OTTO-VON-GUERICKE-UNIVERSITÄT MAGDEBURG

Fakultät für Informatik
Arbeitsgruppe Datenbanken und Software Engineering
Prof. Dr. rer. nat. habil. Gunter Saake
Dr.-Ing. Eike Schallehn

Masterarbeit

Konzept zur Produktdatenintegration für interdisziplinäre Vorgehensmodelle im Ingenieurwesen

vorgelegt von:

Student:	Szur, Marvin Konstantin
Studiengang:	Digital Engineering
Matrikelnummer:	184634
E-Mail:	marvin.szur@st.ovgu.de

Magdeburg, den 21. Januar 2016

Abstract

These days, the development of complex products leads to the integration of several disciplines (e.g. mechanics, electronics or software) rather than having only one discipline involved. All these disciplines view the product (or system) that is being developed from different perspectives. But not only their view of the product is different: they also have varying software tools for modeling and simulation. Thereby, the virtual product models they create using their software tools, differ as well.
The purpose of this master thesis is to develop a concept for the virtual model integration in the multidisciplinary product development.
Therefore, several procedure models for the product development are analyzed in a first step, whereby the focus will be on procedure models based on the V-model of the VDI 2206 guideline. Comparing these procedure models, the most important phases of interdisciplinary product development will be identified.
In the following chapter, different product models suitable for the interdisciplinary product development are being assessed. They are divided into four categories to give a better understanding of the different product models types and to explain the selection of analyzed product models. The four categories are: general engineering views on a system, data models and data formats, data integration concepts and ontology-based concepts.
In a last step, a concept for model integration is proposed for the data integration using earlier established „core models“ as a framework. This concept will then be compared with a use case which is published in technical journals.

Inhalt

Verzeichnis der Abbildungen

Verzeichnis der Tabellen

Verzeichnis der Abkürzungen

CAD	Computer-Aided Design
CAE	Computer-Aided Engineering
CAM	Computer-Aided Manufacturing
CAx	Computer-Aided x (x für die jeweilige rechnergestützte Technologie)
CPM2	Core Product Model 2
DM	Datenmanagement
ER	Entity Relationship
ERP	Enterprise Ressource Planning
FEM	Finite-Elemente-Methode
ISO	International Standardization Organization
MBSE	Model-based Systems Engineering
MTS	Mechatronische Systeme
NIST	National Institute of Standards and Technology
OAM	Open Assembly Model
OMG	Object Management Group
PDM	Product Data Management
PE	Produktentwicklung
PLM	Product Lifecycle Management
SE	Systems Engineering
UML	Unified Modeling Language
VDI	Verein deutscher Ingenieure
VHDL	VHSIC (Very High Speed Integrated Circuit) Hardware Description Language
XMI	XML Metadata Interchange
XML	Extensible Markup Language

.

1 Einleitung

1.1 Motivation

Die Entwicklung von technischen Systemen oder Produkten ist heutzutage meist durch einen domänenübergreifenden Charakter geprägt, hauptsächlich durch die Fachdomänen Mechanik bzw. Maschinenbau, Elektrik bzw. Elektronik sowie Softwareentwicklung. Das Zusammenwirken dieser Domänen wird oft auch als Mechatronik bezeichnet. In der domänenübergreifenden Produktentwicklung existieren eine Vielzahl von modellhaften Produktbeschreibungen (hier Produktmodelle genannt), um den Produktentwicklungsprozess zu unterstützen und das zu entwickelnde Produkt rechnergestützt abbilden zu können. Viele der Produktmodelle decken jedoch meist nur eine Domäne, einige bestimmte Entwicklungsphasen oder beides ab, jedoch nicht den gesamten Entwicklungsprozess, sofern dies möglich ist.

Ziel dieser Arbeit ist die Erarbeitung eines Konzept werden, mit dem der in den Vorgehensmodellen beschriebene Entwicklungsprozess möglichst umfassend anhand von bereits vorhandenen Produktmodellen unterstützt werden kann. Eine derartige Untersuchung ist nach gemäß der Recherchearbeiten für diese Arbeit noch nicht vorgenommen worden.

1.2 Fokus und Aufbau der Arbeit

Diese Arbeit untersucht den Einsatz von Produktmodellen zur modellbasierten Datenintegration. In Kapitel 2 werden dazu sowohl grundlegende Begriffe erklärt, als auch eine thematische Eingrenzung vorgenommen. Kapitel 3 untersucht mehrere Vorgehensmodelle, welche den Produktentwicklungsprozess beschreiben und unterstützen. Diese Vorgehensmodelle bilden den thematischen Kontext und stellen die Phasen der Produktentwicklung heraus, für die im anschließenden Kapitel 4 die Produktmodelle untersucht werden. In Kapitel 5 findet eine Konzeptvorstellung zur modellbasierten Datenintegration auf Grundlage von Kernmodellen statt.

Themenfelder wie Product Lifecycle Management (PLM), Systems Engineering (SE), Projektmanagement, semantische Integration oder Wissensmanagement werden in dieser Arbeit höchstens am Rande thematisiert. Auch Versions-/Konfigurationsmanagement, Produktdatenmanagement (PDM) bzw. typische PDM-Systeme sollen we-

niger Fokus dieser Arbeit sein. Eine genauere Eingrenzung erfolgt in den nachfolgenden Kapiteln.

2 Grundlegende Begriffe

In diesem Kapitel werden zum besseren Verständnis der Thematik einige Begriffe näher erläutert. Hierdurch sollen Inhalt und Fokus der Arbeit besser in den Vordergrund rücken. So werden bei den Begriffserläuterungen auch thematische Einschränkungen und Ausrichtungen vorgenommen, um besser klarzustellen, welche thematische Richtung diese Arbeit verfolgt.

Zunächst wird auf die Produktentwicklung eingegangen sowie auf die damit in Zusammenhang stehenden und an diese Arbeit angrenzenden Themenfelder des Product Lifecycle Management und Systems Engineering. Danach werden die Modellbegriffe – Vorgehensmodell und Produktmodell – erklärt. Die Begriffsklärungen erfolgen einerseits allgemein für diese Arbeit, andererseits konkrete die Modelle für das methodische Vorgehen in der Produktentwicklung (Vorgehensmodelle) und die Modelle, Abbildungen und Ähnliches für zu entwickelnde Produkte oder Systeme (Produktmodelle).

2.1 Produktentwicklung

Produktentwicklung ist ein Teil der Produktentstehung, welche wiederum ein Teil des Produktlebenszyklus ist. Der Schwerpunkt der Betrachtung in dieser Arbeit liegt auf der Produktentwicklung von physischen Produkten bzw. Produkten, welche vorwiegend mechanischer und elektrischer bzw. elektronischer Art sind. Entwicklung von reinen Software-Produkten oder sehr softwarelastigen ist nicht Teil der Betrachtung. Im Folgenden werden zum besseren Verständnis des Begriffes einige Definitionen zitiert.
Ganz allgemein beschrieben, wird nach Pahl/Beitz in [PBFG13] wird die Produktentwicklung als Entwicklungs- und Konstruktionsprozess bezeichnet, welcher „die Summe aller Arbeitsschritte von der Produktidee bis zur Erzeugung der Fertigungsunterlagen“ umfasst. Weiterhin wird das Anfallen von Produktdaten sowie der Einsatz von Modellen in der Produktentwicklung thematisiert.

Martin Eigner, Daniil Roubanov und Radoslav Zafirov liefern eine spezifischere Definition in [ERZ14] für Produktentwicklung:

> *Die integrierte, multidisziplinäre Produktentwicklung umfasst alle Tätigkeiten und Disziplinen, die das Produkt und sein zur Produktion, Betrieb und Entsorgung benötigtes Umfeld (Werkzeuge, Vorrichtungen, Maschinen, Anlagen,...) über den Produktlebenszyklus, alle beteiligten Disziplinen und die Zuliefererkette beschreiben. Das Ergebnis ist eine vollständige Produktdefinition („Intellectual Product“), die aus sichten-*

> *und phasenorientierten Produktstrukturen und allen zugehörigen Dokumenten und Konfigurationen besteht.*

Zur Produktdefinition können beispielsweise Dokumente und Zeichnungen eingesetzt werden. Durch die Rechnerunterstützung ist zunehmend der Einsatz von modellhafter Produktdefinition notwendig oder zumindest vorteilhaft. Daher definieren Eigner et al. auch die „Modellbasierte Virtuelle Produktentwicklung“, deren Definition speziell in dieser Arbeit essenziell ist:

> *Modellbasierte Virtuelle Produktentwicklung (MVPE) ist die durchgehende, rechnerunterstützte, formale Modellbildung und Dokumentation entlang aller entwicklungsrelevanten Phasen des Produktlebenszyklus mit der Zielsetzung der Weitergabe des Modells in die nächste Entwicklungsphase sowie der Weiterverwendung dieser Modelle für Simulation, Validierung und Verifikation. Ziel ist die frühe Erarbeitung des Produkt- und Produktionswissens und damit das frühzeitige Optimieren von Produkteigenschaften im Sinne einer ganzheitlichen Optimierung des gesamten Produktlebenszyklus sowie die drastische Reduzierung von physischen Prototypen.*

Auf diese Definition zur MVPE stützt sich der Aufbau und das Vorgehen in dieser Arbeit. Modelle zur (partiellen oder möglichst vollständigen) Abbildung von Produkten werden in dieser Arbeit einen Schwerpunkt darstellen, worin sich auch die Betrachtung und Analyse der Produktmodelle in Kapitel 4 begründet.

Für diese Arbeit wird die Produktentwicklung gemäß dieser Definitionen verstanden, speziell mit den Aspekten der Multidiszplinarität (bzw. Interdisziplinarität oder domänenübergreifend, in dieser Arbeit gleichbedeutend verwendet) sowie dem Einsatz von Modellen zur Abbildung des zu entwickelnden Produktes.

2.2 Verwandte und angrenzende Themenfelder

Sowohl zur Begriffsklärung als auch zur Eingrenzung und besseren Verständnis der in dieser Arbeit behandelten Thematik sollen in diesem Unterkapitel die Begriffe des Systems Engineering (SE), des Product Lifecycle Management (PLM) sowie des Product Data Management (PDM) erläutert werden.

Antti Saaksvuori und Anselmi Immonen definieren den PLM-Begriff in [SI08] wie folgt:

> *PLM is a holistic business concept developed to manage a product and its lifecycle including not only items, documents, and BOM's, but also analysis results, test specifications, environmental component information, quality standards, engineering requirements, change orders,*

> *manufacturing procedures, product performance information, component suppliers, and so forth.*

Gemäß dieser Definition greift PLM auf einer größeren und allgemeineren Ebene an als es in dieser Arbeit das Ziel ist, da z.B. Informationen über die Fertigung oder Zulieferer abseits der Betrachtung in dieser Arbeit liegen.

Der Begriff PDM, welcher bisweilen synonym mit PLM verwendet wird, beschreibt hingegen laut nur eine Teilmenge des PLM [SI08]. Meist kommen eigens dafür entwickelte Software-Werkzeuge zum Einsatz, welche allgemein als PDM-Systeme bekannt sind. Bei den typischen Grundfunktionen bzw. die zu verwaltenden Aspekte eines PDM-Systems handelt es sich nach Martin Eigner und Ralph Stelzer in [ES09] dabei u.a. um:

- Stamm und Struktur eines Produktes
- Dokumentenmanagement
- Projektmanagement
- Workflow-Management
- Freigabe- und Änderungswesen
- Viewing, Redlining und Digital Mock-Up (digitales Versuchsmodell)
- Input/Output-Management
- Archivierung, Sicherung
- Integration

Diese Definition zeigt, dass selbst der Begriff des PDM für diese Arbeit zu weit greift, da der für diese Arbeit relevante Themenbereich der Integration von Produktdaten, die in der Entwicklung anfallen, lediglich ein Teilbereich der PDM darstellt.
Das Systems Engineering ist laut Tim Weilkiens in [We09] eine Disziplin, die über den gängigen Disziplinen wie Elektronik oder Mechanik steht. Zunächst bezieht er sich die Definition des Technical Board International Council on Systems Engineering (INCOSE), welche wie folgt lautet:

> *Das Systems Engineering konzentriert sich auf die Definition und Dokumentation der Systemanforderungen in der frühen Entwicklungsphase, die Erarbeitung des Systemdesigns und die Überprüfung des Systems auf Einhaltung der gestellten Anforderungen unter Berücksichtigung der Gesamtproblems: Betrieb, Zeit, Test, Erstellung, Kosten & Planung, Training & Support und Entsorgung.*

Insgesamt beschreibt Weilkiens das SE als eine Metadisziplin, die technische aber auch wirtschaftliche organisatorische Aspekte mit einschließt. Diese Disziplin reicht

über die Produktentwicklung hinaus und schließt u.a. die Produktion und Betriebsphase mit ein. Als konkrete Aufgabenbereiche nennt Weilkiens dabei:

- Projektmanagement
- Anforderungsanalyse
- Anforderungsmanagement
- Systemdesign
- Systemverifikation
- Systemvalidierung
- Systemintegration
- Anforderungsdefinition
- Risikomanagement

Aus diesen Aufgabenbereichen wird deutlich, dass das SE in einem größeren Maßstab angesiedelt ist. Jedoch sind einige Aspekte und Gedanken des SE für die Produktentwicklung durchaus von Relevanz. Erwähnenswert für diese Arbeit ist speziell das „Model-based Systems Engineering“ (MBSE), welches den Einsatz von Modellen anstelle von Dokumenten vorsieht. Derartige Konzepte werden in den späteren Kapiteln ausführlicher beschrieben.

2.3 Modellbegriff

Modelle dienen prinzipiell zur vereinfachten Darstellung von Sachverhalten oder Zusammenhängen, welche entweder bereits real existieren oder in die Realität umgesetzt werden. In der Regel erfolgt eine Reduktion auf die für einen bestimmten Zweck wichtigen Eigenschaften des realen Systems, Zusammenhangs oder Gedankenkonstruktes. Ist beispielsweise nur die grobe Geometrie eines realen Systems von Bedeutung, kann eine Handskizze aussagekräftig genug sein. Andererseits können auch Modelle zum Einsatz kommen, wenn viele Details noch nicht bekannt sind und zunächst nur die bisher bekannten Sachverhalte und Eigenschaften abgebildet werden können oder sollen.

Generell lässt sich sagen, dass Modelle sich in praktisch allen Fällen im Spannungsfeld zwischen Reduktion auf das Wesentliche und hinreichender Aussagekraft (für einen bestimmten Anwendungszweck) befinden. Bei rechnerinternen Modellen kommen noch Aspekte wie Speicherplatzbelegung und Verarbeitung hinzu.

Grundsätzlich können Modelle in folgender Art und Weise existieren:

- gedanklich
- textlich
- grafisch, visuell
- logisch, mathematisch
- physikalisch
- real, physisch, materiell

Die nachfolgenden Definitionen stellen den Modellbegriff sowohl allgemein als auch speziell für diese Arbeit dar. Lindemann definiert in [Lin09] ein Modell dabei ganz allgemein als:

> *Gegenüber einem Original zweckorientiert vereinfachtes, gedankliches oder stoffliches Gebilde, das Analogien zu diesem Original aufweist und so bestimmte Rückschlüsse auf das Original zulässt.*

Eigner et al. geben in [ERZ14] eine konkretere und für diese Arbeit wichtige Definition:

> *Ein Modell ist ein Abbild bzw. ein Vorbild für ein System oder einen Prozess. Ein Modell kann ein begriffliches (z. B. mathematisches, informationstechnisches) oder ein physisches (z. B. maßstäblicher stofflicher Prototyp) Gebilde sein.*
> *Nach Stachowiak wird ein Modell durch die folgenden drei Merkmale charakterisiert [Sta73]:*
>
> - *Abbildungsmerkmal: Jedes Modell ist Abbild oder Vorbild.*
> - *Verkürzungsmerkmal: Jedes Modell [ist] abstrahiert.*
> - *Pragmatisches Merkmal: Jedes Modell wird im Hinblick auf einen Verwendungszweck geschaffen.*

Zusammenfassend lässt sich festhalten, dass Modelle vereinfachte Abbildung von Produkten, Systemen, Sachverhalten oder sonstigen Zusammenhängen sind, welche entweder bereits realisiert sind oder noch realisiert werden sollen bzw. können. Es kann sich dabei um Zusammenhänge aus der Technik, aber auch beispielsweise aus Biologie oder Wirtschaftswissenschaft handeln, wobei in dieser Arbeit der Fokus auf technischen Systemen liegt.

2.4 Vorgehensmodelle

Vorgehensmodelle dienen in der Produktentwicklung dazu, in Form einer Richtlinie den Entwicklungsprozess zu unterstützen, bei dem verschiedene zu durchlaufende Phasen definiert sind [ERZ14].

Vorgehensmodelle in der Produktentwicklung sind einerseits auf zeitliche Phasen begrenzt und andererseits besitzen sie eine Genauigkeit bzw. eine Stufe der Abstraktion oder Konkretisierung, welche eine Aussage darüber liefert, in welcher Größenordnung ein Vorgehensmodell angesiedelt ist. Letzteres wird in Abbildung 1 von [Lin09] ersichtlich, nach welcher die Vorgehensmodelle im Spannungsfeld zwischen Mikro- und Makrologik sich klassifizieren lassen. Die in dieser Arbeit thematisierten Vorgehensmodelle liegen dabei vorwiegend im Bereich der Phasen/Arbeitsabschnitte. Ansätze wie PLM oder SE sind mehr auf der Makrologik- bzw. Gesamtprojekt-Ebene angesiedelt, weshalb diese hier nicht im Fokus liegen.

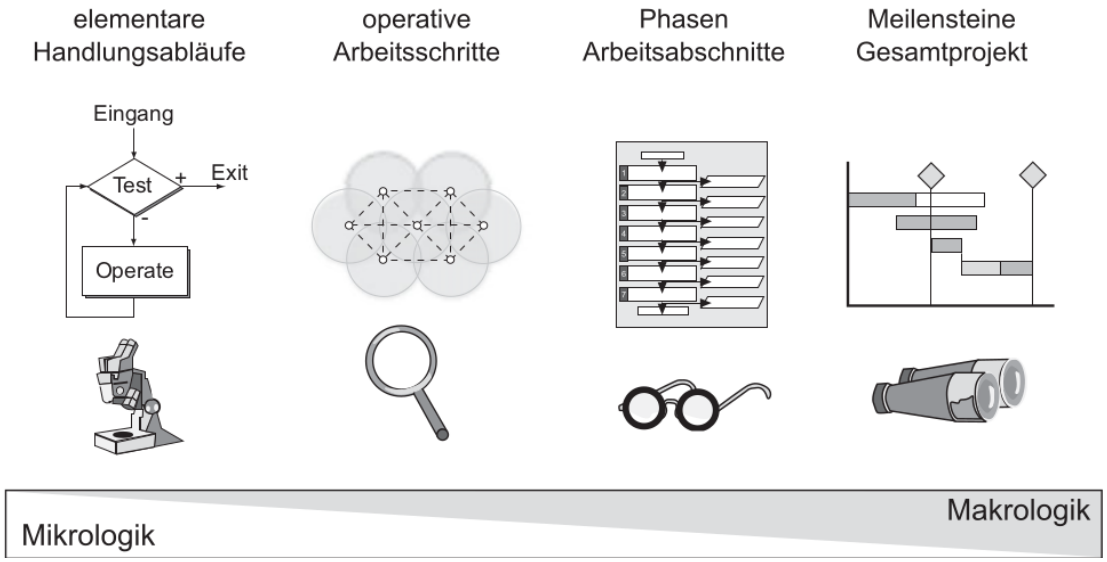

Abbildung 1: Diverse Auflösungsgrade des Produktentwicklungsprozesses [Lin09]

Ein in verschiedenen technischen Disziplinen verwendeter Typ von Vorgehensmodellen ist das V-Modell, welches seinem Namen vorwiegend seiner optischen Anmutung zu verdanken hat. Als das erste V-Modell gilt das, welches laut [Ba98] von Barry Boehm Anfang der 1980er Jahre veröffentlicht wurde, mit der Motivation, die Qualitätssicherung zu integrieren. Bei diesem wie auch anderen V-Modellen nimmt von links nach rechts der Reifegrad des Produktes bzw. zu entwickelnden Systems zu und von oben nach unten der Detaillierungsgrads, wodurch sich eine V-förmige Anordnung der Entwicklungsphasen ergibt.

Des Weiteren charakteristisch sind die sich auf der linken Seite befindlichen Synthese-Phasen und die dazu jeweils zugeordneten Testphasen zur Qualitätssicherung auf

der rechten Seite. Die V-Form verdeutlicht damit einerseits, dass die jeweils zusammengehörenden Synthese- und Testphasen auf dem gleichen Detaillierungsgrad des zu entwickelnden Produktes (physisch oder virtuell) angesiedelt sind, anderseits jedoch mit steigendem Detaillierungsgrad zeitlich näher beieinander liegen, wie in Abbildung 2 gezeigt ist. So liegen beispielsweise die Anforderungsdefinition und der Abnahmetest auf einem geringen Detaillierungsgrad, haben aber den größten zeitlichen Abstand zueinander.

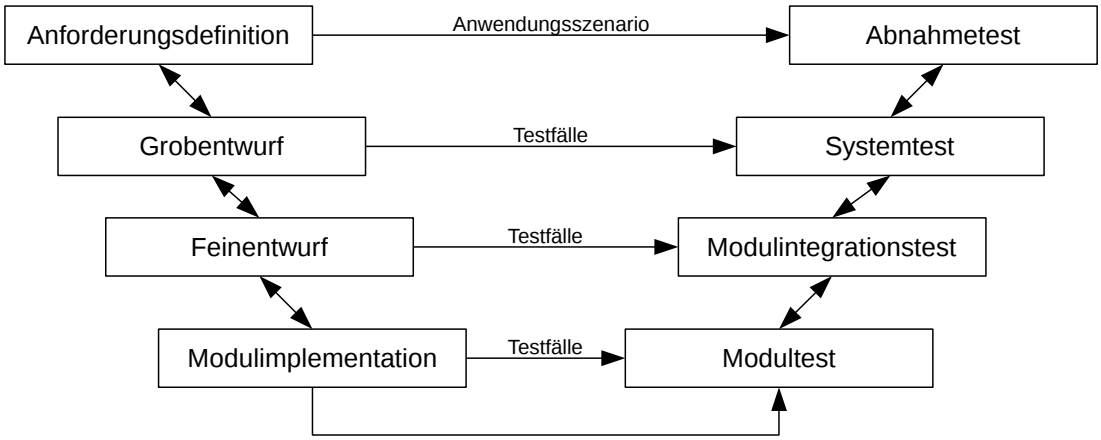

Abbildung 2: V-Modell nach [Ba98]

Dieses Konzept wurde auch von anderen Fachgebieten (vor allem in der Mechatronik und dem Systems Engineering) sowie von staatlichen Behörden für standardisierte Vorgehensmodelle aufgegriffen und bildet somit eine weit verbreitetes Grundkonzept für Vorgehensmodelle. Auf dieser Tatsache stützt sich die Auswahl der Produktmodelle, welche in Kapitel 3 genauer dargelegt wird.

2.5 Produktmodell

In dieser Arbeit dient der Begriff „Produktmodell“ als Sammelbegriff für Repräsentationen, Abbildungen und Dokumentationen von existierenden oder in Entwicklung befindlichen Produkten (bzw. Systemen). Für diese Arbeit sind dabei vor allem Produktmodelle relevant, die vorwiegend (oder sogar ausschließlich) mit Rechnern darstellbar sind. Der Vollständigkeit halber werden auch Produktmodelle untersucht, die im Grunde auch rein handschriftlich erstellt und verwendet werden können.

Dieser Begriff des Produktmodells soll ebenfalls mit einigen Definitionen aus der Fachliteratur für diese Arbeit deutlicher herausgestellt werden. So wird gemäß der

Konstruktionslehre von Pahl/Beitz in [PBFG13] ein Produktmodell folgendermaßen definiert:

> *Ein Produktmodell ist in diesem Zusammenhang eine rechnerinterne Darstellung, die alle relevanten Informationen über ein Produkt in hinreichender Vollständigkeit enthält. Ein Produktmodell enthält neben geometrischen Daten auch technisch-funktionale, technologische und baustrukturelle Informationen sowie auch solche zum Konstruktions- und Fertigungsprozess. Für die Abbildung dieser Informationsinhalte eigenen sich Partialmodelle (auch Phasenmodelle genannt), die zweckmäßig ausgegliederte Bestandteile eines Produkts repräsentieren.*

Teilweise wird das zu entwickelnde Produkt auch „System“ genannt (in dieser Arbeit werden daher - sofern nicht anders erwähnt - beide Begriffe synonym verwendet) und so definieren Eigner et al. in [ERZ14] ein Systemmodell für das modellbasierte Systems Enginering (MBSE) in Anlehnung an [FMS12] und [Hyb09]:

> *Das Systemmodell ist eine explizite Annäherung an, Repräsentation oder Idealisierung von ausgewählten Aspekten der Anforderungen, der Struktur, des Verhaltens, der Parameter, des Kontexts, der Validierung und Verifikation oder anderer Charakteristiken, die mit einem oder mehreren Systemen assoziiert sein können.*

Der Systemgedanke wird auch in der VDI-Richtlinie 2206 [VDI04] thematisiert und verdeutlicht ebenfalls den Nutzen von Produktmodellen bzw. die Notwendigkeit, eine gesamte Sicht auf ein Produkt zu erlangen und zu verwalten:

> *Erst der Systemgedanke, das heißt die funktionale und räumliche Integration vieler unterschiedlicher Technologien zu einem komplexen System und dessen ganzheitliche Betrachtung, ergibt eine neue Qualität und ist unabdingbar für die erweiterte mechatronische Gesamtfunktionalität.*

In der Produktentwicklung können Modelle zur Synthese oder zur Analyse verwendet werden [ERZ14]. In der VDI 2206 findet eine ähnliche Unterscheidung bei der Modellanalyse hinsichtlich der Zielrichtungen statt: Entweder findet eine Analyse zur Feststellung des Ist-Zustandes statt oder zur Analyse des möglichen Verhaltens. Ersteres deckt sich mit dem Analysemodell und Letzteres mit dem Synthesemodell.

Während die Synthesemodelle beschreiben, wie ein System oder Produkt realisiert werden soll (Spezifikation), dienen Analysemodelle dazu, ein bereits (real oder virtuell) entwickeltes System hinsichtlich bestimmter Eigenschaften zu untersuchen. Bedingt durch den Einsatz der Modelle und bedingt durch die unterschiedlichen in der Produktentwicklung vorhandenen Sichtweisen, Entwicklungsphasen und Domänen entstehen diverse unterschiedliche Modelle. Diese Modelle sollten möglichst konsis-

tent bzw. inhaltlich widerspruchsfrei sein und möglichst weitreichend einsetzbar sein, um den Einsatz verschiedener Produktmodelle gering zu halten. Genaueres dazu folgt im nächsten Unterkapitel.

2.6 Produktdaten- und Modellintegration

Bei Produktdaten handelt es sich grundsätzlich um jene Daten, welche über ein Produkt im Laufe des gesamten Lebenszyklus anfallen oder gezielt erhoben werden. Eine prinzipielle Unterscheidung der Produktdaten (auch als Produktinformationen bezeichnet) haben Antti Saaksvuori und Anselmi Immonen in [SI08] anhand von drei Gruppen vorgenommen (siehe Tabelle 1). Besondere Aufmerksamkeit gilt hierbei den Definitionsdaten innerhalb der Entwicklung eines Produktes.

Definitionsdaten des Produkts	Beschreibt physikalische und/oder funktionale Eigenschaften des Produktes (z.B. Form oder Funktion) sowohl in exakter technischer als auch in abstrakter konzeptueller Form.
Lebenszyklusdaten des Produktes	Daten im Zusammenhang mit dem Produkt und einer Lebenszyklusphase, z.B. Fertigung, Entwicklung, Nutzung oder Entsorgung.
Metadaten, die Produkt- und Lebenszyklusdaten beschreiben	Informationen über die Informationen: Art der Informationen, Ablageort, von welcher Person erstellt, Informationen über Zugriff etc.

Tabelle 1: Einteilung der Produktdaten nach [SI08]

Produktdaten und -informationen werden in der Regel in Form eines Modells dargestellt. Saaksvuori und Immonen beschreiben ein Produktdaten-/Informationsmodell dabei wie folgt:

> *A product data or information model is a conceptual model of the product in which information on the product and the connections between various information elements and objects are analyzed at a general, generic level.*

Zwischen verschiedenen Firmen, Organisationen, Fachbereichen bzw. Teilbereichen selbiger muss bei der Kooperation praktisch zwangsläufig irgendeine Form von Integration stattfinden, um diese Kooperation zu ermöglichen. Die Arten der Integration lassen sich dabei nach dem Referenzmodell des KOMFORCE-Arbeitskreises einordnen, welches in Abbildung 3 dargestellt ist. KOMFORCE steht für „Kommunikations- und Forschungskreis für Integrationstechnologien im Computer-Aided Design und Engineering“. Während die beiden oben Ebenen (verfahrenstechnische und prozesstech-

nische Integration) eher organisatorischer Natur sind, sind die beideren unteren Ebenen (modelltechnische und Systemtechnische Integration) technischer Natur.

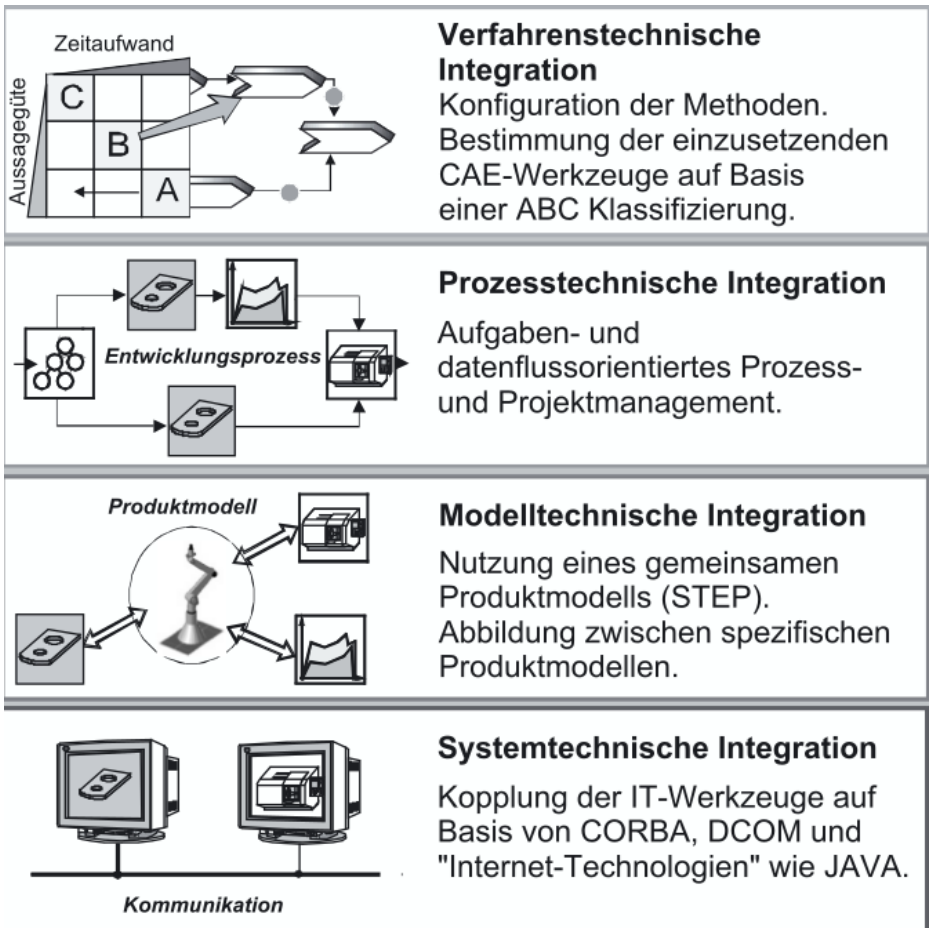

Abbildung 3: Integrationsebenen nach dem KOMFORCE-Referenzmodell [VDI04]

Für diese Arbeit ist aber die Modellintegration bzw. die modelltechnische Integration von zentraler Bedeutung, da die in Tabelle 1 geschilderten Daten sowie auch generell die Darstellung von zu entwickelnden Produkten oder System mittels Modellen umgesetzt sind. Die Relevanz von Produktmodellen wurde auch vom „sendler\circle“, einer in Deutschland ansässigen Interessengruppe aus der PLM-Branche, im Mai 2014 im bayrischen Hechenberg hervorgehoben. Zusammengefasst wurden in Form der sogenannten Hechenberger Thesen, welche in [We14] wie folgt lauten:

1. *Die Grundlage innovativer, “intelligenter”, vernetzter Produkte sind digitale Produktmodelle.*
2. *Das digitale Produktmodell muss alle Elemente der Mechanik, Elektrik, Elektronik und Software enthalten und ihr Zusammenwirken virtuell spiegeln können.*

3. *Digitale Modelle machen Entwicklung, Produktion und Betrieb komplexer Produkte beherrschbar.*
4. *Das durchgängige Management der digitalen Produktmodelle über ihren gesamten Lebenszyklus ist eine wichtige Voraussetzung für die Realisierung von Industrie 4.0.*

Bei der im Titel dieser Arbeit erwähnte Datenintegration handelt es sich für diese Arbeit somit um eine Integration von Produktmodellen, kurz Modellintegration. Der Entwicklungsprozess dient in Form der Vorgehensmodelle als Kontext.

Die Integration mittels Produktmodellen wurde bereits von Wenzel Seiler in [Sei85] beschrieben, wie Abbildung 4 anhand von zwei prinzipiellen Ansätzen zeigt: Entweder findet eine Kopplung der Modelle durch Transformation (Modelltransformation) statt oder alle Modelle sind mittels eines zentralen Modells kohärent (Modellkohärenz). Allerdings ist laut Seiler nur die Modellintegration für die Produktentwicklung geeignet, da in bei der Modelltransformation ein erhöhter Aufwand durch die zusätzlich anfallenden Transformationsprozesse notwendig ist.

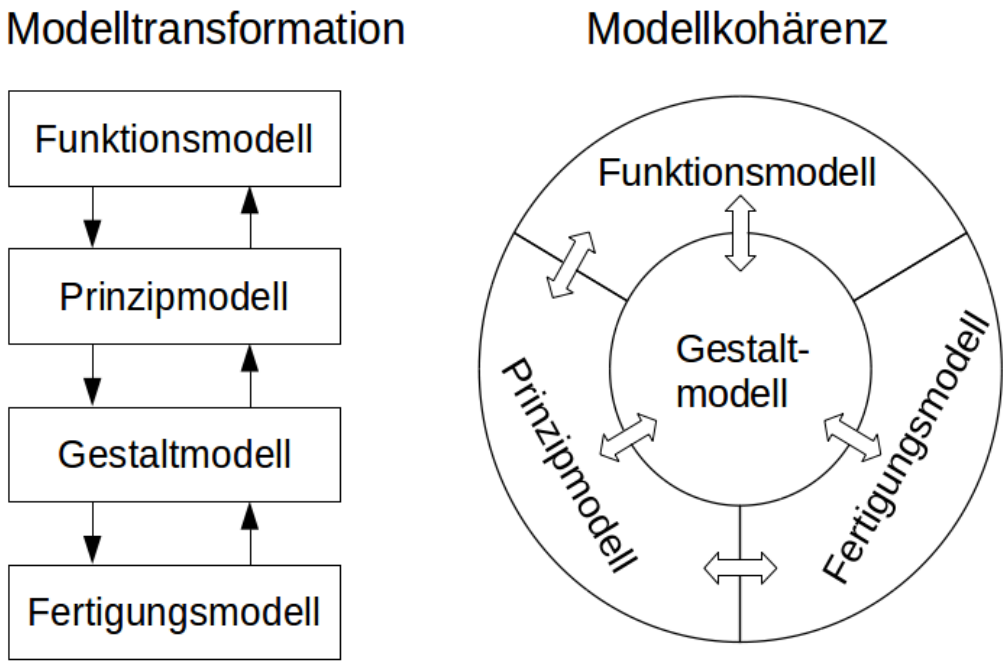

Abbildung 4: Modelltransformation und Modellkohärenz in Anlehnung an [Sei85]

Für die Modellintegration werden dafür in Kapitel 4 Produktmodelle untersucht, welche zumindest innerhalb einiger Phasen der Produktentwicklung möglichst als kohärente Produktmodelle dienen können. Im Vorfeld dazu werden zunächst die Vorge-

hensmodelle der interdisziplinären Produktentwicklung untersucht, für die es gemeinsame, kohärente Produktmodelle zu finden gilt.

3 Vorgehensmodelle

In diesem Kapitel werden die Vorgehensmodelle untersucht, welche als inhaltlicher Rahmen der Konzeptentwicklung dienen. Die Untersuchung hat das Ziel, die typischen und wichtigsten Phasen der interdisziplinären bzw. mechatronischen Produktentwicklung zu identifizieren, aber auch zu ermitteln, welche Rahmenbedingungen gelten oder inwiefern diese Vorgehensmodelle Aspekte wie Zusammenarbeit oder Rechnereinsatz berücksichtigen.

Zunächst wird eine thematische Eingrenzung vorgenommen, welche die Auswahl der untersuchten Vorgehensmodelle erläutert. Anschließend werden die Analysekriterien vorgestellt, mit denen die Untersuchung durchgeführt wird.

3.1 Eingrenzung und Unterteilung der Vorgehensmodelluntersuchung

In der Produktentwicklung existiert über verschiedene Fachdomänen hinweg eine Vielzahl von Vorgehensmodellen. Die nachfolgende Tabelle 2 zeigt die Vorgehensmodelle, die bei der Recherche dieser Arbeit identifiziert wurden. Es handelt sich dabei mehrheitlich um Vorgehensmodelle aus der Mechatronik sowie dem SE, da diese die Zusammenarbeit der verschiedenen fachlichen Bereiche, Domänen oder Disziplinen stark berücksichtigen, wenn nicht sogar als inhärente Eigenschaft besitzen. In der linken Spalte ist der Name der jeweiligen Methodik aufgeführt, bei der SE für Systems Engineering jeweils steht. Die zweite Spalte zeigt die an der Entwicklung maßgeblich beteiligten Personen, Organisationen oder beides sowie die dazugehörige Quelle. Im Falle der Quellen zu den SE-Vorgehensmodellen sei zusammenfassend auf die von der von Object Management Group (OMG) erstellte Liste der SE-Methoden in [www12] verwiesen, da diese Vorgehensmodelle hier nicht eingehend betrachtet werden.

Name des Vorgehensmodells	Personen, Organisation	Jahr	Domäne
VDI 2206	Heinz-Nixdorf-Institut [VDI04]	2004	Mechatronik
3-Ebenen-Modell nach Bender	Bender [Ben05]	2005	Mechatronik
W-Modell	Anderl, Nattermann [NA10]	2010	Adaptronik
Erw. V-Modell mit SE-Aspekten	Graessler, Hentze [GH15]	2015	Mechatronik, SE
RFLP (in CATIA V6)	Kleiner, Kramer [KK12]	2013	Mechatronik
MVPE/SysLM-Vorgehensmodell	Eigner et al. [Eig13] [Eig+14]	2012	Mechatronik
Vorgehen nach Isermann	Isermann [Iser08]	1999	Mechatronik
VDI 2221	Pahl/Beitz [PBFG13]	1993	Maschinenbau
Partitionierungsmethode	Jansen [Ja06]	2006	Mechatronik
Design Process of intelligent MTS	Oestersötebier et al.[Oe+12]	2012	Mechatronik
Salminen und Verho	Salminen, Verho [SV92]	1992	Mechatronik
Spiralmodell	Chan, Leung [CL96]	1996	Mechatronik
Kallenbach	Kallenbach [KBSS97]	1997	Mechatronik
Integrierter Entwicklungsprozess	Schön [Sch99]	1999	Mechatronik
Lückel	Lückel [LKS00]	2000	Mechatronik
Spezifikationstechnik	Gausemeier et al. [GFDK08]	2009	Mechatronik
Hierarchical Design Method	Hehenberger et al.	2010	Mechatronik
Sequential Design Process	Shetty, Kolk [SK98]	1998	Mechatronik
Simulationsbasiertes Vorgehen	Fabio Dohr [Doh14]	2014	Mechatronik
INERELA-Vorgehensmodell	Gausemeier, Feldmann [GF06]	2004	Mechatronik
Agile Design Methods for Mechatronics System Integration	Bricogne et al. [BTRE13]	2013	Mechatronik
X-Modell	Eigner et al. [ERZ14]	2014	Mechtronik
V-Modell XT 2.0	Weit e.V. [Weit06]	2006	Allg. Systeme
Telelogic Harmony-SE	Douglass (IBM)	2005	SE
Rational Unified Process for SE	Cantor et al. (IBM)	1996	SE
Object-Oriented Systems Engineering Method (OOSEM)	Friedenthal et al. (INCOSE)	1998	SE
Vitech MBSE	Long (Vitech Corp.)	2000	SE
JPL State Analysis	Dvorak et al.	2000	SE
Object Process Methodology	Dori	2002	SE
SYSMOD	Weilkiens	2008	SE
Fernandez ISE & Process Pipelines in OO Architectures	Fernandez	1998	SE
ISO-15288, OOSEM and Model-Based Submarine Design	Pearce, Hause	2012	SE
Alstom ASAP methodology	Ferrogalini	2012	SE
Pattern-based SE	Schindel	2010	SE
Arcadia	Polarsys/Capelle	2010	SE

Tabelle 2: Im Rahmen der Recherche identifizierte Vorgehensmodelle

Die Vorgehensmodelle des Systems Engineering werden in dieser Arbeit außen vor gelassen, da Systems Engineering in seiner Sichtweise und Herangehensweise auf die Produktentwicklung deutlich weiter gefasst ist und gemäß Abbildung 1 mehr auf dem Level des Gesamtprojektes angesiedelt ist. Bei den Vorgehensmodellen der Mechatronik hat sich vor allem das Vorgehen nach VDI 2206 als „de facto-Standard“ etabliert, da auf dessen Grundlage diverse weitere Vorgehensmodelle entstanden sind. In der Praxis scheint das V-Modell weiterhin eine hohe Relevanz zu besitzen, so setzt beispielsweise die von Siemens PLM Software entwickelte Methode des „Systems Driven Product Development“ auf das V-Modell der VDI 2206 [www7]. Außerdem kommt von Ulrich Sendler in [www8] das V-Modell heutzutage gemeinhin in der Produktentwicklung zum Einsatz.

Dadurch lässt sich darauf schließen, dass dieses Vorgehen nicht nur in der Forschung, sondern auch in der Anwendung eine große Zustimmung erhalten hat. Das V-Modell der VDI 2206 sowie die weiteren auf dieser Grundlage entstandenen Vorgehensmodelle können als eine Kategorie weit verbreiteter und vom Grundaufbau ähnlicher Vorgehensmodelle betrachtet werden, die als thematischer Rahmen für diese Arbeit dient. Die auswählten Vorgehensmodelle und ihr Jahr der Veröffentlichung lauten dabei wie folgt:

- Vorgehen nach Isermann (1999)
- Entwicklungsmethodik für mechatronische Systeme nach VDI 2206 (2004)
- 3-Ebenen-Modell nach Bender (2005)
- W-Modell für die Entwicklung adaptronischer Systeme nach Anderl und Natterman (2010)
- MBSE/RFLP-Methode nach Kleiner und Kramer (2012)
- Erweitertes V-Modell für MBSE/SysLM nach Eigner et al. (2012)
- Erweitertes V-Modell mit Aspekten des Systems Engineering nach Graessler und Hentze (2015)

Es existieren neben den oben aufgeführten Vorgehensmodellen noch weitere dem V-Modell ähnliche Entwicklungsmodelle wie z.B. das V-Modell des US-Transportministeriums [USDOT07] oder das V-Modell XT Bund von deutschen Bundesbehörden [BSI13]. Allerdings wurden diese nicht in die Betrachtung mit aufgenommen, da sie (ähnlich den SE-Vorgehensmodellen) sehr allgemein und weit gefasst und somit als

weniger geeignet für diese Arbeit gesehen werden. In den nachfolgenden Unterkapiteln werden die ausgewählten Vorgehensmodelle genauer beschrieben.

3.1.1 Vorgehen nach Isermann

Rolf Isermann stellt in [Iser08] ein Vorgehensmodell vor, welches das Älteste der in dieser Arbeit untersuchten Vorgehensmodelle ist, da es spätestens 1999 in der ersten Auflage von [Iser08] bereits vorgestellt wurde. Im Vergleich zur VDI 2206 oder dem 3-Ebenen-Modell ist es auch in deutlich mehr Schritte unterteilt, wie aus Abbildung 5 zu entnehmen ist.

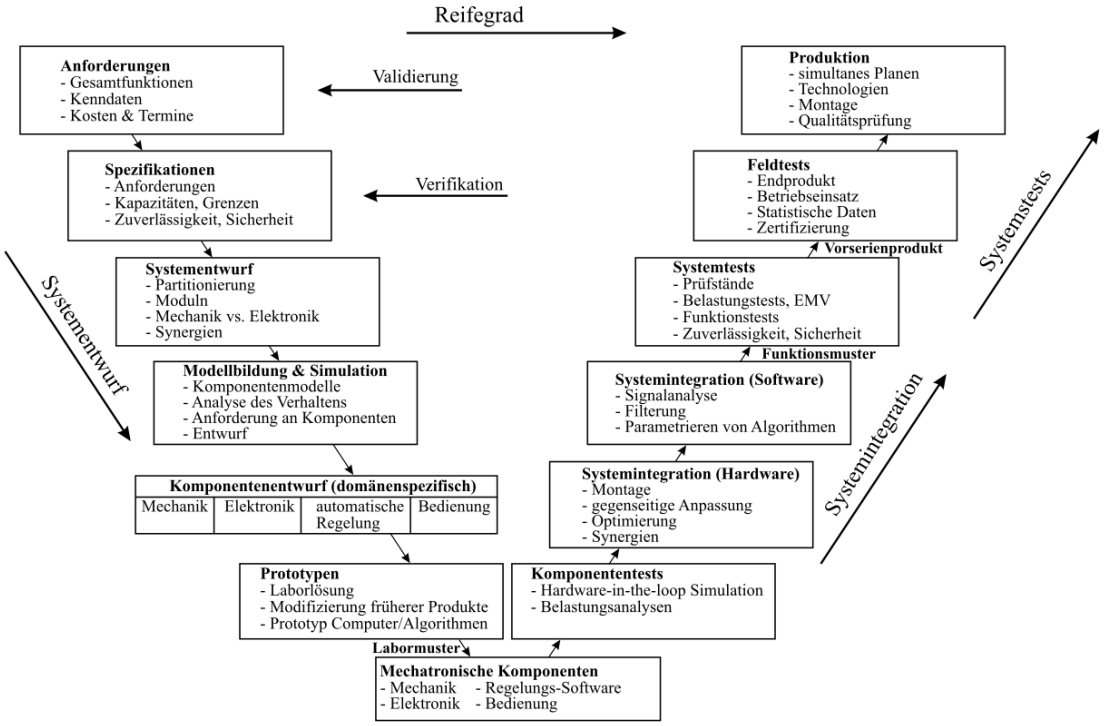

Abbildung 5: Vorgehensmodell nach Isermann [Iser08]

Das Vorgehensmodell ist in 13 Schritte unterteilt:

1. Anforderungen an die Entwicklung
2. Spezifikation
3. Systementwurf
4. Modellbildung und Simulation
5. Komponentenentwurf (Mechanik, Elektronik, automatische Regelung und Bedienung)
6. Prototypen

7. Mechatronische Komponenten
8. Komponentenprüfung
9. Systemintegration (Hardware)
10. Systemintegration (Software)
11. Systemtest
12. Feldtests
13. Produktion

Auffallender Unterschied zu den anderen, vorwiegend der VDI 2206 angelehnten, Vorgehensmodellen ist bei diesem, dass die Integration auf zwei Phasen verteilt ist. Weiterhin existieren explizite Phasen für Tests, welche in den anderen Vorgehensmodellen meist mit den Integrationsphasen zusammengefasst sind.

Nicht erkennbar hingegen ist das das Vorhandensein von Iterationen. Auch die Modellunterstützung ist nur wenig ausgeprägt, so werden in den ersten drei Phasen explizit Dokumente als Ergebnisse benannt und in den späteren Phasen wird nur vereinzelt der Einsatz von Modellen (meist zur Simulation) erwähnt. Eine in anderen untersuchten Vorgehensmodellen überhaupt nicht explizit thematisierte Fachdomäne ist die Bedienung, welche Isermann gleichberechtigt neben den anderen drei üblichen Domänen in seinem Vorgehensmodell darstellt, was diesem Vorgehensmodell ein gewisses Alleinstellungsmerkmal verleiht. Ebenso ist es als einziges vor der VDI 2206 erschienen, die für alle anderen hier betrachteten Vorgehensmodelle die Vorlage bildet.

3.1.2 VDI-Richtlinie 2206: Entwicklungsmethodik für mechatronische Systeme

Die im Jahr 2004 vom Verein Deutscher Ingenieure (VDI) veröffentlichte Richtlinie [VDI04] zur Entwicklung mechatronischer Systeme wurde mit den Ziel entwickelt, die bis dato entstandene Vielfalt an Erkenntnissen in einer Methode bereitzustellen. Sie basiert dabei auf den VDI-Richtlinie 2221 („Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte“) sowie der VDI-Richtlinie 2422 („Entwicklungsmethodik für Geräte mit Steuerung durch Mikroelektronik“).

Als Makrozyklus dient dabei ein Vorgehen in V-Form, wie Abbildung 6 zeigt. Auf der Mikroebene ist ebenfalls ein dem Systems Engineering entlehnter Problemlösezyklus vorgesehen.

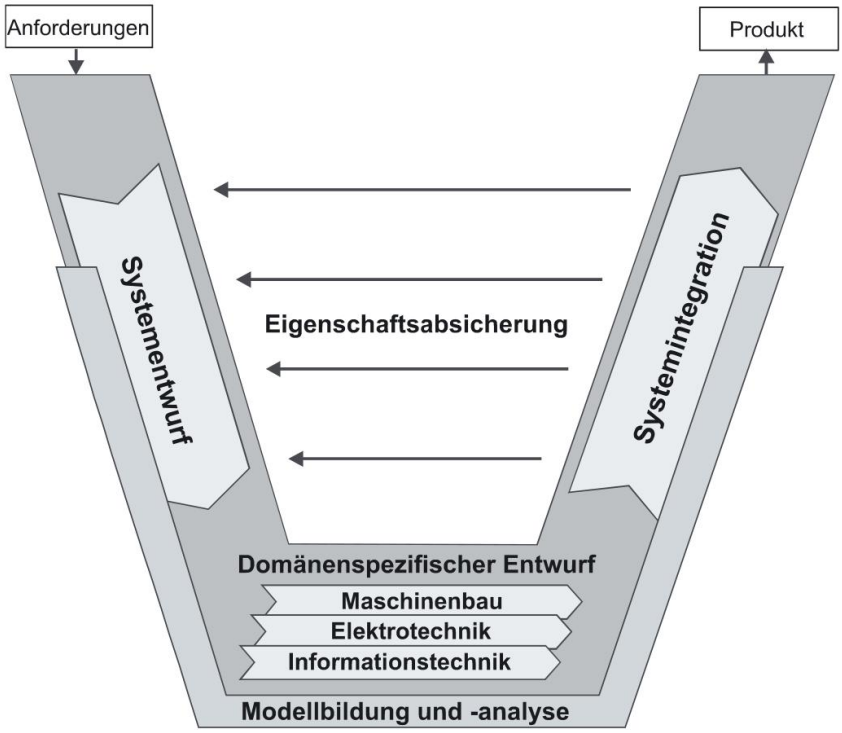

Abbildung 6: Darstellung des V-Modells als Makrozyklus der VDI 2206 [VDI04]

Das V-Modell der VDI 2206 ist in nachfolgend aufgeführte und erläuterte Schritte unterteilt:

Anforderungen: Die Anforderungen bilden den Ausgangspunkt der Entwicklung. Aber auch in späteren Phasen dienen sie als Bewertungsgrundlage des jeweiligen Fortschritts des Entwicklungsprozesses.

Systementwurf: Diese Phase beinhaltet die Entwicklung eines domänenübergreifenden Modells. Dabei wird das Gesamtsystem in die wichtigsten Teilfunktionen zerlegt, um die physikalischen und logischen Wirkungsweisen zu beschrieben:

- Erstellung einer Funktionsstruktur (nach Pahl/Beitz) für Gesamt- und Teilfunktionen (aufgliedern bis Wirkprinzipien/Lösungselemente gefunden werden können).
- Wirkstruktur: Finden und Zuweisen von Wirkprinzipien für jede Funktion (z.B. mittels Katalogen), um die Funktionsstruktur zu konkretisieren.
- Baustruktur: Quantifizierung der Wirkstruktur durch überschlägige Berechnung und grobe Dimensionierung der Geometrie; Einbettung der Lösungselemente in ein Stütz- bzw. Hüllsystem.

- Lösungskonzept: Konkretisierung zu prinzipiellen Lösungsvarianten, beispielsweise mittels Berechnung und Simulation

Domänenspezifischer Entwurf: Basierend auf dem Systementwurf werden die elektrischen, mechanischen oder steuerungstechnischen Teilsysteme im Detail entwickelt. Dazu erfolgt eine Partitionierung des Gesamtsystems entsprechend der Funktionserfüllung der involvierten Fachdomänen. Innerhalb dieser Entwurfsphase können in den Fachdomänen die domänenspezifischen Vorgehensmodelle angewandt werden.

Systemintegration: Abschließend erfolgt die Integration der parallel entwickelten Teilsysteme zu einem Gesamtsystem. Es wird dabei unterschieden zwischen den Arten der Integration:

- Integration verteilter Komponenten: Verbinden der Komponenten (z.B. Aktoren, Sensoren oder Leistungsstellglieder) mittels Energie- und Signalfluss. Zum Einsatz kommen dafür beispielsweise Kabel, Steckverbindungen, Schläuche oder Datenbus.
- Modulare Integration: Das Gesamtsystem ist in Module unterteilt, welche miteinander gekoppelt werden sollen. Jedes Modul hat eine definierte Funktionalität und kann durch einheitliche Schnittstellen mit anderen Modulen verbunden werden. Durch die Modularisierung kann eine höhere Flexibilität und Funktionsvielfalt erreicht werden. Allerdings wird dieser Vorteil mit erhöhtem Raum-, Material- und Bauaufwand erzielt (im Vergleich zu räumlich integrierten Systemen).
- Räumliche Integration: die Komponenten sind räumlich zu einer komplexen Funktionseinheit zusammengefasst, womit diese Integrationsart das Gegenteil der modularen Integration bildet. Die hier erreichbaren Ersparnisse an Raum und Material bergen im Gegenzug jedoch die Gefahr unerwünschter Wechselwirkungen wie beispielsweise Wärmestau, magnetische Streufelder oder mechanische Schwingungen.

Eigenschaftsabsicherung: Die Eigenschaftsabsicherung subsumiert die Begriffe Validierung und Verifikation. Es erfolgt ein durchgehender Abgleich der Anforderungen mit dem jeweiligen Entwicklungsfortschritt, damit die gewünschten Systemeigenschaften auch realisiert werden. Dies geschieht über Hardware-in-the-Loop (HIL), bei der reale Komponenten in die Simulationsumgebung eingebunden werden oder über

Software-in-the-Loop (SIL), bei der es sich um eine Integration der Systemmodelle in die Simulationsumgebung handelt.

Modellbildung und -analyse: Begleitung der Phasen durch flankierende Abbildung und Analyse der Systemeigenschaften mittels Modellen und rechnergestützten Simulationswerkzeugen (aufgrund komplexer Struktur und domänenübergreifendem Charakter von MTS). Eine zentrale Rolle wird dabei den Verhaltensbeschreibungen beigemessen, da sie zur Beschreibung des domänenübergreifenden funktionalen Zusammenhang verwendet werden können.

Produkt: Resultat des Makrozyklus ist das fertige Produkt in Form eines beispielsweise virtuellen oder realen Prototypen, welcher in weiteren Makrozyklen konkretisiert wird.

Modellbasierte Entwicklung und Modellintegration in der VDI 2206

Durch die Rechnerunterstützung zur Erstellung von Modellen in den unterschiedlichen Domänen wird in der Richtlinie auch die Notwendigkeit einer Integration dieser Modelle erwähnt. Dazu verweist die Richtlinie auf eine Integration mittels mathematischer Beschreibungen, da diese als allgemeingültig und flexibel für diesen Zweck angesehen wird. Für die Wiederverwendbarkeit von Modellen wird auch die Relevanz der „Durchgängigkeit“ erwähnt, um Modelle möglichst über viele Domänen und Entwicklungsphasen nutzen zu können. Ein Modellbildungsprozess ist vorgegeben sowie dazugehörige Modelle mit unterschiedlichem Abstraktionsgrad:

1. Topologisches Modell: Anordnung und Verknüpfung funktionserfüllender Elemente. Ein Element repräsentiert die kinematische Funktion, dynamische Funktion (z.B. Bewegung von Massen unter Krafteinwirkung) sowie die mechatronische Funktion (Regelung, Überwachung, Bahnplanung).
2. Physikalisches Modell: basiert auf dem topologischen Modell und beschreibt die domänenspezifischen Eigenschaften des Systems. Es ist durch der jeweiligen Domäne angepassten Größen (z.B. Längen und Massen bei Mechanik oder Widerstände und Induktivitäten bei Elektrik) definiert.
3. Mathematisches Modell: das physikalische Modell wird in eine abstrakte, systemunabhängige Darstellung überführt und integriert die unterschiedlichen domänenspezifischen Modelldarstellungen. Die physikalischen Effekte des physikalischen Modells werden mittels mathematischer Ausdrücke beschrieben.

Die Modellierungstiefe variiert dabei je nach verwendetem Modell für die jeweiligen physikalischen Effekte (Reibung, Biegung etc.)

4. Numerisches Modell: wenn das mathematische Modell mit rechnergestützten Algorithmen verarbeitet werden soll, ist eine Überführung in ein numerisches Modell notwendig. Das numerische Modell enthält konkrete Zahlenwerte, welche vorher z.B. experimentell oder analytisch ermittelt werden können.

Das V-Modell bildete die Grundlage für diverse weiter V-Modelle in der mechatronischen bzw. multidisziplinären Produktentwicklung, welche nachfolgend erläutert werden.

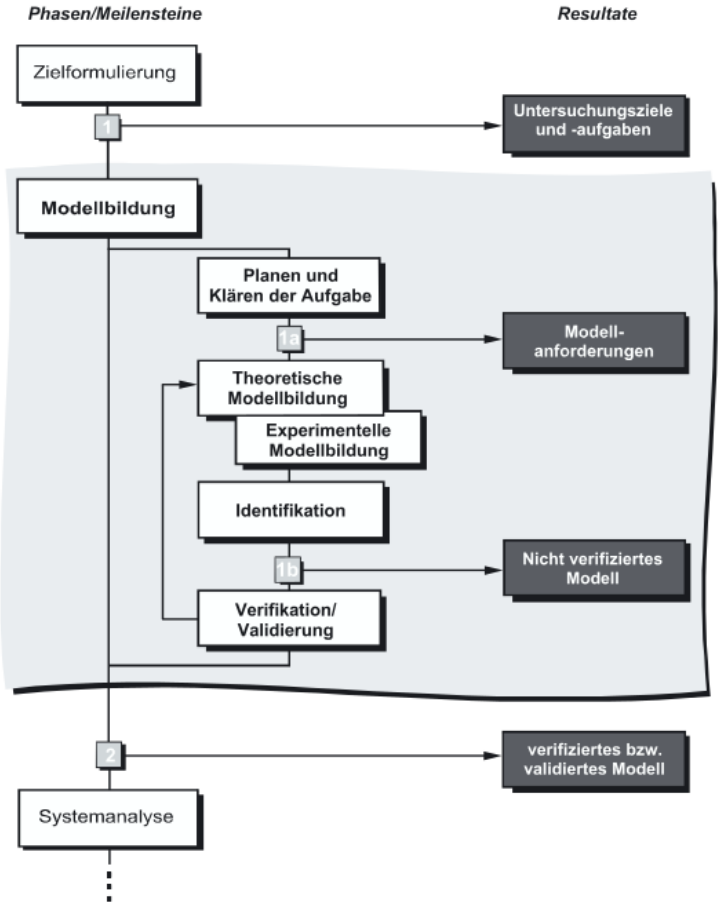

Abbildung 7: Modellbildungsprozes der VDI2206 [VDI04]

3.1.3 3-Ebenen-Vorgehensmodell nach Bender

Klaus Bender stellt in [Ben05] ein Vorgehensmodell für die Mechatronik vor, welches dem V-Modell der VDI 2206 sehr ähnlich kommt, jedoch eine Konkretisierung dessen ist und sich vor allem durch seine namensgebende Einteilung in drei Ebenen unter-

scheidet. Die drei Ebenen repräsentieren dabei den Auflösungsgrad der Systembetrachtung während der Entwicklung:

- Systemebene
- Subsystemebene
- Komponentenebene

Auf der Systemebene findet eine Gesamtbetrachtung des zu entwickelnden Systems statt, weshalb dort auch keinerlei Unterscheidung der Domänen (IT-Software, IT-Hardware und Mechanik) existiert. Diese findet erst in der Subsystemebene statt zum Zwecke der Unterteilung bzw. der Integration. Dort findet sich im Vergleich zur VDI 2206 eine Eigenheit derart, dass die Domänen Elektrik und Software teilweise zusammengefasst sind: In der Subsystemebene findet auf der linken Seite die Entwicklung von IT-Software und -Hardware zunächst gemeinsam statt bevor sie unterteilt wird, während die Mechanik in der Subsystemebene von Beginn an separiert ist. Eine vollständige Unterteilung der drei Domänen findet in der Komponentenebene statt, wo wie in der VDI 2206 die eigentliche Entwicklung des Produktes angesiedelt ist. Im rechten Teil finden die Tests statt, bei denen jede Test bzw. Integration – entsprechend dem „V-Charakter“ einem Entwurf aus der linken Seite zugeordnet ist.

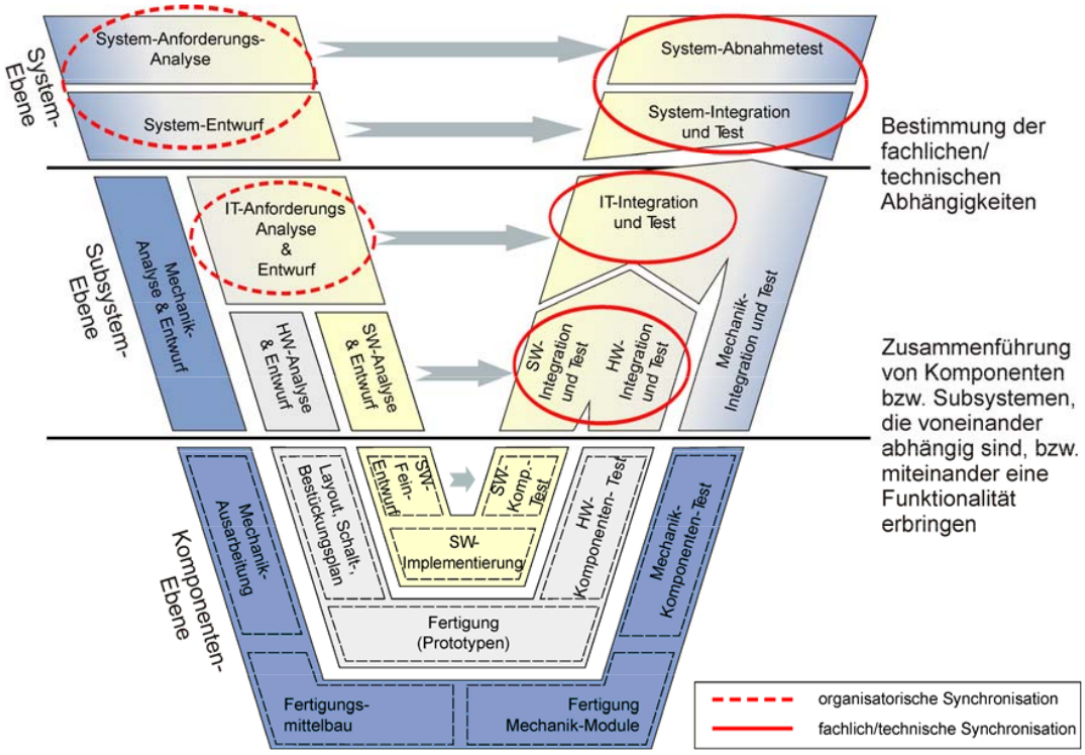

Abbildung 8: Darstellung des 3-Ebenen-Vorgehensmodells [Gau08]

In den domänenübergreifenden Phasen werden von Jürgen Gausemeier in [Gau08] Synchronisationspunkte technischer und organisatorischer Art identifiziert. In der linken Seite sind diese organisatorischer Natur, da in dieser Phase mehrheitlich Spezifikationen und Erstellung von Konzepten durchgeführt werden. In der rechten Seite hingegen sind diese technisch/fachlicher Natur, da dort die bereits entwickelten Teilsysteme integriert und getestet werden müssen.

3.1.4 W-Modell nach Anderl und Nattermann

Über eine Konkretisierung der VDI 2206 hinaus gehen Roland Nattermann und Reiner Anderl mit ihrem erstmals in [NA10] vorgestelltem und in [NA11], [NA13a] sowie [NA13b] weitergehend erläutertem W-Modell für die Entwicklung von adaptronischen Systemen. Adaptronische Systeme werden von Natterman und Anderl als die logische Fortsetzung von mechatronischen Systemen gesehen – in Richtung einer vollständigen strukturellen Integration von Komponenten, wie beispielsweise die Vereinigung von Sensor und Aktor in einem Bauteil.

Intention für die Entwicklung dieses Vorgehensmodellen waren Probleme, welche bei der Anwendung des Vorgehensmodell nach VDI 2206 identifiziert wurden, wie beispielsweise bei der Entwicklung hochgradig integrierter Systeme in der Adaptronik oder mangelnder Kommunikation zwischen den beteiligten Domänen. Des Weiteren werden im W-Modell auch Datenmanagement (DM) sowie Simulation stärker berücksichtigt. Ersteres spiegelt sich in der Nutzung von PDM-Systemen wieder aber auch in der Entwicklung eines Datenmodells mit mehreren Packages (*ActiveSystem*, *SystemModel*, *SystemSimulation*, *SystemParameters*, *SystemProperties* und *ParameterPropertyRelation*), auf die in späteren Kapiteln genauer eingegangen wird.

Analog zum Vorgehensmodell von Bender findet auch im W-Modell eine Unterteilung in drei Ebenen statt: System, Subsysteme und disziplin-spezifische Komponenten. Das größte Alleinstellungsmerkmal sind die fünf Entwicklungsphasen des W-Modells, bei dem der dritte Schritt inhaltlich (wie auch optisch in der Darstellung des W-Modells) der wichtigste und daher namensgebende Schritt ist:

1. **Systemanalyse**: Das zu entwickelnde Produkt wird (ähnlich zur VDI 2206 oder dem 3-Ebenen-Modell) analysiert und die einzelnen Anforderungen werden strukturiert. Aus der Anforderungsstruktur wird eine erste funktionale

Struktur erstellt. Zur Vorbereitung für die nächsten Phasen werden die Anforderungen und Funktionen auf die einzelnen Domänen verteilt.

2. **Analyse von Lösungen und Abhängigkeiten**: ebenfalls analog zur VDI 2206 werden in dieser Phase parallel die domänenspezifischen Teilsysteme ausgearbeitet. Darüber hinaus wird eine Betrachtung der Lösungen durchgeführt hinsichtlich kritischer Systemparameter sowie interner und externer Abhängigkeiten. Das dadurch entstehende domänenspezifische Parameter-Eigenschafts-Netzwerk (unter Nutzung der Packages *SystemParameters* und *SystemProperties* des Datenmodells) wird im nachfolgenden Schritt für die Erstellung eines multidisziplinärem Abhängigkeits-Netzwerks verwendet.
3. **Virtuelle Systemintegration**: Ziel dieses Schrittes ist ein holistisches, bidirektionales Systemmodell. Dazu wird aus dem Lösungsraum der einzelnen Domänen der Produktlösungsraum gebildet. Dieser wird analysiert und mögliche Lösungsvarianten abgeleitet. Für jede Variante können generische Systemmodelle erstellt werden, die jeweils mit der Anforderungsstruktur und der funktionalen Struktur verbunden sind, wodurch das holistische, bidirektionale Systemmodell entsteht. Die variantenspezifischen Systemmodelle werden mit entsprechenden Eigenschaften und Parametern erweitert, wofür das in Schritt 2 aufgestellte Netzwerk zum Einsatz kommt.

   Zur Bewertung der bisherigen Entwicklungsergebnisse werden Simulationen zum Systemverhalten durchgeführt. Die Simulationen finden auf der (Sub-)System-Ebene statt und sind nicht an domänenspezifische Software-Tools o.ä. gebunden, wodurch auf domänenübergreifendender Ebene eine Aussage über den Reifegrad und die Anforderungserfüllung des Produktes getroffen werden kann.
4. **Modellanalyse und detaillierte Entwicklung**: Ausgehend vom Systemmodell in Schritt 3 wird in diesem Schritt die detaillierte Entwicklung vorgenommen. Völlig unabhängig geschieht dies jedoch nicht aufgrund der permanenten Verlinkungen zu den domänenspezifischen Datensätzen des Systemmodells (virtuelle Verifikation). Dies geschieht durch inneren parametrischen Strukturen der jeweiligen Autorensysteme.
5. **Systemintegration**: Der letzte Schritt ist wieder mit der VDI 2206 identisch, jedoch wird dieser nicht als absolut notwendig erachtet. Denn durch die bereits

angesprochene permanente virtuelle Verifikation in Schritt 4 ist der Schritt 5 eher als optional zu sehen.

Diese Schritte müssen nicht immer linear durchgeführt werden, es sind auch Sprünge und Iterationen vorgesehen. So können Änderungen in den Anforderungen oder der funktionalen Struktur direkt in die detaillierte domänenspezifische Entwicklung einfließen ohne zusätzliche Iterationen zur Integration und Analyse. Dadurch soll das W-Modell besser für die Entwicklung aktiver Systeme geeignet sein als das V-Modell der VDI 2206.

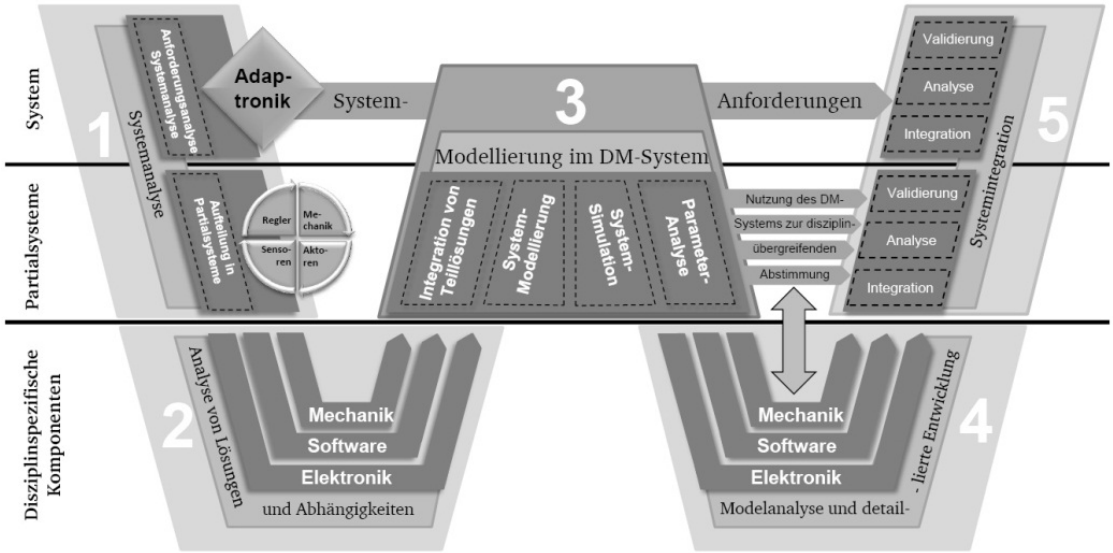

Abbildung 9: W-Modell für die Entwicklung adaptronischer Systeme [www1]

3.1.5 MBSE/RRFLP-Methode nach Kleiner und Kramer

Der RFLP-Ansatz von Sven Kleiner und Christoph Kramer aus [KK12] und [KK13] steht für Requirements (Anforderungen), Functions (Funktionen in Form der Funktionsstruktur), Logical (logische Struktur) und Physical (physische Gestalt), welche wichtige Artefakte in der Produktentwicklung darstellen. Dieser Ansatz basiert auf der Nutzung kommerzieller Software der Firma Dassault Systèmes, dem „V6 integrated information model“ sowie auf dem V-förmigen Vorgehensmodell aus dem Systems Engineering. Abbildung 10 zeigt das Vorgehensmodell mitsamt der RFLP-Sichtweise sowie der Unterteilung in die drei Hauptphasen Systemanalyse, physikalische Entwicklung und Systemintegration. In der Hauptphase der Systemanalyse findet eine theoretische Produktbeschreibung statt, in der physikalischen Entwicklung werden die Produktentwicklungsdaten erstellt.

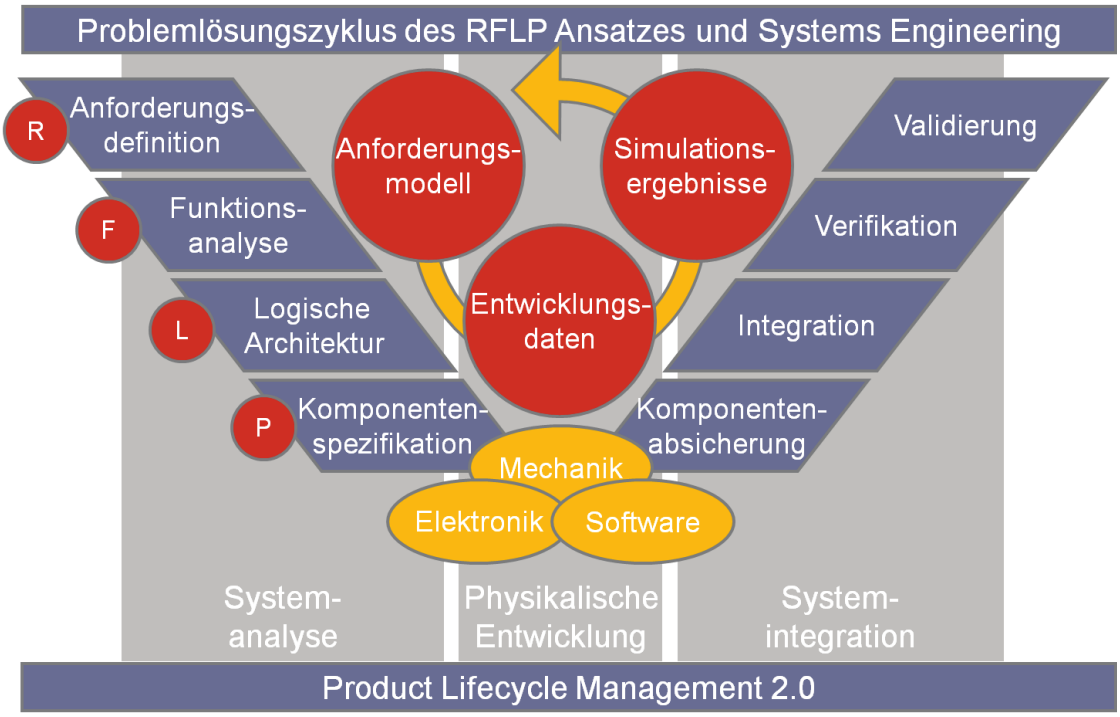

Abbildung 10: Vorgehensmodell der MBSE/RFLP-Methode [KK12]

Nach Kramer und Kleiner können bisherige IT-Systeme die SE-Methodik nur bedingt beschreiben, da sie ein föderatives oder integriertes Informationsmodell über PE-Phasen und Domänen hinweg als nicht gegeben sehen. Diese Lücke soll der RFLP-Ansatz schließen, welche im Entwicklungsprozess am linken Ast des V-Modells entlang platziert ist.

Die Methode von Kleiner und Kramer stützt sich, wie bereits erwähnt, auf kommerzielle Software-Lösungen, welche wiederum konkrete Produktmodelle besitzen, die nachfolgend aufgeführt werden.

Anforderungen (R): Erfassung und Verwaltung in Anlehnung an VDI 2206 und 2221 mittels Textdokumenten (durch Microsoft Word) und Einbindung in das PDM-System ENOVIA.

Funktionen (F): grafische Darstellung der Funktionsstruktur gemäß Pahl/Beitz. Die Funktionalitäten des Systems werden dabei als Blöcke dargestellt, welche mittels Energie-, Signal- oder Materialfluss miteinander verbunden sind.

Logische Struktur (L): Logisches Modell mit Simulationssystem Dymola erstellen, Verhaltensbeschreibung mit offener Modelliersprache Modelica. Jeder logischen Systemkomponente ist ein modelliertes Systemverhalten hinterlegt.

Physikalische Struktur (P): Erstellung von 3D-Modellen mit CAD-System „CATIA“. Die 3D-Modelle werden anhand der CATIA-Modelica-Integration in die Dymola-Simulationsumgebung integriert, womit die Verhaltensbeschreibung mit physikalischen Eigenschaften auf Grundlage des 3D-Modells erweitert wird. Die CAD-Modelle sind mit der logischen Verhaltensbeschreibung des zu entwickelnden Produktes verbunden, sodass Änderungen am CAD-Modell direkten Einfluss auf die Simulation haben.

3.1.6 SysLM-Vorgehensmodell nach Eigner et al.

Auch Martin Eigner, Torsten Gilz und Radoslav Zafirov haben ein erweitertes V-Modell für MBSE in [EGZ12] vorgestellt, welches in [ERZ14] als MVPE-Vorgehensmodell für die modellbasierte, virtuelle Produktentwicklung (MVPE) bezeichnet wird. Sie begründen ihr Konzept damit, dass aktuelle PLM- und ERP-Konzepte für die interdisziplinäre PE in den frühen Phasen nicht geeignet sind: Zum einen sind die Datenmodelle zu starr, zum Anderen ist die Trennung in Systeme für Entwicklung, Produktion und andere Phasen des Produktlebenszyklus zu monolithisch und anachronistisch, da die Erfassung der komplexen Zusammenhänge der Produkte mit ihren Wechselwirkungen in den frühen Phasen der Produktentwicklung nicht mehr gegeben ist.

Für diese Phasen sehen Eigner et al. unzureichende integrative Methoden, Prozesse sowie Software-Lösungen in administrativer (PLM, ERP) wie anwendungsorientierter Hinsicht. Es fehlt ein durchgängiges Modellierungskonzept von Anforderungen, Funktionen sowie logischer und physischer Beschreibung des zu entwickelnden Produktes. Für eine derartige Integration greifen sie daher ebenfalls auf den Ansatz des Systems Engineering zurück, um damit das weit verbreitete V-Modell in der Mechatronik zu einem Vorgehensmodell für das MBSE zu erweitern, welches von PLM unterstützt wird. In [Eig+14] wurde dies zum Vorgehensmodell für das Systems Lifecycle Management (SysLM) erweitert. SysLM wird von Eigner et al. als allgemeine Informationsverwaltungslösung beschrieben, die die PLM um die frühen Entwicklungsphasen und alle Diszplinen erweitert. Dazu werden die Autorensysteme (beispielsweise CAD-Systeme) via SysML-Datenbackbone zur Integration miteinander verbunden. Dargestellt ist dieses Konzept in Abbildung 11.

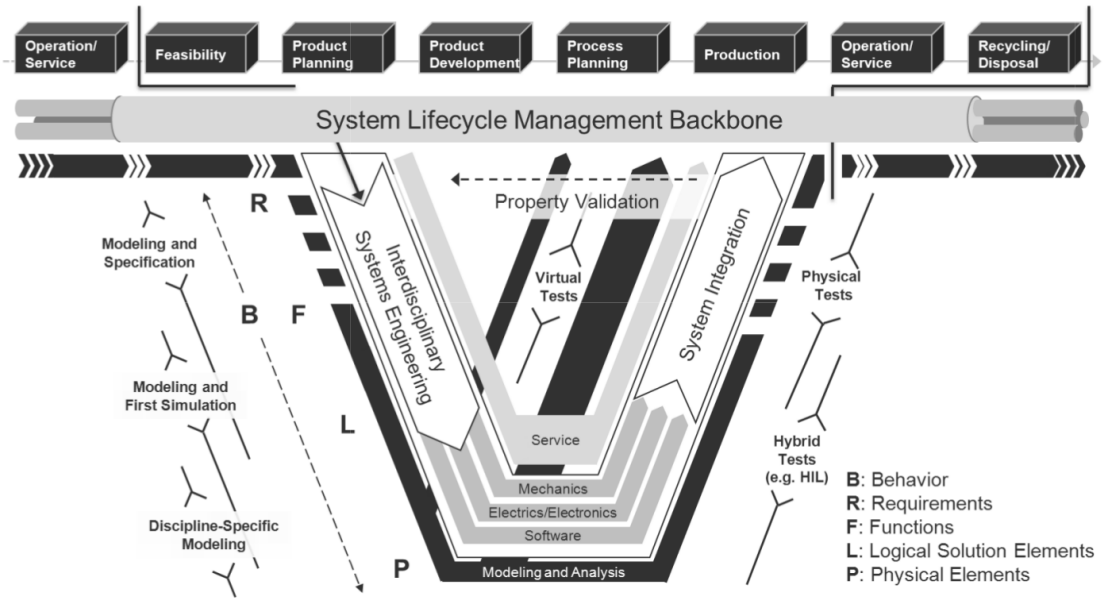

Abbildung 11: erweitertes V-Modell für SysLM/MBSE [Eig+14]

Da laut den Autoren dieses Vorgehensmodells ist die modellgestützte Entwicklung unzureichend berücksichtigt ist, wurde hier – ähnlich der Methode von Kleiner und Kramer – der RFLP-Ansatz gewählt, allerdings um das Verhalten (engl. behaviour, kurz B) erweitert, für das eine Modellierung sowie Analyse vorgesehen ist [Eig+14].

Zu Beginn werden die Anforderungen analysiert, sodass konsistente, technische Anforderungsmodelle entstehen, welche mit „A“ gekennzeichnet sind. Ist dies geschehen, werden darauf basierend die Funktionen (F) beschrieben. Zunächst kann dies in allgemeiner und grober Form zur Beschreibung von Funktion und Verhalten geschehen, und wird dann sukzessive in die Teilfunktionen verfeinert. Das Ergebnis ist eine erste Spezifikation des zu entwickelnden Systems in einer neutralen und von den Fachdisziplinen unabhängigen Form. Das Beschreiben des Verhaltens und der funktionalen Elemente durch die Logischen Lösungselemente (L) bildet den letzten Teil der disziplinübergreifenden, lösungsneutralen Entwicklung. Die disziplinspezifische Entwicklung beispielsweise von Hardware-Komponenten oder auch Software-Code beginnt mit der Entwicklung der physikalischen Elemente (P). Laut Eigner et al. setzen hier erst viele CAx-Prozesse in der virtuellen Produktentwicklung erst an [EGZ12].

Für die verschiedenen Phasen definieren sie unterschiedliche Arten von Modellen:

1. Modellbildung und Spezifikation: qualitative Modelle zur reinen Beschreibung von Anforderungen, Funktions- oder Systemstrukturen. Beispielhafte Sprachen sind dafür SysML oder ModelicaML.

2. Modellbildung und Simulation: meist quantitative, simulierbare und disziplinübergreifende Modelle, wie z.B. multiphysikalische Simulationsmodelle. Beispielhafte Simulationswerkzeuge: Matlab/Simulink, Simscape, Dymola oder Modelica.
3. Disziplinspezifische Modellbildung und Simulation: Geometrie- oder CAE-Modelle, in der Regel genauere quantitative Modelle. Beispielhafte Werkzeuge: disziplinspezifische Software für CAD, CAM oder andere numerische Simulationen (FEM, CFD etc.).

Zwischen den verschiedenen Modellelementen findet eine Rückverfolgbarkeit (sog. „traceability“) statt, welche durch semantische Verlinkungen gewährleistet. Genauer wird dies in Kapitel 4 beschrieben.

3.1.7 Erweitertes V-Modell mit SE-Aspekten

Den aktuellsten Ansatz in der Erweiterung des V-Modells zeigen Iris Graessler und Julian Hentze in [GH15] unter der Nutzung von Aspekten des Systems Engineering (SE). Das SE ist unter anderem auf die frühen Phasen der Produktentwicklung gerichtet, welche vor allem im Zusammenhang mit der Anforderungsanalyse stehen. Genau diese Tätigkeiten sind nach Graessler und Hentze in der VDI 2206 unzureichend berücksichtigt, da die Anforderungen dort als bereits bekannt und nicht mehr veränderlich gelten. Aufgrund der hohen Bedeutung dieser frühen Phasen für die gesamte Produktentwicklung (auch als sog. „Frontloading“ bezeichnet) wurde das Vorgehensmodell mit folgenden neuen Phasen zu Beginn erweitert: Projektplanung, Definition der Stakeholder, Anforderungserhebung sowie die Anforderungsanalyse. Um eine Betrachtung des gesamten Produktlebenszyklus umfassender zu berücksichtigen, befinden sich im Anschluss an die Entwicklung in exemplarischer Form die Phasen des Einsatzes, der Wartung und der Entsorgung des zu entwickelnden Systems.

Die Abbildung 12 zeigt das V-Modell aus der VDI 2206 mit den Erweiterungen von Graessler und Hentze. Dem Vorgehensmodell übergeordnet sind dabei die Projektphasen, welche sich entsprechend Abbildung 1 in der Makroebene sich befinden dürften. Auf der linken Seite befinden sich die durch das Frontloading bedingten bereits erwähnten zusätzlichen Entwicklungsphasen, welche entsprechend dem Prinzip des V-Modells (bis auf die Projektplanung) vertikal angeordnet sind, weil diese mittels Eigenschaftsabsicherung mit der rechten Seite des V-Modells in Zusammenhang stehen.

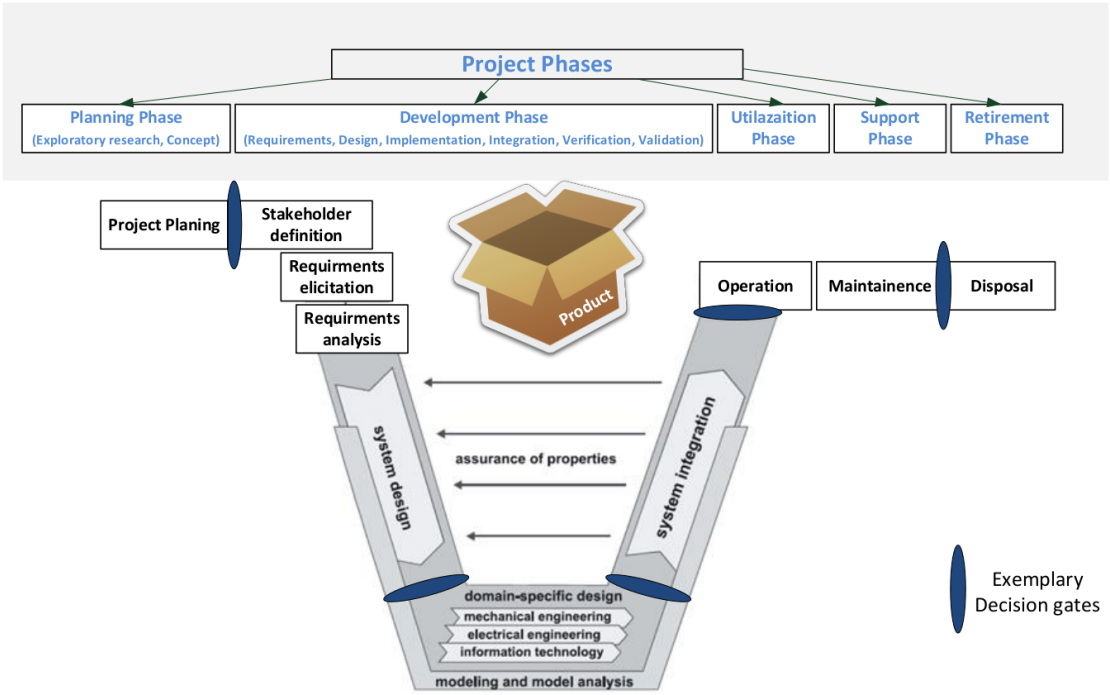

Abbildung 12: Das V-Modell der VDI 2206 erweitert mit SE-Aspekten [GH15]

Weitere aus dem SE übernommene Aspekte sind das Systemdenken sowie Entscheidungstore (sog. „decision gates“), welche passiert werden müssen. Ersteres bedeutet eine gesamte Sicht auf das System, seiner Grenzen sowie die Schnittstellen mit der Umgebung und den Subsystemen. Zur Unterstützung dieses Systemdenkens schlagen Graessler und Hentze vor, das zu entwickelnde System in der Mitte des V-Modells abzubilden, wie in in diesem durch die in der Mitte der Abbildung 12 dargestellte Box angedeutet ist. Die Entscheidungspunkte sind zwar auch in der VDI 2206 vorgesehen, aber im V-Modell nicht explizit gekennzeichnet, weshalb diese von Graessler und Hentze im erweiterten V-Modell nachträglich hinzugefügt wurden.

3.2 Analysekriterien

Die Analyse der Vorgehensmodelle hat das Ziel, einen Überblick darüber zu bieten, wie sehr die etablierten V-Modelle den Ansprüchen der heutigen Zeit genügen sowie das Herausstellen der typischen Entwicklungshasen. Die Analysekriterien sowie die eigentliche Analyse der Vorgehensmodelle sind daher teilweise angelehnt an Chen Zheng, Matthieu Bricogne, Julien Le Duigou und Benoît Eynard aus [ZBDE14] sowie an Fabio Dohr aus [Doh14]. Zheng et al. haben ihre Analyse im Hinblick auf Concurrent Engineering sowie Kollaboration durchgeführt (Kriterium 9 bis 11), was für für

diese Arbeit ebenfalls als eine geeignete Untersuchung erachtet wird. Dohr (Kriterium 3 bis 8) hat im Rahmen seiner Dissertation unterschiedliche Vorgehensmodelle hinsichtlich des Einsatzes von Simulation in der mechatronischen Produktentwicklung untersucht. Zwar ist Simulation bzw. simulationsgestützte Produktentwicklung nicht direkt ein essenzieller Bestandteil dieser Arbeit, jedoch kommen in Simulationen ebenfalls Produktmodelle zum Einsatz, welche in dieser Arbeit durchaus auch Gegenstand der Untersuchung sind. Darüber hinaus sind diese Kriterien für die Betrachtung generell geeignet, zu veranschaulichen, inwiefern die Vorgehensmodelle den aktuellen Realitäten wie beispielsweise dem zunehmenden Einsatz von Rechnern und Simulation oder der verteilten Zusammenarbeit unter Nutzung des Internets gerecht werden. Hinzu kommen weitere Kriterien, sodass die Analysekriterien insgesamt wie folgt lauten:

1. Anforderungen
2. Explizite Produktmodelle
3. Eigenschaftsanalyse
4. Umgang mit Analyseergebnissen
5. Simulation
6. Rechnerunterstützung
7. Iterationen
8. Umgang mit Analyseergebnissen
9. Concurrent Engineering
10. Makro-Level-Kollaboration, Interdisziplinarität
11. Mikro-Level-Kollaboration

Die ersten beiden Kriterien stammen weder von Zheng et al. oder Dohr, sondern wurden zusätzlich aufgestellt für die Betrachtung. In den nachfolgenden Textabschnitten erfolgt eine Erläuterung der Kriterien.

Anforderungen: Vorhandensein von Anforderungsanalyse, Requirement Engineering, Spezifikationstechnik und ähnlichen Tätigkeiten um möglichst gezielt die Anforderungen oder andere implizit wie explizit geforderte Eigenschaften des zu entwickelnden Produktes zu ermitteln und zu dokumentieren.

Explizite Produktmodelle: Vorhandensein von explizit benannten Produktmodellen, die den Entwicklungsprozess unterstützen.

Eigenschaftsanalyse: Prüfen, ob das realisierte (Teil-)System die gestellten Anforderungen erfüllt, z.B. durch Berechnung, Schätzung, Vergleiche, Experimente oder Simulation.

Simulation: Einsatz von Simulationssystemen, welcher ebenfalls wichtiger Bestandteil einer möglichst vollständig virtuellen und rechnergestützten Produktentwicklung ist.

Rechnerunterstützung: Integration der Rechnerunterstützung durch beispielsweise CAD, CAE oder PDM-Systeme. Dieses Kriterium besitzt inhaltliche Schnittmengen mit dem Kriterium für Simulationen, da diese mit Rechnerunterstützung durchgeführt werden, jedoch ist dieses Kriterium weiter gefasst.

Iterationen: Berücksichtigung von Iterationen sowie Unterstützung dahingehend, wann eine Iteration nötig ist, wie sie gestaltet ist und was ihre Auswirkungen sind.

Umgang mit Analyseergebnissen: Aus Analysen sollen Informationen über das zu entwickelnde Produkt gewonnen werden. Für die Vorgehensmodelle bedeutet dies beispielsweise, ob eine Iteration durchgeführt werden muss oder ob regulär im Vorgehensmodell fortgefahren werden kann.

Concurrent Engineering: zeitgleiches Arbeiten mehrerer in der Produktentwicklung involvierter Personen. Zwar sind prinzipiell ohnehin schon Vorgehensmodelle ausgewählt, die dies berücksichtigen, dennoch wird betrachtet, in welchem Umfang dieses Prinzip berücksichtigt oder gar unterstützt wird.

Makro-Level-Kollaboration, Interdisziplinarität: Berücksichtigung der Zusammenarbeit unterschiedlicher Fachdomänen. Dies schließt einerseits die Zusammenführung der Subsysteme einzelner Domänen ein und andererseits die Schnittstellen zwischen den Systemen.

Mikro-Level-Kollaboration: Berücksichtigung der Zusammenarbeit einzelner Personen, z.B. Austausch und Teilen von Daten. In der Regel geschieht dies auf informellem Wege, beispielsweise mittels E-Mail, Telefon oder persönlichem Gespräch.

3.3 Analyse der Vorgehensmodelle

Das Ergebnis der Analyse zeigt Tabelle 3. In der ersten Zeile sind die Kriterien und in der ersten Spalte sind die Vorgehensmodelle aufgelistet. Ab der zweiten Zeile bzw. zweiten Spalte erfolgt die dreistufige Bewertung. Die grau unterlegten Felder sind Be-

wertungen, die den bereits erwähnten Quellen von Dohr sowie Zheng et al. entnommen sind.

	Anforderungsanalyse	Produktmodelle u. -repräsentationen	Eigenschaftsanalyse	Umgang mit Analyseergebnissen	Simulation	Rechnerunterstützung	Iterationen	Interdisziplinarität	Concurrent Engineering	Makro-Level Kollaboration	Mikro-Level Kollaboration
VDI 2206	-	o	o	-	o	+	o	+	+	o	-
3-Ebenen-Modell (Bender)	-	-	o	-	-	o	-	o	+	o	-
W-Modell (Anderl, Nattermann)	-	+	+	o	+	+	o	+	+	+	+
Erweitertes V-Modell mit SE-Aspekten (Graessler, Hentze)	+	-	o	o	o	+	o	+	+	o	-
MBSE/RFLP-Methode (Kleiner, Kramer)	o	+	+	-	+	+	-	+	+	+	o
SysLM-Vorgehensmodell (Eigner et al.)	+	+	+	-	o	+	o	+	+	+	+
Vorgehen nach Isermann	-	-	o	-	o	o	-	+	+	+	-

Tabelle 3: Vergleich der betrachteten Vorgehensmodelle, teilweise (grau unterlegte Felder) nach [Doh14] und [ZBDE14] (+: stark ausgeprägt; o: mittelmäßig ausgeprägt; -: schwach ausgeprägt)

Der Vergleich zeigt die Unterschiede in Umfang und Detailtiefe zwischen den Vorgehensmodellen, die teilweise auch dem Alter der Vorgehensmodelle geschuldet sein dürfte. Während beispielsweise frühere Vorgehensmodelle, wie die VDI 2206 oder das 3-Ebenen-Modell eher als allgemeine Rahmenwerke fungieren, werden mit dem späteren W-Modell oder dem erweiterten V-Modell für MBSE deutlich weiterführende Schritte vorgenommen. Dies führt bis zu konkreter Unterstützung durch Modelliersprachen und Software-Werkzeuge, wie es unter anderem im Vorgehensmodell von Kleiner und Kramer der Fall ist. Dieser höhere Grad an Fortschrittlichkeit und Ausgereiftheit dürfte mit großer Wahrscheinlichkeit in der Tatsache begründet liegen, dass bei der Entwicklung dieser Vorgehensmodelle auf vorangehende Vorgehensmodelle sowie damit einher gegangene Erfahrungen in der Anwendung selbiger aufgebaut werden konnten. Weiterhin haben sich in der Zeit die CAx-Systeme und andere Software-Werkzeuge sowie die Rechnerhardware weiter entwickelt.

Es wurden bei den untersuchten Vorgehensmodellen Phasen identifiziert, die sie nahezu alle gemein haben, die jedoch unterschiedlich stark ausgeprägt sind. Die in den sieben V-Modellen identifizierten sieben „allgemeinen Phasen des V-Modells“ lauten wie folgt:

1. Anforderungsphase
2. Funktionale Phase
3. Logische Phase, Verhalten
4. Physische bzw. domänenspezifische Phase
5. Integration der Teilsysteme
6. Integration des Gesamtsystems
7. Verifikation, Validierung, Test

Als aggregiertes V-Modell sind diese Phasen in Abbildung 13 dargestellt. Diese Phasen, sollen in der Produktentwicklung mit Produktmodellen und anderen Abbildungen respektive Repräsentationen des jeweiligen Produktes prinzipiell adressiert und unterstützt werden. Zu diesem Zweck findet im nachfolgenden Kapitel die Auswahl, die Einordnung sowie eine Untersuchung von mehreren Produktmodellen statt.

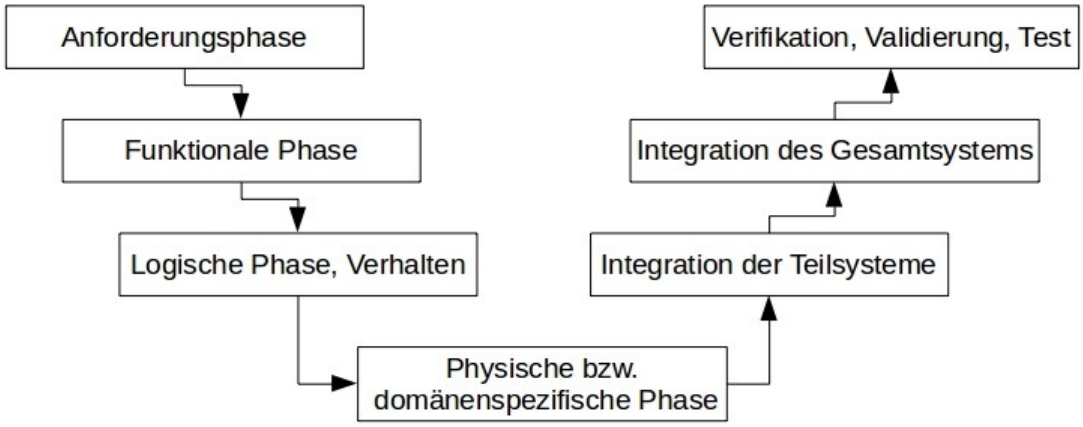

Abbildung 13: Aus den untersuchten Vorgehensmodellen aggregiertes V-Modell

4 Produktmodelle

Dieses Kapitel stellt die untersuchten Produktmodelle sowie anderweitige Repräsentationen und Abbildungen realer oder zu realisierender technischer Produkte bzw. Systeme vor. Im Folgenden werden die Produktmodelle und -repräsentationen zusammenfassend nur noch als Produktmodelle bezeichnet.

Die hier gebotene Auswahl hat nicht den Anspruch, die komplette Bandbreite in der Produktentwicklung nötiger Produktmodelle zu zeigen, sondern soll zeigen, welche Modelle als zentrale Modelle dienen können, in Abhängigkeit derer weitere Modelle eingesetzt werden können bzw. davon abgeleitet werden können.

4.1 Eingrenzung und Unterteilung der Produktmodelluntersuchung

Bei der Untersuchung der Produktmodelle wurden aufgrund prinzipieller Unterschiede zwischen den Produktmodellen vier Kategorien aufgestellt, in die sich alle identifizierten Produktmodelle einordnen ließen, um somit einen besseren Überblick und Verständnis über die Produktmodelle zu bieten:

A. **Ingenieurwissenschaftliche Modellsichten auf ein System:** Modelle, die der ingenieurwissenschaftliche Denk- und Arbeitsweise in der Produktentwicklung dienen bzw. entspringen. Diese Modelle stammen teilweise aus einer Zeit, in der Rechnerunterstützung in der Produktentwicklung lediglich in geringem oder praktisch gar keinem Maße vertreten war. Daran wird ersichtlich, dass der Einsatz von Modellen nichts zwangsläufig durch das Aufkommen von Rechnern in der Produktentwicklung relevant war. Beispiele sind dafür zwei- oder dreidimensionale Geometriemodelle, mathematische Modelle oder Prinzipskizzen (Funktionsstruktur, Schaltpläne). Die hierzu untersuchten Modelle sind:
   1. Funktionsstruktur: grafische Darstellung gemäß der Konstruktionslehre von Pahl/Beitz, einem Standardwerk des Maschinenbaus.
   2. Bondgraphen von Henry Paynter: eine domänenneutrale Darstellung für technische Systeme, entwickelt am Massachusetts Institute of Technology
   3. Systems Modeling Language (SysML) der (OMG), eine vielseitige grafische Notation, die speziell für das Systems Engineering entwickelt wurde.

B. **Dateiformate und Datenmodelle:** teilweise auf Basis wie der in Kategorie A angesprochenen Denk- und Sichtweisen ist in den vergangenen Jahrzehnten

eine Vielzahl ingenieurwissenschaftliche Anwendungsprogramme entstanden. Jedes dieser Software-Werkzeuge nutzt für das Abbilden des Produktes ein Datenmodell, welches in der Regel in einem nativen Dateiformat gespeichert wird. Zum Export und Import können auch (mehr oder weniger standardisierte) Dateiformate zum Einsatz kommen oder auch Programmierschnittstellen (meist als Application Programming Interface, kurz API bezeichnet). Generell zählen zu dieser Kategorie Modelle, welche erst durch den Einsatz von Rechnern in der Produktentwicklung entstanden sind bzw. durch den Rechnereinsatz erst sinnvoll nutzbar wurden, wie z.B. Dateiformate zum Speichern und Austausch. Hierfür werden betrachtet:

1. Auswählte Anwendungsprotokolle (AP) des „Standard for the Exchange von Product Data“ (STEP): Der umfangreiche, seit den 1980er Jahren fortwährend entwickelte Standard für Produktdatenabbildung und -austausch.
2. Core Product Model 2: ein generisches Datenmodell, das vom US-amerikanischen National Institute of Standards and Technology (NIST) standardisiert ist.
3. Modelica von der Modelica Association: ein Simulationswerkzeug für vielfältige physikalische Systeme. Es ist zwar weniger etabliert als das kommerzielle Konkurrenzprodukt „Matlab“ der Firma Mathworks, besitzt aber ein großes Potenzial u.a. durch seinen freien Charakter sowie seine Nutzung und Förderung durch profitorientierte wie gemeinnützige Organisationen.

C. **Integrationskonzepte und Datenmodelle:** bereits entwickelte oder gar implementierte Konzepte zur Datenintegration gehören vorwiegend zu dieser Kategorie. Meist geschieht dies unter Nutzung einer sogenannten Middleware, welche als Vermittlungsplattform zwischen den verschiedenen Anwendungen fungiert. Diese Middleware kann beispielsweise eine Datenbank, die über verschiedene Schnittstellen mit den Anwendungsprogrammen mit dessen Modellen (siehe Kategorie B) zwecks Datenaustausch gekoppelt ist. Derartige Konzepte gehen über die Produktmodelle hinaus und versuchen, mehrere davon miteinander in Abhängigkeit zu setzen, was auch mit dieser Arbeit angestrebt werden soll. Daher bietet eine Betrachtung der folgenden Konzepte auch einen Blick auf den Stand der Forschung den folgenden Beispielen:

1. Metadatenmodell zum W-Vorgehensmodell von Anderl und Nattermann
2. Multidisziplinäre Modellierung und Simulation für mechatronischen Entwurf nach Lefèvre et al.
3. SysLM inkl. Datenmodell für das MBSE nach Eigner et al.

D. **Ontologien und andere semantische Ansätze:** über die Verwaltung und Integration von Daten und Produktmodellen aus Kategorie C hinaus gehen Ansätze, die Verarbeitung des Inhalts und der Bedeutung der Daten eingehen. Beispielhaft für derartige Ansätze sei hierfür auf [Tu06], [DBPH07] oder [SB11] verwiesen.

Schwerpunkt der Betrachtung in dieser Arbeit bilden Produktmodelle und Konzepte der Kategorien A und B, welche eine multidisziplinäre Arbeit ermöglichen und unterstützen. Ferner werden auch Konzepte aus der Kategorie C betrachtet, auch wenn diese sich bisweilen aus Lösungen der Kategorie A zusammensetzen. Eine Betrachtung dieser bietet jedoch die Möglichkeit, den Stand der Forschung auf dem Gebiet der Integration von Produktdaten und -modellen näher zu betrachten. Dies unterstützt die Konzeptfindung in Kapitel 5, wenn ähnliche Konzepte bereits untersucht worden sind.

4.2 Produktmodelle

In diesem Unterkapitel werden die für die interdisziplinäre Produktentwicklung als relevant identifizierten Produktmodelle erläutert. Der Schwerpunkt der Erläuterung liegt auf der Beschreibung der Produktmodelle aus einer anwendungsorientierten Sicht. Wichtig sind dabei die adressierten Fachdomänen, die Phasen der Produktentwicklung sowie sonstige Aktivitäten, mit welchen die Zusammenarbeit verschiedener Fachdisziplinen ermöglicht und unterstützt werden können.

4.2.1 Funktionsstruktur nach Pahl/Beitz

Nach Pahl/Beitz in [PBFG13] existieren mindestens zwei Sichtweisen auf eine Produkt: die funktionale sowie die physische Sicht, welche zusammen die die Produktarchitektur bilden. Erstere stellt dar, welche Funktionen das Produkt bieten soll. Letztere stellt dar, wie die geforderten Funktionen in technisch-physischer Hinsicht erfüllt werden sollen. Für die funktionale Beschreibung gibt es gemäß Pahl/Beitz in [PBFG13] die Funktionsstruktur, die eine lösungsneutrale und vergleichsweise leicht verständliche Darstellungsform dafür bietet, was ein Produkt tut bzw. tun soll. Dazu wird zwi-

schen den Hauptfunktionen und Nebenfunktionen unterschieden, welche durch die drei Flussarten Energie, Stoff und Signal miteinander in Verbindung stehen (siehe Abbildung 14).

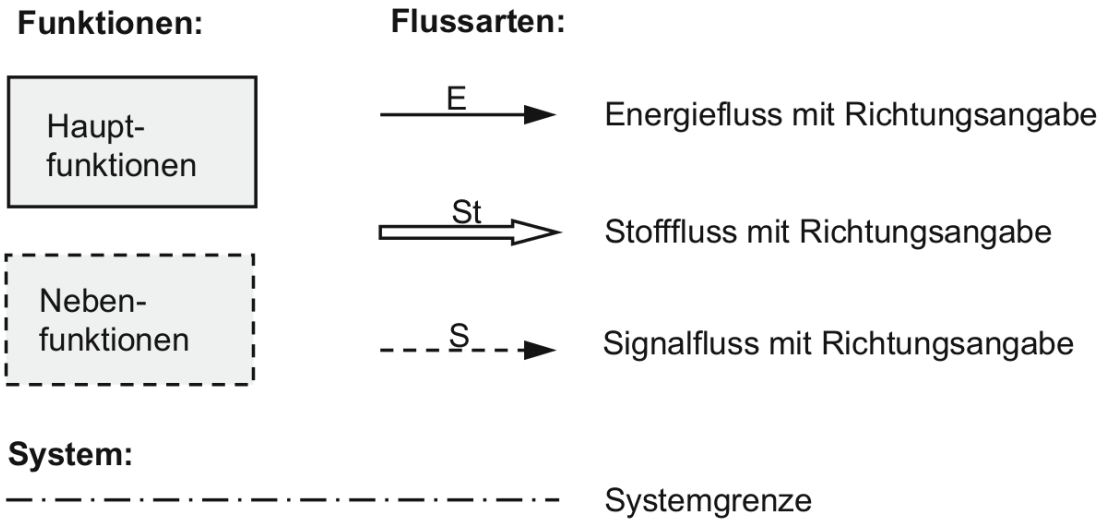

Abbildung 14: Symbole der Funktionsstruktur [PBFG13]

Je nach Komplexität und Betrachtungsmaßstab lässt sich die Funktionsstruktur darüber hinaus auch unterteilen. Die Gesamtfunktion bzw. der Zweck des Produktes wird unterteilt in mehrere Teilfunktionen, wie Abbildung 15 zeigt.

Vorteilhaft an dieser Darstellungsform ist, dass sie eine lösungsneutrale Darstellung der geforderten respektive beabsichtigten Funktionen bietet und vor allem vergleichsweise leicht zu erstellen und lesen ist. Die Funktionsstruktur bildet gemäß der in [PBFG13] gezeigten Methodik die Grundlage für das Finden von lösungsneutralen physikalischen Effekten (z.B. Reibung, Hebelwirkung) sowie darauf basierender Wirkprinzipien. Jedoch werden auch in anderen Methodiken Funktionsstrukturen erstellt, wie in der bereits geschilderten RFLP-Methode von Kleiner und Kramer. Allerdings ist sie im Vergleich zu strenger und umfassender formalisierten Darstellung wie z.B. der SysML mit ihren unterschiedlichen Diagrammtypen deutlich weniger umfangreich und aussagekräftig. Dennoch bietet sie eine gute Möglichkeit, schnell auch für fachlich Außenstehende die Funktionen festzuhalten und verständlich zu machen, welche anschließend bei Bedarf in SysML oder andere detailliertere Darstellungsformen überführt werden können.

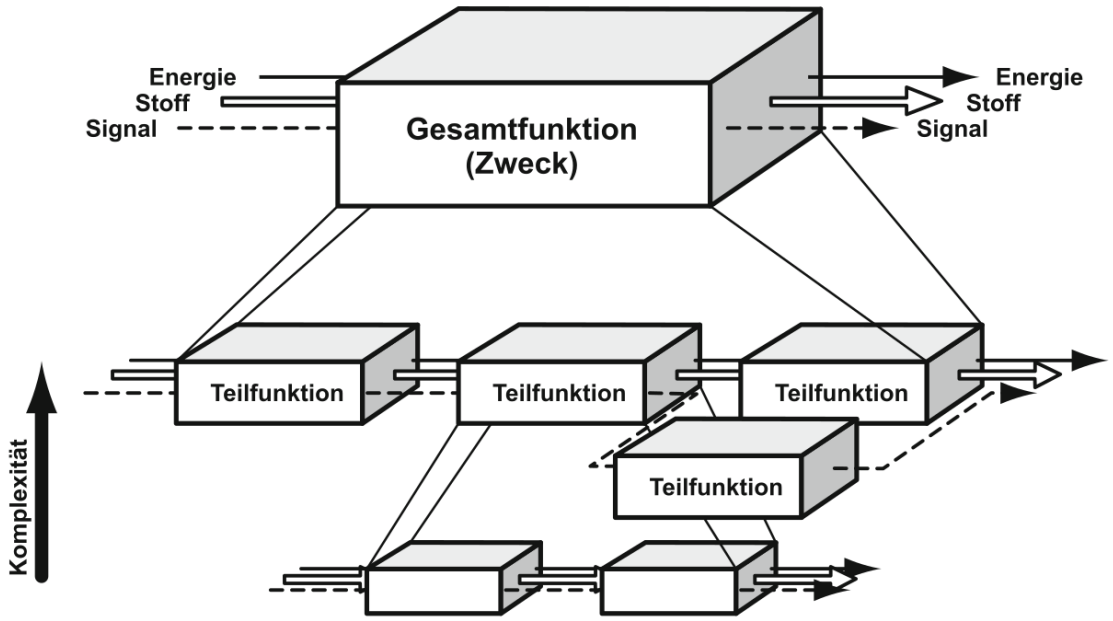

Abbildung 15: Unterteilung der Gesamtfunktion in Teilfunktionen [PBFG13]

4.2.2 Bondgraphen

Bondgraphen sind eine grafische Beschreibungsmethode mit dem Ziel, eine einheitliche Beschreibungsform sämtlicher physikalischer Systeme zu ermöglichen, welche hier basierend auf der Beschreibung von Werner Roddeck in [Ro12] erläutert werden. Entwickelt wurde diese Methode 1959 von Henry Paynter am Massachusetts Institute of Technology. Die Intention für die Entwicklung der Bondgraphen war der zentrale Gedanke, dass die grundlegende Eigenschaft bei dynamischen Systemen der Energiefluss (Leistung) zwischen den Komponenten eines Systems ist. Die Leistung setzt sich immer in den jeweiligen Fachdomänen aus zwei Größen zusammen, bei denen eine in der Bondgraphen-Darstellung als „Effort“ (Aufwand, Anstrengung) bezeichnet wird und die andere als „Flow“ (Fluss). Die nachstehende Tabelle 4 zeigt die jeweiligen Größen aus den typischen Fachgebieten

Fachgebiet	**Effort**	**Flow**
Elektrotechnik	Potential (Spannung)	(elektrischer) Strom
Mechanik	Kraft	Geschwindigkeit
Fluidtechnik	Druck	Volumenstrom
Chemie	Chemisches Potential	Molarer Fluss
Thermodynamik	Temperatur	Entropiefluss

Tabelle 4: Übersicht über die Effort- und Flow-Größen von Bondgraphen [Ro12]

Im Gegensatz zur Funktionsstruktur gibt es auch nur zwei Arten von Pfeilen zwischen den Komponenten: Energie und Information. Von den Komponenten hingegen existieren feste Kategorien, je nach der Funktion, die sie ausüben:

- Quelle (Source) für Effort und Flow
- Speicher für Effort Flow
- Senke bzw. Verlust von Leistung
- Umwandlung von Leistung
- Verzweigung von Effort und Flow

Die nachfolgende Auflistung beschreibt stichpunktartig diese oben erwähnten Komponenten bzw. Elementen der Bondgraphen.

1-Port-Element:

- R-Element: elektrischer Widerstand, mechanischer Dämpfer (bei Ausblendung der produzierten Abwärme)
- C-Element (C für Capacity oder Compliance): verlustfreie Speicherung von Energie, z.B. elektrischer Kondensator oder elastische Feder
- I-Element: z.B. elektrische Induktivität oder mechanische Masse
- Quellen:
  - Effortquelle: Der ist Effort vorgegeben und der Flow ergibt sich aus umgebenen System (z.B. elektrische Spannungsquelle, mechanische Masse mit Schwerkraft)
  - Flowquelle: Der Flow vorgegeben und Effort ergibt sich aus dem System (z.B. elektrische Konstantstromquelle oder mechanische Zahnradpumpe)

2-Port-Element:

- Transformer: Umwandlung der Leistung gemäß eines Verhältnisses aus Eingangs- und Ausgangsflow bzw. Eingangs- und Ausangseffort (elektrischer Trafo, mechanisches Getriebe)
- Gyrator: Leistungsumwandlung als Verhältnis zwischen
  - Eingangseffort und Ausgangsflow oder
  - Eingangsflow und Ausgangseffort
  - Beispiele: Kreisel, Gleichstrommotor

MultiPorts

- Elemente mit drei oder mehr Ports, daher auch als Junction bezeichnet

- kein Speichern von Flow oder Effort
- prinzipiell wird zwischen zwei Arten unterschieden:
  - Flow-Junction (auch bekannt als Parallel-Junction oder 0-Junction)
    - mindestens je ein Input- und Output-Bond für die Leistung nötig
    - algebraische Summe der Flows muss Null sein
  - Effort-Junction: wie Flow-Junction, nur mit vertauschten Rollen von Effort und Flow

Abbildung 16 zeigt beispielhaft die Erstellung eines Bondgraphen ausgehend von einem elektrischen Schaltplan eines Serienschwingkreises. Die griechischen Buchstaben stellen dir Ports zwar, in denen der Flow hinein bzw. hinaus geht. U ist die Spannungsquelle, R ist der elektrische Widerstand, L die Induktivität und C die Kapazität. In a) werden zunächst die Punkte zwischen mittels griechischer Buchstaben angetragen, in b) die jeweiligen elektrischen Komponenten durch ihre Pendants der Bondgraphen-Darstellung und schlussendlich findet in c) eine Vereinfachung bzw. Zusammenfassung statt, indem die Null-Ports entfernt werden.

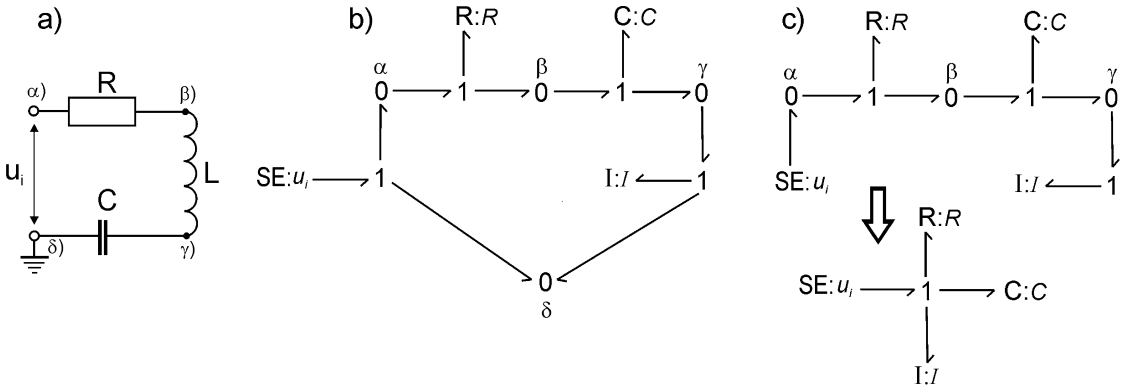

Abbildung 16: Erstellung eines Bondgraphen aus einem Serienschwingkreis [Ro12]

Bondgraphen bieten eine gute Grundlage zur topologischen und mathematischen Beschreibung von technischen Systemen losgelöst von ihrer Domäne. Allerdings Beschreibt Roddeck nur die Umwandlung von domänenspezifischen Modellen in die domänenübergreifende Bondgraph-Darstellung. In früheren Phasen der Produktentwicklung kann oder soll die physikalische Umsetzung allerdings noch offen bzw. lösungsneutral sein. Beispielsweise ob ein technisches System oder die (Funktion eines technischen Systems elektrisch oder thermisch) realisiert werden soll, weshalb sie für die

frühen Phasen weniger geeignet sind bzw. nur umgekehrt eingesetzt werden können, indem eine allgemein Bondgraph-Darstellung ein physikalisches System (elektrische Schaltung, mechanisches System) überführt wird.

4.2.3 STEP-Anwendungsprotokolle

STEP steht für „Standard for the Exchange of Product model data“ und ist ein Standard zur Beschreibung von Produktdaten zum Zwecke des Austausches (zwischen Anwendungsprogrammen oder Organisationen) oder zur Archivierung. Standardisiert ist STEP gemäß ISO 10303 und wird in [We09] folgendermaßen zusammenzufassend beschrieben:

> *Es ist eine Art Baukasten bestehend aus mehreren Dokumenten. Hierzu gehören:*
>
> - *die Sprache EXPRESS zur Beschreibung von objektorientieren Datenmodellen*
> - *Implementierungsmethoden zur Umsetzung der Datenmodelle, z.B. ein Textformat, (ISO 10303-21), XML-Format (ISO 10303-28) oder eine API (10303-22)*
> - *Basismodelle für Datenklassen, z.B. Produktidentifikation und -konfiguration (ISO 10303-41), visuelle Darstellung (ISO 10303-46) oder mathematische Beschreibungen (ISO 10303-51)*
> - *Anwendungsprotokolle als Erweiterung der Basismodelle, z.B. für finite Elemente Methoden (ISO 10303-104) oder Kinematik (ISO 10303-105)*
> - *Anwendungsprotokolle für Beschreibung von Produktdaten unter einem spezifischen Aspekt, z.B. ISO AP-214 zur Beschreibung von Produktdaten im Automotive-Bereich (ISO 10303-214)*

Besonders sind die Anwendungsprotokolle (AP) in dieser Arbeit Bedeutung, die dem interdisziplinärem Produktentwicklungsprozess unterstützen. Für diese Arbeit wurden daher folgende APs als relevant identifiziert:

- AP209 Edition 2: Multidisciplinary Analysis and Design
- AP233: Systems Engineering [AP233]
- AP239: Product life cycle support [AP239]
- AP242: Managed model-based 3D engineering [AP242]

STEP AP209

Das AP209 ist laut [www6] ein Standard für den Austausch und die Langzeitarchivierung von Informationen zwischen iterativen Entwurfs- und Analyseschritten im

Produktlebenszyklus, wie es beispielsweise der Austausch von Informationen zwischen CAD- und CAE-Systemen der Fall sein kann. Konkret formuliert ist das Ziel, die Bereiche CAD, CAE, PDM und Simulationsdatenmanagement zu integrieren. Vorwiegend im Fokus stehen dabei Branchen wie Schiffbau, Luftfahrt, Automobilindustrie oder andere Branchen mit komplexer und stark auf Simulation gestützter Produktentwicklung . AP209 Edition 2 deckt folgende Anwendungsbereiche ab:

- Finite-Elemente-Analyse für Strukturmechanik
- Produktdefinition: Struktur und Form, Materialeigenschaften
- Numerische Strömungsmechanik (Computational Fluid Dynamics, CFD)
- Kinematik
- Simulationsdatenmanagement mittels der Software SimPDM
- Materialdatenbank

Das AP209 hat gemeinsame Module mit dem AP242 (Erläuterung folgt), wodurch eine Integration der beiden APs gewährleistet werden soll, wie Abbildung 17 veranschaulicht. Das AP242 kann dabei als Teilmenge des AP209 gesehen werden. Weiterhin ist auch das AP237 für Strömungsmechanik eine Teilmenge des AP209. Die gleiche Produktstruktur wie das AP209 besitzen außerdem das AP233 und 239, weshalb das AP209 zum besseren Verständnis hier mit aufgeführt ist, jedoch nicht direkter Teil der untersuchten Produktmodelle sein soll.

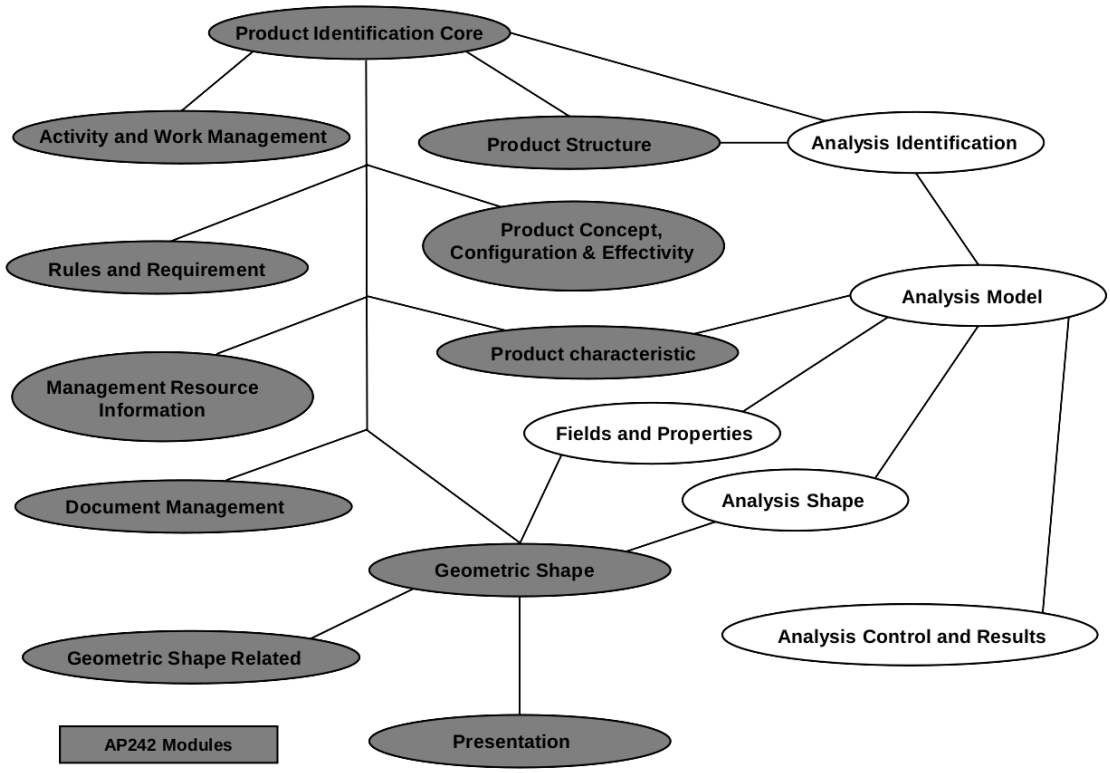

Abbildung 17: Module von AP209 Edition 2 und Teile davon als AP242 [www6]

STEP AP233

Das AP233 ist nach [Sie09] von diversen Organisationen, wie z.B. Object Management Group (OMG), OASIS (Organization for the Advancement of Structured Information Standards), der INCOSE (International Concil on Systems Engineering) sowie der ISO für das Systems Engineering entwickelt worden und im Jahre 2012 von ISO als Standard veröffentlicht worden. Gemäß der Definition des SE werden nicht nur technische sondern auch administrative und organisatorische Aspekte adressiert. Diesen Aspekten soll das AP233 Rechnung tragen, in dem es diese beiden Themenfelder als wichtigste Kategorien für seine Inhalte nennt, wie Abbildung 18 mit den zwei Hauptbereichen „Program management“ und „System Requirements/Design“ zeigt.

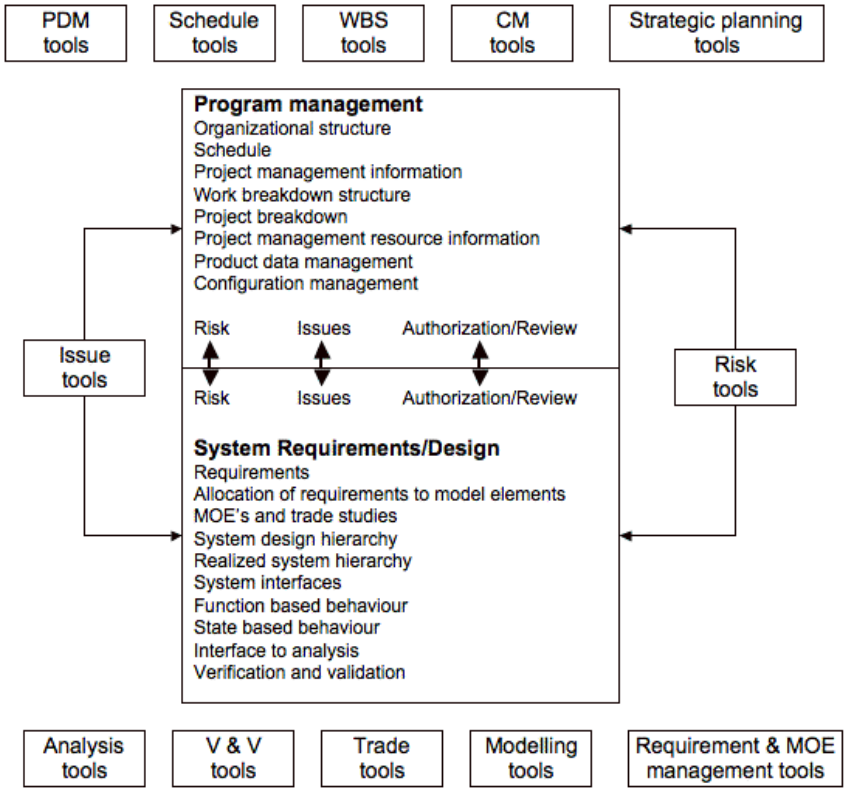

Abbildung 18: Inhalt des STEP AP233 aus [AP233]

Diese beiden Hauptbereiche sind verbunden via risk analysis, issue resolution sowie management authorization und review. Um die Hauptbereiche herum sind die Arten der wichtigsten Software-Werkzeuge abgebildet, die für das AP233 eingesetzt werden können. Oben stehen die administrativen Werkzeuge für PDM, Projektstrukturplan (Work Breakdown Structure, WBS), Konfigurationsmanagement (CM) und strategische Planung. Auf der unteren Seite befinden sich die Werkzeuge für Analyse, V&V (Verifikation und Validierung), Handel, Modellierung sowie für Management von Anforderungen und MOE (measure of effectiveness). Da die verschiedenen Anwendungsprotokolle laut [AP233] modular aufgebaut sind und möglichst viele gemeinsame Module nutzen, ist somit auch eine Möglichkeit gegeben, SE-Software mit anderen Software-Werkzeugen zu verbinden. Somit ist es laut ISO-Standard möglich, das AP233 als ein „integrierendes AP“ über andere APs hinweg zu nutzen. Die oberste Ebene der Module des AP233 sind in Abbildung 19 schematisch dargestellt. Gut erkennbar sind die Module für organisatorische (linke Seite) sowie eher technische Aspekte (rechte Seite).

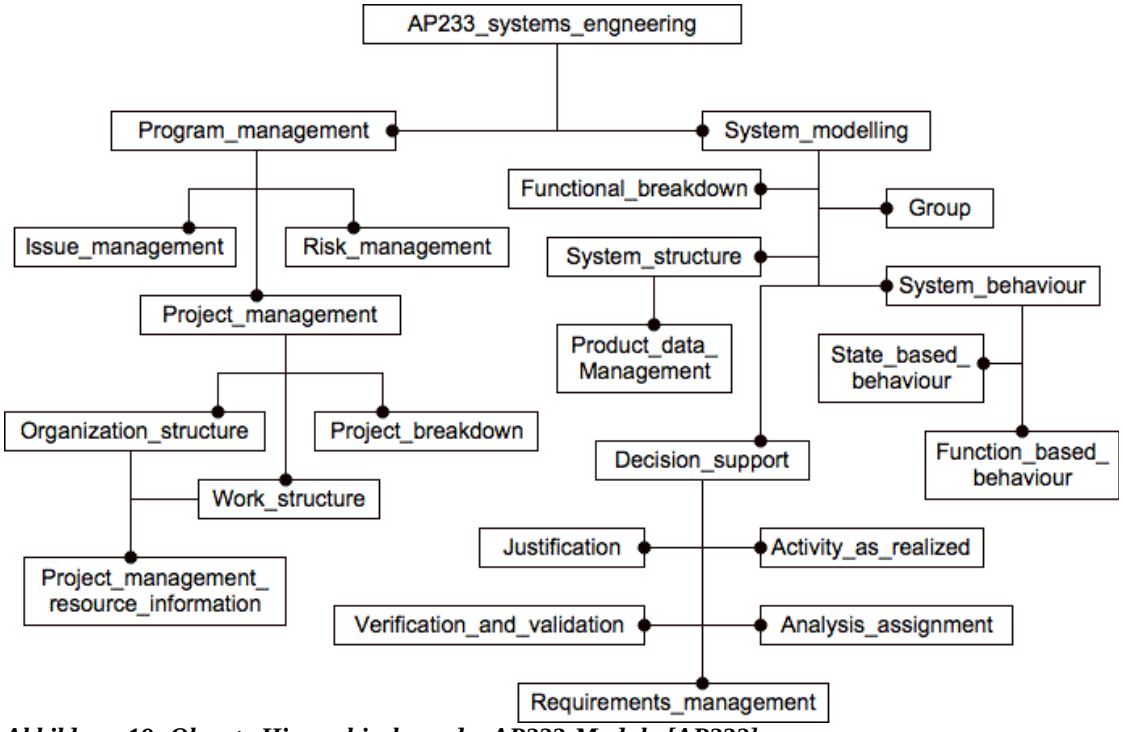

Abbildung 19: Oberste Hierarchieebene der AP233-Module [AP233]

STEP AP239

Das im Jahre 2012 standardisierte AP239 hat das Ziel, eine Unterstützung im Produktlebenszyklus zu bieten, wie das Bild der zugrunde liegenden Idee in Abbildung 20 zeigt. Angestrebte Branchen und Produkte sind vor allem diejenigen, die komplex sind oder anspruchsvollen Wartung und Reparatur benötigen, wie beispielsweise in der Luftfahrt, im Schiffbau, in der Kraftwerkstechnik oder Militärfahrzeuge. Die Intention ist, dass die Daten, welche vorwiegend in der Entwicklung des Produktes entstehen, über die Entwicklung und Fertigung hinaus im gesamten Produktlebenszyklus verfügbar sind. So können für die Nutzung und Wartung des Produktes Daten aus den „Life Cycle Shared Data“ angefordert werden oder auch für Rückmeldungen aus dem Einsatz des Produktes sowie Anweisungen zur Produktänderung den Produktdaten hinzugefügt werden.

Die Einsatzmöglichkeiten des AP239 für das Requirement Engineering wurden gemeinsam mit dem AP233 entwickelt, da beide APs insbesondere für komplexe Systeme gedacht und entwickelt worden sind.

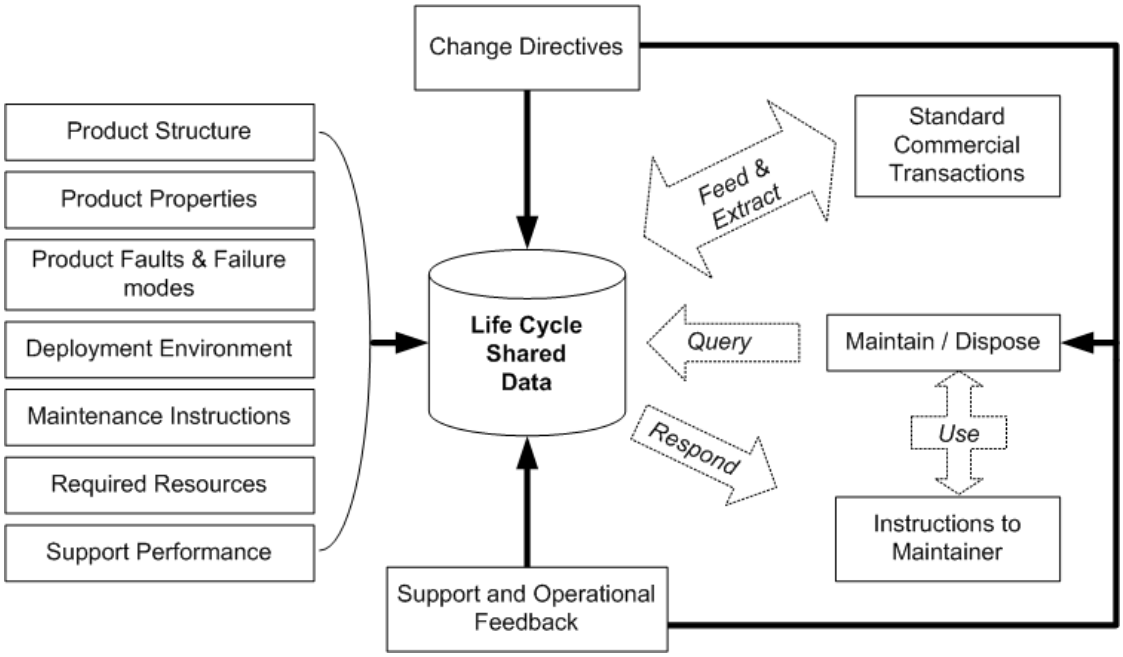

Abbildung 20: Produktlebenszyklus-Unterstützung des STEP AP239 [AP239]

STEP AP 242

Auf Initiative der Luftfahrt- und Automobilindustrie wurde das AP242 entwickelt und laut der Website des Projektes [www2] vereinigt es die beiden STEP AP203 („Configuration Controlled 3D Design“) sowie STEP AP214 („Core data for automotive mechanical design processes”) vereinigt. Ziel ist es den CAD/CAE- sowie PDM-Datenaustausch zu unterstützen. Beabsichtigte Anwendungsszenarien sind dreidimensionales, modellbasiertes Konstruieren, Langzeitarchivierung sowie Visualisierung von dreidimensionalen Modellen. Die wichtigsten Funktionalitäten des AP242 sind in Abbildung 21 nach Themenbereichen gezeigt. Es ist anhand der Darstellung ersichtlich, dass der Fokus zu einem großen Anteil auf mechanischer Entwicklung und Konstruktion liegt. Durch die zusätzliche Unterstützung von PDM-Funktionalitäten, Prozessplanung und Anforderungen ist in gewissem Umfang eine domänen- und phasenübergreifende Unterstützung gewährleistet. Das AP242 kann als das Komplement zu anderen ISO-standardisierten Formaten wie dem PRC (Product Representation Compact) oder JT (Jupiter Tesselation) gesehen werden, da diese ebenfalls standardisiert und im Maschinenbau für die Visualisierung von 3D-Modellen verbreitet sind. Allerdings besitzen diese einen anderen Schwerpunkt im Einsatzbereich sowie deutlich weniger Informationsgehalt [ISO12] [www13].

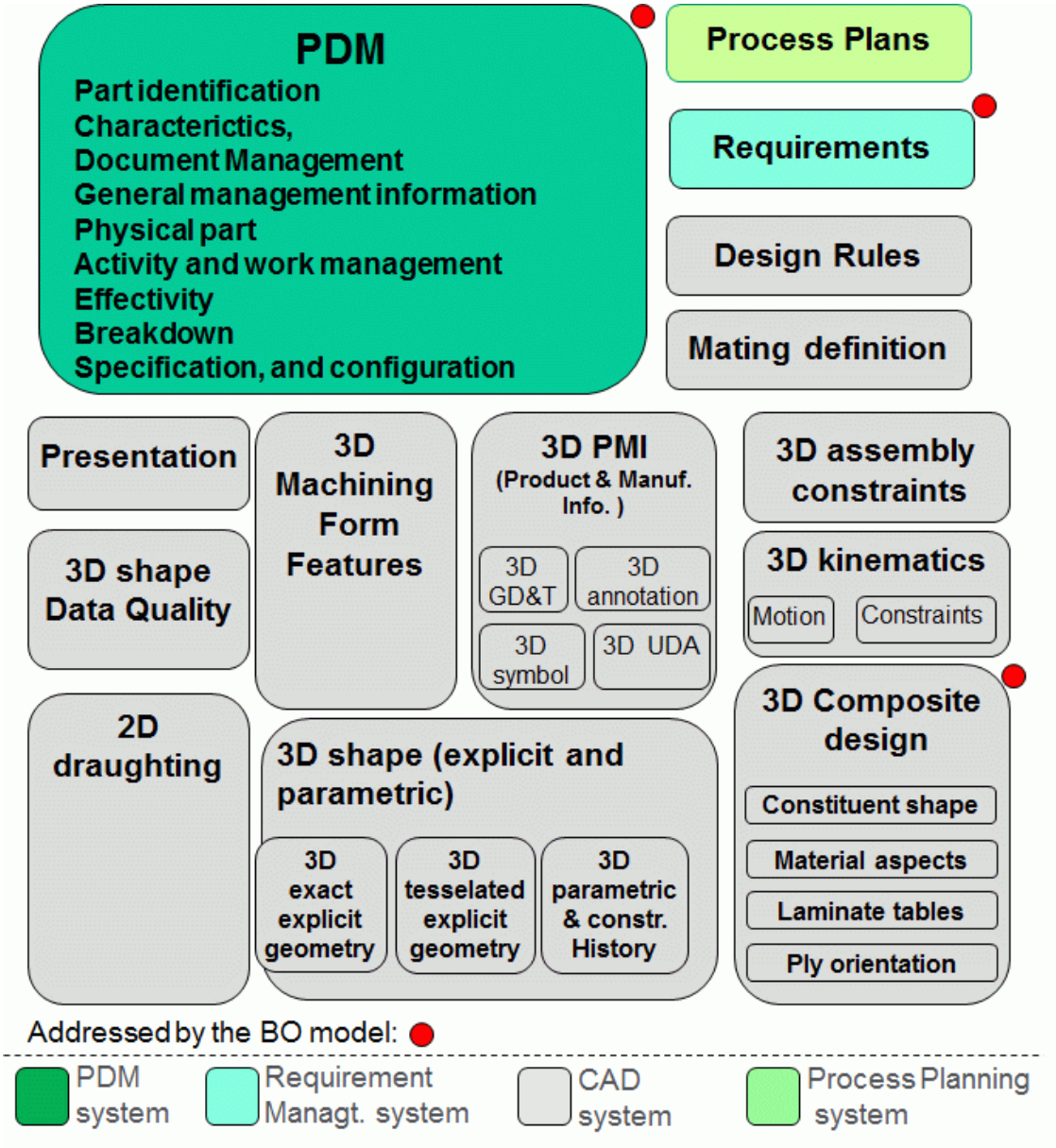

Abbildung 21: Übersicht des Informationsmodells für das AP242 aus [AP242]

Zu den hier betrachteten Anwendungsprotokollen wurde nichts Vergleichbares identifiziert, weshalb sie als sehr geeignetes Produktmodell erscheinen für die multidisziplinäre Produktentwicklung – sofern diese Anwendungsprotokolle von den CAx-Systemen unterstützt werden, beispielsweise in Form von Dateiformaten zum Datenaustausch.

Zusammenfassend ist die Bedeutung dieser beschriebenen Anwendungsprotokolle sowie die Schritte in der Produktentwicklung, welche unterstützt werden können, in Abbildung 22 ersichtlich.

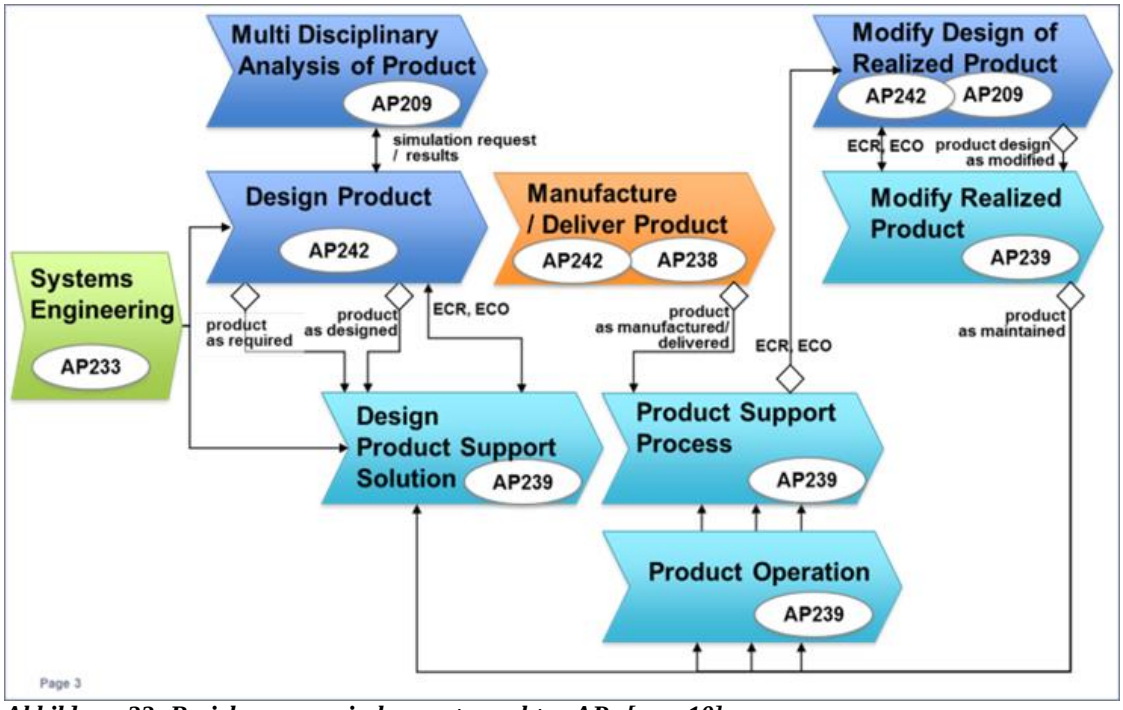

Abbildung 22: Beziehungen zwischen untersuchten APs [www10]

Auch die Einsetzbarkeit der APs über diverse Phasen des Produktlebenszyklus, wie in Abbildung 23 dargestellt, zeugt von einer hohen Ausgereiftheit. Lediglich in der Konzeptphase ist ausschließlich das AP233 einsetzbar, die Bereiche Simulation und Definition sind hingegen vergleichsweise stark adressiert.

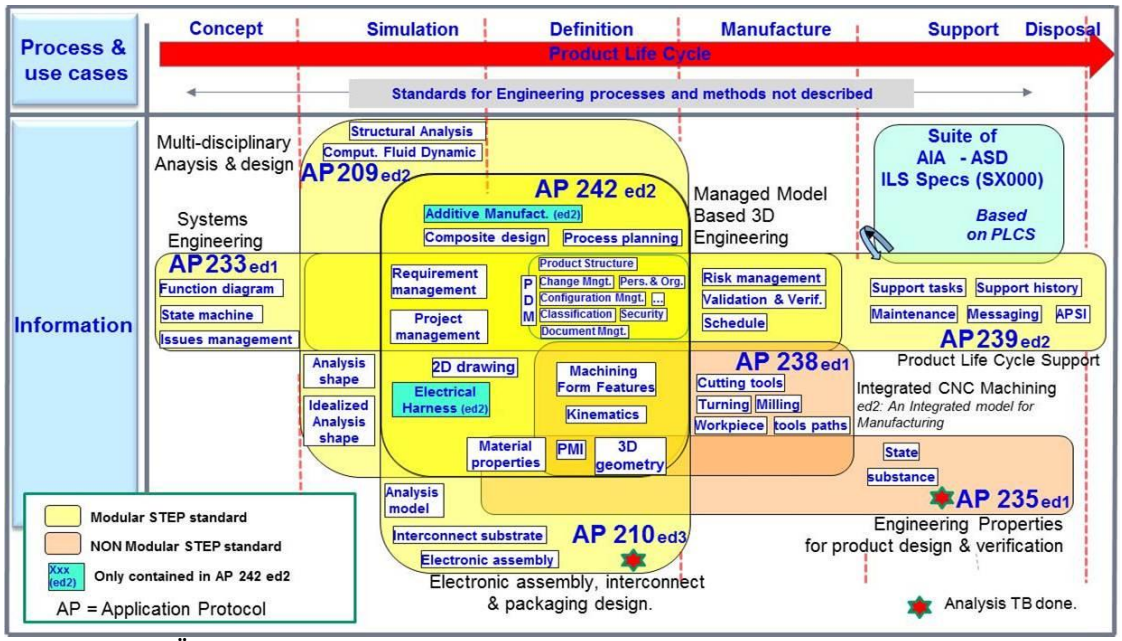

Abbildung 23: Übersicht über STEP APs für PLM-Interoperabilität [AP239ed3]

4.2.4 SysML

Die Systems Modeling Language (SysML) ist eine in Abwandlung der Unified Modeling Language (UML) entwickelte grafische Modellier- und Beschreibungssprache. Entwickelt wurde sie gemeinsam von der Object Management Group (OMG) und dem International Council on Systems Engineering (INCOSE) und wurde im März 2003 veröffentlicht [www5].

Im Gegensatz zur vorwiegend auf Software-Entwicklung ausgerichteten UML ist die SysML gezielt für das Systems Engineering entwickelt worden, wofür einige Diagramme aus der UML verändert, weggelassen oder Diagramme vollständig neu erschaffen wurden. Die unten stehende Tabelle 5 zeigt eine Übersicht über die Diagramme und Modelle der SysML nach Tim Weilkiens [We09]. Die wichtigsten beiden Sichtweisen auf ein System (bzw. auf ein Produkt) in der SysML sind dabei die Struktur und das Verhalten, wofür die beiden Spalten als kategorische Einordnung der meisten Diagramme und Modelle dienen. Eine Übersicht mit allgemeinen Beispielen für die SysML-Notationen ist von Weilkiens frei verfügbar [We13].

	Struktur	**Verhalten**	**Sonstiges**
Diagramm	Blockdefinitionsdiagramm Internes Blockdiagramm Zusicherungsdiagramm Paketdiagramm	Aktivitätsdiagramm Anwendungsfalldiagramm Zustandsdiagramm Sequenzdiagramm	Anforderungsdiagramm, Profil und Stereotyp, Zuteilung, Modellsicht, Dateiaustauschformate STEP AP233 und XMI
Modell	Strukturmodell	Verhaltensmodell	

Tabelle 5: Aufbau der SysML nach [We09]

Die SysML ist im Gegensatz zu einigen anderen in dieser Arbeit vorgestellten Produktmodellen eine reine Beschreibungssprache, die im Grunde auch ohne Rechnerunterstützung genutzt werden kann. Eine Rechnerunterstützung bietet dafür aber leichtere Erstellung, Bearbeitung, sowie Austausch und es können möglichst automatisiert andere Produktmodelle mit SysML-Diagrammen in Abhängigkeit gesetzt werden. Des Weiteren bezieht die SysML den Einsatz eines Dateiformates für den neutralen Dateiaustausch explizit mit ein, und zwar das in der Tabelle aufgeführte STEP AP233 für Systems Engineering. SysML und das AP233 haben inhaltlich starke Überschneidungen, jedoch auch Unterschiede wie Abbildung 24 der OMG zeigt. Die Entwicklung von STEP AP233 und der SysML wurde laut [We09] gemeinsam durchgeführt und aufein-

ander abgestimmt, um SysML-Modelle in andere SE-Software übertragen zu können und umgekehrt.

Der Austausch der Modelle erfolgt gemäß der STEP-Implementierungsmethoden via der von der OMG standardisierten XMI-Spezifikation (XML Metadata Interchange) oder mittels einer Programmierschnittstelle (API). XMI ist ein offenes, standardisiertes Austauschformat für den Austausch von Metadaten zwischen Software-Entwicklungswerkzeugen auf Grundlage der textlichen Beschreibungssprache XML (Extensible Markup Language).

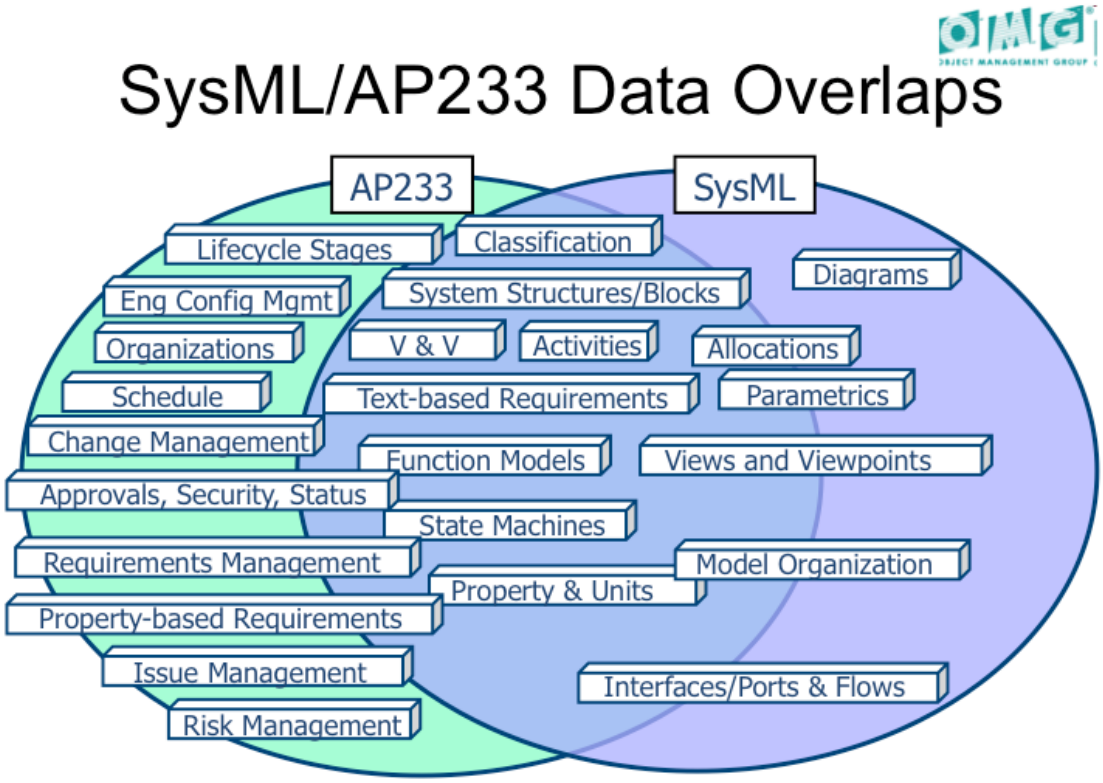

Abbildung 24: Überschneidungen von SysML und STEP AP233 aus [www3]

Falls nicht schon geschehen, ist die SysML ist dabei, zur wichtigsten Beschreibungssprache für komplexe technische Systeme zu avancieren, wie beispielsweise in [AZ13] oder [Bia15] zum Ausdruck kommt. Darüber hinaus steigert die gemeinsam mit dem STEP AP233 abgestimmte Entwicklung die vielseitige Einsetzbarkeit dieser standardisierten Modelliersprache in einem erheblichen Maße.

4.2.5 Modelica

Modelica ist nach Eigendarstellung auf der Website der Modelica Association [www4] eine Beschreibungssprache für das dynamische Verhalten komplexer physikalischer Systeme. Modelica wird seit 1996 von der Modelica Association entwickelt und auch seitdem gefördert. Zusätzlich existieren dazu noch Gruppen von Modelica nutzenden Personen in diversen Ländern. Basierend auf dieser freien Sprache wurden diverse Simulationsumgebungen entwickelt, einerseits kommerzielle Software-Lösungen von Entwicklern bekannter CAx-Systeme, andererseits auch Open Source Software. In der Methode von Kleiner und Kramer wird auf den Einsatz des kommerziellen CAx-Systems CATIA V6 des Herstellers Dassault Systèmes verwiesen, welches Modelica verwendet.

Die Sprache ist nicht-proprietär, objektorientiert sowie gleichungsbasiert und besitzt Software-Bibliotheken für folgende Fachdomänen:

- Elektrik, Elektronik, Magnetismus und Digitaltechnik
- Mechanik: eindimensionale Rotation und Translation, dreidimensionale mechanische Systeme, Fahrzeugdynamik, Getriebe und Antriebsstränge,
- eindimensionale Fluidmechanik: Rohr-Netzwerke mit Behältern, Pumpen etc.
- Thermodynamik
- Regelungstechnik
- mathematische Funktionen

Die Bibliotheken unterliegen meist freien Lizenzen, jedoch existieren auch kommerzielle Bibliotheken oder Bibliotheken, die von einzelnen Personen beigesteuert wurden und teilweise keiner Lizenz unterliegen.

Die Modellierung erfolgt entweder mittels Blockschaltbilder (in freien oder kommerziellen Simulationsumgebungen) oder direkt in der freien textuellen Beschreibungssprache, wie in Abbildung 25 exemplarisch illustriert ist. Es ermöglicht die Simulation eines multidisziplinären Systems auf ebenfalls unterschiedliche Art und Weise in Form von Quelltext (links), 3D-Grafik (im Hintergrund) sowie diversen Blockschaltbildern.

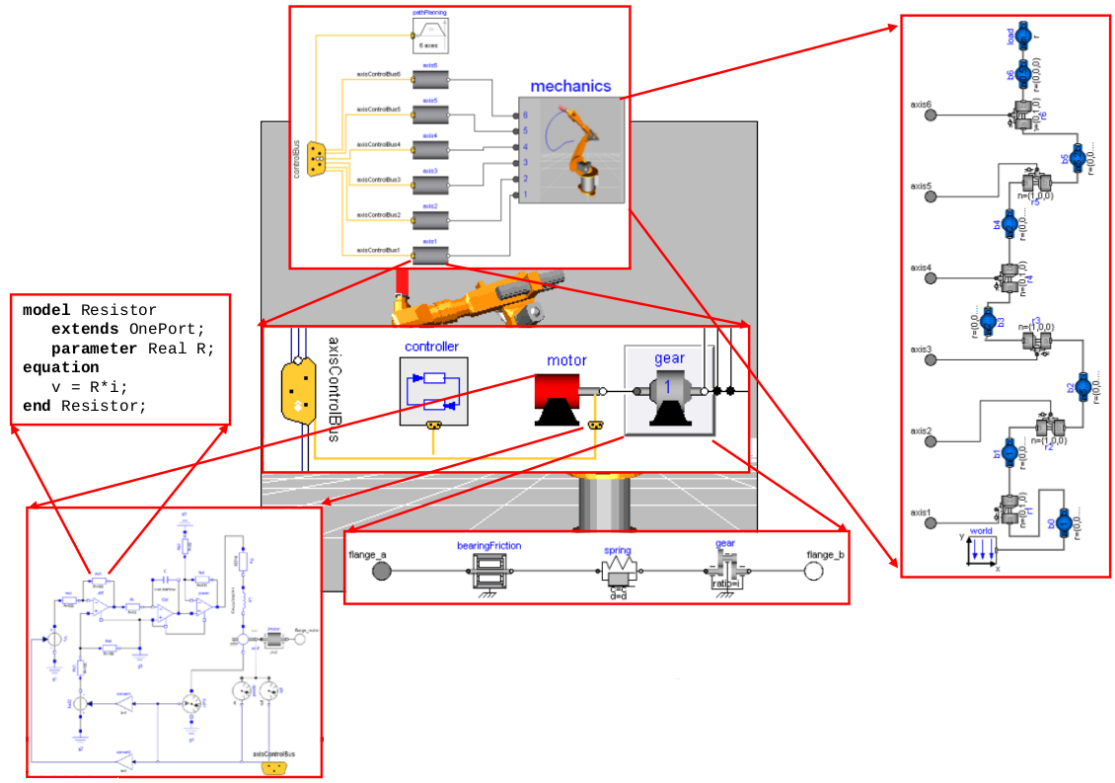

Abbildung 25: Domänenübergreifende Simulation in Modelica [OW13]

Modelica besitzt auch Kompatibilität zur anderen Beschreibungsformen und Programmiersprachen. So existieren Input- und Output-Funktionsblöcke für die Very High Speed Integrated Circuit Hardware Description Language (kurz: VHDL), eine nach IEEE standardisierte, textbasierte Hardwarebeschreibungssparache für digitale Systeme, sowie Digitaltechnikkomponenten gemäß dem VHDL-Standard. Es können darüber hinaus Funktionen anderer Programmiersprachen wie C, Java oder Fortran innerhalb von Modelica aufgerufen werden. Weitere Schnittstellen sind eine XML-Repräsentation [PF03] sowie das Functional Mock-up Interface (FMI) [www11]. Das FMI ist eine standardisierte Schnittstelle zur Kopplung verschiedener Simulationssoftware (Co-Simulation) oder zum Austausch von Modellen. Entwickelt wird das FMI ebenfalls unter der Schirmherrschaft der Modelica Association. Nach [BO11] lässt sich die Idee hinter dem FMI mittels Abbildung 26 beschreiben. Links ist jeweils das Software-Werkzeug gezeigt („Tool“) und rechts die Functional Mock-up Unit („FMU“) mit dem Simulationsmodell und ggf. dem Solver, der die Berechnung durchführt.

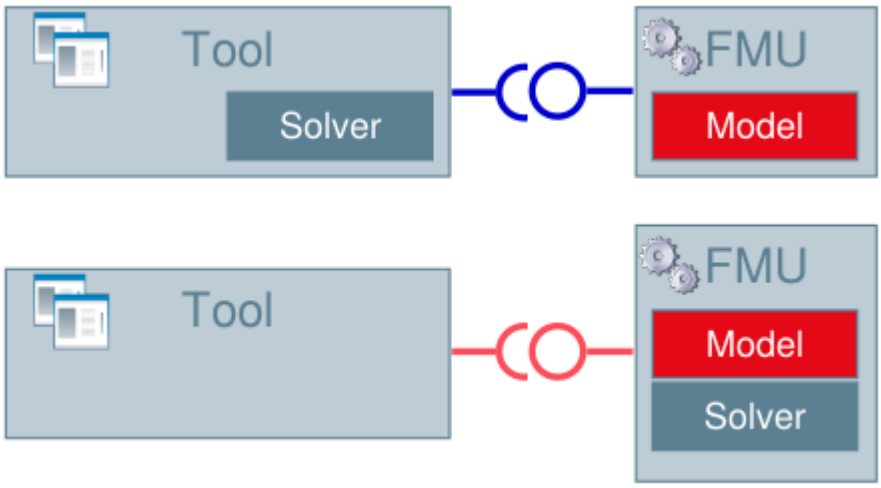

Abbildung 26: Modellaustausch (o.) und Co-Simulation (u.) des FMI [BO11]

Die FMU ist eine gepackte Datei und beinhaltet die Beschreibungsdatei im XML-Format sowie Implementation als Quellcode oder in Binärform. Im Falle das Modellaustausches wird die Simulation mit dem Solver des jeweiligen Tools ausgeführt, während bei der Co-Simulation der Solver der jeweiligen FMU genutzt wird.

Mit seiner Offenheit, einer Vielzahl an Bibliotheken sowie Unterstützung durch eine internationale Gemeinde, der Forschung sowie der Industrie (sowohl in der Nutzung als auch in der Einbeziehung in eigene kommerzielle Software-Produkte) hat sich Modelica als vielseitige Grundlage zur Beschreibung und Simulation technischer Systeme etabliert. Bei Bedarf lassen sich mittels FMI viele weitere Simulationswerkzeuge zusätzlich einsetzen.

4.2.6 CPM2

Das Central Product Model 2 (CPM2) ist ein Datenmodell, welches vom US-amerikanischen National Institute of Standards and Technology (NIST) entwickelt und in [FFBS08] vorgestellt wurde. Das CPM2 hat das Ziel, die PLM-Unterstützung über den gesamten Lebenszyklus hinweg zu ermöglichen. Es handelt sich dabei um ein generisches, abstraktes Modell für vorwiegend elektromechanische Systeme. Das CPM2 basiert auf zwei Prinzipien:

1. Das Schlüsselobjekt ist das Artefakt, welches z.B. ein Bauteil oder eine (Unter-)Baugruppe repräsentieren kann

2. Das Artefakt ist eine Aggregation aus drei Objekten, welche die drei prinzipiellen Aspekte des Artefakts repräsentieren:
    1. Funktion (beabsichtigtes Verhalten)
    2. Form (Geometrie und Material)
    3. Verhalten (mittels Simulation und/oder Experiment analysiert und mit Funktion verglichen)

Das CPM2 besitzt zwei Arten von Klassen: *object* und *relationship*, welche entsprechend dem Entity Relationship-Modell (ER-Modell, siehe Abbildung 27) gebildet sind und in vier Gruppen eingeteilt sind:

- abstrakte Klassen zur Organisation des Modells ohne Instanzen
- Objektklassen zur Repräsentation von Entitäten wie im ER-Modell
- Beziehungsklassen zur Repräsentation von Beziehungen gemäß ER-Modell
- Utility-Klassen für zusätzliche Informationen

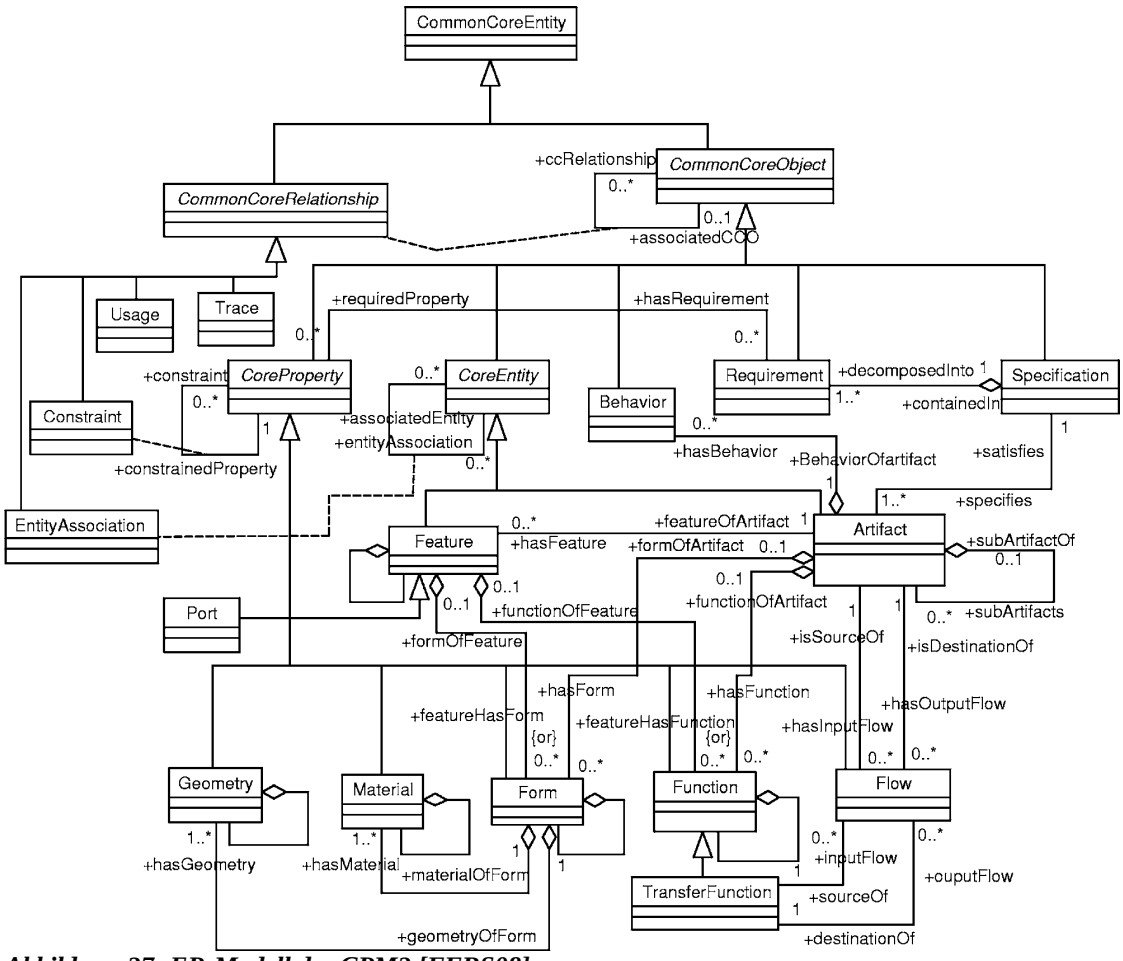

Abbildung 27: ER-Modell des CPM2 [FFBS08]

Bei den Klassen *CommonCoreEntity*, *CommonCoreRelationShip*, *CommonCoreObjekt*, *CoreEntity* sowie *CoreProperty* handelt es sich um abstrakte Klassen. Die Klassen *Artifact*, *Feature*, *Function*, *Form*, *Behavior*, *Geometry*, *Material* dienen zur Beschreibung der beiden oben genannten Prinzipien. *Specification* und *Requirement* dienen für Beschreibung von Spezifikation und Anforderungen, während *Port*, *TransferFunction* sowie *Flow* zur Definition von Schnittstellen durch die Beschreibung des dazugehörigen Energie-, Material- oder Signalflusses dienen.

Darüber hinaus existieren noch Klassen, welche in der Abbildung weniger ersichtlich sind, und zwar die Beziehungsklassen (*Constraint*, *EntityAssociation*, *Usage*, *Trace*) sowie für zusätzliche Informationen die Utility-Klassen (*Information*, *ProcessInformation*, *Rationale*).

CPM2-Modelle können auf drei Stufen der Abstraktion bzw. Konkretisierung existieren, je nach Fortschritt der Entwicklung des jeweiligen Produktes:

Conceptual Model: domänenübergreifende, generische Produktinformationen ohne domänen- oder objektspezifischen Attribute.

Intermediate Model: Besteht typischerweise aus wenigen Instanzen und Attributen, welche sukzessiv ausgebaut werden. Es setzt sich zusammen aus: Informations- und Typ-Attribut. So kann beispielsweise ein Artefakt vom Typ „Zylinderstift“ existieren, den Eigenschaften in den Informationen „Länge: 50mm; Durchmesser 5mm“.

Implementation Model: Ist ein Teil der „Model Driven Architecture“ (MDA) oder Object Management Group (OMG). Die MDA ermöglicht die Konvertierung von plattformunabhängigen Modellen (wie CPM2) in plattformspezifische Modelle.

Das CPM2 bleibt in seiner Beschreibung meist recht allgemein und abstrakt. Vergleichsweise direkte Anwendung ermöglicht das CPM2 aber in der Darstellung von Anforderungen, Spezifikation, Funktionen oder der Produktstruktur, wodurch es durchaus vergleichbar mit der SysML oder der Funktionsstruktur ist.

Weiterhin existieren diverse Erweiterungen des ersten CPM. Erwähnenswert sind in diesem Zusammenhang unter anderem das Open Assembly Model (OAM) oder das Mechatronic Device Model. Das OAM ist als Standard-Repräsentation und Austauschprotokoll für Baugruppen entwickelt worden, das für die Datenstruktur STEP nutzt. Das OAM unterstützt Toleranz- oder Kinematik-Analysen und ist offen für weitere Anwendungen wie FEM, Prozessplanung oder virtuelle Montage mittels Virtueller Realität. Das Mechatronic Device Model hingegen ist als unterstützendes Framework für

die konzeptionelle Entwicklung von „multiple interaction-state mechatronic devices“ gedacht, um Entwürfe beispielsweise auszutauschen oder auf Anforderungserfüllung zu prüfen.

4.2.7 Meta-Datenmodell und Integrationskonzept von Anderl und Nattermann

Zusätzlich zu dem in Kapitel 3 vorgestellten W-Modell haben auch Anderl und Natterman in [NA10], [NA11], [NA13a] sowie [NA13b] ein Datenmodell sowie ein Datenmanagementsystem (DM-System) beschrieben, welche die für die Adaptronik typische, starke Integration von Komponenten beinhaltet. Das entwickelte Konzept besteht aus mehreren Ebenen (sog. Layer), die vor allem unterschiedliche Abstraktionsgrade darstellen und in Tabelle 6 strukturiert aufgeführt und erläutert sind. Die oberste Ebene bildet eine sehr abstrakte Systemmodellierung ab, während die beteiligten Fachdisziplinen die unterste Ebene bilden, welche für die physikalische Umsetzung zuständig sind. Dieses Layer-Konzept veranschaulicht gut, dass die einzelnen Domänen lediglich eine untergeordnete Rollen spielen und Aspekte wie das Verhalten oder die Parameter essenziell sind. Durch die automatische Simulation mitsamt Auswertung ist auch keine explizite Integrationsphase mehr nötig, da diese fortwährend durchgeführt wird. Für das Layer-Konzept wurde ein entsprechendes Metadatenmodell entwickelt und in [NA13b] vorgestellt, welches folgende Pakete enthält:

ActiveSystem: Setzt sich zusammen aus unterteilt in fünf Typen von Komponenten zusammen: Funktionen, Anforderungen (im Requirements Interchange Format, ein von der OMG standardisiertes XML-Dateiformat für Anforderungen), Parametern und dem Systemmodell.

SystemModel*:* Die Systemdefinition erfolgt durch Elemente mit gemeinsamen Beziehungen sowie Verlinkung von Systemelementen zu Parametern und Eigenschaften. Parameter sind hier unabhängig von anderen Werten und frei definierbar.

SystemSimulation: Die Simulation ist modularisiert und es sind Modulbibliotheken für typische Systemelemente verfügbar. Die Inputs und Outputs der Module sind mit Parametern und Eigenschaften verlinkt.

SystemParameter, **SystemProperties**: Es wird explizit zwischen Parametern und Eigenschaften unterschieden. Parameter sind frei modifizierbare Werte und von anderen Werten unabhängig, während Eigenschaften vollständig von anderen Eigenschaften

und Parametern abhängen und somit nicht frei definiert werden können. In diesen beiden Paketen repräsentiert das Metadatenmodell das zentrale Parameter-Eigenschafts-Netzwerk.

ParameterPropertyRelation: Hier bei handelt es sich um die Repräsentation der Beziehung zwischen Systemeigenschaften und -beziehungen. Das Metadatenmodell nutzt das sog. Property-Driven-Design, bei dem Parameter in Eigenschaften transferiert werden.

Die Simulation erfolgt dabei stets mittels Matlab, während das PDM-System auch ein anderes, gängiges und am Markt befindliches System sein kann. Auch im Integration Layer ist eine gewisse Flexibilität ermöglicht, wodurch dieses Layer-Konzept auch als Grundlage für Erweiterungen und Abwandlungen dienen kann. Allerdings ist immer die Datenbank für das Datenmanagement nötig, bei der es sich um eine Eigenentwicklung handeln muss, da diese nicht weiter beschrieben wird.
Eine Simulation mit Modelica anstelle von Matlab würde das Konzept auf einen offenen Standard basieren lassen und somit sicherlich die Verbreitung dieses Konzeptes vorantreiben, denn über einen Einsatz in der industriellen Praxis konnte im Rahmen der Recherche für diese Arbeit nichts gefunden werden. Allerdings ist es das einzige identifizierte Konzept, welches gezielt auf die Eigenheiten adaptronischen Komponenten mit deren hochgradiger Integration eingeht.

Systemmodellierung

- Paket: ActiveSystem
- Unterteilung des aktiven Systems in Ansteuerung, mechanische Struktur, Aktuatorsystem, Sensorsysteme, Signalverarbeitung

Parameter-Analyse

- Pakete: SystemParameter, SystemProperties
- gewichtetes Netzwerk aus Parametern
- Beschreibung der Integration von Simulations-Approaches
- konsistent mit der Systemaufteilung des Datenmodells durch Verbindung von Komponenten aus DM-Systemdatenbank mit Teilen des Matlab-Modells (Parameter und Eigenschaften werden aus der Datenbank in das Matlab-Modell übertragen)
- automatische Simulation sowie Benachrichtigung an die User über Auswirkungen auf Gesamtsystem anhand der Beziehungen zwischen den verschiedenen Parametern (Letztere werden manuell via Graphical User Interface eingegeben)

Systemsimulation

- abstrakte Simulation des globalen Systemverhaltens, basierend auf Herold-Modell
- Durchführung und Test automatisiert und standardisiert
- Software-Tool: Matlab/Simulink
- modular, verschiedene Abstraktionsgrade
- in Änderungs- und Veröffentlichungs-Managent des DM-s

Systemzuverlässigkeit

- wird aus Zuverlässigkeit der Einzelkomponenten und jeweilig verbundener Berechnungsmethode abgeleitet
- automatische Berechnung und Analyse bzgl. Anforderungen

PDM-Funktionalität

- Funktionen eines gängigen PDM-Systems
- Eingesetztes PDM-System: ENOVIA SmarTeam V5

Integration-Layer

- Variante 1: standardisierte Konnektoren (sog. Connectors)
  - Vorteil: nur ein Service nötig für Datenintegration
  - potenzieller Nachteil: Anzahl der nötigen Konnektoren steigt mit Anzahl eingesetzter Software-Tools
- Variante 2: SOA (Serivce-oriented Architecture)
  - Software richtet sich nach Interessen und Prozessen der jeweiligen Organisation
  - Anwendungen werden als Dienstleistungen bereitgestellt, weshalb Hard- und Software in den Prozessen irrelevant sind

Mechanik	Elektronik	Software	sonstige

Tabelle 6: Layer-Konzept für die Entwicklung adaptronischer Systeme [NA10] [NA13a] [NA13b] [NA11]

4.2.8 Multidisziplinäre Modellierung und Simulation für mechatronische Entwicklung

Jérémy Lefèvre, Sébastien Charles, Magali Bosch-Mauchand, Benoît Eynard und Éric Padiolleau stellen in [LCBME14] ein Konzept zur Integration mehrerer Produktmodelle und Simulationen vor. Als Vorgehensmodell liegt das V-Modell zugrunde und bei den Produktmodellen wird ebenfalls auf bereits vorhandene Entwicklungen zurückgegriffen. Besonders an dem Konzept von Lefèvre et al. ist die Verwendung eines globalen Modells sowie mehrerer davon abhängiger Sub-Modelle (siehe vereinfachtes UML-Klassendiagramm in Abbildung 28).

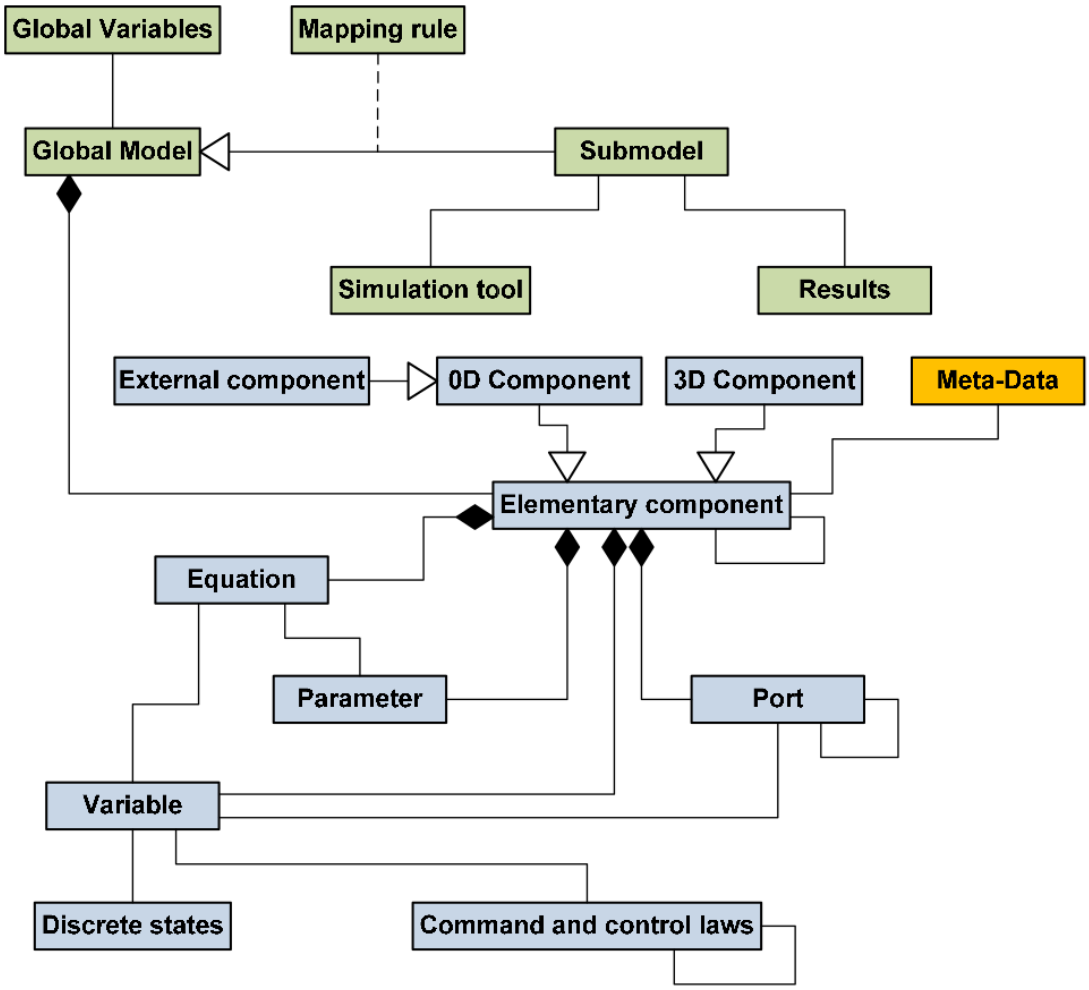

Abbildung 28: UML-Diagramm des Ansatzes von Lefèvre et al. [LCBME14]

Das Prinzip, mehrere Produktmodelle in Abhängigkeit voneinander einzusetzen, wurde bereits in Abbildung 4 dargestellt. Ein Verweis auf die Modellkohärenz nach [Sei85] geschieht seitens Lefèvre et al. nicht, es ist jedoch durchaus möglich, das die-

ses Konzept von anderen in gleicher oder ähnlicher Form entwickelt wurde. Implementiert wurde dieses Konzept in einer von Lefèvre entwickelten PLM-Plattform.

Erwähnenswert ist im Konzept von Lefèvre et al. des Weiteren der explizite Einsatz von Simulation. Für die Modellierung und Simulation des globalen Systemverhaltens kommen dabei SysML und Modelica zum Einsatz. Für die Interoperabilität zwischen den genutzten (und meist kommerziellen) CAE-Tools wird entweder das als High-Level-Konzept eingestufte „mesh based parallel code coupling interface“ (MpCCI) oder das als Low-Level-Konzept eingestufte STEP verwendet, da dort der Datenaustausch dateibasiert stattfindet. Bei MpCCI handelt es sich um einen Server, der auf verschiedenen Betriebssystemen (Microsoft Windows, Linux) installiert werden kann und einige kommerzielle CAE-Systeme mit einbezieht, um die Simulationscodes in Echtzeit zu koppeln.

Eine Übersicht über diese IT-Architektur ist in Abbildung 29 gezeigt. Grundsätzlich wird bei der IT-Architektur zwischen einer modellbasierten und einer dreidimensionalen Simulation unterschieden. Erstere setzt auf SysML und Modelica, Letztere auf STEP. Zur Kopplung von SysML und Modelica wird ein Integrationskonzept von Paredis et al. [Pa+10] eingesetzt, welches eine Modelltransformation vorsieht. Eine Transformation bedeutet in diesem Zusammenhang die automatische Erstellung eines Zielmodells aus einem Quellmodell. Darüber hinaus kommt das STEP AP239 auch über den gesamten Lebenszyklus hinweg für den Datenaustausch und die Datenverwaltung zum Einsatz.

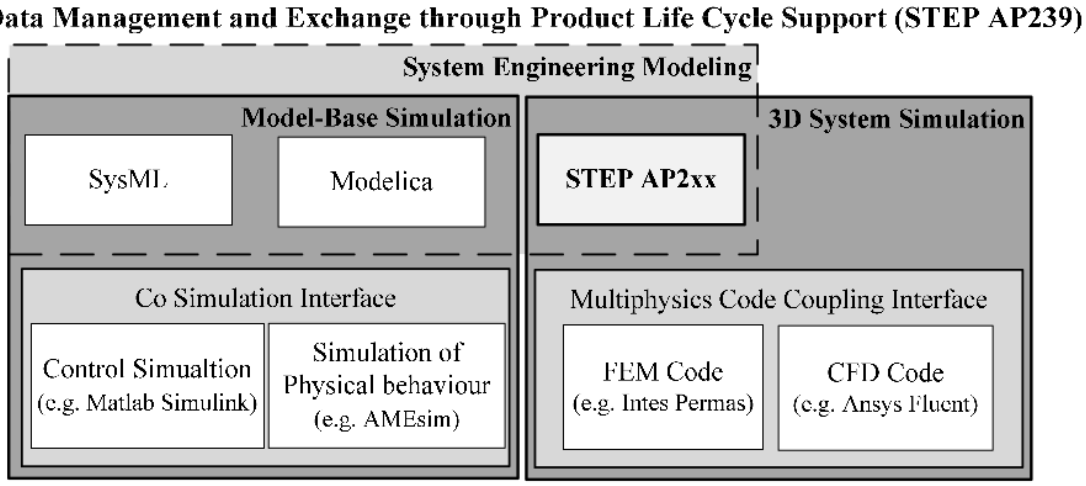

Abbildung 29: IT-Architektur für multidisziplinäre Simulation [LCBME14]

Insgesamt handelt es sich nicht um ein neues Produktmodell, sondern um ein auf bereits vorhandenen Modellen gestütztes Integrationskonzept. Hervorzuheben ist hierbei die Nutzung etablierter und offener Standards, zumindest für die Modellierung und den Datenaustausch. Allerdings sehen die Entwickler für dieses Konzept noch Forschungsbedarf für die Definition eines Daten-Backbones zwischen Systemmodellierung und multidisziplinärer Simulation.

4.2.9 SysLM-Datenmodell für MBSE nach Eigner et al.

Eigner et al. haben für deren entwickeltes MBSE/MVPE-Vorgehensmodell ein sogenanntes SysLM-Datenmodell für MBSE entwickelt, welches im Jahre 2012 erstmals vorgestellt wurde und unter anderem in [Eig13] und [EGZ12] beschrieben ist. SysLM steht dabei für Systems Lifecycle Management, welches eine Erweiterung des PLM-Datenmodells um die systemtechnischen Elemente bedeutet.

Um die erstellten SysML-Modelle in ein PDM-System zu integrieren, wurde ein Datenschema zur Abbildung auf XML mit unterschiedlichen Modellierungselementen vorgeschlagen:

- **Hierarchien**: hierarchische Abbildung von Anforderungs-, Funktions- und logischen Elementstrukturen. Die einfachen, stücklistenartigen Formen von Produktstrukturen (Baumstrukturen), welche in PDM-Systemen existieren, können dafür als Grundlage verwendet werden.
- **Querverweise**: Beziehungen zwischen Modellelementen und modellgrenzenübergreifendenden Elementen, welche mit Zuweisungen in SysML modelliert werden.
- **Typisierte interne Verbindungen** zwischen Modellelementen in einem Modell, z.B. mit Modelica oder dem internen Blockdiagramm aus der SysML.

Eine vereinfachte Darstellung des Datenschemas ist in Abbildung 30 gezeigt. Logische Systemelemente repräsentieren die Realisierung von Funktionen mittels eines physikalischen Effektes. Ein Systemelement kann auch externe Modelle referenzieren, z.B. in Simulationswerkzeugen wie Modelica, Matlab (inkl. Simulink oder Simscape) oder VHDL. Weiterhin ist ein logisches Systemelement mit den Stammdaten und der Stückliste verbunden, die von verschiedenen CAD-Systemen für Mechanik und Elektrik referenziert wird. Die Integration der funktionalen Produktbeschreibung als Teil des PLM-Konzeptes wird in Abbildung 31 gezeigt.

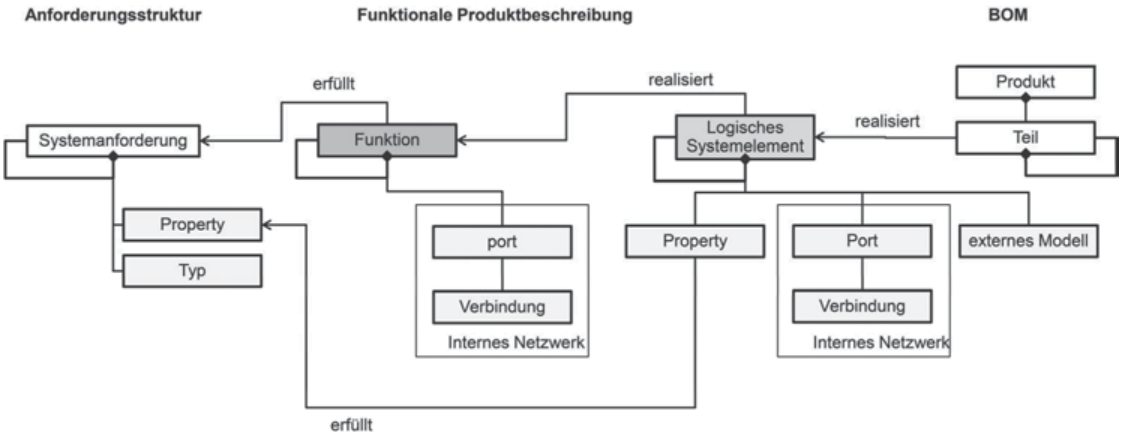

Abbildung 30: Vereinfachtes Datenschema für den MBSE/SysLM-Ansatz [Eig13]

Grundlage bildet ein PDM-System (in der Abbildung als PLM bezeichnet), welches für die jeweiligen Phasen (Anforderungen, Funktionen, logische Elemente sowie physikalische Komponenten) bestimmte Produktstrukturen abbildet und eine Rückverfolgbarkeit („Traceability“) von Änderungen ermöglicht (dargestellt durch die geschwungenen Pfeile in der oberen Hälfte der Abbildung). Die untere Hälfte des Bildes stellt die Abbildung des Produktes in den jeweiligen Autorensystemen dar, wie z.B. Software-Werkzeuge für das Anforderungsmanagement oder verschiedene CAx-Systeme.

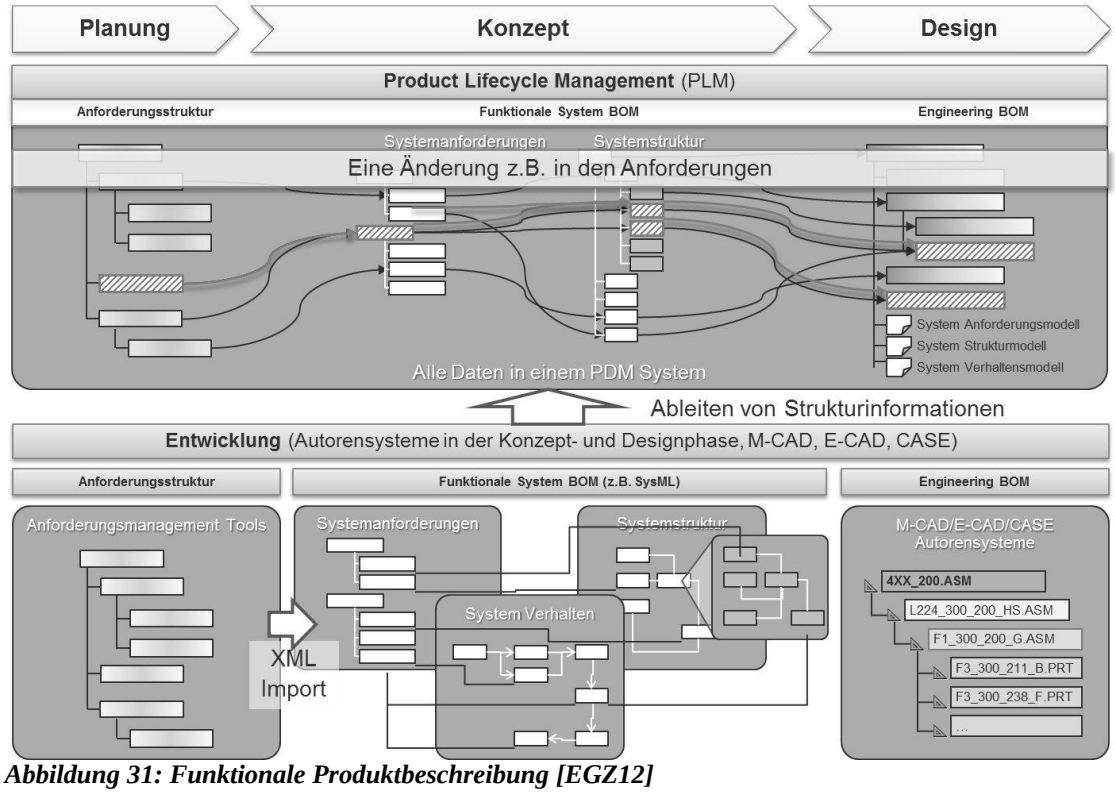

Abbildung 31: Funktionale Produktbeschreibung [EGZ12]

Auffallender Unterschied bei diesem Konzept ist, dass keine spezielle oder gar eigens dafür entwickelte Software benötigt wird, sondern auf am Markt typische PDM- und CAx-Systeme genutzt werden können. Dadurch ist dieses Konzept leichter und vielseitiger umsetzbar, da die Auswahl der Software eine gewisse Freiheit und Flexibilität bietet. Im Gegenzug dazu ist die Kopplung der unterschiedlichen Software-Werkzeuge auch weniger eng im Vergleich zum Konzept von Anderl und Nattermann, bei dem automatisch eine regelmäßige Simulation des Systems durchgeführt wird oder im Konzept von Lefèvre et al. mit der Kopplung von unterschiedlichen Simulationen.

4.3 Analysekriterien

Nachfolgend sind die Kriterien aufgelistet und beschrieben, auf deren Vorhandensein und Ausprägung die Produktmodelle untersucht werden. Die Auswahl der Kriterien erfolgte unter anderem im Hinblick darauf, wie vielseitig diese Modelle einsetzbar sind, weshalb Aspekte wie Interoperabilität oder Standardisierung bewertet werden.

1. **Rechnerunterstützung**: Ermöglichung, Föderung sowie Unterstützung des Einsatzes von Rechnern.
2. **Abstraktionsgrade**: Ermöglichung von unterschiedlichen Abstraktions- oder Konkretisierungsgrade zur Produktdarstellung.
3. **Standards**: Standardisierung des Produktmodells bzw. Einbeziehung oder Nutzung von anderweitigen standardisierten Konzepten oder Modellen.
4. **Interoperabilität**: existieren standardmäßig oder zusätzlich verfügbare Methoden und Werkzeuge zum Austausch oder zur Transformation in andere Modelle, Sprachen oder andere Darstellungsformen.
5. **explizite Software notwendig**: dieses Kriterium geht über die Rechnerunterstützung hinaus, da es dazu dient, ob das Produktmodell eine explizite Software nutzt. Gerade im Fall von proprietärer Software kann dies die Nutzung des Produktmodells unter Umständen sehr einschränken, wenn der Einsatz einer bestimmten Software nicht möglich oder nicht erwünscht ist.
6. **Phasenübergreifende Kopplung**: Einsetzbarkeit des Produktmodells über mehrere Phasen hinweg. Dies ist dahingehend von Vorteil, dass nicht für jede Phase ein eigenes Produktmodell verwendet werden muss, was den Aufwand an eventuellen Modelltransformationen sowie den Bedarf unterschiedlicher Produktmodelle minimiert.

7. **Sonstige Aspekte**: primär ist in dieser Arbeit bei den Produktmodellen die Abbildung der Produkteigenschaften von Relevanz. Dennoch soll am Rande berücksichtigt werden, inwiefern das Modell anderweitige Aspekte und Phasen des Produktlebenszyklus berücksichtigt.

4.4 Analyse der Produktmodelle

Die nachfolgende Tabelle 7 zeigt die Untersuchung der Produktmodelle hinsichtlich der erläuterten Kriterien. Die oberste Zeile zeigt die Analysekriterien und die nachfolgenden Zeilen jeweils drei Produktmodelle aus den Kategorien entsprechend der Einordnung, die in Kapitel 4.1 vorgenommen wurde.

			Rechnerunterstützung	Abstraktionsgrade	standardisiert, Nutzung von Standards	Interoperabilität	explizite Software notwendig	phasenübergreifend	sonstiges Aspekte
A	Funktionsstruktur		-	+	-	-	nein	-	-
	Bondgraph		-	-	-	-	nein	-	-
	SysML		o	+	ja	STEP, XMI	nein	o	Organisation
B	Modelica		+	+	ja	SysML, XML u.a.	nein	+	Physikalische Simulation
	STEP	AP 233	+	o	ja	SysML, STEP	nein	o	Organisation
		AP 239	+	o	ja	STEP	nein	o	Aufgaben- planung
		AP 242	+	o	ja	STEP	nein	o	Teilw. PDM u. Prozessplanung
	CPM2		+	+	ja	OAM nutzt STEP	nein	+	Sehr allgemein, OAM ist STEP-basiert
C	Anderl, Nattermann		+	+	-	k.A.	Eigenentwicklung	+	
	Lefevre et al.		+	+	SysML, STEP, Modelica	STEP u.a.	PLM-Plattform	+	Echtzeit-Kopplung der Simulationen
	Eigner et al.		+	k.A.	SysML, XMI	XML	Gängige PDM/CAx-Systeme	o	

Tabelle 7: Gegenüberstellung der Produktmodelle (+: stark ausgeprägt; o: mittelmäßig ausgeprägt; -: schwach ausgeprägt)

Tabelle 8 stellt dar, welche der Produktmodelle für die Phasen geeignet sind, die in Kapitel 3.3 identifiziert wurden. Die Tabelle dient dazu, einen Überblick zu bieten, in welchen Phasen der Produktentwicklung die Produktmodelle prinzipiell einsetzbar sind.

		Anforderungen	Funktionen	Logische Struktur	Verhalten	Physikalische Struktur, domänenspezifische Entwicklung	Integration von Teilsystemen	Integration des Gesamtsystems	VerifikationValidierung,	Produktion	Einsatz, Wartung, Entsorgung
Funktionsstruktur		+	+	o	-	-	-	-	-	-	-
Bond Graph		-	-	o	-	+	o	o	-	-	-
SysML		+	+	+	o	-	-	o	o	-	-
Modelica		-	-	o	+	+	+	+	+	-	-
STEP	AP 233	+	+	o	+	o	-	-	+	-	-
	AP 239	o	-	+	-	+	-	-	+	o	+
	AP 242	o	-	o	o	+	+	-	o	o	-
CPM2		+	+	+	+	o	-	-	o	-	-
Anderl, Nattermann		o		+	+	o	+	+	+	-	-
Lefevre et al.		+	+	+	+	+	-	-	-	-	-
Eigner et al.		+	+	+	+	+	-	-	-	-	-

Tabelle 8: Phasenabdeckung der Produktmodelle (+: explizit ausgeprägt; o: mittelmäßig ausgeprägt; -: schwach ausgeprägt bzw. keine Angaben)

Es ist fällt auf an der Tabelle, dass viele Modelle und Konzepte den Fokus auf die linke Hälfte des V-Modells legen. Einerseits liegt es darin begründet, dass in diesen Phasen die wichtigsten Entscheidungen getroffen werden, was von Graessler und Hentze mit dem Begriff des „Frontloading“ in der Produktentwicklung beschrieben wurde [GH15]. Andererseits liegt es auch daran begründet, dass die linke Hälfte hauptsächlich von Syntheseschritten geprägt ist.

Spätestens ab der Mitte des V-Modells (Entwicklung der physikalischen Struktur bzw. domänenspezifische Entwicklung) teilt sich die Entwicklung auf in die einzelnen Domänen, sodass ein paralleles Arbeiten beginnt und es wird Koordination sowie Informationsaustausch zwischen den Anwendungen notwendig wird. Laut Eigner et al.

setzen hier viele CAx-Prozesse erst hier an in der virtuellen Produktentwicklung [EGZ12]. Unterstützt werden kann diese Phase durch diverse Methoden und Werkzeuge zur Kopplung von CAE- und CAD-Systemen (z.B. durch eine sog. Middleware zwischen den CAx-Systemen), um die Entwicklungen in den einzelnen Domänen möglichst aufeinander abzustimmen und zu koordinieren. Dazu kann ein Produktmodell als zentrales Produktmodell fungieren oder es erfolgt ein direkter Austausch zwischen den CAx-Systemen (vgl. Abbildung 4). Diese Kopplung kann möglichst automatisiert geschehen, wie im Konzept von Anderl und Nattermann oder auf manuelle Art und Weise, beispielsweise in Form von Datei- oder Dokumentenaustausch (z.B. via E-Mail) und Absprachen zwischen den Fachbereichen und einzelnen Personen. Damit der Austausch zwischen den zeitlich parallel eingesetzten Systemen möglich ist, wird im „Interface“-Magazins der Firma Siemens in [Int12] eine Offenheit von Systemen gefordert, welche anhand von vier Schlüsselmerkmalen definiert wurde:

1. *Ein leistungsstarkes Interface für den dynamischen Datenaustausch mit anderen Anwendungen – entweder dateibasiert (XML) oder vorzugsweise servicebasiert (SOA)*
2. *Unterstützung offener Standards, wie JT, STEP AP210 und 212 u.ä.*
3. *Offenheit für Integration auf unterschiedlichen Ebenen – Client, Web-Client, Business Logic*
4. *Leistungstransparenz anhand reibungslos arbeitender Exchange Interface-Installationen*

Konkrete Möglichkeiten dafür sind Kopplungen zwischen CAx-Systemen, welche möglichst automatisiert dafür sorgen, dass die Informationen in den unterschiedlichen Domänen konsistent miteinander sind. Exemplarisch erwähnt seien hierfür das am Georgia Institute of Technology entwickelte Konzept des Constraint Modeling (publiziert u.a. in [CPS08] und [CBPS09]) oder eine vom Fraunhofer IFF entwickelten Datenbank als Middleware zwischen CAx-Systemen, vorgestellt in [BMKS13].

In der rechten Hälfte konnte nur eine geringe Unterstützung mittels Produktmodellen festgestellt werden. Dies lässt darauf schließen, dass in der rechten Hälfte des V-Modells ist die Unterstützung durch spezielle Produktmodelle entweder weniger notwendig ist oder zur Zeit weniger stark entwickelt ist. Jedoch sollten sich die Phasen der Systemintegration einfacher und effizienter gestalten, sofern die Integration in den vorangehenden Phasen in hinreichender Qualität geschehen ist, da im Idealfalle die parallel getätigten Entwicklungsschritte vollständig zueinander konsistent sind. So verweisen Anderl und Nattermann in ihrem Konzept darauf, das theoretisch keine explizi-

te Integrationsphase nötig ist, dies bereits automatisch während der parallelen Entwicklungsschritte umgesetzt wird. Prinzipiell können für diese Phasen der Qualitätskontrolle in einer Systemebene die jeweiligen Systhesemodelle der gleichen Systemebene verwendet werden. Für die Integration eines mechanischen Teilsystem können so beispielsweise die 3D-Modelle verwendet werden (um diese virtuell zusammenzufügen) oder für die Bewertung der gesamten Funktionalität des entwickelten Produktes kann zum Beispiel die Funktionsstruktur herangezogen werden.

5 Konzept zum integrierten Produktmodelleinsatz

Die Analyse der Produktmodelle hat unter anderem gezeigt, dass nur ein Produktmodell kaum in der Lage ist, die Phasen der Produktentwicklung vollständig zu adressieren. Aus diesem Grunde bietet sich die Entwicklung eines umfassenden Konzeptes auf Grundlage bereits existierender Produktmodelle und Integrationskonzepte, welche in Kapitel 4 untersucht wurden. Dazu wurde zunächst aufgezeigt, welche Phasen der untersuchten Vorgehensmodelle mit den Produktmodellen prinzipiell abgedeckt werden können. Im nächsten Unterkapitel wird das Integrationskonzept vorgestellt und im dritten und letzten Unterkapitel wird ein in Fachpublikationen vorgestellter Anwendungsfall beschrieben, ausgewertet sowie mit dem entwickelten Konzept verglichen.

5.1 Schnittstellen zwischen den Produktmodellen

Es existieren bereits Schnittstellen und Kompatibilitäten zwischen den einzelnen Produktmodellen, welche entweder bei der Entwicklung gezielt berücksichtigt und beabsichtigt waren (wie beispielsweise bei SysML und STEP oder bei OAM und STEP) oder im Nachhinein von anderer Seite her ermöglicht wurden (beispielsweise bei SysML und Modelica [Pa+10]). Schematisch dargestellt ist dies mittels der Prinzipskizze in Abbildung 32. Die dicken Linien zeigen die Schnittstellen zwischen den Produktmodellen (elliptische Objekte), während die Pfeile darstellen, welche Produktmodelle von den vorgestellten Integrationskonzepten (rechteckige Objekte) genutzt werden. So kommt in der Methode von Kleiner und Kramer die Funktionsstruktur sowie Modelica zum Einsatz und darüber hinaus wird eine Nutzung von SysML gefordert, was durch eine gestrichelte Linie kenntlich gemacht ist. Die Schnittstelle zwischen Bondgraphen und Modelica hingegen existiert derart, dass es Bibliotheken für die Erstellung von Bondgraphen gibt.

Diese Darstellung kann nach Bedarf mit weiteren Produktmodellen und Integrationskonzepten beliebig erweitert werden. Durch diese Netzwerk-ähnliche Darstellung ist relativ schnell ersichtlich, welche Modelltransformationen vorgenommen werden können zwischen Produktmodellen. Produktmodelle mit vielen Schnittstellen eignen sich als zentrale Modelle gemäß dem Prinzip der Modellkohärenz von Seiler (vgl. Abbildung 4).

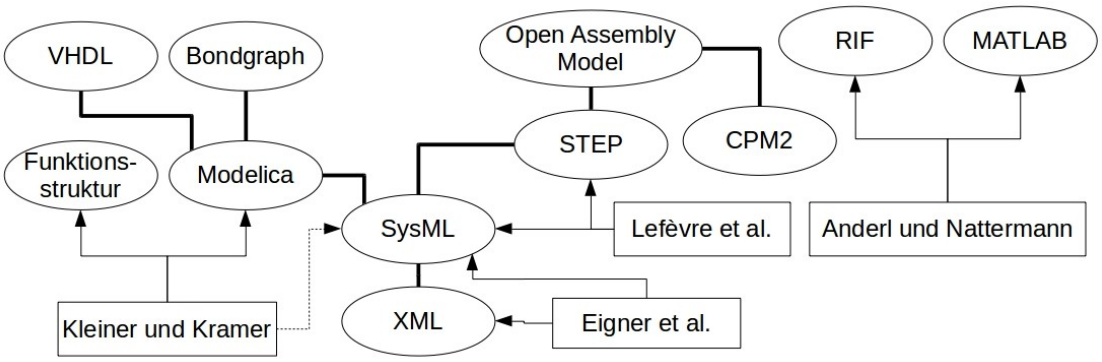

Abbildung 32: Schnittstellen zwischen den Produktmodellen

5.2 Kernmodelle als Integrationsbasis

Basierend auf den Erkenntnissen, die im vorherigen Unterkapitel geschildert wurden, wird hier ein Konzept zur Modellintegration vorgeschlagen entsprechend der modelltechnischen Integration des KOMFORCE-Referenzmodells (vgl. Abbildung 3) und der Modellintegration von Wenzel Seiler (vgl. Abbildung 4). Allerdings ist diese Modellintegration zweistufig gestaltet unter der Einbeziehung mehrerer Produktmodelle, welche hier als Kernmodelle bezeichnet werden sollen. Dargestellt ist das Konzept in Abbildung 33. Es umfasst die SysML als den „inneren Kern“, um den wiederum die Produktmodelle XML, STEP und Modelica sich befinden und damit den „äußeren Kern“ bilden. Die Doppelpfeile zwischen den Produktmodellen stellen die Schnittstellen zwischen diesen Produktmodellen dar.

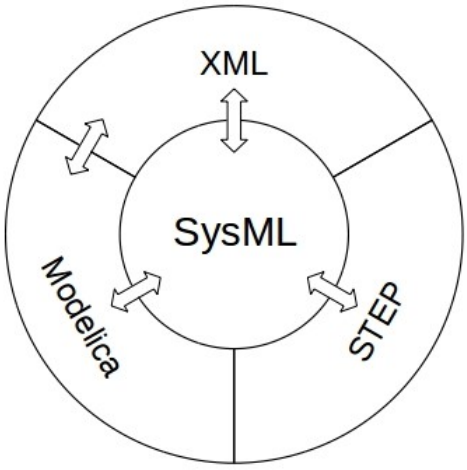

Abbildung 33: Kernmodelle als Integrationsbasis

Die Bezeichnung Kernmodelle rührt daher, dass in Abhängigkeit dieser Modelle weitere Produktmodelle eingesetzt werden können, sofern dies nötig ist. Die Vorteile dieser als Kernmodelle ausgewählten Produktmodelle lassen sich wie folgt zusammenfassen:

- Standardisiert, offen, nicht-proprietär
- Einsetzbar mittels diverser (v.a. auch freier) Software-Werkzeuge → Erhöhung des Kreises von anwendenden Personen und Organisationen, da je nach Präferenzen verschiedene Software-Werkzeuge eingesetzt werden können und bei freier Software Lizenzkosten vermieden werden können
- hohe Kompatibilität bzw. Interoperabilität zwischen den Modellen möglich
- große Abdeckung der Produktentwicklungsphasen möglich → wenig Bedarf für weitere Produktmodelle, wobei dies je nach Branche, Unternehmensgröße und anderen Rahmenbedingungen variieren kann

5.3 Vergleich mit einem Anwendungsfall

An der Hochschule für angewandte Wissenschaften in München wurde unter der Leitung von Vahid Saledi im Jahr 2015 der von Siemens PLM Software entwickelte Entwicklungsansatz namens „Systems-Driven Product Development“ (SDPD) untersucht. Dies wurde im Zuge eines Forschungsprojekt exemplarisch durchgeführt und untersucht. Vorgestellt wird dieses Forschungsprojekt unter anderem in [DE15], [Int15] und [Gr15].

Bei dem eingesetzten Vorgehensmodell handelt es sich um das V-Modell aus der VDI 2206. Die Entwicklung wurde in vier Ebenen unterteilt: Business Level, Functional Level, Technical Level sowie Component Level. Die nachfolgende Auflistung erläutert die Charakteristiken der jeweiligen Ebenen ausführlicher auf Grundlage von [DE15], [Int15] und [Gr15]:

Business Level:

- Definition von Anforderungen, Kundenbedürfnissen und Interessen der Stakeholder, festgehalten in einer Tabelle unter Nutzung von Microsoft Excel
- Strukturierung und Hierarchisierung der Anforderungen mit dem Teamcenter-Modul für Requirements Management
- Identifikation, Dokumentation und Kontrolle von Änderungen durch Anforderungsmanagement

Functional Level:

- Definieren von Systemstruktur und -verhalten des zu entwickelnden Systems
- Erstellung der Systemarchitektur
- Festlegen relevanter Parameter
- Erstellung mit in Teamcenter integriertem Microsoft Visio: Funktionen besitzen Ports, zwischen denen Objekte und Informationen übertragen werden können (keine Angaben zur Darstellungsform, Visio unterstützt aber etablierte und standardisierte Darstellungen wie UML u.a.)

Technical Level:

- Erstellung der technischen Systemarchitektur zur Beschreibung des logischen Verhaltens
- Nutzung von Matlab/Simulink mit Integration in Teamcenter zur Analyse des dynamischen Verhaltens der geometrisch dargestellten Komponenten
- Erstellung von Systemalternativen
- Bestimmung von technischen Lösungen
- Darstellung der Entscheidungspfade
- Darstellung des morphologischen Baukastens

Component Level:

- Beschreiben der physikalischen Eigenschaften des Systems
- Konstruieren der geometrischen 3D-Modelle
- Erstellung der 3D-Modell mit dem CAD-System „CATIA“ von Dassault Systèmes, für die Zukunft ist allerdings explizit NX (von Siemens PLM Software) geplant mit Schnittstellen zu ECAD-Systemen. Detaillierte Erläuterungen dazu blieben aus.

Über die diversen Level hinweg findet ein Abgleich von Anforderungsstrukturen mit Prozess- oder Produktstrukturen möglich mittels Teamcenter-Modul für Rückverfolgbarkeit.

Auswertung des Anwendungsfalls

Die prinzipielle Einteilung in die vier Level besitzt Ähnlichkeiten zu der RFLP-Unterteilung von Eigner et al. oder von Kleiner und Kramer. Eine Darstellung der Anforderungen in einem Excel-Dokument lässt eine modellhafte Darstellung ausschließen und auf eine eher klassische Anforderungsliste wie z.B. gemäß Pahl/Beitz schlie-

ßen. Eine Darstellung der Anforderungen mittels SysML wäre hier zweckmäßiger, denn in der anschließende funktionalen Ebene werden Modelle genutzt und mittels SysML hätte über diese bedienen Ebenen bereits eine Modellintegration erzielt werden können. Schließlich ermöglicht das an Teamcenter gekoppelte Visio explizit die Darstellung in UML und in anderen graphischen Darstellungen sowie darüber hinaus die Erweiterbarkeit um eigene Visio-Diagrammobjekte [Sie15].

Für die Simulation wurde zwar das sehr verbreitete Matlab/Simulink eingesetzt, welches auch in Teamcenter integriert werden kann. Vorteilhafter ist jedoch eine Nutzung Modelica, zumal Siemens PLM Software nach [www9] eine auf Modelica basierende Simulationssoftware namens „LMS Amesim Modelica Platform“ anbietet. Durch den Einsatz von Modelica wäre eine größere Freiheit in der Nutzung von Simulationsumgebungen sowie eine Interoperabilität zu Darstellungen wie der SysML gewährleistet. Für die Nutzung von CAD-Systemen von Fremdherstellern ist dies möglich, da laut [DE15] das CAD-System CATIA V5 verwendet wird, welches von Dassault Systèmes entwickelt wurde. Für das Konzept der Modellintegration ist grundsätzlich nur wichtig, dass das CAD-System die Nutzung von STEP ermöglicht. Meist ist dies in gängigen CAD-Systemen der Fall, jedoch nicht immer alle APs.

Auch der Abgleich über die unterschiedlichen Systemebenen zum zwecke der Rückverfolgbarkeit wird sich einfacher gestalten, wenn weniger unterschiedliche Produktmodelle dafür zum Einsatz kommen, wie beispielsweise der vorgeschlagene Einsatz von SysML und Modelica.

Dieser exemplarische Anwendungsfall zeigt, dass die dokumentenbasierte Produktentwicklung zumindest in Teilen durchaus noch etabliert ist, selbst bei führenden Anbietern von CAx- und PDM-Systemen. Eine zusammenfassende Gegenüberstellung Produktmodelle aus dem Anwendungsfall und dem Konzept zur Modellintegration aus Kapitel 5.2 ist in Tabelle 9 gezeigt.

Systemebene	**SDPD-Anwendungsfall**	**Modellintegration**
Anforderungen	Tabelle mit MS Excel	SysML
Funktion	Grafisches Modell mit MS Visio	SysML
Logik	Matlab/Simulink	Modelica
Physikalisch	CATIA	Modelica, STEP

Tabelle 9: Gegenüberstellung von Anwendungsfall und Modellintegration

Im Konzept zur Modellintegration wurde auf das Nennen von Software-Tools verzichtet, da einerseits dort eine Freiheit in der Auswahl selbiger herrscht und andererseits ein Vergleich derartiger Software-Werkzeuge nicht Gegenstand dieser Arbeit ist.

6 Fazit und Ausblick

Die Bedeutung von Produktmodellen ist für die heutige Produktentwicklung unerlässlich, wie der zunehmende Einsatz des MBSE deutlich macht. Herzorgangen ist dies ebenfalls aus den Zitaten in Kapitel 2 oder in Form einiger analysierter Vorgehensmodelle, in welchen der Modelleinsatz erhebliche Vorteile mit sich bringt. Es besteht allerdings noch Ausbaupotenzial in der Praxis, wie der untersuchte Anwendungsfall zeigt.

Die Entwicklung der Vorgehensmodelle ist hingegen weit fortgeschritten, denn das Vorgehensmodell der VDI 2206 wurde in den untersuchten Vorgehensmodellen der letzten zehn Jahre in seiner grundlegenden Form meist beibehalten und hinsichtlich Detaillierung oder Entwicklungsphasen ausgebaut. Speziell die Unterteilung der vier Systemebenen in Anforderungen, Funktionen, logische und physikalische Struktur, kurz als RFLP bezeichnet, wird hier als ein sich abzeichnender De-Facto-Standard bzw. „best practice“ für die mechatronische Produktentwicklung gesehen. Für Vorgehensmodelle sowie diese vier Sichtweisen wurde ein Konzept zur Modellintegration entwickelt, um die Produktentwicklung anhand von Produktmodellen sinnvoll zu unterstützen.

Da es sich bei dieser Arbeit um ein Konzept handelt, ergeben sich Möglichkeiten zur Konkretisierung und Erweiterung. Prinzipiell und ohne Anspruch auf Vollständigkeit sollen drei verschiedene Ansätze in Aussicht gestellt werden:

1. **Umsetzung des Konzeptes, Eignung von Software-Werkzeugen:** zur Prüfung der Stärken sowie der generelle Praxistauglichkeit dieses Konzeptes ist eine möglichst reale Umsetzung ein adäquater Ansatz. Ein konkrete zu untersuchender Aspekt wäre z.B. die Eignung verschiedener Software-Werkzeuge für das Konzept der Modellintegration.
2. **Untersuchung der Schnittstellen:** die Untersuchung der Schnittstellen zwischen den Kernmodellen wurden primär auf ihr grundsätzliches Vorhandensein untersucht. Hier bietet sich die Möglichkeit, diese Schnittstellen beispielsweise für bestimmte Branchen zu untersuchen.
3. **Ergänzung um weitere Modelle, Kopplungen von CAx-Systemen:** eine Untersuchung im Hinblick darauf, inwiefern sich verschiedene CAx-Systeme (bzw. deren Produktmodelle) sich mit den Kernmodellen koppeln lassen.

7 Anhang

Literaturverzeichnis

[AP233] International Organization for Standardization (2012): ISO 10303-233:2012(E), First edition 2012-11-15, Industrial automation systems and integration — Product data representation and exchange, Part 233: Application protocol: Systems engineering

[AP239] International Organization for Standardization (2012): ISO 10303-239:2012(E), Second edition 2012-11-15, Industrial automation systems and integration — Product data representation and exchange, Part 239: Application protocol: Product life cycle support

[AP239ed3] asd-ssg.org (2015): White Paper - ISO 10303 (STEP) AP 239 edition 3 - Application Protocol For Product Life Cycle Support (PLCS)

[AP242] International Organization for Standardization (2014): ISO 10303-242:2014(E): Industrial automation systems and integration — Product data representation and exchange, Part 242: Application protocol: Managed model-based 3D engineering

[AZ13] Albert Albers, Christian Zingel (2013): Challenges of Model-Based Systems Engineering: A Study towards Unified Term Understanding and the State of Usage of SysML; Auszug aus: Smart Product Engineering; herausgegeben von Michael Abramovici, Rainer Stark; Springer Heidelberg New York Dordrecht London

[Ba98] Helmut Balzert (1998): Lehrbuch der Software-Technik; Spektrum Akademischer Verlag GmbH, Heidelberg Berlin

[Ben05] Klaus Bender (2005): Embedded Systems - qualitätsorientierte Entwicklung; Springer Verlag Berlin

[Bia15] Alain Biahmou (2015): Systems Engineering; Auszug aus: Concurrent Engineering in the 21st Century - Foundations, Developments and Challenges; herausgegeben von Josip Stjepandić, Nel Wognum, Wim J.C. Verhagen; Springer Cham Heidelberg New York Dordrecht London

[BMKS13] Eric Bayrhammer, Sebastian Möser, Matthias Kennel, Ulrich Schmucker (2013): Durchgängige Produktentwicklung im Sondermaschinenbau für kleine und mittlere Unternehmen; 11. Magdeburger Maschinenbau-Tage 25.-26.09.2013

[BO11] Torsten Blochwitz, Martin Otter (2011): The Functional Mockup Interface for Tool independent Exchange of Simulation Models (Presentation); URL: https://trac.fmi-standard.org/export/700/branches/public/docs/Modelica2011/The_Functional_Mockup_Interface.pdf (letzter Zugriff: 05.01.2016)

[BSI13] 4Soft GmbH, akquinet AG und MID GmbH (2013): V-Modell XT Bund -- Teil 1: Grundlagen des V-ModellsVersion 1.1 (Basis V-Modell XT 1.4);

[BTRE13] Matthieu Bricogne, Nadège Troussier, Louis Rivest, Benoît Eynard (2013): Agile Design Methods for Mechatronics System Integration; Auszug aus: Product Lifecycle Management for Society - 10th IFIP WG 5.1 International Conference, PLM 2013, Nantes, France, July 6-10, 2013, Proceedings; herausgegeben von Alain

Bernard, Louis Rivest, Debasish Dutta; Springer Heidelberg New York Dordrecht London

[CBPS09] Kenway Chen, Jonathan Bankston, Jitesh H. Panchal and Dirk Schaefer (2009): A Framework for Integrated Design of Mechatronic Systems; Auszug aus: Collaborative Design and Planning for Digital Manufacturing; herausgegeben von Lihui Wang, Andrew Y.C. Nee; Springer-Verlag London Limited

[CL96] L.W. Chan, T.P. Leung (1996): Spiral Design Model for Consumer Mechatronic Products; Mechatronics Volume 6, Issue 1

[CPS08] Kenway Chen, Jitesh Panchal, Dirk Schaefer (2008): An integrated approach to design mechatronic systems cross-disciplinary constraint modelling;

[DBPH07] Violeta Damjanović, Wernher Behrendt, Manuela Plössnig, Merlin Holzapfel (2007): Developing Ontologies for Collaborative Engineering in Mechatronics; Auszug aus: The Semantic Web: Research and Applications, 4th European Semantic Web Conference, ESWC 2007, Innsbruck, Austria, June 3-7, 2007. Proceedings; herausgegeben von Enrico Franconi, Michael Kifer, Wolfgang May ; Springer-Verlag Berlin Heidelberg

[DE15] Validierung eines ins PLM-System integrierten Ansatzes für das Systems Engineering: http://www.digital-engineering-magazin.de/fachartikel/validierung-eines-ins-plm-system-integrierten-ansatzes-fuer-das-systems-engineering (letzter Zugriff: 29.12.2015)

[Doh14] Fabio Dohr (2014): Methodik für die simulationsbasierte Entwicklung mechatronischer Systeme; Dissertation an der Universität des Saarlandes

[EGZ12] Martin Eigner, Torsten Gilz, Radoslav Zafirov (2012): Interdisziplinäre Produktentwicklung - Modellbasiertes Systems Engineering; URL: http://www.plmportal.org/de/forschung-detail/interdisziplinaere-produktentwicklung-modellbasiertes-systems-engineering.html (letzter Zugrff: 03.01.2016)

[Eig+14] Martin Eigner, Thomas Dickopf, Hristo Apostolov, Patrick Schaefer,Karl-Gerhard Faißt, Alexander Keßler (2014): System Lifecycle Management - Initial Approach for a Sustainable Product Development Process Based on Methods of Model Based Systems Engineering; Auszug aus: Product Lifecycle Management for a Global Market - 11th IFIP WG 5.1 International Conference, PLM 2014; herausgegeben von Shuichi Fukuda, Alain Bernard, Balan Gurumoorthy, Abdelaziz Bouras;

[Eig13] Martin Eigner (2013): Modellbasierte Virtuelle Produktentwicklung auf einer Plattform für System Lifecycle Management; Auszug aus: ; herausgegeben von Ulrich Sendler; Springer-Verlag Berlin Heidelberg

[ERZ14] Martin Eigner, Daniil Roubanov, Radoslav Zafirov (2014): Modellbasierte Virtuelle Produktentwicklung; Springer-Verlag Berlin Heidelberg

[ES09] Martin Eigner, Ralph Stelzer (2009): Product Lifecycle Management - Ein Leitfaden für Product Developmentund Life Cycle Management (2., neu bearbeitete Auflage); Springer-Verlag Berlin Heidelberg

[FFBS08] Steven J. Fenves, Sebti Foufou, Conrad Bock, Ram D. Sriram (2008): CPM2: A Core Model for Product Data;

[FMS12] Friedenthal, S., Moore, A., Steiner, R. (2012): A Practical Guide to SysML. The Systems Modeling Language;

[Gau08] Jürgen Gausemeier (2008): Domänenübergreifende Vorgehensmodelle;

[GF06] Jürgen Gausemeier, Klaus Feldmann (2006): Integrative Entwicklung räumlicher elektronischer Baugruppen; Carl Hanser Verlag München

[GFDK08] Jürgen Gausemeier, Ursula Frank, Jörg Donoth, Sascha Kahl (): Spezifikationstechnik zur Beschreibung der Prinziplösung selbstoptimierender Systeme des Maschinenbaus; Konstruktion Ausgabe 7/8-2008

[GH15] Iris Graessler, Julian Hentze (2015): Enriching mechatronic V-Model by Aspects of Systems Engineering; 7th ECCOMAS Thematic Conference on Smart Structures and Materials

[Gr15] Systems Engineering: Von der Forschung zur Praxis: http://www.plm-it-business.de/plm/systems-engineering--von-der-forschung-zur-praxis.htm (letzter Zugriff: 29.12.2015)

[Hyb09] Hybertson, D.W. (2009): Model-Oriented Systems Engineering Science. A Unifying Framework forTraditional and Complex Systems; CRC Press, Boca Raton

[Int12] Interface - Das Magazin für Product Lifecycle Management: Herstelleralbtraum: Ein Produkt mit Funktionsfehlern - Produktvielfalt und internationale Zusammenarbeit beherrschen; Ausgabe 1-2012, Siemens Industry Software GmbH & Co. KG

[Int15] Interface - Das Magazin für Product Lifecycle Management: Systems Engineering verändert Vorgehensweisenbei der Produktentstehung; Ausgabe 1-2015, Siemens Industry Software GmbH & Co. KG

[Iser08] Rolf Isermann (2008): Mechatronische Systeme - Grundlagen - 2. vollständig neu bearbeitete Auflage; Springer-Verlag Berlin Heidelberg

[ISO12] International Standardization Organization (2012): ISO 14306:2012(E), First Edition: Industrial automation systems and integration — JT file format specification for 3D visualization

[Ja06] Sebastian Jansen (2006): Eine Methodik zur modellbasierten Partitionierung mechatronischer Systeme; Dissertation an der Ruhr-Universität Bochum

[KBSS97] E. Kallenbach, O. Birli, E. Saffert, C. Schäffel (1997): Zur Gestaltung integrierter mechatronischer Produkte; VDI Berichte Ausgabe 1315

[KK12] Sven Kleiner, Christoph Kramer (2012): Entwerfen und Entwickeln mit Systems Engineering auf Basis des RFLP-Ansatzes in V6; Whitepaper 05/2012

[KK13] Sven Kleiner, Christoph Kramer (2013): Model Based Design with Systems Engineering Based on RFLP Using V6; Auszug aus: Smart Product Engineering; herausgegeben von M. Abramovici and R. Stark; Springer-Verlag Berlin Heidelberg

[LCBME14] Jérémy Lefèvre, Sébastien Charles, Magali Bosch-Mauchand, Benoît Eynard, Éric Padiolleau (2014): Multidisciplinary modelling and simulation for mechatronic design; Journal of Design Research Volume 12, Issue 1-2

[Lin09] Udo Lindemann (2009): Methodische Entwicklung technischer Produkte; Springer-Verlag Berlin-Heidelberg

[LKS00] J. Lückel, T. Koch, J. Schmitz (2000): Mechatronik als integrative Basis für innovative Produkte; VDI Berichte Ausgabe 1533

[NA10] Roland Nattermann, Reiner Anderl (2010): Approach for a Data-Management-System and a Proceeding-Model for the Development of Adaptronic Systems; Proceedings for the ASME International Mechanical Engineering Congress & Exposition, Vancouver, British Columbia, Canada November 12-18, 2010

[NA11] Roland Nattermann, Reiner Anderl (2011): Simulation Data Management Approach for Developing Adaptronic Systems - The W-Modell Methodology; World Academy of Science, Engineering and Technology Vol. 5 2011-03-28

[NA13a] Roland Nattermann, Reiner Anderl (2013): The W-Model - Using Systems Engineering for Adaptronics; Procedia Computer Science

[NA13b] Roland Nattermann, Reiner Anderl (2013): Meta-data-Model für the Development of Adaptronic Systems; Auszug aus: Smart Product Engineering - Proceedings of the 23rd CIRP DesignConference, Bochum, Germany,March 11th–13th, 2013; herausgegeben von Michael Abramovici and Rainer Stark; Springer-Verlag Berlin Heidelberg

[Oe+12] Felix Oestersötebier, Viktor Just, Ansgar Trächtler, Frank Bauer, Stefan Dziwok (2012): Model-based Design of mechatronic Systems by means of smeantic web ontologies and reusable solution elements; Proceedings of the ASME 2012 11th Biennial Conference on Engineering Systems Design and Analysis

[OW13] Martin Otter, Dietmar Winkler (2013): Modelica Overview; URL: https://www.modelica.org/education/educational-material/lecture-material/english/ModelicaOverview.pdf (letzter Zugriff 08.12.2015)

[Pa+10] Chris Paredis, Yves Bernard, Roger Burkhart, Hans-Peter de Koning, Sanford Friedenthal, Peter Fritzson, Nicolas Rouquette, Wladimir Schamai (2010): An overview of the SysML-Modelica transformation specification

[PBFG13] Jörg Feldhusen, Karl-Heinrich Grote (2013): Pahl/Beitz Konstruktionslehre - Methoden und Anwendung erfolgreicher Produktentwicklung; Springer-Verlag Berlin Heidelberg

[PF03] Adrian Pop, Peter Fritzson (2003): ModelicaXML: A Modelica XML Representation with Applications; Proceedings of the 3 International Modelica Conference, Linköping, November 3-4, 2003

[Ro12] Werner Roddeck (2012): Einführung in die Mechatronik, 4., überarbeitete Auflage; Springer Vieweg Verlag

[SB11] Dieter Spath, Stefanie Bunzel (2011): Ontology-based Technology Model for the Use in the Early Stage of Product Development;

[Sch99] Achim Schön (1999): Konzept und Architektur eines Assistenzsystems für die mechatronische Produktentwicklung; Dissertation an der Universität Erlangen-Nürnberg

[Sei85] Wenzel Seiler (1985): Technische Modellierungs- und Kommunikationsverfahren für das Konzipieren und Gestalten auf der Basis der Modell-Integration; Dissertation an der Universität Karlsruhe

[SI08] Antti Saaksvuori, Anselmi Immonen (2008): Product Lifecycle Management - Third Edition; Springer-Verlag Berlin Heidelberg

[Sie09] Georg Siebes (2009): AP233 - An Information Model for Systems Engineering; 11th NASA/ESA Workshop on Product Data Exchange 2009

[Sie15] Systems Engineering Fact Sheet: A whole system approach to improving product development: http://www.plm.automation.siemens.com/de_de/products/teamcenter/systems-engineering-software/index.shtml#lightview%26uri=tcm:73-4887%26title=Systems-Engineering-Fact-Sheet-5244%26docType=pdf (letzter Zugriff: 12.01.2016)

[SK98] 1998 (Devdas Shetty, Richard A. Kolk): Mechatronics System Design; Cengage Learning

[Sta73] Stachowiak, H. (1973): Allgemeine Modelltheorie; Springer-Verlag, Wien

[SV92] V. Salminen, A. Verho (1992): Systematic and innovative design of a mechatronic product; Mechatronics Volume 2, Issue 3

[Tu06] Tania Tudorache (2006): Employing Ontologies for an Improved Development Process in Collaborative Engineering; Dissertation an der Technischen Universität Berlin

[USDOT07] United States Department of Transportation - Federal Highway Administration (2007): Systems Engineering for Intelligent Transportation Systems;

[VDI04] Verein Deutscher Ingenieure (2004): VDI-Richtlinie 2206: Entwicklungsmethodik für mechatronische Systeme

[We09] Tim Weilkiens (2009): Systems Engineering mit SysML/UML - Modellierung, Analyse, Design - 2., aktualisierte u. erweiterte Auflage; dpunkt.verlag

[We13] OMG SysML 1.3 Notationsübersicht: http://model-based-systems-engineering.com/wp-content/uploads/2012/03/sysmod-sysml-1.3-notations%C3%BCbersicht-weilkiens.pdf (letzter Zugriff: 15.01.2016)

[We14] Tim Weilkiens (2014): Das digitale Produktmodell – Die Hechenberger Thesen; URL: http://www.plmportal.org/de/oose-innovative-informatik.html (letzter Zugriff: 07.01.2016)

[Weit06] Christian Bartelt, Otto Bauer, Gerd Beneken, Klaus Bergner, Ulrich Birowicz, Thomas Bliß, Nils Cordes, David Cruz, Patrick Dohrmann, Jan Friedrich, Michael Gnatz, Ulrike Hammerschall, Istvan Hidvegi-Barstorfer, Helmut Hummel, Dirk Israel, Thomas Klingenberg, Klaus Klugseder, Inga Küffer, Marco Kuhrmann, Michael Kranz, Wolfgang Kranz, Hans-Jürgen Meinhardt, Michael Meisinger, Sabine Mittrach, Hans-Joachim Neußer, Dirk Niebuhr, Klaus Plögert, Doris Rauh, Andreas Rausch, Thomas Rittel, Winfried Rösch, Erik Saas, Joachim Schramm, Marc Sihling, Thomas Ternité, Sascha Vogel, Marion Wittmann (2006): V-Modell XT -- Das deutsche Referenzmodell für Systementwicklungsprojekte, Version: 2.0

[www1] Das W-Modell - Systems Engineering in der Entwicklung aktiver Systeme: http://www.plmportal.org/files/plm/img/profile/Forschung_Lehre/DiK/Systems%20Engineering%20mit%20dem%20W-Modell.jpg (letzter Zugriff: 19.10.2015)

[www10] ASD Strategic Standardization Group Website: http://www.asd-ssg.org/step-ap239-ed3-wp (letzter Zugriff: 13.01.2016)

[www11] Functional Mock-up Interface Website: https://www.fmi-standard.org/ (letzter Zugriff: 13.01.2016)

[www12] omgwiki.org - List of Methodologies and Methods: http://www.omgwiki.org/MBSE/doku.php?id=mbse:methodology#List%20of%20MBSE%20Methodologies (letzter Zugriff: 17.01.2016)

[www13] PRC Format Specification: http://help.adobe.com/livedocs/acrobat_sdk/9/Acrobat9_HTMLHelp/API_References/PRCReference/PRC_Format_Specification/ (letzter Zugriff: 20.01.2016)

[www2] STEP AP242 Project Website: http://www.ap242.org/ (letzter Zugriff: 12.01.2016)

[www3] SysML/AP233 Data Overlaps: http://www.omgwiki.org/OMGSysML/doku.php?id=sysml-ap233:mapping_between_sysml_and_ap233 (letzter Zugriff: 16.11.2015)

[www4] Modelica Website: https://www.modelica.org/ (letzter Zugriff: 19.11.2015)

[www5] Object Management Group Website: http://www.omgsysml.org (letzter Zugriff: 23.11.2015)

[www6] STEP AP 209 Website: http://www.ap209.org (letzter Zugriff: 30.11.2015)

[www7] Validierung eines ins PLM-System integrierten Ansatzes für das Systems Engineering: http://www.digital-engineering-magazin.de/fachartikel/validierung-eines-ins-plm-system-integrierten-ansatzes-fuer-das-systems-engineering (letzter Zugriff: 21.12.2015)

[www8] Was ist Systems Engineering?: http://www.plmportal.org/de/grundlegendes-aus-sicht-des-plmportals.html (letzter Zugriff: 21.12.2015)

[www9] LMS Amesim Modelica Platform: http://www.plm.automation.siemens.com/en_us/products/lms/imagine-lab/amesim/platform/modelica.shtml (letzter Zugriff: 11.01.2016)

[ZBDE14] Chen Zheng, Matthieu Bricogne, Julien Le Duigou, Benoît Eynard (2014): Survey on mechatronic engineering: A focus on design methods and product models; 24th CIRP Design Conference

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Schriften entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden.

Magdeburg, den 21. Januar 2016

_____

Szur, Marvin Konstantin