstandard QVT[480] umgesetzt werden, die alle auf der gleichen technologischen Basis MOF zur Metamodellierung basieren. Der Datenaustausch kann über XMI-Technologien und unter Anwendung der STEP-Standardfamilie realisiert werden. Die IT-seitige Integration kann beispielsweise durch Enterprise Architecture Integrations (EAI)[481] oder Serviceorientierte Architekturen (SOA)[482] geschehen, worauf hier jedoch ebenso nicht näher eingegangen wird, wie auf die Möglichkeit zur Integration von Produktdaten- und –lebenszyklusmanagement-Systemen (PDM/PLM)[483].

6.6 Diskussion und Zwischenfazit

Die in diesem Kapitel vorgestellte Modellierungstechnik ist in der Lage, die fachdisziplinübergreifend relevanten Elemente von Ziel- und Objektsystem multidisziplinärer technischer Systeme zu beschreiben und zu vernetzen. Die modellierten Informationen können durch Sichten in Form von Diagrammen anwender- und aufgabenspezifisch extrahiert und dargestellt werden. Durch die getroffenen Erweiterungen der Modellierungssprache SysML unter Integration der Syntax und der semantischen Zusammenhänge der zuvor definierten gemeinsamen Sprache der Produktentwicklung können insbesondere mechanische Systeme eindeutiger beschrieben werden. Die Integration erster Konzepte des Contact & Channel – Ansatzes ermöglicht bereits jetzt die durchgängige, funktionszentrierte Analyse und Synthese technischer Systeme, unterstützt durch den realisierten technischen Demonstrator in Form eines Plug-Ins mit Sprachprofil und Automatismen. Ein Vorgehensmodell und Modellierungsregeln helfen dem Anwender bei der zielführenden Verwendung der Modellierungssprache. Durch die Vorstellung

[477] Business Process Modeling Notation, vgl. Kapitel 2.6.4
[478] Vgl., Bild 6-1
[479] Web Ontology Language, eine semantische Beschreibungssprache, vgl. Kapitel 2.7.2
[480] Query View Transformation, eine Beschreibungssprache für die Mdoelltransformation
[481] Siehe bspw. Conrad et al. (2005)
[482] Siehe bspw. Starke und Tilkov (2007)
[483] Vertiefende Informationen bspw. in Eigner und Stelzer (2009), Feldhusen und Gebhardt (2008)

eines Konzepts zur Integration der Modellierungssprache in eine durchgängige, weitgehend auf existierenden Standards basierenden, IT-Entwicklungsumgebung mit ersten prototypischen Demonstratoren wird die technische Machbarkeit einer ganzheitlichen IT-basierten Unterstützung des Produktentstehungsprozesses aufgezeigt. Darüber hinaus wurde ein Konzept vorgestellt, welches das Potential einer Integration der Technik in einen übergeordneten Produktportfolio-lenkungsprozess darlegt.

Trotz all dieser erreichten Ziele erfordert eine Realisierung dieser Gesamtvision einer ganzheitlichen, durchgängigen und anwenderfreundlichen modellbasierten Entwicklungssystematik noch zahlreiche weitere Forschungs- und Entwicklungsarbeiten. Aufgrund der sich rasant weiterentwickelnden Modellierungsstandards sind die Softwareanbieter gezwungen, ihre Modellierungswerkzeuge permanent anzupassen und deren Funktionsumfang auszuweiten. Dies erklärt auch zahlreiche Einschränkungen von SysML-Modellierungswerkzeugen in der Bedienerfreundlichkeit, die jedoch häufig einen Hauptgrund für Akzeptanzprobleme bei Anwendern in der Industrie darstellten, wie Interviews und Umfragen des Autors ergaben[484]. Auch die SysML selbst bietet noch viel Potential zur Steigerung der Effizienz in der Modellierung verschiedener Aspekte. Dies beruht auch darauf, dass erst nach und nach die „Altlasten“ ihres Ursprungs in der UML und damit der Softwareentwicklung überholt oder abgelegt werden. Beispielsweise wurden in der SysML V1.3 die Schnittstellen in Strukturdiagrammen komplett überarbeitet, um komplizierte, aber technisch zusammengehörige Schnittstellen besser darstellen zu können.

Jedoch sind einige insbesondere für physische Systeme relevante Informationen nach wie vor nicht oder nur sehr umständlich modellierbar. So bietet die SysML kein Diagramm, das geometrische Zusammenhänge darstellen kann. Die Beschreibung von Eigenschaften und Merkmale von Wirkflächen, die nicht Teil der übertragenen Objektflüsse sind (z.B. Oberflächenrauigkeit, Formen, Härtung von Oberflächen etc.), ist erstmals seit der SysML Version 1.3 möglich. Dies konnte im Rahmen dieser Arbeit aufgrund der fehlenden Implementierung in einem der verwendeten Modellierungswerkzeuge jedoch nur unzureichend auf Eignung getestet werden. Die hinreichende Modellierung von Testfällen physischer Systeme außerhalb des Softwarebereichs ist derzeit noch nicht mit den bordeigenen Mitteln der SysML in einer Form möglich, die Testingenieure hinreichend unterstützt bzw. Simulationsmodelle automatisiert parametrierbar macht. Weiterhin ist es nicht

[484] Siehe Albers und Zingel (2013a). Auch Broy et al. (2010) stellen fest, dass die Anwendbarkeit einer Methode maßgebend von der Qualität und Leistungsfähigkeit der Werkzeugunterstützung abhängt.

möglich, Kräfte- oder Momentengleichgewichte ohne echte Flüsse von Stoff, Energie oder Information zu modellieren, was unter anderem an einem fehlenden, bidirektionalen Objektfluss in Aktivitätsdiagrammen liegt, der hierzu erforderlich wäre. Die parametrische Modellierung eignet sich nur sehr bedingt zur Modellierung naturwissenschaftlicher Effekte, zudem kennt die Sprache keine mathematischen Operatoren zur Formulierung von Gleichungen. Dies lässt sich jedoch teilweise durch die Existenz deutlich leistungsfähigerer Modellierungswerkzeuge wie Matlab/Simulink oder Modelica für die physikalische Modellierung rechtfertigen.

Die aus wissenschaftlicher Sicht größte Herausforderung liegt jedoch darin, eine Methode zur Identifikation des zielführendsten und effizientesten Übergangs zwischen fachdisziplinübergreifender und disziplinspezifischer Modellierung zu entwickeln. Wird die Modellierung in SysML zu extensiv betrieben, mindert dies den Mehrwert, da der Modellier- und Modellpflegeaufwand massiv steigt und somit die Handhabbarkeit und Akzeptanz beeinträchtigt werden. Andererseits würde eine nur teilweise Verwendung der Modellierungstechnik auch deren Mehrwert mindern, da nicht alle fachdisziplinübergreifend relevanten Informationen gekoppelt werden können und somit die bisherige, dokumentenbasierte Entwicklung nicht vollständig abgelöst werden kann. Für eine derartige Methode sind zahlreiche Randbedingungen zu berücksichtigen wie beispielsweise die fachliche Zusammensetzung und Größe des Projektteams bzw. Unternehmens, Art und Vielfalt der eingesetzten und zu koppelnden disziplinspezifischen Modellierungswerkzeuge oder auch die Komplexität und Anzahl des zu modellierenden Produktportfolios.

7 Validierung der Modellierungstechnik

Für die Validierung der am Beispiel der Hybridantriebsstrangmodellierung evaluierten Modellierungstechnik wurde eine andere Entwicklungsdomäne gewählt, um die Übertragbarkeit der Erkenntnisse nachzuweisen. Da das IPEK über zahlreiche Prüfumgebungen und umfassende Kompetenzen in der Antriebssystementwicklung verfügt, wurde als primäre Validierungsanwendung die Modellierung einer Gesamtfahrzeug-Entwicklungsumgebung gewählt. Aufgrund der zahlreichen in deren Entwicklung partizipierenden Fachbereiche und der großen Teilproduktvielfalt eignet sich dieses Anwendungsfeld besonders gut zur Validierung einer fachdisziplinübergreifenden Modellierungstechnik. Darüber hinaus konnte die Validierung in Kooperation mit einem Automobilzuliefererunternehmen[485] erfolgen, da die Modellierungstechnik hier aktiv bei der Entwicklung einer Automatisierungsplattform für Prüfstände zum Einsatz kam. Zunächst wird der fachliche Hintergrund der zu modellierenden Systeme kurz vorgestellt, gefolgt von den gewonnenen Erkenntnissen an einigen Beispielen aus Elementen des Zielsystems und des Objektsystems einer Gesamtfahrzeug-Entwicklungsumgebung, die abschließend diskutiert werden.

Ferner wurde die Modellierungstechnik im Rahmen einer Diplomarbeit auch auf die Beschreibung von Laserschneidmaschinen übertragen. Auch dort konnte ein Nachweis über den Mehrwert modellbasierter Systembeschreibung erbracht werden[486]. Ein Überblick über die Ergebnisse wird in Kapitel 7.4 vorgestellt.

7.1 Einführung in den fachlichen Kontext der Validierungsbeispiele

Das Automobilzuliefererunternehmen entwickelt und vertreibt Systemlösungen sowie Entwicklungsprozesse für die Validierung sämtlicher Systemebenen in der Automobilentwicklung (vgl. XiL-Ebenen). Das Spektrum reicht von Friktionsprüfständen über verschiedenste Motoren-, Kupplungs-, Getriebe-, Batterie- und Antriebsstrangprüfstände bis hin zu Rollenprüfständen. Darüber hinaus werden auch umfangreiche Automatisierungsplattformen zur Regelung der Prüfstände und der Restsystem- und Umgebungssimulation, Messtechnik für verschiedenste Anwendungen von Emissions- und Verbrauchsmesstechnik über Komfort- und

[485] Hier: AVL List GmbH, Graz
[486] Martini (2012)

Fahrbarkeitsanalysesysteme bis hin zu EMV- und Akustikmesseinrichtungen sowie Bewertungs-, Kalibrierungs- und Optimierungswerkzeuge angeboten. Insgesamt umfasst das Portfolio über 6.000 Einzelprodukte, die kundenspezifisch in unzähligen, individuellen Systemkonfigurationen kombiniert werden können.

Aufgrund des großen Bedarfs des Kooperationspartners sowie der fachlichen Erfahrungen des Autors dieser Arbeit im Umgang mit Rollenprüfständen aus vorangegangenen Kooperationsprojekten heraus[487] wurde aus diesem Produktportfolio das übergeordnete Validierungsbeispiel einer neu entwickelten Gesamtfahrzeug-Entwicklungsumgebung gewählt. Auch dieser Teilbereich der Produktpalette des Unternehmens ist in sich noch von einer extrem hohen Variantenvielfalt geprägt, da neben den zahlreichen Konfigurationsmöglichkeiten über verschiedene Untersuchungsziele nicht nur Rollenprüfstände für Personenfahrzeuge, sondern auch für alle anderen Straßen- und Geländefahrzeuge verfügbar sind (siehe Bild 7-1).

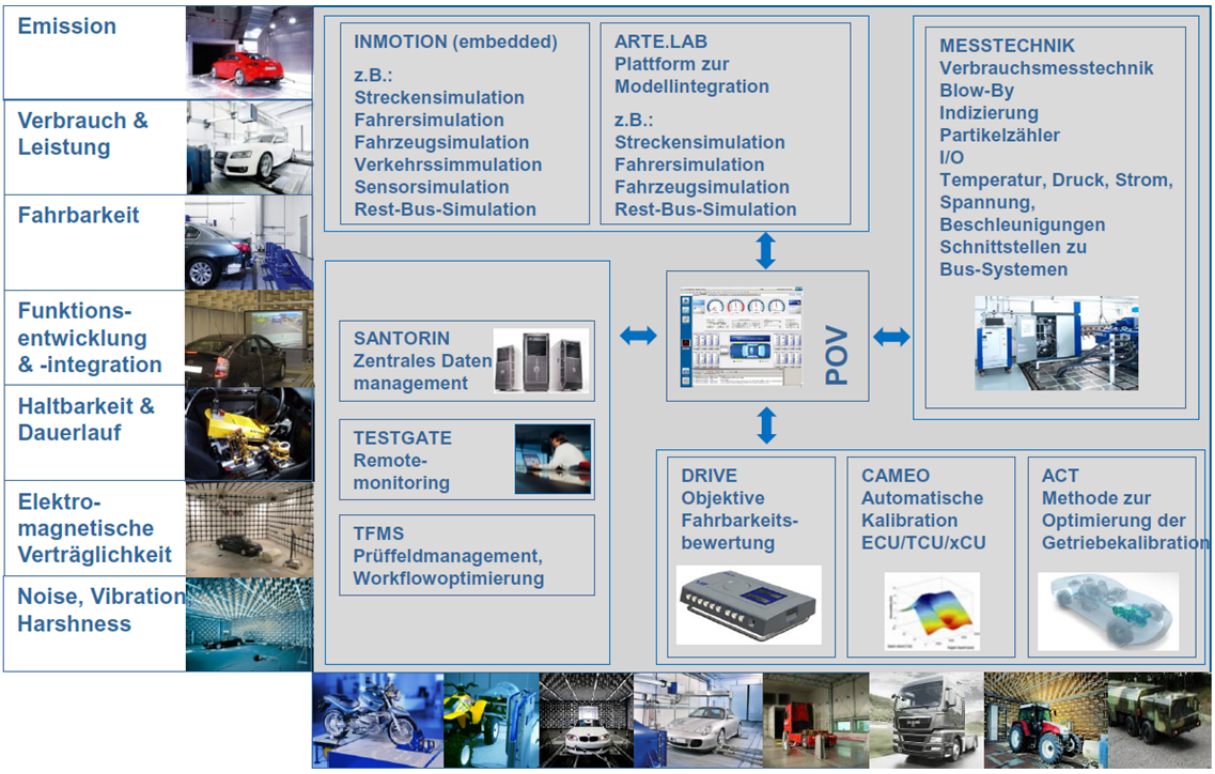


Bild 7-1: Produktspektrum des Automobilzulieferunternehmens im Bereich der Gesamtfahrzeug-Entwicklungsumgebungen[488]

Die Grafik zeigt einerseits die Hauptanwendungsfälle für Rollenprüfstände auf der linken Seite sowie die durch Grafiken dargestellten möglichen Prüflinge unten. Im

[487] Vgl. Kapitel 4.2.1
[488] aus Düser et al. (2012)

Zentrum der Prüfstandsregelung steht die Automatisierungsplattform[489], die für sämtliche Sekundärsysteme wie Simulationstools, Datenmanagement, Messtechnik sowie Bewertungs-, Kalibrierungs- und Optimierungstools entsprechende Dienste zu deren Betrieb bereitstellt. Ein Vorläufer der hier vorgestellten Modellierungstechnik wurde zur Entwicklung der Automatisierungsplattform für Rollenprüfstände eingesetzt, die inzwischen kurz vor der Marktreife steht. Die Ergebnisse aus der Validierung in diesem Kontext werden ebenfalls in den nachfolgenden Kapiteln vorgestellt.

Das Ziel des Einsatzes der Modellierungstechnik in diesem Systemkontext war, eine funktionsbasierte Beschreibung und Strukturierung der zahlreichen, möglichen Anwendungsfälle von Rollenprüfständen herbeizuführen und diese zur Konfiguration von kundenspezifischen Systemlösungen einzusetzen. Hierzu wurde der Schwerpunkt auf die Anwendungsfallanalyse inklusive deren Ausdetaillierung durch Ziel-Funktionen (*Target Functions**) sowie die Ableitung von Anforderungen gelegt. Ferner wurden erste Produkte des Unternehmens modelliert, um beispielhafte Sichten auf Systemarchitekturen der aus der funktionalen Analyse resultierenden Systemlösungen zu modellieren und anwendergerecht zu visualisieren.

7.2 Modellierung von Gesamtfahrzeug-Entwicklungsumgebungen

Die Modellierung der Gesamtfahrzeug-Entwicklungsumgebungen wurde mit dem Ziel einer funktionsbasierten Vorgehensweise bei der Konfiguration von kundenspezifischen Systemlösungen entwickelt. Dabei sollte ein Systemmodell auf Basis der Haupt- und Teilfunktionen des Produktportfolios des Unternehmens entstehen. Damit sollen Missverständnisse seitens des Kunden bei der Zusammenstellung einer für ihn optimalen Produktlösung vermieden werden. Bisher konnte bei der Auswahl anhand von Komponenten passieren, dass ein Kunde eine Produktkombination wählt, die für seine Anwendungsfälle ungeeignet oder teurer als notwendig ist. Unter anderem wurden so auch im Rahmen vergangener Projekte für die eigentlich ungeeignete Systemkonfiguration Zusatzfunktionen entwickelt, die eine andere Systemkonfiguration bereits abgedeckt hätte. Durch gezieltes Erfragen der gewünschten Funktionen kann aus dem Portfoliomodell die für den Kunden am besten geeignete Systemlösung ausgewählt werden.

Bei der Modellierung wurde einerseits eine Anwendungsfallstruktur erarbeitet und mit Ziel-Funktionen ausdetailliert, die es ermöglicht, die entsprechende Systemlösung für

[489] POV steht für PUMA Open Vehicle, dem Namen der Automatisierungsplattform des Unternehmens

einen Kunden auf funktionaler Basis zu konfigurieren. Andererseits wurden die bereits realisierten Funktionen des Unternehmensproduktportfolios modelliert und den ausführenden Komponenten zugewiesen. Somit können nun die erforderlichen Komponenten und Teilprodukte für eine zur Funktionserfüllung optimalen Systemlösung ausgelesen werden. Hierbei können neben den Anwendungsfällen auch nichtfunktionale Anforderungen, Ziele und Randbedingungen einbezogen werden.

Die Anwendungsfallanalyse wurde nach einer einwöchigen SysML-Schulung durch den Autor, welche auf Ergebnissen der vorliegenden wissenschaftlichen Arbeit beruht, weitgehend selbstständig durch Anwender beim Projektpartner durchgeführt. Sie verfügten zum damaligen Zeitpunkt jedoch nur über Teile der Erweiterungen der SysML in Form eines Vorläufers der hier vorgestellten Modellierungstechnik. Das resultierende Modell wurde in einem Projekt mit dem Ziel einer anwendergerechteren Visualisierung weiter ausgearbeitet. Hierzu wurde unter anderem ein Paketdiagramm erstellt, das die Navigation durch das Modell erleichtert und eine Übersicht der modellierten Aspekte bietet (siehe Bild 7-2). Das Diagramm repräsentiert den Modellaufbau durch dessen Paketstruktur und ist über *Viewpoints* mit entsprechenden Diagrammen verlinkt, auf die man durch Doppelklick auf das Element navigieren kann.

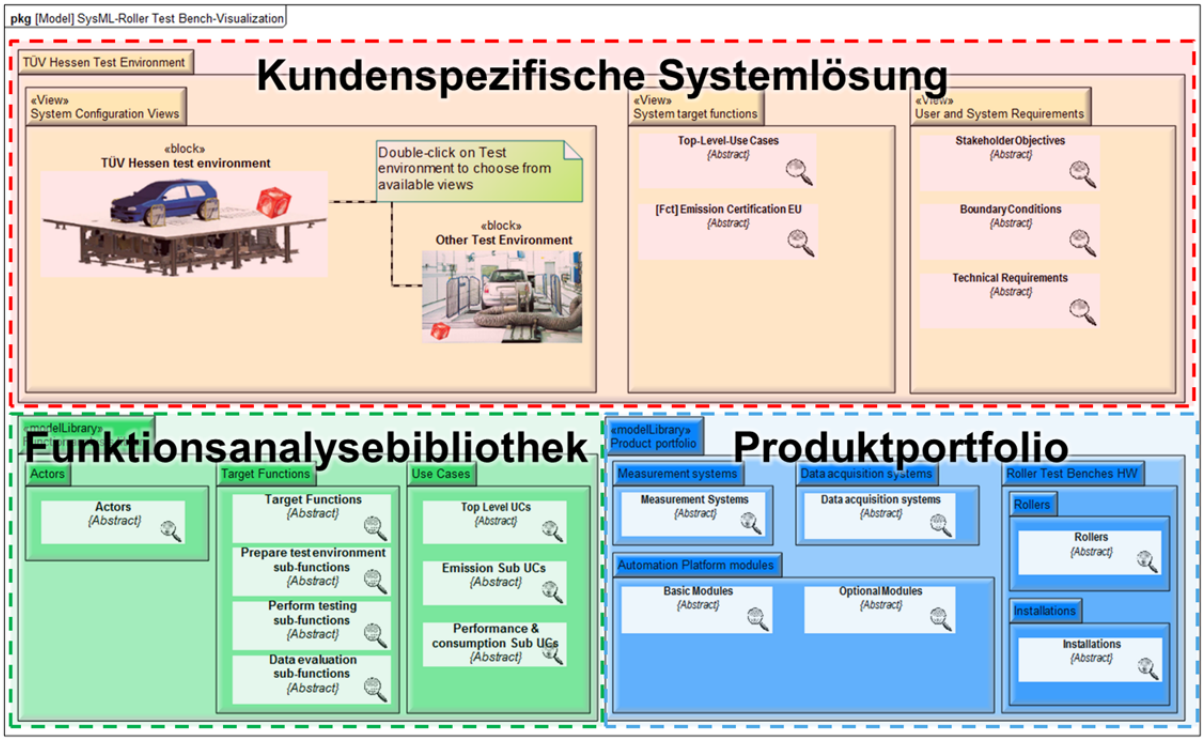


Bild 7-2: Übersichtsdiagramm AVL Rollenprüfstände

Wie der Abbildung zu entnehmen ist, gliedert sich das Modell in drei Hauptbereiche. Die **Funktionsanalysebibliothek** beinhaltet mögliche *Actors* (z.B. mögliche Units

under Test - UUT), *Use Cases* und Ziel-Funktionen (*Target Functions**), die mit Rollenprüfständen des Unternehmens durchgeführt werden können. Das **Produktportfolio** beinhaltet die Produkte des Unternehmens im Bereich Gesamtfahrzeug-Entwicklungsumgebungen, gegliedert nach ihren Hauptfunktionen (z.B. Messtechnik, Automatisierungssysteme etc.). Der obere Bereich **Kundenspezifische Systemlösung** beinhaltet die ausgewählten Anwendungsfälle und Ziel-Funktionen, definierte Anforderungen (*Stakeholder Objectives**, *Boundary Conditions** und *Technical Requirements**) sowie Sichten auf die resultierende Systemarchitektur einer konkreten, kundenspezifischen Produktkonfiguration.

Bild 7-3 zeigt exemplarisch einige mögliche *Actors* (z.B. mögliche UUT, Kunden oder interagierende Normen und Standards) als Teil der Funktionsanalysebibliothek.

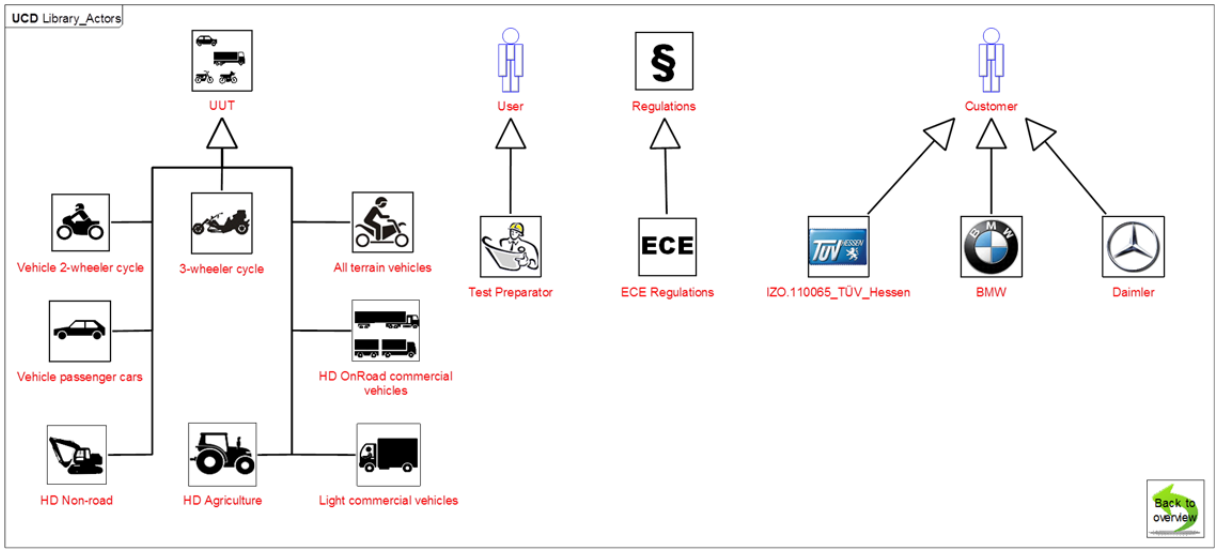


Bild 7-3: Durch Piktogramme visuell aufbereitete Actor-Bibliothek im *Use Case Diagram*

Aus dieser Bibliothek können neben UUTs beispielsweise einzuhaltende Regulierungen für Anwendungsfälle (*Use Cases*) ausgewählt werden. Die Anwendungsfälle selbst sind hierarchisch gegliedert, wobei die oberste Ebene an die Referenzanwendungsfelder aus Bild 7-1 angelehnt ist (siehe Bild 7-4).

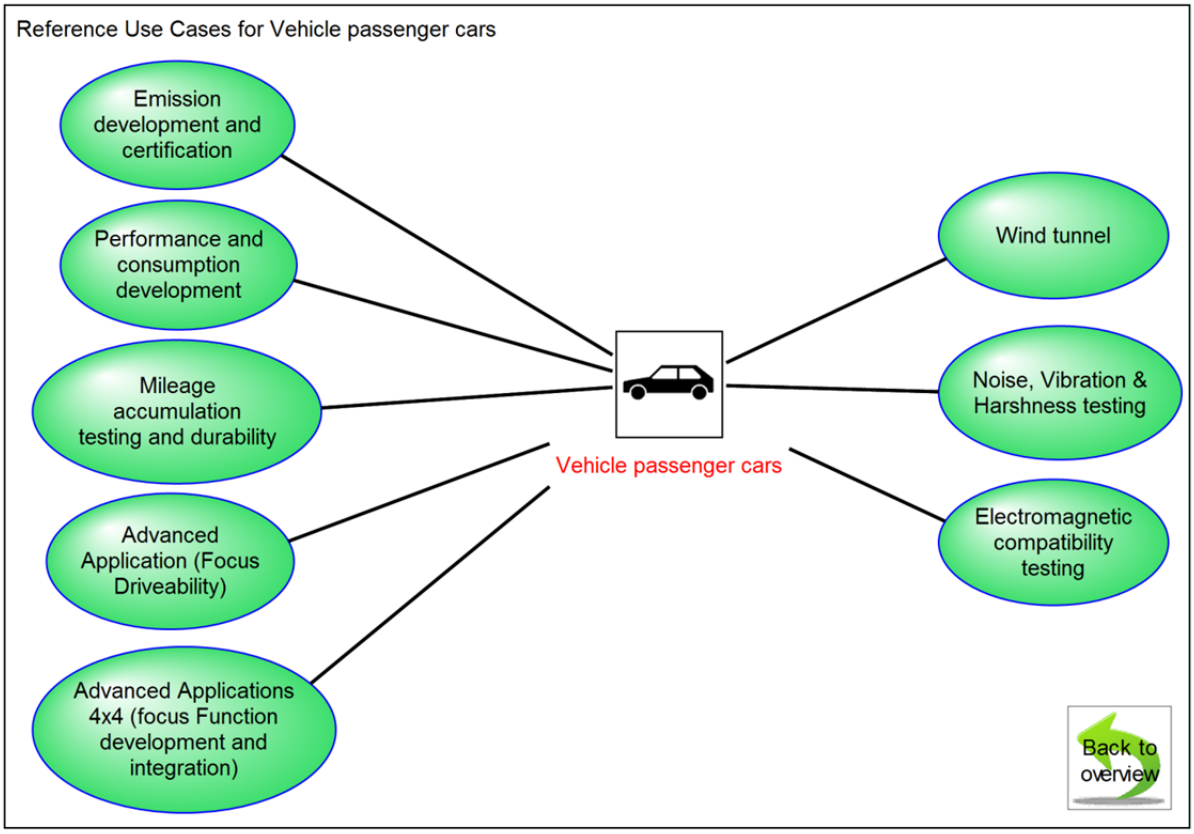


Bild 7-4: Referenzanwendungsfälle für Rollenprüfstände

Diese sind weiter in Unteranwendungsfälle und einzelne Ziel-Funktionen (*Target Functions**) dekomponiert (siehe Bild 7-5).

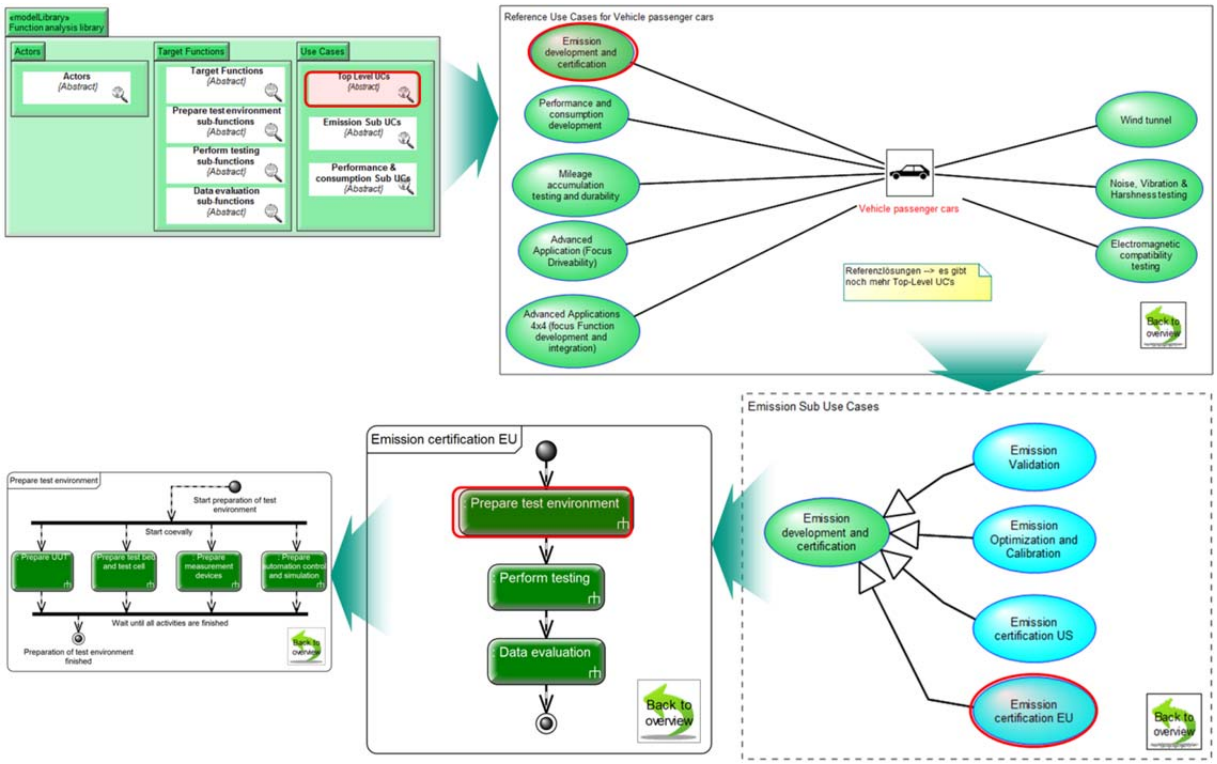


Bild 7-5: Dekomposition der Referenzanwendungsfälle in Ziel-Funktionen

Aus dieser funktionsbasierten Beschreibung können nun in Absprache mit dem Kunden die zu realisierenden Funktionen ausgewählt und im Paket „Kundenspezifische Systemlösung“ abgelegt werden. Dort können den Funktionen dann die entsprechenden Komponenten aus der Produktportfoliobibliothek, von der Bild 7-6 einen Auszug zeigt, über *Activity Partitions* zugewiesen werden[490].

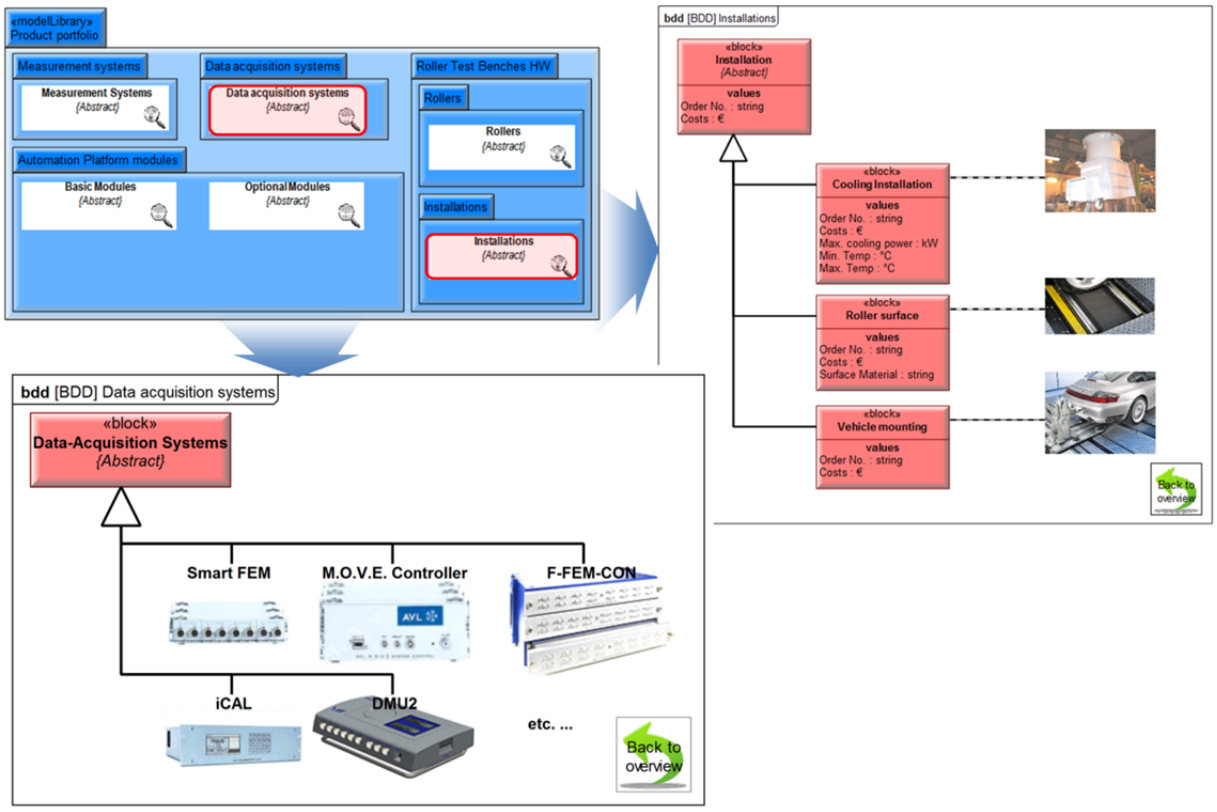


Bild 7-6: Auszug aus der Produktportfoliobibiothek

Die Verwendung der mit dem Produktportfolio vernetzten Anwendungsfall- und Funktionsbibliothek ermöglicht die Auswahl der passenden Produkte für die Systemlösung, wobei neben deren Funktionen auch die Bereitstellung geforderter Schnittstellen, Merkmale und Eigenschaften (z.B. Welle-Nabe-Verbindung, Datenbus, Messgenauigkeit, Dynamik, Kosten, Arbeitsbereich etc.) einbezogen werden, die in Anforderungen definiert wurden[491]. Da die Systemarchitektur meist sehr umfangreich ist, kann ihre Gesamtheit auf mehrere Diagramme verteilt werden. Bild 7-7 zeigt dies am Beispiel der beteiligten Komponenten und ihrer Schnittstellen für Fahrbarkeitsmessungen auf dem Rollenprüfstand.

---

[490] Vgl. Kapitel 6.3.3

[491] Auf grafische Beispiele für Anforderungsdiagramme wurde hier verzichtet, da deren optische Aufbereitung analog zu den Beispielen aus den Kapiteln 6.2.2 - 6.2.6 ist.

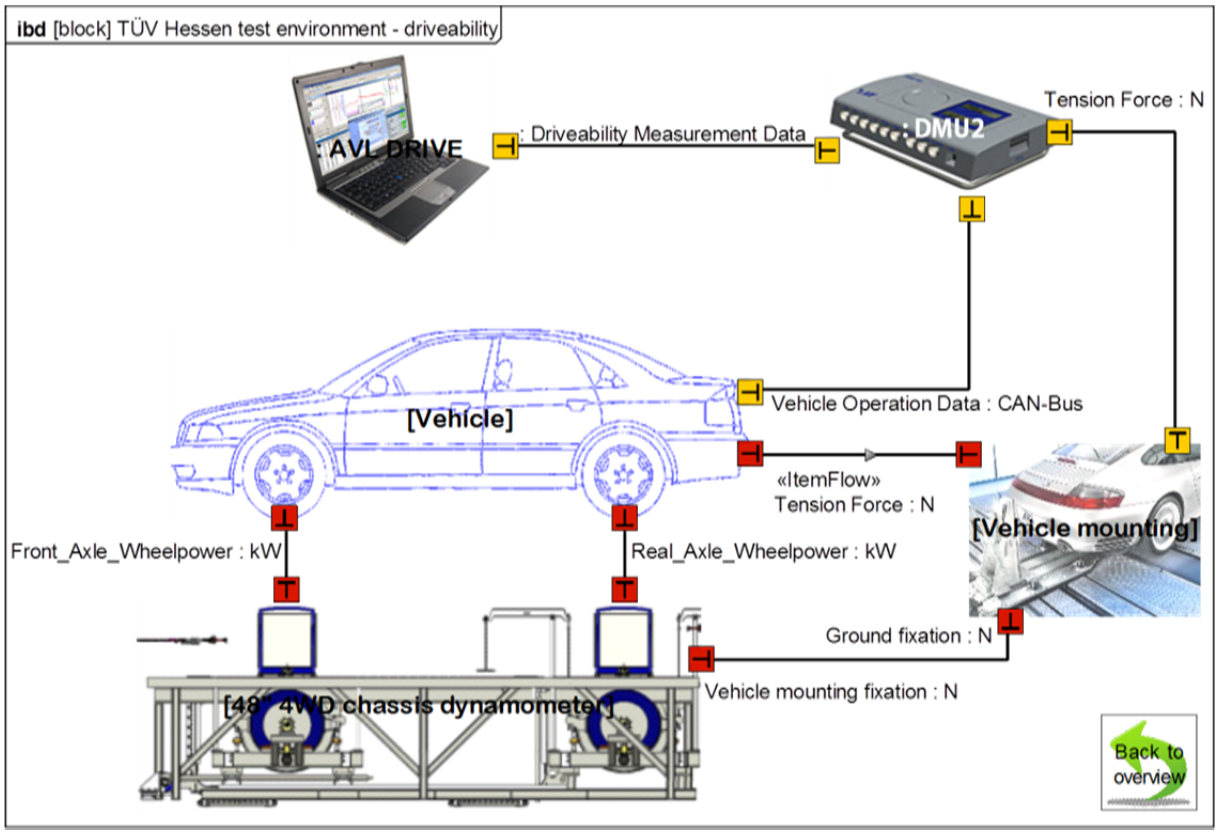


Bild 7-7: Systemkonfiguration für Fahrbarkeitsmessungen auf dem Rollenprüfstand

Die Modellierungstechnik wurde hier für die retrospektive Modellierung der Produkte des Unternehmens und ihrer Funktionen zur Konfiguration von Gesamtfahrzeug-Entwicklungsumgebungen eingesetzt. Die Konfiguration einer kundenspezifischen Systemlösung entspricht nicht dem vollen Produktentwicklungsprozess, sondern beschränkt sich auf die Aktivitäten der Zielsystemspezifikation und die anschließende Auswahl der Systemkomponenten auf Basis ihrer Funktionen. Demzufolge kann hier vielmehr der funktionsbasierte Produktportfolio-Lenkungsprozess (vgl. Bild 6-32) zugrunde gelegt werden, denn aus Sicht des Unternehmens ist entscheidend, ob mit dem bestehenden Produktportfolio bereits alle Kundenanwendungsfälle und Interessensvertreterziele erfüllbar sind oder ob eine individuelle Lösung entwickelt wird bzw. sogar das Produktportfolio entsprechend erweitert werden sollte. Es ist folglich zu entscheiden, mit welchem Aufwand eine Kundenlösung realisiert wird: mit geringerem Aufwand individuell für den einen speziellen Kunden oder die deutlich aufwändigere Anpassung des Produktportfolios. Diese Art von Entscheidung ist Teil des Portfoliolenkungsprozesses und daher ein gutes Anwendungsbeispiel des in Kapitel 6.5.3 vorgestellten Frameworks der funktionsorientierten Lenkung mechatronischer Produkte. Die Notwendigkeit der dort angestrebten Modellkopplung ergibt sich aus der Feststellung, dass die Modellierungstechnik für SysML allein zwar die Konfiguration einer Systemlösung unterstützt, jedoch nicht alle erforderlichen

Informationen für strategische Entscheidungen wie die Portfoliolenkung bereitstellen kann.

Dennoch hat das Feedback der Anwender im Partnerunternehmen ergeben, dass ein signifikanter Mehrwert einer solchen Modellierungstechnik gegeben ist. Dies belegen auch die folgenden, anonymisierten Zitate von unternehmensinternen Modellanwendern[492]:

- „Eine Use Case Analyse bei komplexen Projekten ist sehr sinnvoll, da man nur so ein Verständnis bekommt, was der Kunde wirklich will und wie die produkt- und applikationsübergreifenden Anwendungsfälle sind."
- „Das Wissen, was in der Köpfen einzelner Leute ist wird nachhaltig, modellbasiert gespeichert und ist somit wieder abrufbar."
- „Ein Use Case Modell hilft auch bei der Entwicklung von mechatronischen Systemen. Was und wie der Kunde verschiedene Dinge mit dem System tun möchte sind bedeutend – eine strukturierte Erfassung essentiell."
- „Wechselwirkungen oder Überlappungen bei Anwendungsfällen können einfach erkannt werden."

Bisher wurden nur einige Anwendungsfälle, Funktionen und Produkte des Unternehmens mit der Modellierungstechnik abgebildet, jedoch plant das Unternehmen aufgrund seiner guten Erfahrungen, dies mittelfristig für das gesamte Produktportfolio umzusetzen. Derzeit laufen weitere Pilotprojekte in verschiedenen Anwendungsbereichen mit bislang stark überwiegend positivem Anwenderfeedback.

7.3 MMI-Modellierung eines Rollenprüfstands-Automatisierungssystems

Die retrospektive Modellierung des Mensch-Maschine-Interfaces (MMI) des derzeitigen Automatisierungssystems wurde als Teil der Zielsystembeschreibung der neuen Prüfstandsautomatisierungsplattform eingesetzt. Sie ist damit eine konsequente Fortführung der im vorangegangenen Kapitel vorgestellten Validierung, die auf einem relativ geringen Detaillierungsgrad zwar den Anwendungsfall der Systemkonfiguration adäquat unterstützt, jedoch in der Form zu generisch wäre, um beispielsweise das Zielsystem eines neu zu entwickelnden Produkts ausreichend detailliert zu beschreiben. In diesem Kapitel wird die Modellierungstechnik daher auf einem sehr hohen Detaillierungsgrad eingesetzt und hinsichtlich ihrer Eignung für diesen Anwendungsfall validiert. Da ein reales Zielsystem für ein so umfangreiches

[492] aus Düser (2012)*, eine nicht veröffentlichte, unternehmensinterne Ergebnispräsentation

Produkt wie die neue Automatisierungsplattform jedoch sehr umfangreich ist, wurden hier nicht alle Aktivitäten der Modellierungstechnik angewendet, sondern eine Fokussierung auf die Beschreibung von MMI-Funktionalitäten des bisherigen Rollenprüfstands-Automatisierungssystems vorgenommen. Die aus dem entstehenden Modell resultierenden Diagramme (bzw. Sichten) wurden durch Entwickler im Unternehmen in die Anforderungsliste eingepflegt. Diese Beschreibung sollte dem Aufbau eines Systemverständnisses der Softwareentwickler der neuen Automatisierungsplattform bezüglich der zu überführenden Funktionen dienen. Ziel ist, das bisher isolierte Automatisierungssystem in die neue Plattform zu integrieren, die auch die Basis für Automatisierungssysteme anderer Prüfstandarten des Unternehmens wie beispielsweise Motoren- oder Antriebsstrangprüfstände ist. Da die Entwickler der neuen Plattform nicht an der Entwicklung der alten Plattform beteiligt waren, mussten diese rein auf Basis der modellierten Diagramme nachvollziehen, wie die derzeitige Regelung eines Rollenprüfstandes softwareseitig abläuft.

Zunächst wurde hierzu eine Paketstruktur des Modells aufgebaut, die nach Funktionsbereichen untergliedert ist. Hierzu gehören beispielsweise Parametrierungen, Automatisierte Prüfläufe, Performance Tests, Servicefunktionen und viele weitere. Darunter wurden dann die Funktionen als *Activities* in *Block Definition Diagrams* abgelegt, wie Bild 7-8 beispielhaft zeigt.

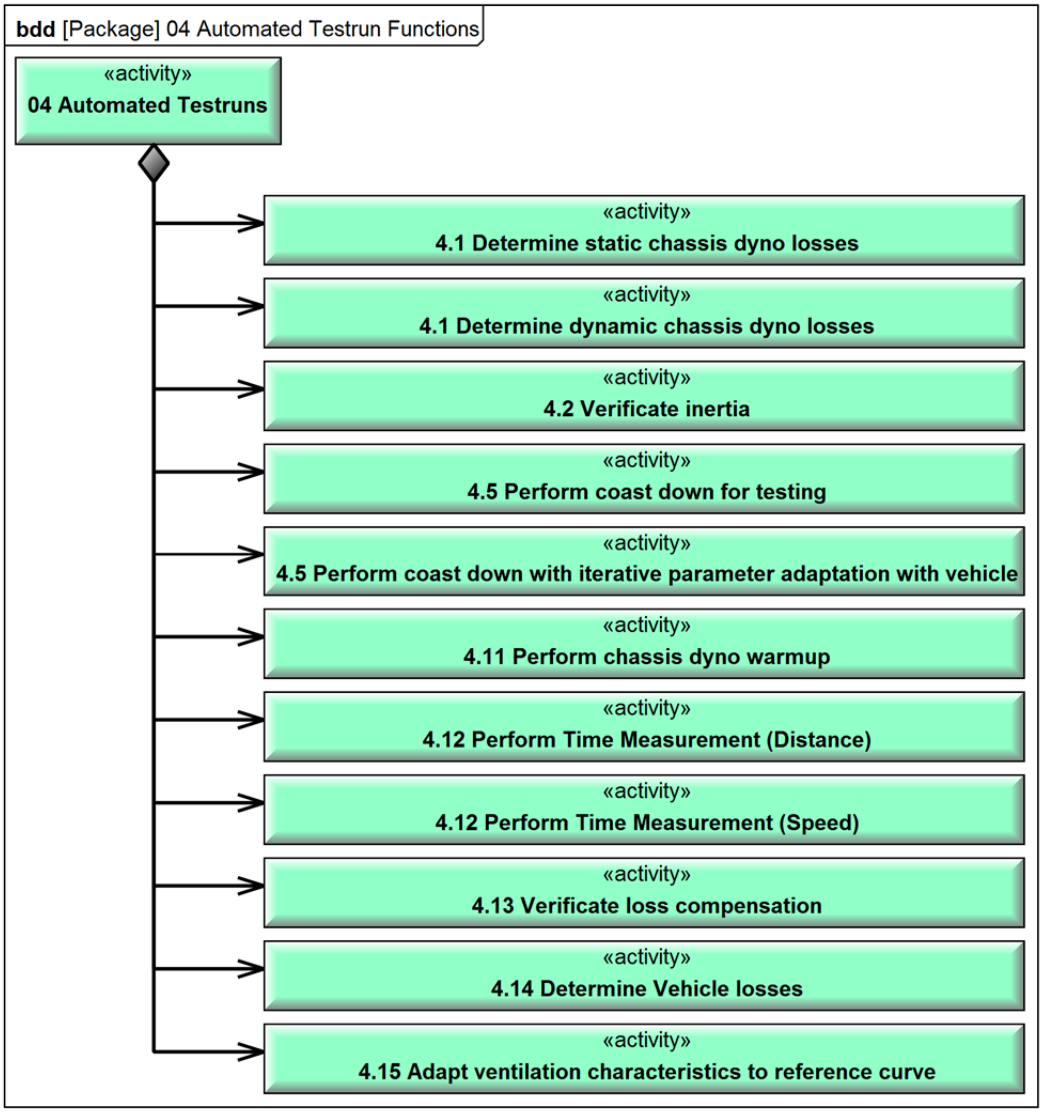


Bild 7-8: Hierarchische Struktur der Funktionen des bisherigen Prüfstandsautomatisierungssystems

Jede dieser Funktionen wurde anschließend in Activity Diagrams ausmodelliert, wobei wiederkehrende bzw. mehrfach verwendete Subfunktionen in einer Bibliothek abgelegt wurden und bei jeder Verwendung in einem neuen Kontext neu als *Call Behavior Action* instanziiert wurden. So konnten Doppelentwicklungen eigentlich identischer Funktionen bereits im Vorhinein vermieden werden. Beispielsweise wird innerhalb automatisierter Prüfläufe wie dem Warmlauf („Chassis Dyno Warmup“) öfters der Betriebsmodus des Prüfstands gewechselt (hellblau im in Bild 7-9 gezeigten, zugehörigen *Activity Diagram* hervorgehoben).

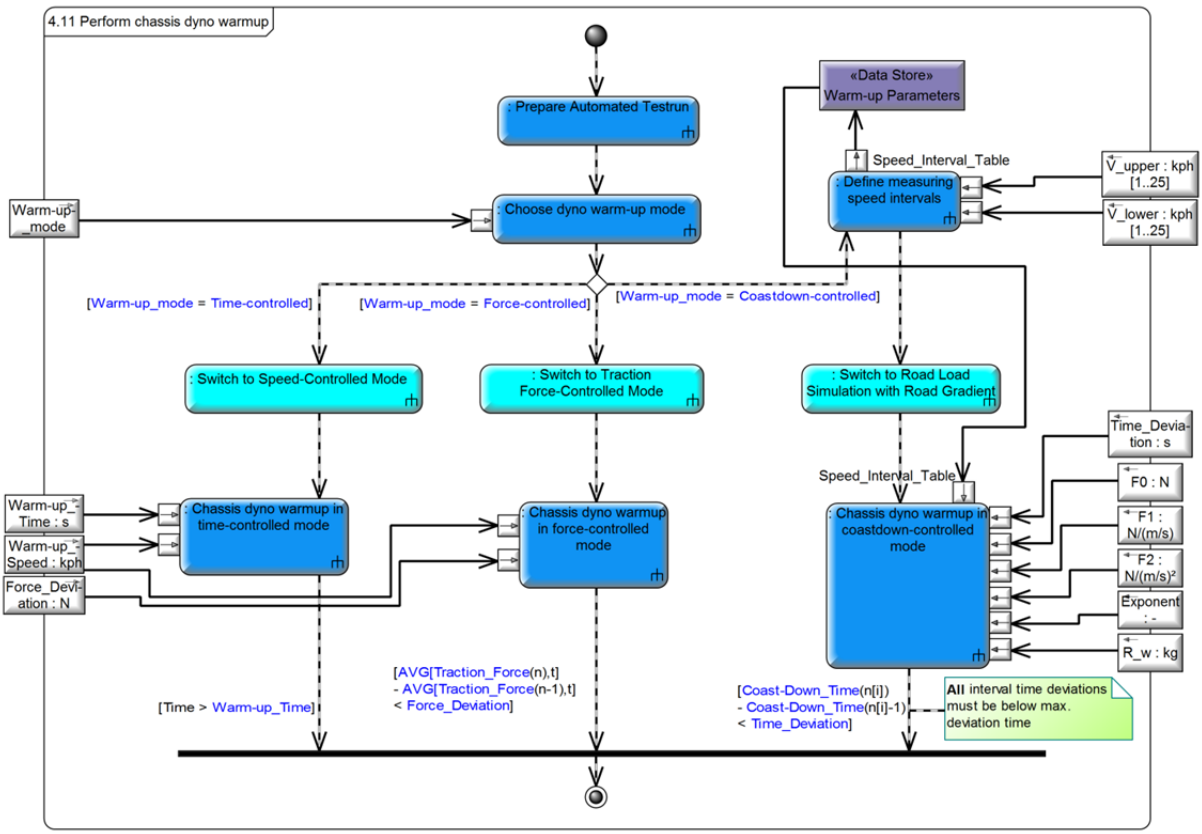


Bild 7-9: Beschreibung der Kontroll- und Objektflüsse der Funktion „Chassis Dyno Warmup“

Das Diagramm beschreibt neben den Kontrollflüssen, also dem logischen Ablauf der Aktivitäten innerhalb der Funktion „Chassis Dyno Warmup“, auch die Objektflüsse sowie Bedingungen für den logischen Ablauf. Auf diese Weise konnten viele für die Softwareentwickler relevanten Informationen beschrieben werden. Insgesamt wurden auf diese Weise fast 70 Diagramme mit teilweise deutlich mehr Elementen als das gezeigte Beispiel erzeugt, die andernfalls manuell in Visio oder anderen Werkzeugen hätten erstellt werden oder gar textuell beschrieben werden müssen. Die Gefahr von Redundanzen, Widersprüchen und Fehlern ist auf dieser Detaillierungsebene besonders hoch. Daher war der Mehrwert sowohl seitens der Anwender, die unter Zuhilfenahme der Diagramme die Anforderungsspezifikation erstellten, als auch seitens der Softwareentwickler, die diese Diagramme zum Aufbau des Systemverständnisses einsetzen konnten, schnell ersichtlich, wie auch die nachfolgenden, anonymisierten Zitate erneut belegen[493].

- „Man ist gezwungen, Funktionalitäten konsistent zu beschreiben. Wiederverwendung von Subfunktionen ist transparent und nachvollziehbar.“

[493] aus Düser (2012)*, eine nicht veröffentlichte, unternehmensinterne Ergebnispräsentation

- „Da ich schon manche URS[494] gelesen habe, bin ich vom neuen Approach mit Flussdiagrammen hellauf begeistert. Da ich selbst von der Entwicklerseite komme, sind diese SysML-Diagramme die reinste Wohltat. Der logische Ablauf ist leicht verständlich dargestellt und Unklarheiten sind überaus schnell geklärt. Alles in allem eine extreme Aufwandseinsparung (hauptsächlich zeitlich und programmiertechnisch) mit großem Nutzen."

7.4 Funktionale Modellierung von Laserschneidmaschinen

Neben den verschiedenen zuvor vorgestellten Anwendungsbereichen wurde die Modellierungstechnik in einem weiteren Kontext bei einem anderen Industriepartner[495] eingesetzt. Im Rahmen der Diplomarbeit von MARTINI wurde eine Methode zur funktionalen Produktarchitekturmodellierung am Beispiel einer 2D-Flachbettlasermaschine auf Basis eines Entwicklungszwischenstandes der hier vorgestellten Modellierungstechnik erarbeitet[496]. Die Herausforderung in diesem Anwendungsfeld besteht einerseits in der Komplexität der mechatronischen Systeme, in denen besonders der Bereich Optik aufgrund der eingesetzten, verschiedenen Lasertechniken eine zentrale Rolle in der Funktion der Schneidmaschinen spielt. Andererseits gibt es zahlreiche Varianten, die letztlich der Grund ist, warum nahezu jede verkaufte Maschine eine individuelle Kundenlösung ist. Die Zielsetzung des Unternehmens ist der Übergang weg vom baugruppenorientierten hin zu einem für Kunden transparenteren und funktionsbasiert strukturierten Produktbaukastensystem, der neben einer klaren funktionalen Unterscheidbarkeit von Baureihen und Varianten auch den Entwicklungsaufwand zukünftiger Systeme reduzieren soll. Die Arbeit wurde begleitend zur Konzeptphase eines Lead-Entwicklungsprojekts einer neuen Produktgeneration, die erstmals auf Baukastenbasis entstehen soll, durchgeführt und konnte so die tatsächlichen Herausforderungen der Entwickler erfassen.

Der erste Arbeitsschritt war die retrospektive, funktionsbasierte Modellierung eines bestehenden Produkts, um daraus einerseits eine modulare Funktionsbibliothek und andererseits eine Methode zur prospektiven, funktionsbasierten Modellierung neu zu entwickelnder bzw. zu konfigurierender Produkte abzuleiten (siehe Bild 7-10).

[494] User Requirement Specification = Anwenderanforderungsspezifikation
[495] TRUMPF Laser- und Systemtechnik GmbH, Ditzingen
[496] Martini (2012)

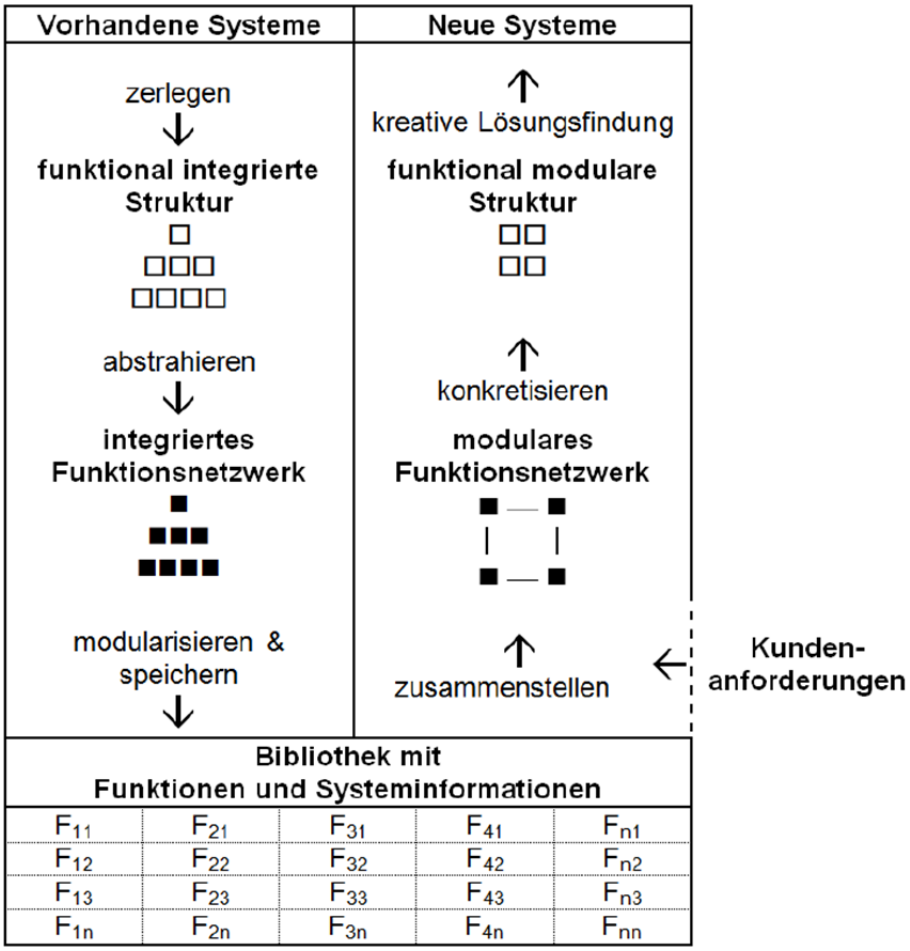


Bild 7-10: Vorgehensweise zur Erzeugung von Bibliothekselementen durch Zerlegung vorhandener Systeme und Generierung neuer Systeme durch Zusammenstellen von Bibliothekselementen[497]

Weiterhin wurde eine Methode erarbeitet, wie eine modellbasierte Entwicklung im Unternehmen eingeführt und angewendet werden kann. Dabei lag der Fokus auf dem durchgängigen Einsatz von fachdisziplinübergreifenden Modellen über mehrere Projekte hinweg, um den Nutzen einer kontinuierlich wachsenden Funktionsbibliothek ausschöpfen zu können. Auf diesem Weg soll der Wissensspeicher kontinuierlich ausgebaut werden. Bild 7-11 zeigt den schrittweisen Aufbau von der Einführung über den Aufbau von Funktionsbibliotheken, die Produktentwicklung bis hin zur kontinuierlichen Pflege und Erweiterung der hinzukommenden Produktgenerationen und Produktvarianten.

[497] Martini (2012)

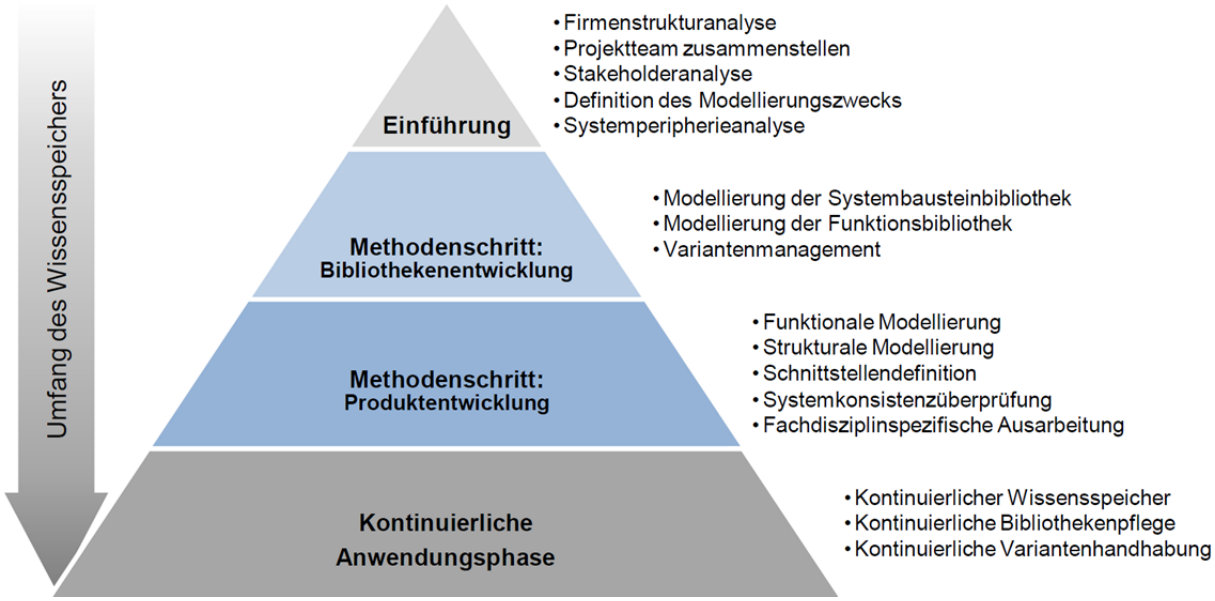
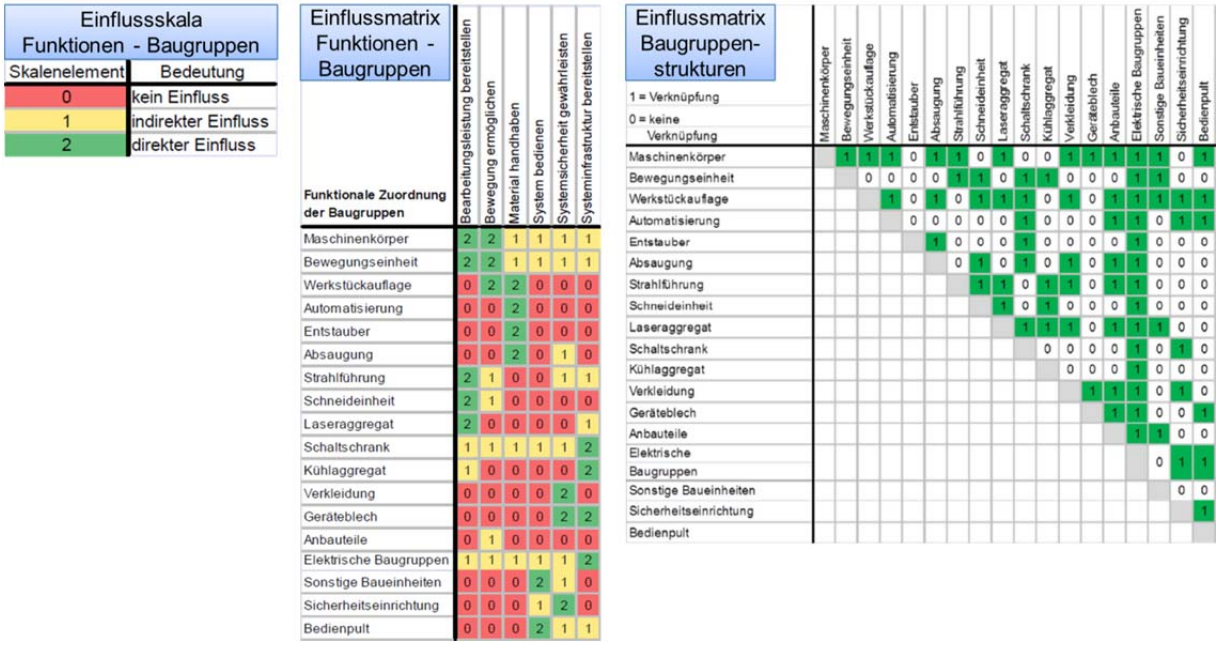


Bild 7-11: Aufbau der Methode von der Einführung bis hin zur kontinuierlichen Nutzung[498]

Der Aufbau einer modularen Produktstruktur auf Basis der Funktionen einzelner Baugruppen wurde im Rahmen der retrospektiven Modellierung durch Einflussmatrizen strukturiert, mittels denen qualitativ indirekte und direkte Wechselwirkungen zwischen Baugruppen und Hauptfunktionen sowie strukturelle Abhängigkeiten zwischen Baugruppenstrukturen erfasst wurden (siehe Bild 7-12).



Einflussskala Funktionen - Baugruppen

Skalenelement	Bedeutung
0	kein Einfluss
1	indirekter Einfluss
2	direkter Einfluss

Einflussmatrix Funktionen - Baugruppen

Funktionale Zuordnung der Baugruppen	Bearbeitungsleistung bereitstellen	Bewegung ermöglichen	Material handhaben	System bedienen	Systemsicherheit gewährleisten	Systeminfrastruktur bereitstellen
Maschinenkörper	2	2	1	1	1	1
Bewegungseinheit	2	2	1	1	1	1
Werkstückauflage	0	2	2	0	0	0
Automatisierung	0	0	2	0	0	0
Entstauber	0	0	2	0	0	0
Absaugung	0	0	2	0	1	0
Strahlführung	2	1	0	0	1	1
Schneideinheit	2	1	0	0	0	0
Laseraggregat	2	0	0	0	0	1
Schaltschrank	1	1	1	1	1	2
Kühlaggregat	1	0	0	0	0	2
Verkleidung	0	0	0	0	2	0
Geräteblech	0	0	0	0	2	2
Anbauteile	0	1	0	0	0	0
Elektrische Baugruppen	1	1	1	1	1	2
Sonstige Baueinheiten	0	0	0	2	1	0
Sicherheitseinrichtung	0	0	0	1	2	0
Bedienpult	0	0	0	2	1	1

Einflussmatrix Baugruppen-strukturen

1 = Verknüpfung
0 = keine Verknüpfung

	Maschinenkörper	Bewegungseinheit	Werkstückauflage	Automatisierung	Entstauber	Absaugung	Strahlführung	Schneideinheit	Laseraggregat	Schaltschrank	Kühlaggregat	Verkleidung	Geräteblech	Anbauteile	Elektrische Baugruppen	Sonstige Baueinheiten	Sicherheitseinrichtung	Bedienpult
Maschinenkörper		1	1	1	0	1	1	0	1	0	0	1	1	1	1	0	1	
Bewegungseinheit			0	0	0	0	1	1	0	1	1	0	0	0	1	1	0	0
Werkstückauflage				1	0	1	0	1	1	1	0	1	0	1	1	1	1	1
Automatisierung					0	0	0	0	0	1	0	0	0	1	1	0	1	1
Entstauber						1	0	0	0	1	0	0	0	0	1	0	0	0
Absaugung							0	1	0	1	0	1	0	1	1	0	0	0
Strahlführung								1	1	0	1	1	0	1	1	0	0	0
Schneideinheit									1	0	1	0	0	0	1	0	0	0
Laseraggregat										1	1	1	0	1	1	1	0	0
Schaltschrank											0	0	0	0	1	0	1	0
Kühlaggregat												0	0	0	1	0	0	0
Verkleidung													1	1	1	0	1	0
Geräteblech														1	1	0	0	1
Anbauteile															1	1	0	0
Elektrische Baugruppen																0	1	1
Sonstige Baueinheiten																	0	0
Sicherheitseinrichtung																		1
Bedienpult																		

Bild 7-12: Einflussmatrizen zur Modularisierung der Produktstruktur[498]

[498] Martini (2012)

Daraus ergab sich die Struktur der retrospektiv modellierten, alten Produktgeneration „L20“, die in SysML unter Anwendung des damaligen Stands der Modellierungstechnik beschrieben wurde und der Stückliste im SAP-System des Unternehmens mit teilweiser Zusammenfassung in abstrakte Baugruppen entspricht (siehe Bild 7-13).

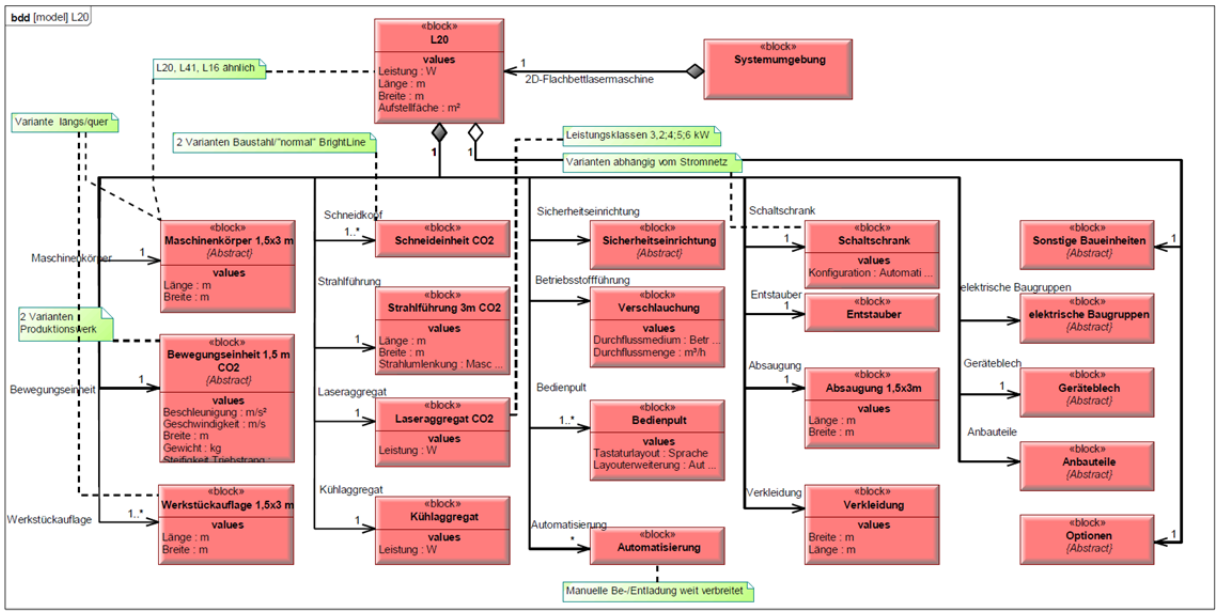


Bild 7-13: Produktstruktur der Generation „L20“ im *Block Definition Diagram*[499]

Aus der Struktur dieser Produktgeneration, deren interne Vernetzung über Schnittstellen auch in *Internal Blockdiagrams* ausgearbeitet wurde sowie aus einer Mindmap-Sammlung von Funktionen konnte eine erste Funktionsbibliothek abgeleitet und in SysML in Form von *Activities* modelliert werden (siehe Bild 7-14).

In der Entwicklung der neuen Produktgeneration auf Basis der Funktionsbibliothek wurden neben den Anforderungen insbesondere Funktionsabläufe in *Activity Diagrams* modelliert. Die modellierten Aspekte eröffneten die Möglichkeit zur Abgrenzung der Qualität der Funktionserfüllung[500] nach Produktvarianten und der Identifikation von zu verbessernden oder konkurrierenden Funktionen in Abhängigkeit der Konfiguration der gewählten oder neu zu entwickelnden Produktkomponenten. Die Möglichkeit zur Modellierung dynamischer Strukturen in Abhängigkeit der Systemzustände (vgl. Kapitel 6.3) war zu diesem Zeitpunkt noch nicht verfügbar, weshalb auf die Modellierung von Zuständen verzichtet wurde. Dies würde in Zukunft die Möglichkeiten zur Strukturierung möglicher bzw. funktional

[499] Martini (2012)

[500] Gemeint ist damit, wie gut eine Funktion ausgeführt wird, bspw. wie genau eine Verfahreinheit mit unterschiedlichen technischen Konzepten agiert oder welche Materialien oder Blechdicken verarbeitet werden können.

sinnvoller Produktkonfigurationen weiter ausbauen. Letztlich wurde für die modellierten Aspekte und Teilsysteme ein besseres Systemverständnis aufgebaut und ein höherer Modularisierungsgrad erreicht.

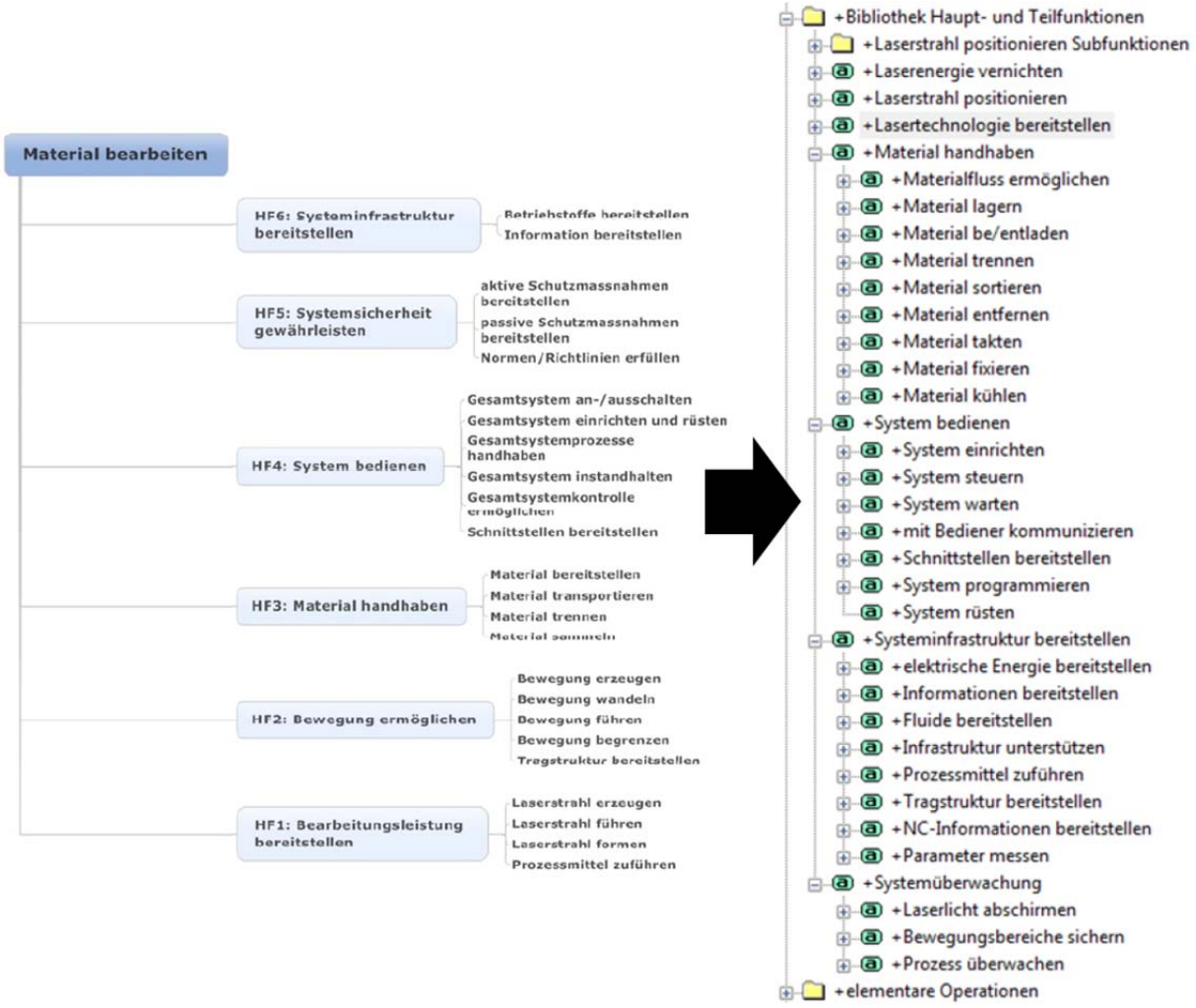


Bild 7-14: Ableitung einer Funktionsbibliothek der alten Produktgeneration[501]

Der identifizierte Mehrwert der modellbasierten Entwicklung lag insbesondere in der Eignung der SysML als Kommunikationswerkzeug im Projektteam, besonders zur Steigerung des Systemverständnisses in der Konzipierung neuer Produkte. Weiterhin wurden die Nachvollziehbarkeit und der Übertragungspfad zwischen modellierten Aspekten wie Anforderungen, Funktionen und Strukturen als zeitsparende Maßnahme bei der Informationsbeschaffung positiv gewertet. Darüber hinaus wurde das Potential zur einheitlichen Beschreibung von Prüfverfahren und Testfällen erkannt, die mögliche, negative Auswirkungen von Änderungen frühzeitiger aufdecken können. Allerdings wurde auch festgestellt, dass das volle

[501] Martini (2012)

Potential einer modellbasierten Entwicklung erst bei konsequentem Einsatz über mehrere Entwicklungsprozesse hinweg ausgeschöpft werden kann, da erst dann der Aspekt der Wiederverwendung von Informationen maßgeblich zum Tragen kommt. Aufgrund signifikanter Kosten durch organisatorische Umstrukturierungen, Softwareanschaffung und Mitarbeiterschulungen bestehen Bedenken, dass heutige Modellierungswerkzeuge den Anforderungen aus hoher Produktkomplexität und heterogenen Entwicklerteams durch unzureichende Funktionalität und Anwenderfreundlichkeit noch nicht gerecht werden und eine Einführung daran scheitern kann.

7.5 Diskussion und Zwischenfazit

Die Validierung der Modellierungstechnik fand größtenteils parallel zu ihrer kontinuierlichen Weiterentwicklung statt, weshalb in allen Anwendungsfällen eine Fokussierung auf Teilaktivitäten der Technik stattfand. Insbesondere das erst kürzlich entwickelte Plug-In für das Modellierungswerkzeug MagicDraw, das maßgebliche Neuerungen durch die Erkenntnisse aus den Validierungsszenarien beinhaltet, konnte noch nicht umfassend validiert werden. Das Potential der Modellierungstechnik wurde dennoch von allen Anwendern erkannt, die Evaluation wird ebenso in beiden Unternehmen weiterverfolgt. Die Hauptkritik wurde weder an der Sprache noch an Vorgehensweisen oder Modellierungsregeln geäußert, sondern insbesondere an den teilweise sehr unterschiedlichen Funktionalitäten der Modellierungswerkzeuge.

Die Modellierungstechnik hat für die retrospektive Modellierung im Rahmen einer Analyse bereits einen hohen Reifegrad erreicht und ist gut auf verschiedene technische Systeme anwendbar. Die Synthese innovativer Systeme unter Einsatz der in Kapitel 6.3 vorgestellten Modellierung von Aktivitäten, deren funktionaler Gruppierung zur Beschreibung der Wirkstruktur und dem Ausleiten der resultierenden physischen Struktur ist noch an realen Systemen umfassend zu validieren. Diese erfordert jedoch weitere Entwicklungsaktivitäten und Tests der hierzu notwendigen Erweiterung der Modellierungswerkzeuge, bevor diese auch die volle Funktionalität ausreichend stabil umsetzen können.

Auch die SysML selbst wird kontinuierlich durch Arbeitsgruppen der INCOSE sowie die OMG weiterentwickelt, was einerseits deren Funktionsumfang erweitert, jedoch andererseits auch die Gefahr birgt, dass zuvor entstandene Modelle nicht mehr funktionieren oder zumindest angepasst werden müssen. So wurde im Juni 2012 mit der SysML 1.3 ein neues Konzept für Ports in Strukturdiagrammen vorgestellt (vgl. Kapitel 2.6.2.4), was je nach Realisierung der neuen Spezifikation in den Modellierungswerkzeugen umfassende Anpassungen existierender Modelle erfordern kann. Die SysML in der Version 1.4 hat das Ziel, die Leistungsfähigkeit der

Variantenmodellierung deutlich zu verbessern. Dies birgt ein großes Potential, besonders im Hinblick auf die Ziele der Unternehmen bezüglich der system- und projektübergreifenden Wiederverwendbarkeit von Informationen und deren effizienter Verwaltung. Auch hier sind wiederum größere Änderungen der Modellierungswerkzeuge zu erwarten, zumal jeder Werkzeuganbieter andere Implementierungsansätze zur Umsetzung der neuen Spezifikation wählt.

Zusammenfassend kann somit festgehalten werden, dass der Übergang von der dokumentenbasierten zur modellbasierten Entwicklung einen zentralen Fortschritt bei der Handhabung der Informationsvielfalt und der durch Heterogenität von Entwicklerteams entstehenden Komplexität darstellen kann. Von essentieller Bedeutung sind eine wohldurchdachte, schrittweise Einführung von modellbasierter Systementwicklung und eine eingehende Evaluation des Modellierungswerkzeugs hinsichtlich der eigenen Anforderungen und Präferenzen. Aus wissenschaftlicher Sicht sind weiterhin zahlreiche Fragestellungen offen, die im nachfolgenden Kapitel im Anschluss an die Zusammenfassung diskutiert werden.

8 Zusammenfassung und Ausblick

In diesem Kapitel werden die Ergebnisse der Arbeit zusammengefasst und der in Kapitel 3.2 formulierten Zielsetzung gegenübergestellt sowie mit dem in Kapitel 4.1 formulierten Zielsystem der Modellierungstechnik abgeglichen.

8.1 Abgleich von Ziel- und Objektsystem der Modellierungstechnik

In den vorangegangenen Kapiteln wurde zunächst eine Basisdefinition für eine gemeinsame Sprache der Produktentwicklung zur Modellbildung technischer Systeme vorgestellt, die insbesondere der Forschungshypothese 1 (vgl. Kapitel 3.2) gerecht werden und einen homogenen Kern fachdisziplinübergreifender Kommunikation bilden soll. Die Evaluationsergebnisse haben gezeigt, dass die Bildung eines einheitlichen Begriffsverständnisses durch Begriffsdefinitionen, grafische Darstellungen mit semantischen Zusammenhängen sowie deren Integration in eine formale Modellierungssprache maßgeblich zu einer eindeutigen und fachdisziplinübergreifend verständlichen Modellierung technischer Systeme beiträgt. Die Unterscheidung in verschiedene, aufeinander aufbauende Aspekte und deren Vernetzung sowie eine grafisch verständliche Aufbereitung konnten dazu beitragen, anwenderfreundliche Modelle zu erzeugen, unter deren Verwendung die nachhaltige Kommunikation – wie in Forschungshypothese 2 gefordert – erreicht werden konnte. Der flexible Charakter der SysML als Basissprache sowie die Kopplung zu weiteren Standards ermöglichen eine flexible Modellierung fachdisziplinübergreifend relevanter Ziel- und Objektsystemelemente (vgl. Kapitel 6.2 bzw. 6.3) und deren Wechselwirkungen (vgl. Kapitel 6.4). Die Integration der Konzepte des Contact & Channel – Ansatzes ermöglicht zudem eine Maximierung des Lösungsraums bei der Synthese technischer Systemarchitekturen. Die darüber hinaus in Forschungshypothese 3 geforderte gezielte Beschränkung auf einen definierten Abstraktionsgrad ist projektspezifisch zu ermitteln. Der Übergang kann jedoch mittels der entwickelten Modellierungstechnik realisiert werden, die durch die vielfältigen, in Kapitel 6.5 vorgestellten Kopplungsmöglichkeiten die Konsistenzerhaltung zwischen Modellen durch Synchronisation mit fachdisziplinspezifischen Modellen ermöglicht. Dadurch wird gleichermaßen Forschungshypothese 4 berücksichtigt, die eine Integration des fachdisziplinübergreifenden Modells in eine bestehende Modelllandschaft anstelle deren Ersatzes fordert. Insbesondere die Maßnahmen zur in Forschungshypothese 5 geforderten anwendergerechteren grafischen Aufbereitung von Sichten auf das fachdisziplinübergreifende Modell haben eine durchweg positive Resonanz erzielt,

wie die Ergebnisse der Validierung in den Kapiteln 7.2 bis 7.4 zeigten. Damit konnten auch die aus den Forschungshypothesen abgeleiteten und in Kapitel 4.1 formulierten Anforderungen erfüllt werden.

Bezüglich des Aufwandes zur Speicherung und Bereitstellung der Informationen ist die Einschränkung anzumerken, dass diese Forderung vor allem durch die Modellierungssoftware und ihren zugehörigen Modellverwaltungssystemen erfüllt wird. Hier konnte am Beispiel des in Kapitel 6.3 vorgestellten Plug-Ins für ein Modellierungswerkzeug aufgezeigt werden, dass durch Automatismen signifikante Zeitersparungen möglich sind. Darüber hinaus können Fehler bei der Modellierung präventiv durch Implementierung von Modellierungsregeln in das Softwarewerkzeug auf ein Minimum reduziert werden.

Zur anwender- und aufgabengerechten Anwendung der Modellierungstechnik wurde in Kapitel 6.1 ein flexibles Anwendungsframework vorgestellt, das neben den weitgehend flexiblen und unabhängig einsetzbaren Aktivitäten der Modellierung auch die Art der Modellnutzung in Abhängigkeit der Rolle eines Anwenders vorschlägt. Dieses Rahmenwerk ist selbstverständlich nicht erschöpfend konkret für die direkte Anwendung in einem Projekt, jedoch kann es durch seinen generischen Charakter wie gefordert generalisierbar auf ein breites Spektrum technischer Produkte angewendet und projektspezifisch konkretisiert werden. Hierzu bietet das Modell in Anlehnung an das Konzept des Sequenzmodells im Integrierten Produktentstehungsmodell iPeM auch die Möglichkeit zur Beschreibung von Referenz-, Implementierungs- und Applikationsmodellen. Diese ermöglichen neben der Planung auch den Abgleich mit den real durchgeführten Aktivitäten bei der Nachbereitung von Projekten und damit die Definition präziserer Handlungsempfehlungen für nachfolgende, ähnlich geartete Projekte.

Trotz der bisher erreichten Ziele verbleiben weitere offene Forschungsfragen, die sich im Rahmen dieser Arbeit herauskristallisierten, jedoch nicht abschließend beantwortet werden konnten bzw. explizit vom Arbeitsumfang abgegrenzt wurden. Das nachfolgende Kapitel greift diese Fragestellungen auf und erläutert kurz deren zentrale wissenschaftliche Herausforderungen.

8.2 Ausblick auf mögliche weiterführende Forschungsthemen

Basierend auf den im Verlauf dieser Arbeit gewonnenen Erkenntnissen werden nachfolgend wissenschaftliche Fragestellungen, teilweise mit ersten konzeptionellen Ansätzen vorgestellt, die nach Meinung des Autors eine sinnvolle Weiterführung der Forschungsarbeiten im Bereich Model-Based Systems Engineering darstellen.

8.2.1 Engere Kopplung von SysML und CAD

Mit der vorgestellten Modellierungstechnik besteht durch die Beschreibung dynamischer Strukturen und die Berücksichtigung funktionsrelevanter Merkmale und Eigenschaften in Form von Parametern die Möglichkeit, Geometrieinformationen im Modell zu hinterlegen und semantisch mit anderen Informationen wie den durch die Gestalt realisierten Funktionen zu koppeln. Allerdings ist eine geometrisch behaftete Darstellung weiterhin nicht möglich, die laut Studien neben Prototypen nach wie vor die primäre von Konstrukteuren verwendete Produktrepräsentation darstellt, insbesondere in der Ideenfindung und Prinzipmodellierung[502]. Der Abstraktionssprung von einer technischen Schnittzeichnung oder einer Skizze zu einem geometrieneutralen, zweidimensionalen Blockdiagramm ist sehr groß, wodurch viele mechanische Funktionen nur schwer darstellbar und nachvollziehbar sind. Ein möglicher Ansatz zu dessen Reduktion wäre das Einfügen einer weiteren Diagrammart zur Visualisierung geometrischer Anordnungen von Strukturen in einer Art „Gestaltdiagramm“, wie es auch die Vision einer integrierten Beschreibung und Darstellung von Funktion und Gestalt nach dem Contact & Channel – Ansatz (C&C²-A) nach ALBERS[503] vorsieht. Dies könnte durch eine Abstraktion von Schnittdarstellungen aus CAD-Modellen geschehen, indem beispielsweise die Darstellungen genormter Gestaltelemente wie Freistiche, Gewinde, Zähne, Welle-Nabe-Verbindungen oder Nuten abstrahiert werden. Weiterhin könnten Gestaltmerkmale für fertigungs- oder montagegerechte Konstruktion ebenfalls auf dieser Ebene abstrahiert werden (z.B. Einführschrägen, Fasen etc.). So würde ausschließlich die in der Nutzungsphase des Systems funktionsbestimmende Gestalt erhalten bleiben, deren Maße insbesondere mit CAD-Skelettmodellen synchronisiert werden. Bauteile von Baugruppen könnten semantisch mit den *Physical Components** der hier vorgestellten Modellierungstechnik gekoppelt werden, Oberflächen mit Ports[504].

Ein einfaches, konzeptionelles Beispiel der Kopplung eines „Shape Diagrams“ mit der physischen Struktur und eine mögliche, abstrahierte Darstellung geometrischer Aspekte zeigt Bild 8-1.

---

[502] Vgl. Hacker et al. (2002), Eckert et al. (2010), Hannah et al. (2012)

[503] Vgl. Kapitel 2.4.9

[504] Die technische Machbarkeit der Kopplung von CAD-Konstruktionselementen wie Oberflächen mit Elementen der SysML wurde in Kapitel 6.5.2 bereits aufgezeigt.

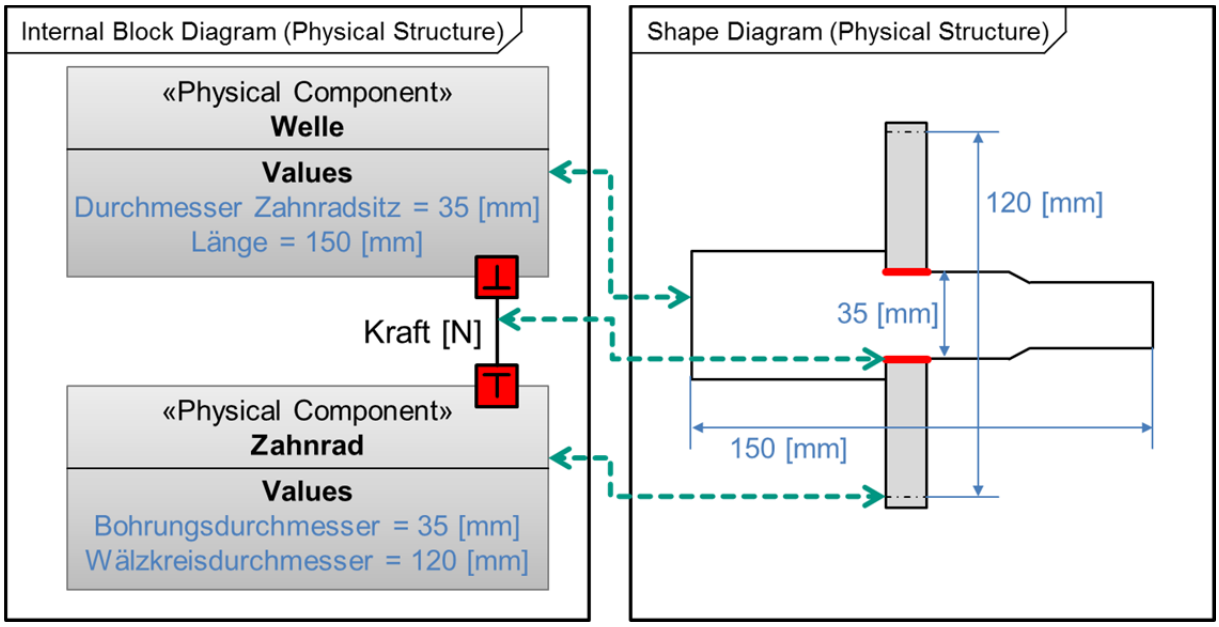


Bild 8-1: Konzept eines "Shape Diagrams"

Die zentrale wissenschaftliche Herausforderung dieses Konzepts liegt im Grad der Abstraktion von CAD zum „Gestaltdiagramm“, insbesondere im Falle komplizierter Freiformgeometrien zur Funktionserfüllung (bspw. Führungsnuten, Oberflächenprofile, Kabelführungen, Strömungskanäle, Kühlungs-, Schmier- und Dichtkonzepte). Weiterhin ist zu prüfen, inwieweit ein Standard wie STEP in der Lage ist, diese Informationen zwischen SysML und CAD mittels seiner heutigen Applikationsprotokolle auszutauschen und wie dieser ggf. weiterzuentwickeln wäre. Auch die Applikationsmöglichkeiten von Transformationsregeln zur Abstraktion oder weiterer Technologien zur Realisierung einer derartigen Modellkopplung wäre eingehend zu evaluieren. Dieses Konzept erstand aus den Schwierigkeiten von Konstrukteuren im Umgang mit SysML, wäre nicht nur eine eingehende wissenschaftliche Validierung des erreichten Mehrwertes notwendig, sondern auch weiterführende Untersuchungen hinsichtlich der Ursache dieser Schwierigkeiten.

Am IPEK wurden bereits verschiedene Vorarbeiten zur Entwicklung einer Softwarelösung für die Anwendung des Contact & Channel – Ansatzes (C&C²-A) durchgeführt[505]. Die Verknüpfung der dort gewonnenen Erkenntnisse mit den Ergebnissen dieser Arbeit im Rahmen einer wissenschaftlichen Weiterentwicklung könnte die Stärken der Ansätze kombinieren und deren Schwächen überwinden.

[505] Albers et al. (2009a), Albers et al. (2010a), Enkler (2010), Albers et al. (2011a), Albers et al. (2011b), Albers und Sadowski (2013)

8.2.2 Integrierte Modellierung des ZHO-Systems

In Kapitel 6.5.4 wurde ein Konzept für ein Rahmenwerk einer durchgängig IT-gestützten, modellbasierten Entwicklungsumgebung vorgestellt. Ein wissenschaftlicher Schwerpunkt liegt dabei in der Weiterentwicklung der hier vorgestellten Modellierungstechnik hinsichtlich einer durchgängigen, semantischen Kopplung von produktbezogenen Informationen (Ziel- und Objektsystem) mit dem Handlungssystem seines Entstehungs- und Lebenszyklusprozesses. Die Nutzung der Business Process Model and Notation (BPMN)[506] ist hier nur ein Vorschlag, da auch diese Sprache – ebenso wie die SysML – auf der Meta-Object Facility (MOF)[507] basiert und entsprechend ähnlich flexible Erweiterungsmechanismen wie die SysML bietet. Damit würden insbesondere die Arbeiten von ALBERS UND BRAUN sowie ALBERS ET AL. fortgeführt und sinnvoll mit den Ergebnissen dieser Arbeit verschmelzen[508]. Auch die Vorarbeiten von ALBERS ET AL. bezüglich der kontinuierlichen Zielsystemspezifikation und dem Umgang mit Unsicherheit und Unschärfe von Zielen in der Entwicklung komplexer Produkte sowie der Repräsentation von Wissen bieten Potential zur Weiterentwicklung der Modellierungstechnik, beispielsweise durch die Kopplung mit Anforderungs- und Wissensmanagementmethoden und –werkzeugen[509].

8.2.3 Modellbasiertes Produktvarianten- und -portfoliomanagement

Die derzeit noch eingeschränkten Möglichkeiten der SysML in der Produktvarianten- und Produktportfoliomodellierung stellen ebenfalls wissenschaftlichen Forschungsbedarf dar. Momentan befindet sich die Version 1.4 der SysML in der Entwicklung durch die SysML Revision Task Force der OMG[510]. Eine der zentralen, geplanten Neuerungen ist auch dort die Weiterentwicklung der Sprache hinsichtlich eindeutiger und durchgängiger Variantenmodellierung. Darüber hinaus präsentierte Weilkiens in seiner Methode SysMOD einen Ansatz zur Variantenmodellierung durch ein erweiterndes Profil[511]. Dies zeigt einerseits den Bedarf dieser Funktionalität auf, andererseits konnte sich bisher aber kein Ansatz verbreitet durchsetzen, was weiterhin bestehenden Forschungsbedarf begründet. Auch das in Kapitel 6.5.3 vorgestellte Rahmenwerk für die funktionale Lenkung mechatronischer Produkte durch einen übergeordneten Portfoliolenkungsprozess erfordert eine entsprechende

[506] Vgl. Kapitel 2.6.4
[507] Vgl. Kapitel 2.8.2
[508] Albers et al. (2011b), Albers et al. (2012a)
[509] Albers et al. (2011c), Albers et al. (2011f), Albers et al. (2012d)
[510] OMG SysML RTF (2012)
[511] Weilkiens (2008), Korff und Weilkiens (2010)

Werkzeugunterstützung, für die im Rahmen des fortlaufenden Forschungsprojekts „Funktionale Lenkung mechatronischer Produkte“ ein Konzept für einen technischen Demonstrator entwickelt wird[512]. Dieses Projekt liefert umfangreiche Basiserkenntnisse und Vorarbeiten, deren Fortführung in weiteren Forschungsprojekten im Hinblick auf die verbesserte Unterstützung von Anwendern beim Management von Produkt- und Technologieportfolios durch ein geeignetes Modellierungswerkzeug großes Potential für eine stärkere Verbreitung modellbasierter Produktentwicklungsmethoden bietet.

8.2.4 Frühzeitige Systemanalysen zur Konzeptabsicherung

Neben der reinen Beschreibung helfen frühzeitige Analysen, Produktentstehungsprozesse effizienter zu managen und Produktkonzepte frühzeitiger zu validieren und deren Funktionserfüllung abzusichern[513]. Bisher etablierte qualitative und quantitative Qualitätssicherungsanalysen wie die Fehlermöglichkeits- und Einflussanalyse (FMEA) oder auch die Fehlerbaumanalyse (FTA) bieten das Potential, auch auf Basis von Modellen ausgeführt zu werden. Erste Ansätze von DAVID ET AL., PIQUES UND ANDRIANARISON oder ARMENGAUD ET AL. zeigen die technische Machbarkeit automatisierter Modellanalysen anhand erster prototypischer Implementierungen auf[514]. Weiterentwicklungspotential liegt insbesondere in der methodischen Unterstützung bei der Anwendung dieser Analysen sowie in deren anwender- und aufgabengerechten Implementierung in fachdisziplinübergreifenden Systemmodellen.

8.2.5 Durchgängige Modellkopplung in Werkzeugketten

Aufbauend auf das in Kapitel 6.5.4 vorgestellte Konzept für ein Rahmenwerk durchgängiger IT-basierter Modellkopplung besteht umfangreicher Forschungsbedarf, wie diese Schnittstellen realisiert werden können und in welcher Aktivität des Produktentstehungsprozesses welche Modelle in welcher Form Informationen austauschen sollten, um die Effizienz der Produktentwicklung maßgeblich zu steigern. Weiterhin ist hierbei eine Methode unter Berücksichtigung von Art und Zweck der eingesetzten Modelle zu entwickeln, um den individuell optimalen Kompromiss für eine maximal effiziente Modellkopplung zu identifizieren. Die besondere Herausforderung liegt darin, einen gewissen Grad der Vereinheitlichung trotz heterogener Modelllandschaft zu realisieren, um die Kompliziertheit der Modellkopplung sowie deren Implementierungsaufwand möglichst

[512] Denger et al. (2012)
[513] Beispielsweise die funktionale Sicherheit von Straßenfahrzeugen, vgl. ISO-26262 (2012)
[514] David et al. (2010), Piques und Andrianarison (2012), Armengaud et al. (2012)

gering zu halten. In diesem Kontext wäre das Applikationsprotokoll 233 der ISO 10303[515] für den Austausch Systems Engineering-relevanter Daten zu analysieren und hinsichtlich seiner Anwendbarkeit zu evaluieren. Hilfreiche Vorarbeiten im Bereich der Modellkopplung sind neben den im Rahmen dieser Arbeit durchgeführten Untersuchungen[516] beispielsweise auch die Forschungsergebnisse von SHAH ET AL.[517] oder BROY ET AL.[518] sowie die Konzepte der im Rahmen des EU-Projekts CESAR entwickelte Reference Technology Platform (RTP)[519]. Ein weiteres, ebenfalls in dieser Arbeit vorgestelltes Werkzeug sind semantische Technologien[520]. Diese können zur übergreifenden Wissensrepräsentation genutzt werden und insbesondere die Kopplung und Extraktion management-relevanter Informationen realisieren. Auch hierzu existieren vielversprechende Vorarbeiten am IPEK[521].

8.2.6 Mensch-zentriertes Advanced Systems Engineering

Die bisher vorgestellten Themenvorschläge zur Fortführung dieser wissenschaftlichen Arbeit im Bereich Model-Based Systems Engineering fokussieren sich alle auf ein konkretes Teilthema zur Verbesserung der Effizienz beim Einsatz von Modellen in der Produktentstehung. Dem übergeordnet sind jedoch immer die Ziele der Anwenderunterstützung durch geeignete Methoden und Werkzeuge und damit einhergehend die Betrachtung des Menschen im Mittelpunkt. Hier wurden bereits umfassende Vorarbeiten am IPEK durchgeführt, deren Weiterführung insbesondere in Bezug auf SysML und angrenzende Technologien des MBSE essentiell für deren langfristigen Erfolg ist[522]. Weiterhin sollten die in Kapitel 11 vorgestellten Modellierungsregeln erweitert werden, um neben einem übergeordneten Vorgehensmodell auch die konkrete Unterstützung der Anwender beim Modellieren zu verbessern und Fehler proaktiv zu vermeiden. Beispiele für Modellierungsregeln, die auch in einem Modellierungswerkzeug als automatisch überprüfbare Regeln implementierbar wären, schlagen u.a. BERTSCHE ET AL. auf Basis ihres Schichtenmodells für Strukturen vor[523].

---

[515] ISO-10303-233 (2012)
[516] Bopp (2010), Gloderer (2010), Kruppok (2012), Bull (2012)
[517] Shah et al. (2009) , Shah et al. (2010)
[518] Broy et al. (2010)
[519] Griessnig et al. (2011), Armengaud et al. (2012)
[520] Vgl. Vorstellung in Kapitel 2.7 und konzeptionelle Integration in Kapitel 6.5.3
[521] Albers et al. (2011f)
[522] Albers et al. (2011c), Albers und Lohmeyer (2012), Albers et al. (2012e), Lohmeyer (2013)
[523] Bertsche et al. (2009)

8.2.7 Ausbildung von Systemingenieuren[524] in der universitären Lehre

Die zahlreichen wissenschaftlichen Herausforderungen im Bereich Systems Engineering und insbesondere Model-Based Systems Engineering als implementierungsnächster Teilbereich des systemischen Entwicklungsparadigmas erfordern interessierten und motivierten wissenschaftlichen Nachwuchs. Hier sieht der Autor der Arbeit noch umfangreichen Handlungsbedarf, die technischen Grundlagen und Fertigkeiten auch in Maschinenbaustudiengängen umfassender in das Lehrkonzept einzubeziehen. Ein moderner Ingenieur ist schon lange nicht mehr reiner Mechanikentwickler und –konstrukteur, sondern wird im beruflichen Alltag nahezu überall mit der Mechatronisierung technischer Systeme konfrontiert. Deren Grundverständnis erfordert neben der Stärkung der Informatik im Maschinenbau insbesondere auch die Vermittlung von Abstraktionskompetenzen und dem entsprechenden Handwerkszeug. Dies findet durch den vom IPEK gelehrten Contact & Channel – Ansatz (C&C²-A) zwar bereits statt, jedoch ist die Akzeptanz bei den Studenten weiterhin verbesserungsfähig, was unter anderem auch daraus resultieren kann, dass es bisher keinerlei etablierte IT-Unterstützung gibt. Dies könnte sich mit der hier vorgestellten Implementierung des C&C-A in der SysML ändern. Ein Grundstein könnte beispielsweise bereits im Grundstudium durch das Vermitteln der Grundlagen objektorientierter Modellierung, verbunden mit anwendungsnahen Lehrmethoden in der Konstruktionslehre, bestehend aus Vorlesung, Übung und IT-gestützten Workshops, bspw. gemäß dem KaLeP nach ALBERS[525], gelegt werden. Beispielsweise könnte die heutige CAD/PDM-gestützte Ausbildung, wie sie auch am IPEK unter Einsatz moderner IT-Infrastrukturen gelehrt wird[526], beispielsweise um die Einführung und Anwendung eines Systemmodellierungswerkzeugs ausgebaut werden.

Basierend auf diesen Grundlagen könnten weiterführende Lehrveranstaltungen fortgeschrittene Kompetenzen zum Paradigma des Systems Engineering, der Systemtheorie, der Modellierung in SysML oder Grundlagen zu angrenzenden Technologien und Standards, wie sie in der vorliegenden Arbeit vorgestellt werden, vermittelt werden. Verschiedene Universitäten und Fachhochschulen bieten bereits heute erste entsprechende Studienfächer an. Auch die Einführung eigener

---

[524] Angelehnt an das Verständnis des Aufgabenspektrums eines Systemingenieurs nach Ropohl (1975). Der Systemkonstrukteur wird hier als Teilgruppe des Systemingenieurs verstanden.

[525] Karlsruher Lehrmodell für Produktentwicklung, vgl. Albers et al. (2006)

[526] Albers et al. (2009c)

Vertiefungsrichtungen oder gar eines eigenen Studiengangs wird an verschiedenen Standorten in Deutschland diskutiert oder bereits angeboten[527].

Seit 2012 bietet die Gesellschaft für Systems Engineering e.V. (GfSE)[528] in Kooperation mit dem TÜV Rheinland als akkreditierte Zertifizierungsstelle eine berufsbegleitende Personalzertifizierung für Systems Engineers als „Certified Systems Engineer (GfSE)®“ an[529]. Darüber hinaus ist die GfSE interessierter Ansprechpartner für die Realisierung von Systems Engineering in der Lehre.

Systems Engineering ist ein Paradigma, das nicht nur die Entwicklungswerkzeuge und Methoden isoliert betrachtet, sondern das System aus Mensch, Technik und Umwelt (Infra- und Superstruktur). Die ganzheitliche, durchgängige und nachhaltige Optimierung dieses Systems mit dem Menschen im Mittelpunkt bietet nach Meinung des Autors dieser Arbeit das Potential für den größten Technologiesprung in der Produktentwicklung seit der Erfindung des Computers.

[527] Beispielsweise TU Chemnitz (2012), Uni Bremen (2012)
[528] Siehe GfSE Website (2012)
[529] Siehe SE-Zert (2012)

9 Literaturverzeichnis

Abiteboul et al (1995) Abiteboul, S.; Hull, R.; Vianu, V.: Foundations of Databases. Addison-Wesley, München, 1995

Abulawi (2012) Abulawi, J.: Ansatz zur Beherrschung der Komplexität von vernetzten 3D-CAD-Modellen. Dissertation im Fachgebiet Maschinenelemente und Rechnergestützte Produktentwicklung an der Helmut-Schmidt-Universität / Universität der Bundeswehr Hamburg. Hamburg, 2012

Ackermann (2007) Ackermann, R.R.: Elektronik bestimmt Innovationen im Auto der Zukunft. In: E&E Select Automotive. 2007

Aigner (1982) Aigner, J.: Zur zuverlässigen Beurteilung von Fahrzeugen. In: Automobiltechnische Zeitschrift (ATZ), Vol. 84, Nr. 9. Springer, Heidelberg, 1982

Albers (2010) Albers, A.: Five hypotheses and a meta model of engineering design processes. In: Proceedings of the 8th International Symposium on Tools and Methods of Competitive Engineering (TMCE ‘10). Ancona, Italien, 2010

Albers (2011) Albers, A.: Der Entwickler im Zentrum des Systems der Produktentstehung. In: Stuttgarter Symposium für Produktentwicklung (SSP). Stuttgart, 2011

Albers et al. (2005) Albers, A.; Burkardt, N.; Meboldt, M.; Saak, N.: SPALTEN Problem Solving Methodology in the Product Development. Proceedings of the 15th International Conference on Engineering Design (ICED ’05). Melbourne, Australien, 2005

Albers et al. (2006) Albers, A.; Burkardt, N.; Meboldt, M.: The Karlsruhe Education Model for Product Development “KaLeP” in Higher Education. In: Proceedings of the 9th International Design Conference (DESIGN ’06). Dubrovnik, Kroatien, 2006

Albers et al. (2008a) Albers, A.; Deigendesch, T.; Alink, T.: Support of Design Engineering Activity – The Contact and Channel Model (C&CM) in the Context of Problem Solving and the Role of Modelling. In: Proceedings of the 10th International Design Conference (DESIGN ’08). Dubrovnik, Kroatien, 2008

Albers et al. (2008b) Albers, A.; Alink, T.; Matthiesen, S.; Thau, S.: Support of System Analyses and Improvement in Industrial Design through the Contact & Channel Model. In: Proceedings of the 10th International Design Conference (DESIGN ’08). Dubrovnik, Kroatien, 2008

Albers et al. (2008c) Albers, A.; Matthiesen, S.; Thau, S.; Alink, A.: Support of Design Engineering Activity through C&CM – Temporal Decomposition of Design Problems. In: Proceedings of the International Symposium on Tools and Methods of Competitive Engineering (TMCE ‘08). Izmir, Türkei, 2008

Albers et al. (2009a) Albers, A.; Braun, A.; Clarkson, P.J.; Enkler, H.G.; Wynn, D.: Contact and channel modelling to support early design of technical systems. In: Proceedings of the 17th International Conference on Engineering Design (ICED ’09), Vol. 5. Stanford University, Kalifornien, USA, 2009

Albers et al. (2009b) Albers, A.; Hessenauer, B.; Maier, T.; Zingel, C.: Optimierungsbasierte Entwicklung von Prüfstandskomponenten unter dynamischen Aspekten. In: FVA – Kongress für Simulation im Produktentstehungsprozess (SIMPEP 2009). Veitshöchheim, 2009

Albers et al. (2009c) Albers, A.; Sauter, C.; Maier, T.; Geier, M.: KALEP – An engineering education model supported by modern IT-solutions. In: Proceedings of the International Conference on Engineering Education & Research (ICEE & ICEER 2009). Seoul, Korea, 2009

Albers et al. (2010a) Albers, A.; Braun, A.; Sadowski, E.; Wynn, D. C.; Wyatt D. F.; Clarkson, P.-J.: Contact and Channel Modeling using part and function libraries in a function-based design approach. Proceedings of the ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE ’10), Montreal, Quebec, Canada, 2010

Albers et al. (2010b) Albers, A.; Braun, A.; Muschik, S.: Uniqueness and the Multiple Fractal Character of Product Engineering Processes. In: Proceedings of the 1st International Conference on Modeling and Management of Engineering Processes (MMEP 2010). Cambridge, UK, 2010

Albers et al. (2010c) Albers, A.; Zingel, C.; Zehetner, J.: Automation of driveability researches in different test-environments using the example of a positive load change-maneuver (“Tip-In”). In: Proceedings des 10. Internationales Stuttgarter Symposium – Automobil- und Motorentechnik. Stuttgart, 2010

Albers et al. (2010d) Albers, A.; Zingel, C.; Zehetner, J.; Meitz, K.: Influence of low-frequency powertrain-vibrations on driveability-assessments. In: Proceedings of the 6th International Styrian Noise, Vibration & Harshness Congress (ISNVH 2010). Graz, Österreich, 2010

Albers et al. (2011a) Albers, A.; Enkler, H.-G.; Ottnad, J.: Managing complex simulation processes: the generalised contact and channel model. In: International Journal of Product Development, Vol. 13, Nr. 3. Inderscience Enterprises, Genf, Schweiz, 2011

Albers et al. (2011b) Albers, A.; Braun, A.; Sadowski, E.; Wynn, D. C.; Wyatt D. F.; Clarkson, P.-J.: System architecture modeling in a software tool based on the Contact and Channel Approach (C&C-A). In: Journal of Mechanical Design, Vol. 133, Nr. 10. ASME, New York, USA, 2011

Albers et al. (2011c) Albers, A.; Lohmeyer, Q.; Ebel, B.: Dimensions of Objectives in Interdisciplinary Product Development Projects. In: Proceedings of the 18th International Conference on Engineering Design (ICED ’11), Vol. 2. Kopenhagen, Dänemark, 2011

Albers et al. (2011d) Albers, A.; Zingel, C.; Maletz, M.: Interdisciplinary Functional Systems Modeling Approach Applied for Hybrid Powertrain Development. In: Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering and Computer Science (WCECS 2011). San Francisco, USA, 2011

Albers et al. (2011e) Albers, A.; Schroeter, J.; Behrendt, M.; Robens, G.; Zingel, C.: Methode zur Bestimmung einer optimierten Schaltstrategie. In: Konferenzband der Getriebe in Fahrzeugen. Friedrichshafen, 2011

Albers et al. (2011f) Albers, A.; Schmalenbach, H.; Lohmeyer, Q.: Ontology development for knowledge representation. In: International Journal of Product Development, Vol. 14, No. 1-4. Inderscience Enterprises, Genf, Schweiz, 2011

Albers et al. (2012a) Albers, A.; Braun, A.; Salomon, S.: Integrated modelling of product- and process-related information. In: Proceedings of the 14th International Dependency and Structure Modelling Conference (DSM ’12). Kyoto, Japan, 2012

Albers et al. (2012b) Albers, A.; Braun, A.; Schmalenbach, H.: An Ontology-Representation of the Integrated Product Engineering Model. In: Proceedings of the 14th International Dependency and Structure Modelling Conference (DSM ’12). Kyoto, Japan, 2012

Albers et al. (2012c) Albers, A.; Schwarz, A.; Zingel, C.; Schröter, J.; Behrendt, M.; Zell, A. et al.: System-oriented Validation Aspects of a Driver Assistance System Based on an Accelerator-Force-Feedback-Pedal. In: Proceedings of the FISITA World Automotive Congress. Peking, China, 2012

Albers et al. (2012d) Albers, A.; Ebel, B.; Lohmeyer, Q.: Systems of objectives in complex product development. In: Proceedings of the 9th International Symposium on Tools and Methods of Competitive Engineering (TMCE 2012). Karlsruhe, 2012

Albers et al. (2012e) Albers, A; Lohmeyer, Q; Radimersky, A: Individuelle und organisatorische Akzeptanz von Methoden des Systems Engineering. In: Tagungsband des Tag des Systems Engineering (TdSE 2012). Paderborn, 2012

Albers et al. (2012f) Albers, A.; Braun, A.; Pinner, T.: Integrated Modelling of Information to Support Product Engineering Processes. In: Proceedings of the 2nd International Workshop on Modelling and Management of Engineering Processes. Cambridge, Großbritannien, 2012

Albers und Braun (2011a) Albers, A.; Braun, A.: A generalised framework to compass and to support complex product engineering processes. In: International Journal of Product Development, Vol. 15, Nr. 1. Inderscience Enterprises, Genf, Schweiz, 2011

Albers und Braun (2011b) Albers, A.; Braun, A.: Der Prozess der Produktentstehung. In: Handbuch Leichtbau - Methoden, Werkstoffe, Fertigung. Carl Hanser Verlag GmbH & CO. KG, München, 2011

Albers und Düser (2007) Albers, A.; Düser, T.: Systematische Driveability Entwicklung von Fahrzeugen mit komplexen Strukturen. In: 2. Internationales Symposium für Entwicklungsmethodik. Wiesbaden, 2007

Albers und Düser (2009) Albers, A.; Düser, T.: Domänenübergreifende Entwicklungsprozesse – Ein modellbasierter Ansatz vom Anwendungsfall bis zur Validierung. In: Proceedings des 2. Grazer Symposiums Virtuelles Fahrzeug. Graz, Österreich, 2009

Albers und Düser (2010) Albers, A.; Düser, T.: A new process for configuration and application of complex validation environments using the example of vehicle-in-the-loop at the roller test bench. In: Proceedings of the 2010 International Mechanical Engineering Congress & Exposition (IMECE2010). Vancouver, British Columbia, Canada, 2010

Albers und Lohmeyer (2012) Albers, A.; Lohmeyer, Q.: Advanced systems engineering – towards a model-based and human-centered methodology. In: Proceedings of the 9th International Symposium on Tools and Methods of Competitive Engineering (TMCE 2012). Karlsruhe, 2012

Albers und Matthiesen (1999) Albers, A.; Matthiesen, S.: Maschinenbau im Informationszeitalter - Das Karlsruher Lehrmodell. In: Proceedings des 44th International Scientific Colloquium Technical University of Ilmenau, 1999

Albers und Meboldt (2007) Albers, A.; Meboldt, M.: SPALTEN Matrix – Product Development Process on the Basis of Systems Engineering and Systematic Problem Solving. In: Krause, F.-L. (Hrsg.): The Future of Product Development. Springer, Berlin, Heidelberg, 2007

Albers und Sadowski (2013) Albers A, Sadowski E: The Contact and Channel Approach (C&C²-A): relating a system's physical structure to its functionality. In: An Anthology of Theories and Models of Design: Philosophy, Approaches and Empirical Explorations (Hrsg. Blessing, L.T.M.; Chakrabarti, A.). Springer, Heidelberg, 2013

Albers und Zingel (2010) Albers, A.; Zingel, C.: Integrated driveability assessment and optimization environment on a roller test bench. In: Proceedings of the Global Powertrain Congress 2010, Troy, USA, 2010

Albers und Zingel (2011) Albers, A.; Zingel, C.: Interdisciplinary Systems Modeling Using the Contact & Channel-model for SysML. In: Proceedings of the 18th International Conference on Engineering Design (ICED'11). Copenhagen, Denmark, 2011

Albers und Zingel (2013a) Albers, A.; Zingel, C.: Challenges of Model-Based Systems Engineering: a study towards unified term understanding and the state of usage of SysML. In: Proceedings of the 23rd CIRP Design Conference. Bochum, 2013

Albers und Zingel (2013b) Albers, A.; Zingel, C.: Extending SysML for Engineering Designers by Integration of the Contact & Channel – Approach (C&C²-A) for Function-Based Modeling of Technical Systems. In: Proceedings of the Conference on Systems Engineering Research (CSER 2013). Atlanta, Georgia, USA, 2013

Alink (2010) Alink, T.: Bedeutung, Darstellung und Formulierung von Funktion für das Lösen von Gestaltungsproblemen mit dem C&C-Ansatz. IPEK Forschungsbericht Band 48. Hrsg.: o. Prof. Dr.-Ing. Dr. h.c. A. Albers, Karlsruher Institut für Technologie (KIT), 2010

Alt (2009) Alt, O.: Car Multimedia Systeme Modell-basiert testen mit SysML. Dissertation der Technischen Universität Darmstadt. Vieweg+Teubner Verlag, 2009.

Alt (2012) Alt, O.: Modellbasierte Systementwicklung mit SysML. Carl Hanser Verlag, München, 2012

Anderl und Trippner (2000) Anderl, R.; Trippner, D. (Hrsg.): STEP Standard for the Exchange of Product Model Data: Eine Einführung in die Entwicklung, Implementierung und industrielle Nutzung der Normenreihe ISO 10303 (STEP). Teubner, Stuttgart, Leipzig, 2000

Andersson et al. (2009) Andersson, H.; Herzog, E.; Johansson, G.; Johansson, O.: Experience from Introducing Unified Modeling Language/Systems Modeling Language at Saab Aerosystems. In: Systems Engineering Journal, Vol. 13, No.4. Wiley Periodicals, Hoboken, USA, 2009

Andrianarison und Piques (2010) Andrianarison, E.; Piques, J.D.: SysML for embedded automotive Systems: a practical approach. In: Proceedings of the Embedded Real Time Software and Systems (ERTS 2010). Toulouse, Frankreich, 2010

Angermann et al. (2011) Angermann, A.; Beuschel, M.; Rau, M.; Wohlfarth, U.: Matlab - Simulink - Stateflow: Grundlagen, Toolboxen, Beispiele. 7. Auflage, Oldenbourg Wissenschaftsverlag, München, 2011

Armengaud et al. (2012) Armengaud, E.; Biehl, M.; Bourrouilh, Q.; Breunig, M.; Farfeleder, S.; Hein, C. et al.: Integrated tool-chain for improving traceability during the development of automotive systems. In: Proceedings of the Embedded Real Time Software and Systems (ERTS² 2012). Toulouse, Frankreich, 2012

ATESST2 Consortium (2010a) ATESST2 Consortium: Electronics Architecture and Software Technology - Architecture Description Language (EAST-ADL). Domain Model Specification of Version 2.1 RC3, 2010

ATESST2 Consortium (2010b) ATESST2 Consortium: Electronics Architecture and Software Technology - Architecture Description Language (EAST-ADL). Profile Specification of Version 2.1 RC3, 2010

AUTOSAR (2011) AUTOSAR development cooperation (Hrsg.): Automotive Open System Architecture (AUTOSAR). Specification of Version 4.0, 2012, available at http://www.autosar.org

Bahill und Gissing (1998) Bahill, A.T.; Gissing, B.: Re-evaluating systems engineering concepts using systems thinking. In: IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, Volume 28, Nr. 4. IEEE, New York, USA, 1998

Bauer und Meerkamm (2007) Bauer, S.; Meerkamm, H.: Decision making with interdependent objectives in Design for X. in: Proceedings of the 16th International Conference on Engineering Design (ICED '07). Paris, 2007

Becker et al. (2013) Becker, C.; Giese, T.; Krakowski, D.; Krittian, S.; Scherer, T.: Efficiency of Model Based Methodologies in Air Systems Engineering. In: Proceedings des Workshops on Aircraft System Technologies (AST 2013). Hamburg, 2013

Beidl et al. (2013) Beidl, C.; Weber, T.; Gietzelt, C.; Düser, T.: Realer Kraftstoffverbrauch und manöverbasiertes Testen. In: Automobiltechnische Zeitschrift (ATZ), Vol. 115, Nr. 4. Springer, Wiesbaden, 2013

Bertalanffy (1949) Bertalanffy, L.: Zu einer allgemeinen Systemlehre. In: Biologia Generalis, Nr. 19. Wien, New York,1949

Bertsche et al. (2009) Bertsche, B.; Göhner, P.; Jensen, U.; Schinköthe, W.; Wunderlich, H.-J.: Zuverlässigkeit mechatronischer Systeme - Grundlagen und Bewertung in frühen Entwicklungsphasen. Springer, Berlin Heidelberg, 2009

Birkhofer (2011) Birkhofer, H.: The Future of Design Methodology. Springer London, 2011

Bleck et al. (1996) Bleck, A.; Goedecke, M.; Huss, M.; Waldschmidt, K.: Praktikum des VLSI-Entwurfs. Teubner Verlag, Wiesbaden, 1996

Booch (1993) Booch, G.: Object-Oriented Analysis and Design with Applications. 2. Auflage, Addison-Wesley Professional, 1993

Broy et al. (2010) Broy, M.; Feilkas, M.; Herrmannsdoerfer, M.; Merenda, S.; Ratiu, D.: Seamless Model-Based Development: From Isolated Tools to Integrated Model Engineering Environments. In: Proceedings of the IEEE, Vol. 98, Nr. 4. IEEE, New York, USA, 2010

Broy et al. (2011) Broy, M.; Kirstan, S.; Krcmar, H.; Schätz, B.: What is the Benefit of a Model-Based Design of Embedded Software Systems in the Car Industry? In: Rech, J.; Bunse, C. (eds.): Emerging Technologies for the Evolution and Maintenance of Software Models. IGI Global, Hershey, Pennsylvania, USA, 2011

Burmester et al. (2005) Burmester, S.; Giese, H.; Tichy, M.: Model-Driven Development of Reconfigurable Mechatronic Systems with MECHATRONIC UML. In: Model-driven Architecture - Lecture Notes of Computer Science, Vol. 3599. Springer, Berlin, Heidelberg, 2005

Cloutier und Bone (2010) Cloutier, R.; Bone, M.: Compilation of SysML Request for Information – Final Report. Stevens Institute of Technology und School of Systems & Enterprises. 2010

Conrad et al. (2005) Conrad, S.; Hasselbring, W.; Koschel, A.; Tritsch, R.: Enterprise Application Integration – Grundlagen - Konzepte - Entwurfsmuster – Praxisbeispiele. Elsevier, Oxford, 2005

Conrad (2010) Conrad, J.: Semantische Netze zur Erfassung und Verarbeitung von Informationen und Wissen in der Produktentwicklung. Dissertation der Universität des Saarlandes, Schriftenreihe Produktionstechnik, Band 49. Saarbrücken, 2010

Cooper und Harper (1969) Cooper, G. E.; Harper, R. P.: The Use of Pilot Rating in the Evaluation of Aircraft Handling Qualities. NASA TN D-5153, 1969

Daenzer und Huber (2004) Daenzer, W. F.; Huber, F. (Hrsg.): Systems Engineering, 12. Auflage. Orell Füssli Verlag Industrielle Organisation, Zürich, 2004

Damerau et al. (2011) Damerau, T.; Kaufmann, U.; Knothe, T.; Stark, R.; Ulbrich, A.: Modellbasierte Prozess- und Systemgestaltung für die Innovationsbeschleunigung - Das Verbundprojekt ISYPROM. In: ZWF – Virtuelle Produktentstehung, Jahrgang 106. Carl Hanser Verlag, München, 2011

David et al. (2010) David, P.; Idasiak, V.; Kratz, F.: Reliability study of complex physical systems using SysML. In: Reliability Engineering and System Safety, Vol. 95. Elsevier, Amsterdam, 2010

Denger et al. (2012) Denger, A.; Fritz, J.; Kissel, M.; Parvan, M.; Zingel, C.: Potentiale einer funktionsorientierten Lenkung mechatronischer Produkte in der Automobilindustrie. In: Tagungsband des Tag des Systems Engineering (TdSE 2012). Paderborn, 2012

Denger et al. (2013) Denger, A.; Fritz, J.; Denger, D.; Zingel, C.; Ölmez, M.; Kissel, M.: Modell-basierter Ansatz zur funktionsorientierten Lenkung von Produkten. In: ProduktDaten Journal, Band 20, Nr. 1. ProSTEP iViP e.V. (Hrsg.), Darmstadt, 2013

Denger und Fritz (2013) Denger, A.; Fritz, J.: Model-based approach for market-oriented positioning of products - Function-oriented system modeling in the context of the product portfolio. Vortrag auf dem ProSTEP Symposium. Hannover, 2013

Deubzer und Lindemann (2009) Deubzer, F.; Lindemann, U.: Networked product modelling - use and interaction of product models and methods during analysis and synthesis. In: Proceedings der 17th International Conference on Engineering Design (ICED '09). Stanford, California, USA, 2009.

DIN 2330 (1979) Deutsches Institut für Normung (Hrsg.): DIN 2330: Begriffe und Benennungen. Beuth Verlag, Berlin, 1979

DIN 66261 (1985) Deutsches Institut für Normung (Hrsg.): DIN 66261: Informationsverarbeitung; Sinnbilder für Struktogramme nach Nassi-Shneiderman. Beuth Verlag, Berlin, 1985

DIN 69901 (2009) Deutsches Institut für Normung (Hrsg.): DIN 69901-1-5, Projektmanagement – Projektmanagementsysteme. Beuth Verlag, Berlin, 2009

DoD (2001) Department of Defense (Hrsg.): Systems Engineering Fundamentals. Defense Acquisition University Press, Fort Belvoir, Virginia, USA, 2001

Dori (2002) Dori, D.: Object-process methodology: a holistics systems paradigm. Springer, Berlin, Heidelberg, New York, 2002

Düser (2010) Düser, T.: X-in-the-Loop – ein durchgängiges Validierungsframework für die Fahrzeugentwicklung am Beispiel von Antriebsstrangfunktionen und Fahrerassistenzsystemen. IPEK Forschungsbericht Band 47. Hrsg.: o. Prof. Dr.-Ing. Dr. h.c. A. Albers, Karlsruher Institut für Technologie (KIT), 2010

Düser (2012)* Düser, T.: AVL-interne Ergebnispräsentation zum Einsatz der SysML für die funktionale Strukturierung des AVL-Produktportfolios im Bereich der Gesamtfahrzeug-Entwicklungsumgebungen. Graz, 2012 (* = nicht veröffentlicht)

Düser et al. (2012) Düser, T.; Schmidt, C.; Schmidt, U.; Pfister, F.: Rollenprüfstände für Fahrzeug- und Antriebskonzepte von morgen. In: ATZ - Automobiltechnische Zeitschrift Ausgabe Nr.: 2012-04. Springer, Wiesbaden, 2012

Dyla (2002) Dyla, A.: Modell einer durchgängig rechnerbasierten Produktentwicklung. Dissertation an der Technischen Universität München (TUM), 2002

Eckert et al. (2010) Eckert, C.; Alink, T.; Albers, A.: Issue driven analysis of an existing product at different levels of abstraction. In: Proceedings of the 11th International Design Conference (DESIGN ’10). Dubrovnik, Kroatien, 2010

Eckert et al. (2011) Eckert, C.; Alink, T.; Ruckpaul, A.; Albers, A.: Different notions of function: results from an experiment on the analysis of an existing product. In: Journal of Engineering Design. Taylor & Francis, London, UK, 2011

Ehrlenspiel (2009) Ehrlenspiel, K.: Integrierte Produktentwicklung – Denkabläufe, Methodeneinsatz, Zusammenarbeit. 4. Auflage, Hanser Verlag, München, Wien, 2009

Eigner und Stelzer (2009) Eigner, N.; Stelzer, R.: Product Lifecycle Management - Ein Leitfaden für Product Development und Life Cycle Management, 2. Auflage. Springer, Berlin, Heidelberg, 2009

Eigner et al. (2012a) Eigner, M; Gilz, T.; Zafirov, R.: Proposal for Functional Product Description as Part of a PLM Solution in Interdisciplinary Product Development. In: Proceedings of the 12th International Design Conference (DESIGN ’12). Dubrovnik, Kroatien, 2012

Eigner et al. (2012b) Eigner, M; Gerhardt, F.; Gilz, T.; Nem, F.M.; Informationstechnologie für Ingenieure. Springer, Berlin, Heidelberg, 2012

EM AG (2012) Engineering Methods AG (Hrsg.): Systems Engineering mit CATIA V6 Systems. Online-Artikel auf den Webseiten der Engineering Methods AG: http://www.em.ag/themenloesungen/systems-engineering, abgerufen am 04.10.2012

Enkler (2010) Enkler, H.-G.: Rechnergestützter Entwurf von Bauteilen mit stark streuenden Leitstützstrukturen am Beispiel hochbelastbarer urgeformter mikromechanischer Systeme. IPEK Forschungsbericht Band 44. Hrsg.: o. Prof. Dr.-Ing. Dr. h.c. A. Albers, Karlsruher Institut für Technologie (KIT), 2010

Erden et al. (2008) Erden, M.; Komoto, H.; van Beek, T.; D'Amelio, V.; Echavarria, E.; Tomiyama, T.: A review of function modeling: Approaches and applications. In: Artificial Intelligence in Engineering Design, Analysis and Manufacturing, Vol. 22. Cambridge University Press, 2008

Erk und Priese (2008) Erk, K; Priese, L.: Theoretische Informatik – Eine umfassende Einführung, 3. Auflage. Springer, Berlin, Heidelberg, 2008.

Estefan (2008) Estefan, J.A.: Survey of Model-Based Systems Engineering (MBSE) Methodologies (Revision B). INCOSE, Seattle, USA, 2008

Fachbach et al. (2012) Fachbach, B.; Bernasch, J.; Schmeja, M.: Herausforderung Virtuelle Fahrzeugentwicklung. In: Virtual Vehicle magazine, Nr. 12, II-2012. Hrsg: Kompetenzzentrum Das Virtuelle Fahrzeug Forschungsgesellschaft mbH (ViF). Graz, Österreich, 2012

Feldhusen und Gebhardt (2008) Feldhusen, J.; Gebhardt, B.: Product Lifecycle Management für die Praxis - Ein Leitfaden zur modularen Einführung, Umsetzung und Anwendung. Springer, Berlin, Heidelberg, 2008

Friedenthal (2009) Friedenthal, S.: SysML: Lessons from Early Applications and Future Directions. In: INCOSE INSIGHT Special Feature: Model-Based Systems Engineering: The new paradigm, Vol. 12, Issue 4. INCOSE, Seattle, USA, 2009

Friedenthal et al. (2011) Friedenthal, S.; Moore, A.; Steiner, R.: A Practical Guide to SysML: The Systems Modeling Language. 2. Auflage, Elsevier, Amsterdam, 2011

Friedrich et al. (2009) Friedrich, J.; Hammerschall, U.; Kuhrmann, M.; Sihling, M.: Das V-Modell XT. In: Das V-Modell® XT, Informatik im Fokus, 2. Auflage. Springer, Berlin, Heidelberg, 2009

FTD (2012) Financial Times Deutschland (Hrsg.): Defekte Fensterheber: Toyota ruft Millionen Autos zurück. Onlineartikel auf: http://www.ftd.de, abgerufen am 10.10.2012

Gausemeier et al. (2009a) Gausemeier, J.; Frank, U.; Donoth, J.; Kahl, S.: Specification technique for the description of self-optimizing mechatronic systems. In: Research in Engineering Design, Volume 20, Issue 4. Springer, 2009

Gausemeier et al. (2009b) Gausemeier, J.; Plass, C.; Wenzelmann, C.: Zukunftsorientierte Unternehmensgestaltung – Stragegien, Geschäftsprozesse und IT-Systeme für die Produktion von morgen. Carl Hanser Verlag, München, 2009

Gausemeier et al. (2010) Gausemeier, J.; Dorociak, R.; Pook, S.; Nyßen, A.; Terfloth, A.: Computer-aided cross-domain modeling of Mechatronic Systems. In: Proceedings of the 11th International Design Conference (DESIGN ’10). Dubrovnik, Kroatien, 2010

Gausemeier et al. (2012) Gausemeier, J.; Lanza, G.; Lindemann, U. (Hrsg.): Produkte und Produktionssysteme integrativ konzipieren – Modellbildung und Analyse in der frühen Phase der Produktentstehung. Carl Hanser Verlag, München, 2012

Gero (1990) Gero, J. S.: Design prototypes: a knowledge representation schema for design. In: AI Magazine, Vol. 11, Nr. 4. AAAI, Palo Alto, USA, 1990

Gero und Kannengiesser (2004) Gero, J.S.; Kannengiesser, U.: The situated function–behaviour–structure framework. In: Design Studies, Vol. 25, Nr. 4. Elsevier, Oxford, 2004

GfSE Website (2012) GfSE: Webseiten der Gesellschaft für Systems Engineering e.V.. URL: http://www.gfse.org/. Abgerufen am 29.10.2012

Graves (2009) Graves, H.: Integrating SysML and OWL. In: Proceedings of OWL: Experiences and Directions, 6th international Workshop (OWLED ‘09). Chantilly, Virginia, USA, 2009

Griessnig et al. (2011) Griessnig, G.; Kundner, I.; Armengaud, E.; Torchiaro, S.; Karlsson, D.: Improving automotive embedded systems engineering at European level. In: Elektrotechnik & Informationstechnik, Vol. 128, Nr. 6. Springer, Berlin, Heidelberg, 2011

Grobshtein und Dori (2011) Grobshtein, Y.; Dori, D.: Generating SysML views from an OPM model: Design and evaluation. In: Systems Engineering Journal, Volume 14, Nr. 3. Wiley Periodicals, Hoboken, USA, 2011

Hacker (2002) Hacker, W.: Denken in der Produktentwicklung – Psychologische Unterstützung der frühen Phasen. vdf Hochschulverlag AG an der ETH Zürich, 2002

Hall (1962) Hall, A.D.: A methodology for systems engineering. Van Nostrand, 1962

Hannah et al. (2012) Hannah, R.; Joshi, S.; Summers, J.D.: A user study of interpretability of engineering design representations. In: Journal of Engineering Design, Vol. 23, Nr. 6. Taylor & Francis, London, UK, 2012

Hansen (1968) Hansen, F.: Konstruktionssystematik : Grundlagen für eine allgemeine Konstruktionslehre. 3. durchges. Auflage, Verlag Technik, Berlin, 1968

Hardt und Große-Rhode (2008) Hardt, M.; Große-Rhode, M.: Funktionsorientierte Systementwicklung mit Autosar - Vom Softwarekomponentenmarkt zu architekturzentrierten Prozessen. In: ATZ-Elektronik Nr.6. Springer Automotive Media, Wiesbaden, 2008

Harold (2002) Harold, E.R.: Die XML Bibel. 2. Auflage, mitp Verlag, Frechen, 2002

Haskins et al. (2011) Haskins, C.; Krueger, M.; Forsberg, K.; Walden, D.; Hamelin, R.D. (Eds.): Systems Engineering Handbook V3.2.2, TP-2003-002-03.2.2. INCOSE, San Diego, USA, 2011

Hirtz et al. (2002) Hirtz, J.; Stone, R.; McAdams, D.; Szykman, S.; Wood, K.: A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts. NIST Technical Note Nr. 1447. Washington, USA, 2002

Hitchins (2007) Hitchins, D.K.: Systems Engineering: A 21st Century Systems Methodology. 1. Auflage, John Wiley & Sons, 2007

Hitzler et al. (2008) Hitzler, P.; Krötzsch, M.; Rudolph, S.; Sure, Y.: Semantic Web – Grundlagen. Springer, Berlin, Heidelberg, 2008

Hohagen und Naumann (1994) Hohagen, F.; Naumann, S.: Parsing: Eine Einführung in die maschinelle Analyse natürlicher Sprache (Leitfäden und Monographien der Informatik). Teubner Verlag, 1994

Hopcroft und Ullmann (1994) Hopcroft, J.E.; Ullmann, J.D.: Einführung in die Automatentheorie, formale Sprachen und Komplexitätstheorie, 3. Auflage. Addison Wesley, Bonn, 1994

Hubka und Eder (1992) Hubka, V.; Eder, W.E.: Einführung in die Konstruktionswissenschaften. Springer, Berlin, Heidelberg, 1992

Immen (2010) Immen, S.: Jahresbericht 2010 des Kraftfahrtbundesamtes, Druckzentrum KBA, 2010

INCOSE (2007) International Council on Systems Engineering (Hrsg.): Systems Engineering Vision 2020, Version 2.03, TP-2004-004-02. INCOSE, San Diego, USA, 2007

INCOSE Website (2012) INCOSE: Websites des International Council on Systems Engineering. http://www.incose.org/. Abgerufen am 29.12.2012

IPEK-PKB (2013) Produktentwicklung Knowledge Base (Wissensdatenbank) des IPEK – Institut für Produktentwicklung. Karlsruher Institut für Technologie, abgerufen im Juni 2013

ISO-10303-203 (2011) ISO - International Organization for Standardization (Hrsg.): ISO 10303-214: Industrial automation systems and integration – Product data representation and exchange. Part 203: Application protocol: Configuration controlled 3D design of mechanical parts and assemblies (2011)

ISO-10303-214 (2010) ISO - International Organization for Standardization (Hrsg.): ISO 10303-214: Industrial automation systems and integration – Product data representation and exchange. Part 214: Application protocol: Core data for automotive mechanical design processes (2010)

ISO-10303-233 (2012) ISO - International Organization for Standardization (Hrsg.): ISO 10303-233: Industrial automation systems and integration – Product data representation and exchange. Part 233: Application protocol: Systems engineering (2012)

ISO-15288 (2008) ISO - International Organization for Standardization (Hrsg.): ISO/IEC 15288:2008 - Systems and software engineering - System life cycle processes, 2008

ISO-19501 (2005) ISO - International Organization for Standardization (Hrsg.): ISO/IEC 19501:2005(E) - Information technology — Open Distributed Processing — Unified Modeling Language (UML) Version 1.4.2, 2005

ISO-26262 (2012) ISO - International Organization for Standardization (Hrsg.): ISO 26262 – Road vehicles – Functional safety, 2012

Jacobson et al. (1992) Jacobson, I.; Christerson, M.; Jonsson, P.: Object-Oriented Software Engineering. Addison-Wesley Professional, 1992

Jaros (2007) Jaros, M.: Integration des STEP-Produktmodells in den Getriebeentwicklungsprozess. Dissertation an der Technischen Universität München (TUM), 2007

Johnson et al. (2008) Johnson, T.; Paredis, C. J. J.; Burkhart, R.M.: Integrating Models and Simulations of Continuous Dynamics into SysML. In: Proceedings of the Modelica Conference. Bielefeld, Germany, 2008

Kajikawa (2008) Kajikawa, Y.: Structure of Knowledge in the Science and Technology Roadmaps. In: Technological Forecasting and Social Change, Volume 75. Elsevier, Oxford, 2008

Karban et al. (2009) Karban, R.; Hauber, R.; Weilkiens, T.: MBSE in Telescope Modeling. In: INCOSE INSIGHT Special Feature: Model-Based Systems Engineering: The new paradigm, Vol. 12, Issue 4. INCOSE, Seattle, USA, 2009

Kasser (2010) Kasser, J.E.: Seven systems engineering myths and the corresponding realities. In: Proceedings of the Systems Engineering Test and Evaluation Conference. Adelaide, Australia, 2010

Kikuchi und Nagasaka (2002) Kikuchi, M.; Nagasaka, I.: On the three Axioms of General Design Theory. In: Proceedings of the International Workshop of Emergent Synthesis (IWES ’02). Kobe, Japan, 2002

Kinkel (2005) Kinkel, S.: Anforderungen an die Fertigungstechnik von morgen. In: Mitteilungen aus der Produktionsinnovationserhebung, Nummer 37. Fraunhofer ISI, Karlsruhe, 2005

Klima und Selberherr (2010) Klima, R.; Selberherr, S.: Programmieren in C, 3. Auflage. Springer, Wien, New York, 2010.

Koller (1998) Koller, R.: Konstruktionslehre für den Maschinenbau, 4. Auflage. Springer, Berlin, 1998

Königs (2009) Königs, A.: Model Integration and Transformation – A Triple Graph Grammar-based QVT Implementation. Dissertation des Fachbereichs Elektrotechnik und Informatik der Universität Darmstadt. 2009

Korff (2008) Korff, A.: Modellierung von eingebetteten Systemen mit UML und SysML. Springer, Heidelberg, 2008

Korff und Weilkiens (2010) Korff, A.; Weilkiens, T.: Toolgestützte Kombination von Modellierungsnotation und Prozess anhand von SysML und SYSMOD. In: Tagungsband des Tag des Systems Engineering (TdSE 2010). München, 2010

Korff et al. (2011) Korff, A.; Lamm, J.G.; Weilkiens, T.: Werkzeuge für den Schmied funktionaler Architekturen. In: Proceedings des Tag des Systems Engineering (TdSE ’11). Hamburg, 2011

Kruse (1996) Kruse, P.J.: Anforderungen in der Systementwicklung – Erfassung, Aufbereitung und Bereitstellung von Anforderungen in interdisziplinären Entwicklungsprojekten. Dissertation, TU Clausthal. VDI-Verlag, Düsseldorf, 1996.

Kruse et al. (2012) Kruse, B.; Münzer, C.; Wölkl, S.; Canedo, A.; Shea, K.: A Model-Based Functional Modeling and Library Approach for Mechatronic Systems in SysML. In: Proceedings of the ASME 2012 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE ‘12). Chicago, IL, USA, 2012

Kulak et al. (2010) Kulak, O.; Cebi, S.; Kahraman, C.: Applications of axiomatic design principles: A literature review. In: Expert Systems with Applications, Volume 37, Nr. 9. Elsevier, Amsterdam, 2010

Lamm und Weilkiens (2010) Lamm, J.G.; Weilkiens, T.: Funktionale Architekturen in SysML. In: Proceedings des Tag des Systems Engineering (TdSE ’10). München, 2010

Lee (2008) Lee, E.A.: Cyber Physical Systems: Design Challenges. In: Proceedings of the 11th IEEE Symposium on Object Oriented Real-Time Distributed Computing (ISORC). Orlando, Florida, USA, 2008

Lindemann (2009) Lindemann, U.: Methodische Entwicklung technischer Produkte. 3. Auflage, Springer, Berlin, Heidelberg, 2009

Lindemann et al. (2009) Lindemann, U.; Maurer, M.; Braun, T.: Structural Complexity Management: An Approach for the Field of Product Design. Springer, Berlin, 2009

Lintermann et al. (2008) Lintermann, F.-J.; Schaefer, U.; Schulte-Göcking, W.; Gettner, K.: Einfache IT-Systeme. Lehr-/Fachbuch, 5. Auflage. Bildungsverlag EINS, 2008

Lohmeyer (2013) Lohmeyer, Q.: Menschzentrierte Modellierung von Produktentstehungssystemen unter besonderer Berücksichtigung der Synthese und Analyse dynamischer Zielsysteme. IPEK Forschungsbericht Band 59. Hrsg.: o. Prof. Dr.-Ing. Dr. h.c. A. Albers, Karlsruher Institut für Technologie (KIT), 2013

Lossack (2006) Lossack, R.-S.: Wissenschaftstheoretische Grundlagen für die rechnerunterstützte Konstruktion. Springer, Berlin, Heidelberg, 2006

Maletz und Zingel (2011)* Maletz, M; Zingel, C.: Projektabschlussbericht Projektnr.: A571003. Kooperationsprojekt mit der AVL List GmbH, Graz, 2011 (* = nicht veröffentlicht)

Manola und Miller (2004) Manola, F.; Miller, E.: Resource Description Framework (RDF). Primer. W3C Recommendation 10 February 2004. Available at http://www.w3.org/TR/rdf-primer/

Marques et al. (2009) Marques, P.A.; Saraiva, P.M.; Requeijo, J.G.; Alink, T.; Albers, A.; Guerreiro, F.F.: Integration of the Contact and Channel Model with Axiomatic Design. In: Proceedings of the International Conference on Axiomatic Design (ICAD ’09). Campus de Caparica, 2009

Matthiesen (2002) Matthiesen, S.: Ein Beitrag zur Basisdefinition des Elementmodells “Wirkflächenpaare & Leitstützstrukturen” zum Zusammenhang von Funktion und Gestalt technischer Systeme. MKL Forschungsberichte, Band 6. Hrsg.: o. Prof. Dr.-Ing. Dr. h.c. A. Albers, Universität Karlsruhe (TH), 2002

Matthiesen und Ruckpaul (2012) Matthiesen, S.; Ruckpaul, A.: New Insights on the Contact & Channel–Approach – Modelling of Systems with Several Logical States. In: Proceedings of the 12th International Design Conference (DESIGN '12). Dubrovnik, Kroatien, 2012

Maurer (2007) Maurer, M.: Structural Awareness in Complex Product Design. Dissertation an Lehrstuhl für Produktentwicklung der TU München, Band 74. München, 2007

Meboldt (2008) Meboldt, M.: Mentale und formale Modellbildung in der Produktentstehung – als Beitrag zum integrierten Produktentstehungs-Modell (iPeM). IPEK Forschungsberichte, Band 29. Hrsg.: o. Prof. Dr.-Ing. Dr. h.c. A. Albers, Universität Karlsruhe (TH), 2008

Mesarovic (1970) Mesarovic, M.D.: Multilevel systems and concepts in process control. In: Proceedings of the IEEE, Volume 58, Issue 1. IEEE, 1970

Modelica Association (2012) Modelica Association: Modelica: A Unified Object-Oriented Language for Physical Systems Modeling. Specification of Version 3.3, 2012, available at http://www.modelica.org

Muschik (2011) Muschik, S.: Development of Systems of Objectives in Early Product Engineering. IPEK Forschungsbericht Band 50. Hrsg.: o. Prof. Dr.-Ing. Dr. h.c. A. Albers, Karlsruher Institut für Technologie (KIT), 2011

Müller und Weichert (2011) Müller, H.; Weichert, F.: Vorkurs Informatik - Der Einstieg ins Informatikstudium, 2. Auflage. Vieweg+Teubner, Springer Fachmedien, Wiesbaden, 2011

Naunheimer et al. (2007) Naunheimer, H.; Bertsche, B.; Lechner, G.: Fahrzeuggetriebe - Grundlagen, Auswahl, Auslegung und Konstruktion, 2. Auflage. Springer, Berlin, 2007

Ohmer (2008) Ohmer, M.: Ein Beitrag zur Synthese technischer Systeme auf Basis des Contact & Channel Model C&CM. IPEK Forschungsbericht Band 32. Hrsg.: o. Prof. Dr.-Ing. Dr. h.c. A. Albers, Universität Karlsruhe (TH), 2008

OMG BPMN (2011) Object Management Group (Hrsg.): Business Process Model and Notation (BPMN). Specification of Version 2.0, 2011, available at http://www.omg.org

OMG MARTE (2011) Object Management Group (Hrsg.): UML Profile for Modeling and Analysis of Real-Time Embedded Systems (MARTE). Specification of Version 1.1, 2011, available at http://www.omg.org

OMG MDA (2003) Object Management Group (Hrsg.): Model-Driven Architecture (MDA) Guideline. Version 1.0.1, 2003, available at http://www.omg.org

OMG ModelicaML (2012) Webseite der OMG-Arbeitsgruppe zur SysML-Modelica-Integration. http://www.omgwiki.org/OMGSysML/doku.php?id=sysml-modelica:sysml_and_modelica_integration

OMG MOF (2011) Object Management Group (Hrsg.): Meta-Object Facility (MOF). Specification of Version 2.4.1, 2011, available at http://www.omg.org

OMG OCL (2012) Object Management Group (Hrsg.): Object Constraint Language (OCL). Specification of Version 2.3.1, 2011, available at http://www.omg.org

OMG QVT (2011) Object Management Group (Hrsg.): Query View Transformation (QVT). Specification of Version 1.1, 2011, available at http://www.omg.org

OMG SysML (2012) Object Management Group (Hrsg.): Systems Modeling Language (SysML). Specification of Version 1.3, 2012, available at http://www.omg.org

OMG SysML RTF (2012) Website der OMG SysML Revision Task Force, http://www.omg.org/issues/sysml-rtf.html, abgerufen am 23.11.2012

OMG UML (2011a) Object Management Group (Hrsg.): Unified Modeling Language (UML). Superstructure Specification of Version 2.4.1, 2011, available at http://www.omg.org

OMG UML (2011b) Object Management Group (Hrsg.): Unified Modeling Language (UML). Infrastructure Specification of Version 2.4.1, 2011, available at http://www.omg.org

OMG XMI (2011) Object Management Group (Hrsg.): XML Metadata Interchange (XMI). Specification of Version 2.4.1, 2011, available at http://www.omg.org

Oesterreich et al. (2012) Oesterreich, B.; Bremer, S.; Scheithauer, A.: Analyse und Design mit der UML 2.5. 10. Auflage, Oldenbourg Wissenschaftsverlag, München 2012

Pahl et al. (2007) Pahl, G.; Beitz, W.; Feldhusen, J.; Grote, K.-H.: Pahl/Beitz Konstruktionslehre - Grundlagen erfolgreicher Produktentwicklung - Methoden und Anwendung. 7. Auflage, Springer, Berlin, Heidelberg, 2007

Paredis et al. (2010) Paredis, C.J.J.; Bernard, Y.; Burkhart, R.M.; de Koning, H.-P.; Friedenthal, S.; Fritzson, P.; Rouquette, N.F.; Schamai, W.: An Overview of the SysML-Modelica Transformation Specification. In: Proceedings of the 20th International Symposium (IS '10). Chicago, USA, 2010

Piques und Andrianarison (2012) Piques, J.D.; Andrianarison, E.: SysML for embedded automotive Systems: lessons learned. In: Proceedings of the Embedded Real Time Software and Systems (ERTS 2012). Toulouse, Frankreich, 2012

Ponn und Lindemann (2011) Ponn, J.; Lindemann, U.: Konzeptentwicklung und Gestaltung technischer Produkte - Systematisch von Anforderungen zu Konzepten und Gestaltlösungen. 2. Auflage, Springer, Berlin, Heidelberg, 2011

Ramos et al. (2010) Ramos, A.L.; Ferreira, J.V.; Barcel´o, J.: Revisiting the SIMILAR process to engineer the contemporary systems. In: Journal of System Science and Systems Engineering, Vol. 19, No. 3. Springer, 2010

Ramos et al. (2012) Ramos, A.L.; Ferreira, J.V.; Barcel´o, J.: Model-Based Systems Engineering: An Emerging Approach for Modern Systems. In: IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, Volume 42, Nr. 1. IEEE, New York, USA, 2012

Reichwein und Paredis (2011) Reichwein, A.; Paredis, C.J.J.: Overview of Architectural Frameworks and Modeling Languages für Model-Based Systems Engineering. In: Proceedings of the ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE 2011). Washington, DC, USA, 2011

Reinhartz-Berger und Dori (2005) Reinhartz-Berger, I.; Dori, D.: OPM vs. UML - Experimenting with Comprehension and Construction of Web Application Models. In: Empirical Software Engineering Journal, Volume 10, Issue 1. Springer, 2005

Rodenacker (1991) Rodenacker, W. G.: Methodisches Konstruieren : Grundlagen, Methodik, praktische Beispiele. Konstruktionsbücher Band 27. Springer, Berlin, Heidelberg, New York, 1991

Ropohl (1975) Ropohl, G.: Systemtechnik - Grundlagen und Anwendung. Carl Hanser Verlag, München, 1975

Ropohl (2009) Ropohl, G.: Allgemeine Technologie : eine Systemtheorie der Technik, 3. Auflage. Universität Karlsruhe Universitätsbibliothek, 2009

Roth (2000) Roth, K.: Konstruieren mit Konstruktionskatalogen, 3. Auflage. Springer, Berlin, Heidelberg, New York, 2000)

Rude (1998) Rude, S.: Wissensbasiertes Konstruieren. Habilitationsschrift. Shaker Verlag, Aachen, 1998

Rumbaugh et al. (1991) Rumbaugh, J.; Blaha, M.; Premerlani, W.; Eddy, F.; Lorensen, W.: Object-Oriented Modeling and Design. Prentice Hall, New York, 1991

Rumbaugh et al. (2004) Rumbaugh, J.; Jacobson, I; Booch, G.: The Unified Modeling Language Reference Manual, 2. Auflage. Addison-Wesley Longman, Amsterdam, 2004

Schamai et al. (2009) Schamai, W.; Fritzson, P.; Paredis, C.J.J.; Pop, A.: Towards Unified System Modeling and Simulation with ModelicaML: Modeling of Executable Behavior Using Graphical Notations. In: Proceedings of the 7th Modelica conference. Como, Italy, 2009

Scheele und Groeben (1988) Scheele, B.; Groeben, N.: Dialog-Konsens-Methoden zur Rekonstruktion Subjektiver Theorien - Die Heidelberger Struktur-Lege-Technik (SLT), konsensuale Ziel-Mittel-Argumentation und kommunikative Flußdiagramm-Beschreibung von Handlungen. Franke Verlag, Tübingen, 1988

Schuh und Klappert (2011) Schuh, G.; Klappert, S.: Technologiemanagement. Handbuch Produktion und Management 2, 2. Auflage. Springer-Verlag, Berlin, 2011

SEBoK (2012) Pyster, A.; Olwell, D.; Hutchison, N.; Enck, S.; Anthony, J.; Henry, D.; Squires, A. (eds): Guide to the Systems Engineering Body of Knowledge (SEBoK), Version 1.0. The Trustees of the Stevens Institute of Technology, Hoboken, NJ, USA, 2012. URL: http://www.sebokwiki.org, abgerufen am 12.10.2012

SE-Zert (2012) Website des deutschen Systems Engineering-Zertifizierungsprogramms der GfSE. URL: http://www.sezert.de/, abgerufen am 23.11.2012

Shah et al. (2009) Shah, A.A.; Schaefer, D.; Paredis, C.J.J.: Enabling Multi-View Modeling With SysML Profiles and Model Transformations. In: Proceedings of the International Conference on Product Lifecycle Management (PLM '09). Inderscience, Genf, Schweiz, 2009

Shah et al. (2010) Shah, A.A.; Kerzhner, A.A.; Schaefer, D.; Paredis, C.J.J.: Multi-view Modeling to Support Embedded Systems Engineering in SysML. In: Graph Transformations and Model-Driven Engineering (Lecture Notes in Computer Science, Volume 5765). Springer, Berlin, 2010

Sharon et al. (2008) Sharon, A.; de Weck, O.L.; Dori, D.: A project-product lifecycle management approach for improved systems engineering practices. In: Proceedings of the 18th Annual INCOSE Conference. Utrecht, Netherlands, 2008

Sharon et al. (2009) Sharon, A.; Dori, D.; de Weck, O.L.: Is there a complete project plan? A model-based project planning approach. In: Proceedings of the 19th Annual INCOSE Conference. Singapore, 2009

Sharon et al. (2011) Sharon, A.; de Weck, O.L.; Dori, D.: Project management vs. systems engineering management: A practitioners' view on integrating the project and product domains. In: Systems Engineering Journal Volume 14, Issue 4. Wiley Periodicals, Hoboken, USA, 2011

Som et al. (2011) Som, O.; Kinkel, S.; Jäger, A.: Innovationsstrategien jenseits von Forschung und Entwicklung. In: Mitteilungen aus der ISI-Erhebung, Nummer 55. Fraunhofer ISI, Karlsruhe, 2011

Staab und Studer (2009) Staab, S.; Studer, R.: Handbook on Ontologies, 2nd Edition. Springer, Berlin, Heidelberg, 2009

Stachowiak (1973) Stachowiak, H.: Allgemeine Modelltheorie. Springer, Wien, 1973

Stahl et al. (2007) Stahl, T.; Völter, M.; Efftinge, S.: Modellgetriebene Softwareentwicklung. Techniken, Engineering, Management. 2. Auflage, Dpunkt Verlag, Heidelberg, 2007

Starke und Tilkov (2007) Starke, G.; Tilkov, S.: SOA-Expertenwissen: Methoden, Konzepte und Praxis serviceorientierter Architekturen. dpunkt Verlag, Heidelberg, 2007

Stöber et al. (2009) Stöber, C.; Gruber, G.; Krehmer, H.; Stuppy, J.; Westphal, C.: Herausforderung Design for X (DfX). In: Proceedings des 20. Symposiums Design for X. Neunkirchen, 2009

Strogatz (2001) Strogatz, S.H.: Nonlinear Dynamics And Chaos: With Applications to Physics, Biology, Chemistry and Engineering. 1. Auflage, Westview Press, 2001

Sturm et al. (2010) Sturm, A.; Dori, D.; Shehory, O.: An Object-Process-Based Modeling Language for Multiagent Systems. In: IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews. Volume 40, Nr. 2, 2010

Suh (1990) Suh, N.P.: The principles of design. Oxford University Press, New York, 1990

Suh (1995) Suh, N.P.: Design and operation of large systems. In: Journal of Manufacturing Systems, Volume 14, Issue 3. Elsevier, Amsterdam, 1995

Suh (1998) Suh, N.P.: Axiomatic Design Theory for Systems. In: Research in Engineering Design, Volume 10, Issue 4. Springer, 1998

Suh / Do (2000) Suh, N.P.; Do, S.-H.: Axiomatic Design of Software Systems. In: CIRP Annals - Manufacturing Technology, Volume 49, Issue 1. Elsevier, Amsterdam, 2000

Tazir (2011) Tazir, N.: Viel diskutiert, selten wirklich angewendet – Systems Engineering. In: Produktdatenjournal Nr. 2/2011, ProSTEP iViP e.V., Darmstadt, 2011

Tazir (2012) Tazir, N.: Systems Engineering – Ohne Methoden kein System. In: Produktdatenjournal Nr. 1/2012, ProSTEP iViP e.V., Darmstadt, 2012

Tomiyama et al. (1989) Tomiyama, T.; Kiriyama, T.; Takeda, H.; Xue, D.; Yoshikawa, H.: Metamodel: A Key to Intelligent CAD Systems. In: Research in Engineering Design, Vol. 1, Nr. 1. Springer, New York, 1989

TU Chemnitz (2012) Webseiten der TU Chemnitz. URL: http://www.tu-chemnitz.de/studium/studiengaenge/diplom/systems_engineering.php, abgerufen am 23.11.2012

Umeda et al. (1990) Umeda, Y.; Takeda, H.; Tomiyama, T.; Yoshikawa, H.: Function, Behaviour, and Structure. In: Applications of Artificial Intelligent in Engineering V. Springer, Berlin, 1990

Umeda und Tomiyama (1995) Umeda, Y.; Tomiyama, T.: FBS Modeling: Modeling Scheme of Function for Conceptual Design. In: Proceedings of the 9th International Workshop on Qualitative Reasoning about Physical Systems. Amsterdam, Niederlande, 1995

Umeda et al. (1996) Umeda, Y.; Ishii, M; Yoskioka, M; Shimomura, Y.; Tomiyama, T.: Supporting conceptual design based on the function-behavior-state modeler. In: Artificial Intelligence for Engineering, Design, Analysis and Manufacturing, Vol. 10, Nr. 4. Cambridge University Press, Großbritannien, 1996

Umeda und Tomiyama (1997) Umeda, Y.; Tomiyama, T.: Functional reasoning in design. In: IEEE Expert, Vol. 12, Nr. 2. IEEE, New York, USA, 1997

Uni Bremen (2012) Webseiten der Universität Bremen. URL: http://www.fb4.uni-bremen.de/studiengaenge/system_engineering/system_engineering.html, abgerufen am 23.11.2012

Valilei und Houshmand (2009) Valilei, O.F.; Houshmand, M.: Advantages of using SysML Compatible with ISO 10303-233 for Product Design and Development based on STEP Standard. In: Proceedings of the World Congress on Engineering and Computer Science (WCECS '09). San Francisco, USA, 2009

Vermaas (2010) Vermaas, P. E.: Technical functions: towards accepting different engineering meanings with one overall account. In: Proceedings des 8th international symposium on Tools and methods of competitive engineering (TMCE'10). Publisher: Delft University of Technology, Ancona, Italy, 2010

VDI-2206 (2004) Verein Deutscher Ingenieure (Hrsg.): Richtlinie VDI 2206, Entwicklungsmethodik für mechatronische Systeme. Beuth Verlag, 2004

VDI-2221 (1993) Verein Deutscher Ingenieure (Hrsg.): Richtlinie VDI 2221, Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte. Beuth Verlag, 1993

VDI-2803 (1996) Verein Deutscher Ingenieure (Hrsg.): Richtlinie VDI 2803, Funktionenanalyse - Grundlagen und Methode. Beuth Verlag, 1996

W3C OWL (2004) World Wide Web Consortium (Hrsg.): Web Ontology Language (OWL) Specification, 2004. URL: http://www.w3.org/2004/OWL/

Weber (2005) Weber, C.: CPM/PDD – An Extended Theoretical Approach to Modelling Products and Product Development Processes. In: Proceedings of the 2nd German-Israeli Symposium on Advances in Methods and Systems for Development of Products and Processes. Stuttgart, 2005

Weber (2007) Weber, C.: Looking at "DFX" and "Product Maturity" from the Perspective of a New Approach to Modelling Product and Product Development Processes. In: Proceedings of the 17th CIRP Design Conference. Heidelberg, 2007

Weber (2013) Weber, C.: Modelling Products and Product Development Based on Characteristics and Properties. In: Proceedings of the International Workshop on Models and Theories (IWMT'13). Bangalore, India, 2013

Weilkiens (2008) Weilkiens, T.: Systems Engineering mit SysML/UML: Modellierung, Analyse, Design. 2. Auflage, dpunkt Verlag, 2008

Weyer (2006) Weyer, J.: Die Zukunft des Autos – das Auto der Zukunft. Arbeitspapier Nr. 14 der Wirtschafts- und Sozialwissenschaftliche Fakultät an der Universität Dortmund. Dortmund, 2006

Wiedner (2013) Wiedner, A.J.: Feldstudie zur Identifikation der von Konstrukteuren praktizierten Handlungsmuster bei der Funktion-Gestalt-Synthese. IPEK Forschungsbericht Band 65. Hrsg.: o. Prof. Dr.-Ing. Dr. h.c. A. Albers, Karlsruher Institut für Technologie (KIT), 2013

Wölkl und Shea (2009) Wölkl, S.; Shea, K.: A Computational Product Model for Conceptual Design using SysML. In: Proceedings of the ASME 2009 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE 2009). San Diego, USA, 2009

Yoshikawa (1981) Yoshikawa, H: General Design Theory and a CAD system. In: Man-Machine Communication in CAD/CAM. North Holland Publishing, 1981

Zingel et al. (2012) Zingel, C.; Albers, A.; Matthiesen, S.; Maletz, M.: Experiences and Advancements from One Year of Explorative Application of an Integrated Model-Based Development Technique Using C&C²-A in SysML. In: IAENG International Journal of Computer Science, Volume 39, Nr. 2, 2012

Zingel (2012)* Zingel, C.: Model-Based Systems Engineering mit SysML - Einführung und Schulung im Werkzeug Artisan Studio. Institutsinterne Schulungspräsentation, IPEK – Institut für Produktentwicklung am Karlsruher Institut für Technologie, Karlsruhe, 2012 (* = nicht veröffentlicht)

10 Studien-, Diplom-, Bachelor- und Masterarbeiten

Nachfolgend werden die durch den Autor betreuten studentischen Arbeiten im Rahmen seiner Forschungsarbeit auflistet. Weiterhin sind auch die eigene Studien- und Diplomarbeit des Autors als Vorarbeiten der vorgestellten Forschungsergebnisse aufgeführt.

Bopp (2010) Bopp, M.: Implementierung und Validierung einer Schnittstelle zwischen einer SysML-Umgebung und Pro/Engineer. Studienarbeit Nr. 1806 am IPEK – Institut für Produktentwicklung am Karlsruher Institut für Technologie, 2010

Bull (2012) Bull, A.: Entwicklung einer Kommunikationsbasis zwischen CAD-Konstruktion und Elektrik-/ Elektronikentwicklung in der Automobilbranche. Studienarbeit Nr. 2067 am IPEK – Institut für Produktentwicklung am Karlsruher Institut für Technologie, 2012

Feierabend (2010) Feierabend, F.: Validierung eines Fahrzeugsimulationsmodells durch Parametervariation am Realfahrzeug. Studienarbeit Nr. 1707 am IPEK – Institut für Produktentwicklung am Karlsruher Institut für Technologie, 2010

Gloderer (2010) Gloderer, M.: Virtual Prototyping: Vernetzung der Metasprache SysML mit einem Simulationsmodell. Studienarbeit Nr. 1824 am IPEK – Institut für Produktentwicklung am Karlsruher Institut für Technologie, 2010

Gnaedinger (2013) Gnaedinger, B.: Entwicklung einer Methode zur funktionalen Analyse technischer Systeme auf Basis des C&C²-A am Beispiel eines elektrisch aktuierten Zweiganggetriebes. Bachelorarbeit Nr. 2362 am IPEK – Institut für Produktentwicklung am Karlsruher Institut für Technologie, 2013

Hauth (2012) Hauth, S.: Modellbasierte, modulare Abbildung einer Gesamtfahrzeug-Prüfumgebung für Produktportfolio-Management. Studienarbeit Nr. 2204 am IPEK – Institut für Produktentwicklung am Karlsruher Institut für Technologie, 2012

Heine (2010) Heine, J.: Aufbau und Validierung einer Prüfumgebung für Fahrbarkeitsbewertungen am Rollenprüfstand. Studienarbeit Nr. 1713 am IPEK – Institut für Produktentwicklung am Karlsruher Institut für Technologie, 2010

Jenz (2010) Jenz, J.: Innovation eines kennfeldbasierten Simulationsmodells anhand Parameter-variation am Gesamtfahrzeug. Studienarbeit Nr. 1758 am IPEK – Institut für Produktentwicklung am Karlsruher Institut für Technologie, 2010

Kruppok (2012) Kruppok, K.: Evaluation einer Softwareschnittstelle zwischen SysML und Simulink anhand eines hybriden Antriebstranges. Studienarbeit Nr. 2113 am IPEK – Institut für Produktentwicklung am Karlsruher Institut für Technologie, 2012

Martini (2012)* Martini, S.M.: Entwicklung einer Methode zur funktionalen Produktarchitekturmodellierung am Beispiel einer 2D-Flachbettlasermaschine. Diplomarbeit Nr. 2227 am IPEK – Institut für Produktentwicklung am Karlsruher Institut für Technologie, 2012 (* = Sperrvermerk bis 10.2015)

Schnurr (2009) Schnurr, C.: Entwicklung einer energieeffizienten Fahrstrategie unter Nutzung eines Aktiven Gaspedals. Studienarbeit Nr. 1658 am IPEK – Institut für Produktentwicklung am Karlsruher Institut für Technologie, 2009

Sleczek (2009) Sleczek, K.: Entwicklung und Konstruktion einer Fahrzeugfesselung für Fahrbarkeitsuntersuchungen am Akustikrollenprüfstand. Studienarbeit Nr. 1636 am IPEK – Institut für Produktentwicklung am Karlsruher Institut für Technologie, 2009

Tschöpe (2010) Tschöpe, T.: Verifizierung des Einsatzspektrums einer hochautomatisierten Fahrbarkeitsprüfumgebung auf dem IPEK-Allradrollenprüfstand. Diplomarbeit Nr. 1874 am IPEK – Institut für Produktentwicklung am Karlsruher Institut für Technologie, 2010

Weigel (2012) Weigel, J.G.: Entwicklung einer Implementierung des Contact & Channel – Approaches (C&C²-A) in der Systems Modeling Language (SysML). Bachelorarbeit Nr. 2246 am IPEK – Institut für Produktentwicklung am Karlsruher Institut für Technologie, 2012

Winter (2010) Winter, R.: Implementierung des C&C-M mittels Erweiterung des SysML-Metamodells und Anwendung am Beispiel einer Fahrbarkeitsprüfumgebung. Studienarbeit Nr. 1755 am IPEK – Institut für Produktentwicklung am Karlsruher Institut für Technologie, 2010

Zingel (2007) Zingel, C.: Modellbasierte Funktionsentwicklung mechatronischer Systeme am Beispiel des Hybridfahrzeugs Toyota Camry. Studienarbeit Nr. 1436 am IPEK – Institut für Produktentwicklung der Universität Karlsruhe (TH), 2007

Zingel (2008) Zingel, C.: Konzeptionierung einer Entwicklungsumgebung für Fahrbarkeitsuntersuchungen am Prüfstand. Diplomarbeit Nr. 08-F-0017 am Lehrstuhl für Fahrzeugtechnik der Universität Karlsruhe (TH), 2008

11 Anhang 1: Modellierungsregeln

Nachfolgend werden die wichtigsten Modellierungsregeln zur Anwendung der Modellierungstechnik nach ihren Aspekten untergliedert aufgeführt. Darüber hinaus sind weiterhin die in der SysML-Spezifikation beschriebenen Modellierungsregeln einzuhalten.

11.1 Anwendungsfallmodellierung

- Die Modellierung des Systemzwecks durch *Use Cases* und *Target Functions** betrachtet das zu entwickelnde System als Black Box und definiert somit die gewünschten Funktionen ausschließlich auf Gesamtsystemebene.
- *Target Functions** beschreiben Ziel-Funktionen und sind ein Element des Zielsystems.
- Zur Beschreibung von Abläufen in Use Cases dürfen ausschließlich *Target Functions** verwendet werden.
- Aus *Target Functions** können ausschließlich *Continuous Function RQs** oder *Discrete Function RQs** abgeleitet werden.
- *Target Functions** können keine Anforderung erfüllen.
- Logische Flüsse (*Control Flows*) und Objektflüsse (*Object Flows*) dürfen in jeweils eigenen *Activity Diagrams* derselben *Target Function** modelliert werden.

11.2 Anforderungsmodellierung

- *Stakeholder Objectives** dürfen nur in Absprache mit dem Auftraggeber modifiziert werden.
- *Stakeholder Objectives** können mittels *refine*-, *containment*- und *trace*-Beziehung strukturiert werden, dürfen aber nicht aus anderen Anforderungsarten abgeleitet werden.
- *Boundary Conditions** sind nur durch den Urheber der Randbedingung veränderbar (beispielsweise das Erscheinen einer neuen Norm, Veränderte Infrastruktur).
- *Boundary Conditions** können sich aus Systemschnittstellen ableiten, nicht jedoch aus Elementen des zu entwickelnden Objektsystems.

- *Technical Requirements** müssen immer aus einem anderen Element mit der *derive*-Beziehung abgeleitet werden, damit ihre Ursache nachvollziehbar ist. Dies vermeidet die Mitführung obsoleter Anforderungen.

11.3 Systemumgebungsmodellierung

- Die Systemumgebung betrachtet das zu entwickelnde System als Black Box. Daher werden hier ausschließlich die unmittelbar interagierenden bzw. wechselwirkenden Nachbarsysteme und die Schnittstellen an der Systemgrenze des zu entwickelnden Systems im *Internal Block Diagram* modelliert.

11.4 Funktionsmodellierung

- *Discrete Functions** dürfen nur in Zustandstransitionen verwendet werden.
- *Discrete Functions** werden durch mindestens einen Eingangsobjektfluss oder ein Ereignis ausgelöst und beenden sich selbst nach Beendigung ihrer Verarbeitungsschritte.
- *Continuous Functions** dürfen nur innerhalb von Systemzuständen verwendet werden.
- *Continuous Functions** werden durch kontinuierliche Eingangsobjektflüsse gestartet und extern z.B. durch ein Ereignis (*Event*) beendet.
- *Continuous Functions** können auch aufeinander folgen, laufen dann aber kontinuierlich parallel ab. Beispielsweise können logisch mehrere Energiewandlungen nacheinander durchgeführt werden, jedoch erfolgt die Gesamtwandlung zeitlich parallelisiert.

11.5 Zustandsmodellierung

- Zustandsdiagramme müssen für jedes System modelliert werden, das verschiedene Zustände einnehmen kann, um eine Zuweisung von zustandsabhängigen Wirkflächenpaaren in dynamischen Strukturen zu ermöglichen. Das zugehörige *State Diagram* sollte entsprechend dem zugeordneten System benannt sein, es kann direkt im betreffenden Strukturelement (*CSS** oder *Physical Component**) abgelegt werden.
- Die Regeln der Funktionsmodellierung sind auch bei der Verwendung in *State Diagrams* einzuhalten.

11.6 Modellierung funktionaler Strukturen

- *CSS** erhalten zur eindeutigeren Darstellung und der Eingrenzung möglicher Wertetypen explizit Wirkflächenarten (*Ports*) entsprechend ihrer Flussart (Stoff, Energie, Information oder Kombinationen daraus).
- Die Art einer Wirkfläche (*Port*) muss der zu übertragen Flussart (dem Wertetyp des *PIN*s der ausgeführten *Activity*) entsprechen.
- *CSS** können Parameter (*Values*) erhalten, die jedoch zwingend zur Beschreibung der Funktion erforderlich sein müssen, welche die *CSS** ausführt.
- Da Wirkflächen (*Ports*) keine Parameter (*Values*) erhalten können, sind für den Fall, dass relevante Parameter zu speichern sind, diese mit entsprechender Kennzeichnung in der *CSS** abzulegen, an der die Wirkfläche angebunden ist.
- Für die Modellierung mechatronischer Systeme wird ausdrücklich die Einhaltung der Regeln gemäß dem Schichtenmodell nach BERTSCHE bei der Modellierung von Wirkflächenpaaren empfohlen[530].

11.7 Modellierung physischer Strukturen

- Physische Strukturen basieren auf den funktionalen Strukturen, können jedoch mehrere *CSS** in einer *Physical Component** vereinen.
- Die Aufteilung einer *CSS** in mehrere *Physical Components** ist strikt untersagt! Es besteht jedoch die Möglichkeit, die *CSS** weiter zu dekomponieren und deren Sub-*CSS** den entsprechenden *Physical Components** zuzuordnen.
- Um Fehler und Mehrfachmodellierung gleicher Sachverhalte zu vermeiden, werden die *Physical Components** per Spezialisierung mit *CSS** verbunden, wodurch die *Phyiscal Components** die Wirkflächen (*Ports*) und Parameter (*Values*) der *CSS** erben. Somit wird gleichzeitig vermieden, dass funktionsrelevante physische Merkmale während der Modellierung der physischen Struktur unzulässig verändert werden.

11.8 Zusicherungsmodellierung

- Zusicherungen (*Constraints*) müssen vor ihrer Verwendung in einem konkreten Kontext immer als Zusicherungsblock (*Constraint Block*) mit ihren Parametern in einem eigenen Ordner (*Package*) im Modell abgelegt werden.

---

[530] Bertsche et al. (2009), S. 89ff.

- Die aus einem Zusicherungsblock (*Constraint Block*) instanziierten Zusicherungen (*Constraints*) werden im Zusicherungsdiagramm, das direkt in einem Block (bzw. einer *CSS** oder einer *Physical Component**) angelegt wird, modelliert. Darin werden die Parameter (*Values*) des Blocks den entsprechenden Parametern der Zusicherung zugeordnet.

11.9 Testfallmodellierung

- Testfälle (*Test Cases*) sind spezialisierte Anwendungsfälle (*Use Cases*), die eine bestimmte Ausprägung eines Anwendungsfalls verifizieren können. Sie können daher zur Verifikation von *Continuous Functional RQ** und *Discrete Functional RQ** verwendet werden.
- Auch *Property RQ** dürfen durch *Test Cases* im Sinne der Eigenschaftsabsicherung verifiziert werden. Diese Testzyklen sind überlicherweise keine normalen Anwendungsfälle, sondern dedizierte und meist auch standardisierte Testzyklen (z.B. Dauerbelastungstests, Crashtests).

11.10 Vernetzung von Zielsystemelementen und Objektsystemelementen

- *Continuous Functional RQ** können nur durch *Continuous Functions** erfüllt werden (*satisfy*).
- *Discrete Functional RQ** können nur durch *Discrete Functions** erfüllt werden (*satisfy*).
- *Property RQ** können nur durch Strukturelemente wie *CSS** oder durch *Phyiscal Components** erfüllt werden (*satisfy*).

12 Anhang 2: Ontologiedarstellungen

Nachfolgend werden weitere Ausschnitte der Ontologie dargestellt.

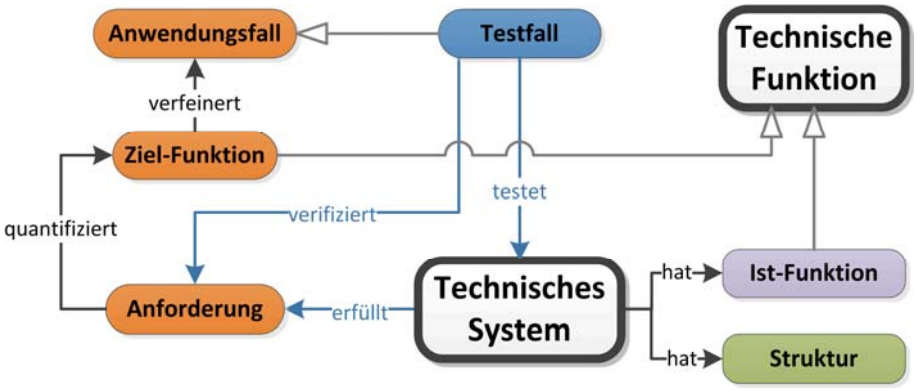


Bild 12-1: Verknüpfung von Ziel- und Objektsystemelementen über Testfall

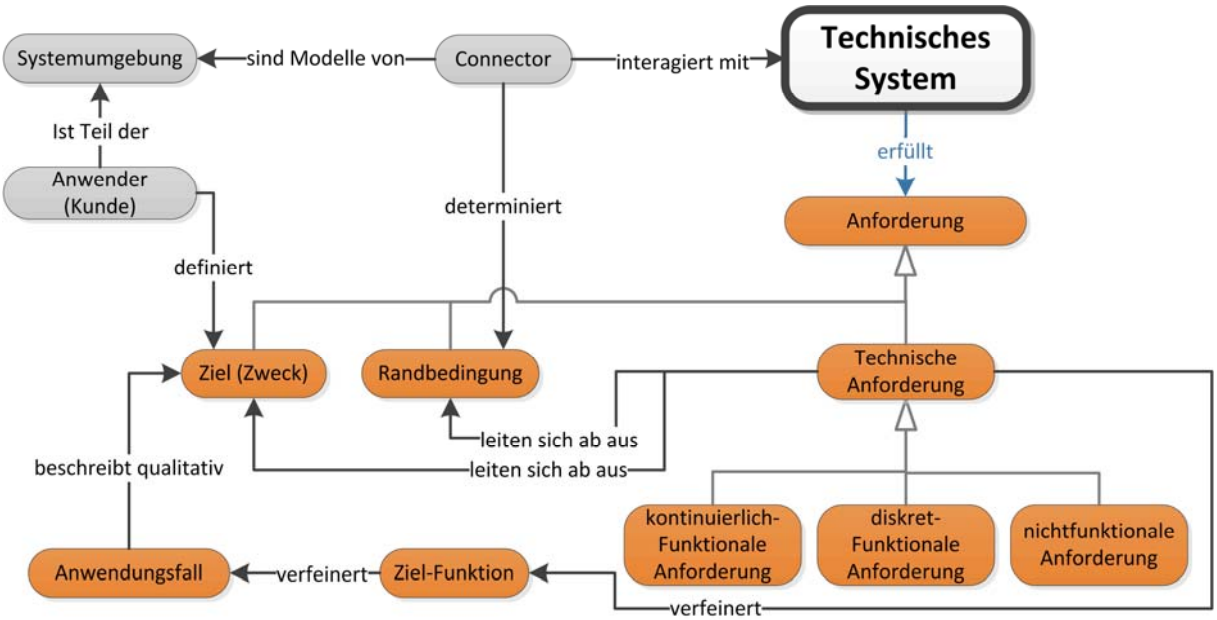


Bild 12-2: Elemente des Zielsystems

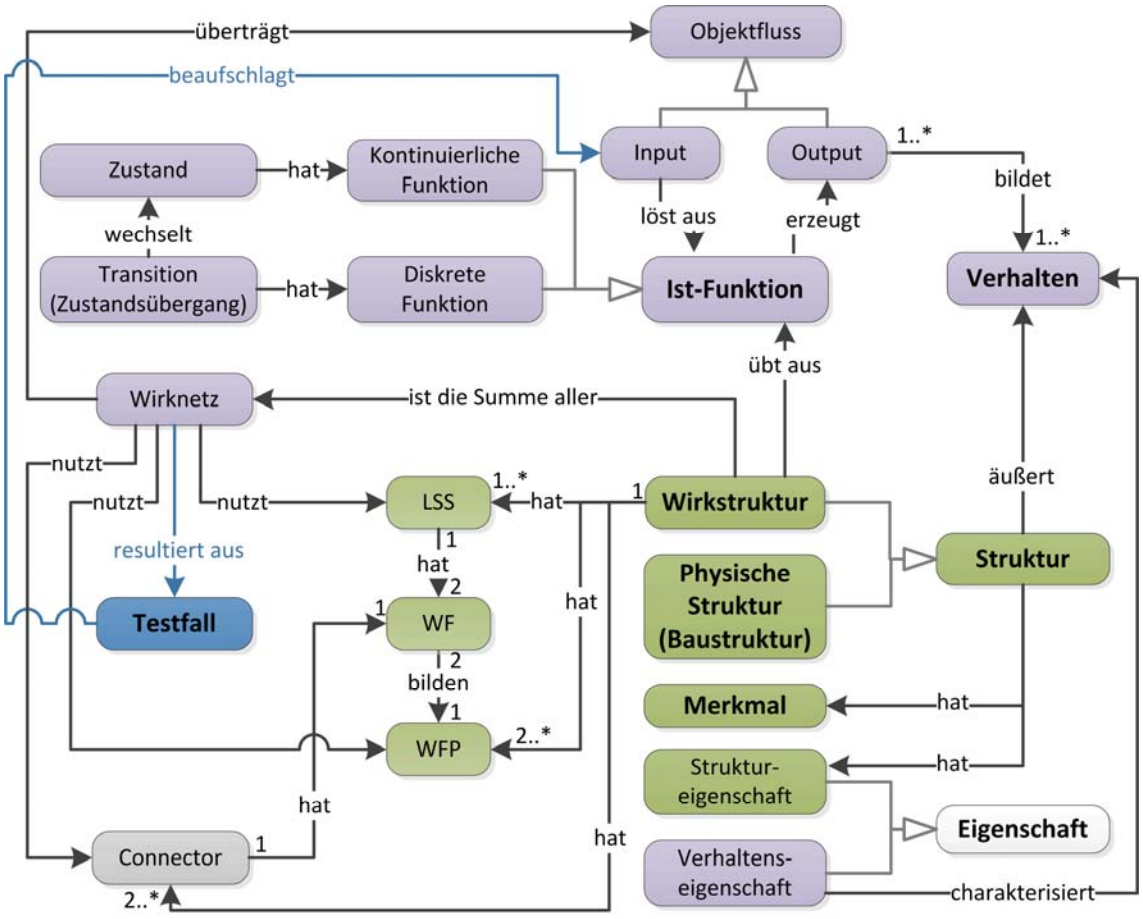


Bild 12-3: Verhaltens- und Strukturelemente des Objektsystems

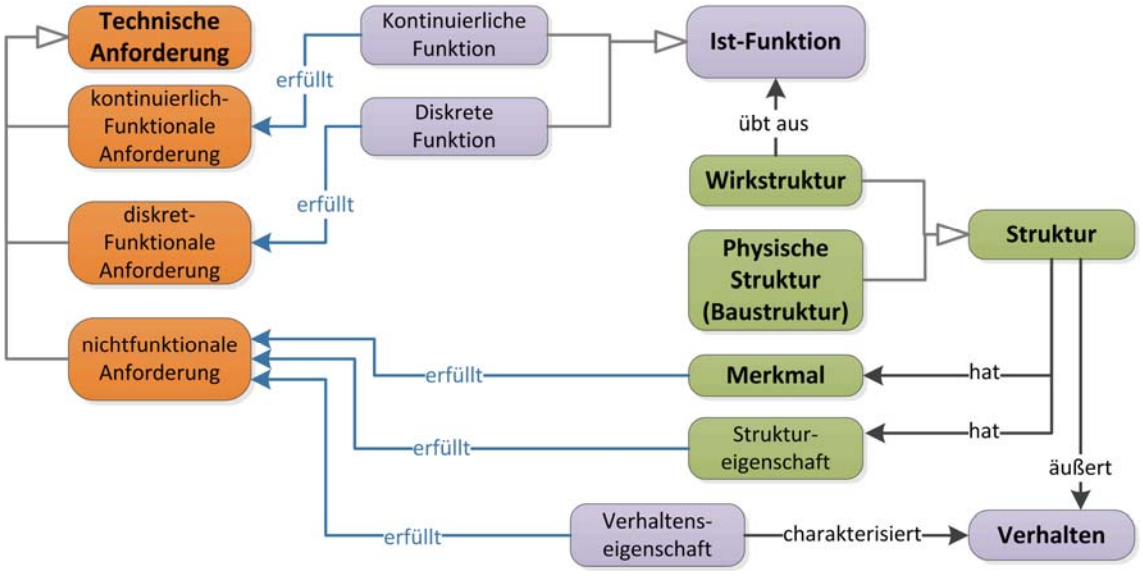


Bild 12-4: Kopplung von Ziel- und Objektsystem (Fokus hier: Erfüllungsbeziehung)