Methodik für die simulationsbasierte Entwicklung mechatronischer Systeme

Dissertation

zur Erlangung des Grades

des Doktors der Ingenieurwissenschaften

der Naturwissenschaftlich-Technischen Fakultät II

– Physik und Mechatronik –

der Universität des Saarlandes

von

Fabio Dohr

Saarbrücken

2014

Tag des Kolloquiums:	16.12.2014
Dekan:	Univ.-Prof. Dr. Georg Frey
Mitglieder des Prüfungsausschusses:	Univ.-Prof. Dr. Michael Vielhaber
	Univ.-Prof. Dr. Martin Eigner
	Univ.-Prof. Dr. Matthias Nienhaus
	Dr.-Ing. Lutwin Klein

Kurzfassung

Die Entwicklung technischer Produkte steht im Spannungsfeld zwischen Kundenansprüchen (insbesondere Produktqualität und -funktionalität) und Marktansprüchen (insbesondere Entwicklungszeit und -kosten). Hierbei stellen mechatronische Produkte aufgrund ihrer Komplexität besondere Ansprüche an den Entwicklungsprozess. Als ein Mittel zur Beherrschung dieser Problematik werden in der Produktentwicklung bereits seit längerem Simulationen angewendet. Ein konsistenter Einsatz im Sinne simulationsbasierter Entwicklung ist hierbei jedoch nicht zu erkennen.

Ziel dieser Arbeit ist die Herleitung einer Methodik für die simulationsbasierte Entwicklung mechatronischer Systeme. Zu diesem Zweck erfolgt zunächst eine ausführliche Analyse existierender Modellierungs- und Simulationstechniken sowie von Entwicklungsmethodiken in Bezug auf die Umsetzung simulationsbasierter Entwicklung. Diese Analyse erfolgt sowohl domänenspezifisch als auch domänenübergreifend und zeigt, dass zwischen technologischen und entwicklungsmethodischen Aspekten ein Missverhältnis besteht, aus dem sich der Handlungsbedarf dieser Arbeit ableitet.

Entsprechend wird im Hauptteil der Arbeit ein Gesamtrahmen für eine Entwicklungsmethodik definiert und eine geeignete Methodik systematisch hergeleitet. Hierbei stehen die Prozessbeschreibung sowie die Unterstützung durch Methoden im Vordergrund. Abschließend wird die entwickelte Methodik an einem mechatronischen Entwicklungsprojekt validiert und veranschaulicht.

Abstract

Product development has to deal with the conflict of satisfying both customer requirements – mainly product quality and functionality – and market requirements – mainly development time and costs. Due to their complexity mechatronic products in particular put high requirements on the development process. Simulation has long been used in product development in order to cope with these challenges. Nevertheless a consistent integration in terms of simulation-based design has not yet been achieved.

This work aims to derive a methodology for simulation-based design of mechatronic systems. Hence a comprehensive analysis is conducted regarding modeling and simulation techniques as well as development methodologies concerning the implementation of simulation-based design. This analysis is performed both domain-specific and across domains. It identifies a gap between technological and methodological aspects which is the rationale for this work.

Consequently, in the main part of this work an overall framework for the development methodology is defined. Based on this framework the structured development of an appropriate methodology is described. The main focus lies on the description of process aspects as well as on methods intended to provide guidance for designers. Finally the developed methodology is validated on and exemplified through a mechatronic design project.

Vorwort

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Konstruktionstechnik der Universität des Saarlandes.

Meinen besonderen Dank möchte ich dem Lehrstuhlinhaber und Betreuer dieser Arbeit, Herrn Professor Michael Vielhaber, aussprechen. Das entgegengebrachte Vertrauen sowie die gebotenen Freiheiten beim Aufbau dieses Forschungsthemas sowie bei allen anderen Tätigkeiten am Lehrstuhl – sowohl fachlich als auch außerfachlich – sind nicht selbstverständlich und haben die Ergebnisse dieser Arbeit sowie das hervorragende Arbeitsklima am Lehrstuhl maßgeblich geprägt. Die inhaltlichen Diskussionen sowie die von ihm eröffneten Möglichkeiten zur Teilnahme an zahlreichen wissenschaftlichen Konferenzen und Veranstaltungen haben wichtige Impulse gegeben, nicht nur für diese Arbeit.

Ebenso möchte ich Herrn Professor Martin Eigner für das Interesse an meiner Arbeit und die inhaltlichen Diskussionen danken.

Allen meinen Kolleginnen und Kollegen am Lehrstuhl möchte ich für das außergewöhnliche Arbeitsklima sowohl in fachlichen als auch in nichtfachlichen Bereichen danken. Über die Arbeit am Lehrstuhl hinaus sind dabei viele freundschaftliche Verhältnisse entstanden. Der gemeinsame Aufbau des Lehrstuhls war in allen Bereichen eine spannende, herausfordernde und vor allem spaßbringende Aufgabe.

Den Studenten, die mich durch ihre Tätigkeiten als studentische Hilfskräfte sowie in studentischen Arbeiten während des Entstehens dieser Arbeit unterstützt haben, bin ich ebenso zu großem Dank verpflichtet. Eine Auflistung dieser Vielzahl von Personen wäre an dieser Stelle zu umfangreich.

Letztendlich verdanke ich die Möglichkeit, diese Arbeit zu verfassen vor allem meinen Eltern und meiner Familie, die mich stets gefördert und mir ein entsprechendes Umfeld geboten haben. Hierbei möchte ich meinem Bruder Frederik nochmals für die Korrektur der Arbeit sowie für die intensive Unterstützung bei der Implementierung der Softwareanteile danken.

I Inhaltsverzeichnis

II Abbildungsverzeichnis

III Tabellenverzeichnis

IV Abkürzungsverzeichnis

ABS	Antiblockiersystem
AMS	Analysemeilenstein
BDD	Blockdefinitionsdiagramm in SysML
BIP	Bruttoinlandsprodukt
CAD	Computer-aided Design
CAE	Computer-aided Engineering
CASE	Computer-aided Software Engineering
CFD	Computational Fluid Dynamics
CPM	Characteristics-Properties-Modeling
DIN	Deutsches Institut für Normung
DMM	Domain Mapping Matrix
DOE	Design of Experiments
DRM	Design Research Methodology
DS-I	Descriptive Study I
DS-II	Descriptive Study II
DSM	Design Structure Matrix
ECAD	Electronic CAD
ESP	Elektronisches Stabilitätsprogramm
FEM	Finite-Elemente-Methode

GUI	Graphical User Interface
HDL	Hardware Description Language
HIL	Hardware-in-the-Loop
IBD	Internes Blockdiagramm in SysML
IEEE	Institute of Electrical and Electronics Engineers
INCOSE	International Council on Systems Engineering
ISO	International Organization for Standardization
LHS	Latin Hypercube Sampling
MBSE	Model-based Systems Engineering
MCS	Monte-Carlo-Simulation
MIL	Model-in-the-Loop
MKS	Mehrkörpersystem
MVM	Münchener Vorgehensmodell
OMG	Object Management Group
PDM	Produktdatenmanagement
PLM	Produktlebenszyklusmanagement
PS	Prescriptive Study
RFLP	Requirements, Functional, Logical, Physical
SE	Systems Engineering
SIL	Software-in-the-Loop
SPICE	Simulation Program with Integrated Circuit Emphasis

SysML	Systems Modeling Language
TOTE	Test-Operate-Test-Exit
UML	Unified Modeling Language
VDI	Verein Deutscher Ingenieure
VHDL	Very High Speed Integrated Circuit Hardware Description Language
VHDL-AMS	VHDL Analog and Mixed-signal
XIL	X-in-the-Loop

V Symbolverzeichnis

B	Verhalten (FBS-Framework)
Be	Gewünschtes Verhalten (FBS-Framework)
Bs	Tatsächliches Verhalten (FBS-Framework)
C_i	Merkmal
F	Funktion (FBS-Framework)
k	Proportionalitätsfaktor des Konfidenzfaktors
MC_j	Modellierungsbedingungen
P_j	Eigenschaft
PR_j	Geforderte Eigenschaft
R_j	Zusammenhang zwischen C_i und P_j
S	Lösungsstruktur (FBS-Framework)
s	Level der Systemhierarchie
SC_j	Simulationsbedingungen
ST_k	Simulationstechnik
ΔP_j	Grenzwert der Eigenschaft
ε_j	Konfidenzfaktor
σ	Standardabweichung

1 Einleitung und Motivation

Die Produktentwicklung als Grundlage des produzierenden Gewerbes stellt einen der bedeutendsten Wirtschaftsfaktoren dar. Dies spiegelt sich auch in den aktuellen Wirtschaftsdaten wider: So lag in Deutschland im Jahr 2013 der Anteil des produzierenden Gewerbes bei 30,2 Prozent des Bruttoinlandsproduktes (BIP) [WWW1], wobei die Exportquote über 50 Prozent des BIP betrug [WWW2]. In Zukunft wird diese Bedeutung weiter zunehmen, um durch innovative und hochqualitative Produkte den Stand der Wirtschaft im internationalen Umfeld langfristig zu gewährleisten. Insbesondere mechatronische Produkte, die auf dem synergetischen Zusammenwirken der klassischen Ingenieurdisziplinen Maschinenbau, Elektronik und Software basieren, bieten vielversprechende Möglichkeiten für Produktinnovationen. Entsprechend gewinnt die Mechatronik immer größere Bedeutung in der Produktentwicklung.

Durch die Globalisierung und den damit steigenden Konkurrenzdruck sehen sich Firmen immer größeren Herausforderungen ausgesetzt. Dabei lassen sich nach [Aber08] drei Hauptaspekte identifizieren: verringerte Entwicklungszeiten, Reduzierung der Entwicklungskosten sowie Forderungen des Marktes nach höherer Produktqualität und -funktionalität.

Klassische Vorgehensweisen in der Produktentwicklung können diesen Anforderungen nicht mehr gerecht werden. So benötigt die Herstellung eines physischen Prototyps je nach Komplexität bis zu 99 Tage und kostet bis zu 1,2 Millionen US-Dollar [Aber06]. Der Einsatz von Simulation hingegen kann Zeit und Kosten für Eigenschaftsabsicherungen signifikant reduzieren und gleichzeitig die Produktqualität erhöhen, da mehr Absicherungszyklen ermöglicht werden und somit die Systemreife erhöht wird. Darüber hinaus werden auch Innovationen gefördert, da durch virtuelle Techniken, darunter Simulation, die natürliche Tendenz von Entwicklern zur Bevorzugung bekannter Lösungen verringert wird [BeSZ05]. Ein großer Vorteil simulationsbasierter Entwicklung besteht darin, dass Simulationen bereits sehr früh im Entwicklungsprozess eingesetzt werden können [PDSK01]. Somit ergibt sich zwar, wie in Abbildung 1.1 dargestellt, ein höherer Entwicklungsaufwand in frühen Phasen, jedoch wird durch das größere Systemwissen der Aufwand in späten Phasen deutlich reduziert [AlNo03]. Dieser steigt in klassischen Entwicklungsprozessen durch den späten und intensiven Einsatz physischer Prototypen deutlich an. Somit unterstützt die simulationsbasierte Entwicklung auch das Konzept des Front-Loadings.

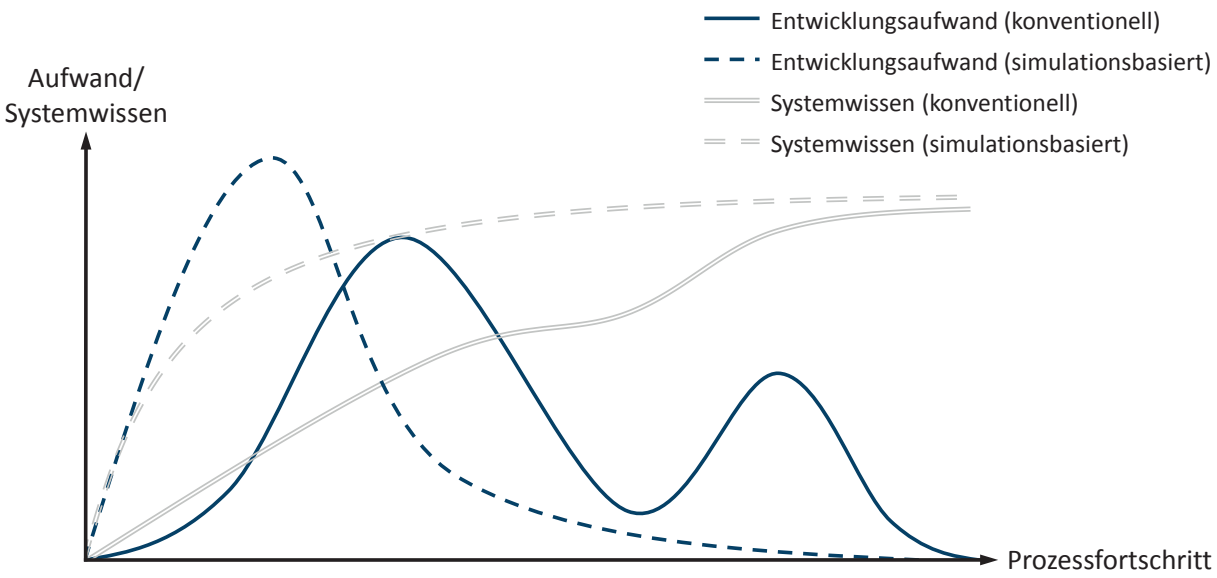


Abbildung 1.1: Entwicklungsaufwand und Systemwissen [DoVi12a], nach [AlNo03]

Diese Vorteile von Simulationen zeigen sich auch in der industriellen Anwendung. So wurden etwa bei der Audi AG bereits 2008 mehrere Hundert Kriterien durch Simulationen abgesichert und auf deren Basis Entscheidungen getroffen [SeRa08]. Karlberg et al. [KLSL13] verweisen zudem auf eine Industriestudie, in der durch den Einsatz von Simulationen eine Reduktion der Entwicklungszeit um 75 Prozent festgestellt wurde.

Dennoch existieren zahlreiche Hindernisse bei der durchgängigen Integration der Simulation in die Produktentwicklung. Seiffert und Rainer [SeRa08] geben an, dass das Wissen über Simulationsmethoden noch wenig verbreitet und der Simulationseinsatz stark segmentiert ist. Das fehlende Wissen bei Entwicklern bezüglich des Potentials der Simulation führt oftmals dazu, dass geeignete Verfahren zu spät im Entwicklungsprozess zum Einsatz kommen [KrFG07]. Bezüglich der Prozessintegration geben beispielsweise Syal und Suyam-Welakwe [SySu10] an, dass im Entwicklungsprozess der Daimler AG keine Festlegung existiert, zu welchem Zeitpunkt virtuelle Absicherungen erfolgen sollen. Das volle Potential simulationsbasierter Entwicklung kann somit nicht ausgeschöpft werden.

1.1 Ziele der Arbeit

Forschungsaktivitäten im Bereich der simulationsbasierten Entwicklung konzentrieren sich aktuell insbesondere auf die Entwicklung und Verbesserung neuer Techniken zur Modellierung und Simulation. Weniger von Interesse waren bisher die methodischen und organisatorischen Aspekte [KAPM11]. Diese Lücke soll im Rahmen dieser Arbeit adressiert werden: Durch die Entwicklung einer Methodik für die simulationsbasierte Entwicklung mechatronischer Systeme

soll die durchgängige Integration der Simulation bereits von frühen Entwicklungsphasen an ermöglicht werden. Hierbei sollen keine neuen Modellierungs- oder Simulationstechniken entwickelt werden. Vielmehr soll die Methodik auf vorhandenen Techniken aufbauen und diese sinnvoll im Entwicklungsprozess integrieren. Entsprechend können die folgenden initialen Forschungsfragen aufgestellt werden:

F1. Welche Möglichkeiten und Ansätze existieren in Bezug auf die simulationsbasierte Entwicklung mechatronischer Produkte?
F2. Wie kann eine Methodik die durchgängige Integration von Simulation in den Entwicklungsprozess mechatronischer Systeme fördern?
F3. Wie muss der Entwicklungsprozess selbst geändert werden, um die simulationsbasierte Entwicklung zu ermöglichen?
F4. Welche Randbedingungen müssen hierbei berücksichtigt werden?

In Abbildung 1.2 ist der Lebenszyklus eines Produktes nach Gamweger et al. [GJSS09] dargestellt. Die in dieser Arbeit zu entwickelnde Methodik soll darin den hervorgehobenen Bereich adressieren. Demnach wird der Ausschnitt der Produktentstehung von der Konzeptentwicklung über die Serienentwicklung bis hin zum Übergang zur Vorserie betrachtet, alle vor- und nachgelagerten Phasen werden als gegeben angesehen. Die Verbindung zu diesen Phasen wird über den gemeinsamen Informationsaustausch hergestellt. Zwar verspricht der Einsatz von Simulation etwa in der Zusammenarbeit von Produktentwicklung und Fertigung große Verbesserungspotentiale, jedoch wird dieser Aspekt im Rahmen der vorliegenden Arbeit bewusst ausgelassen, da zunächst nur die Produktentwicklung selbst im Fokus stehen soll. Die entwickelte Methodik kann jedoch in Zukunft als Grundlage dienen, um diese Zusammenarbeit im Produktentstehungsprozess zu festigen.

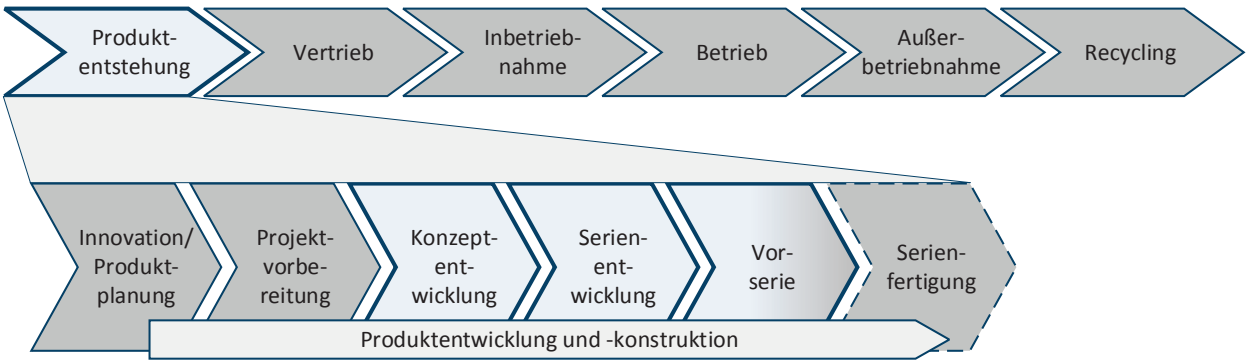


Abbildung 1.2: Einordnung der Entwicklungsmethodik im Produktlebenszyklus, nach [GJSS09]

1.2 Vorgehensweise und Struktur der Arbeit

Die Vorgehensweise in dieser Arbeit richtet sich nach der von Blessing und Chakrabarti vorgeschlagenen Methodik „Design Research Methodolgy (DRM)“ [BlCh09]. Demnach gliedert sich das Vorgehen in drei Hauptteile: Die „Descriptive Study I (DS-I)“ beinhaltet die Analyse existierender Arbeiten und Ansätze und dient der Identifikation des konkreten Handlungsbedarfs. Basierend auf diesen Erkenntnissen wird in der „Prescriptive Study (PS)“ ein Ansatz entwickelt, welcher die identifizierten Schwachstellen und Probleme adressieren soll. In der abschließenden Phase, der „Descriptive Study II (DS-II)“, wird der entwickelte Ansatz auf seine Anwendbarkeit und Wirksamkeit überprüft. Der in Abbildung 1.3 dargestellte Aufbau dieser Arbeit richtet sich somit ebenfalls nach der DRM-Methodik.

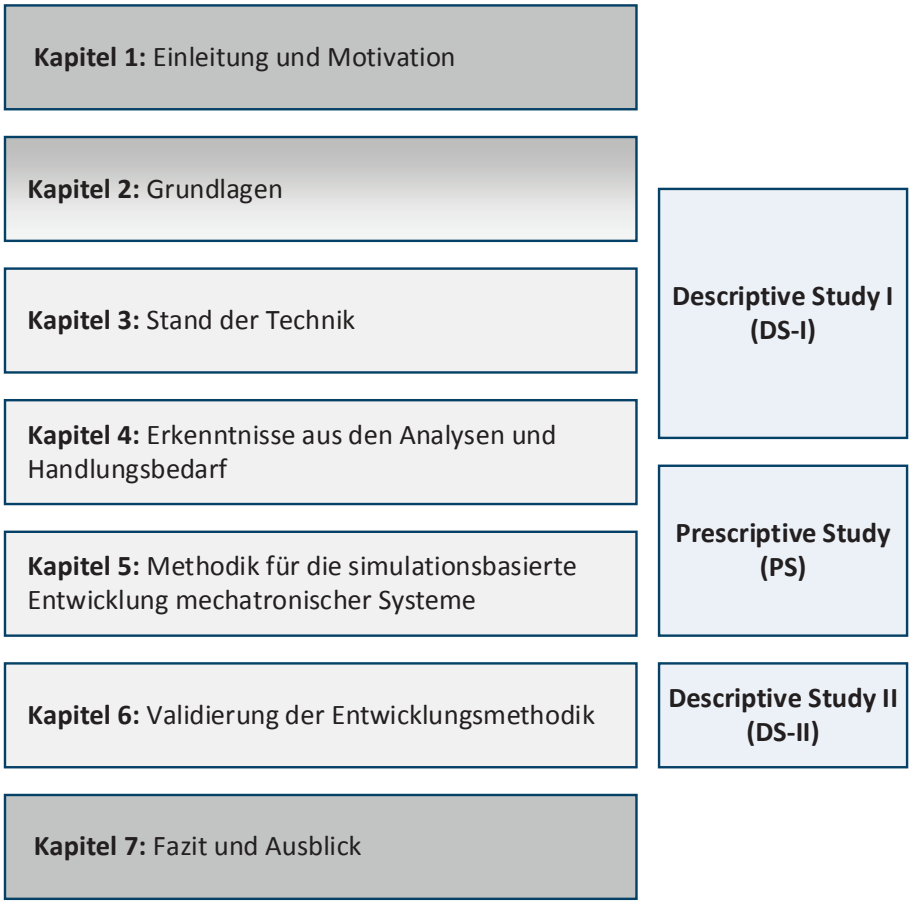


Abbildung 1.3: Aufbau der Arbeit

Entsprechend gliedert sich die vorliegende Arbeit: Anschließend an dieses einleitende Kapitel erfolgt in Kapitel 2 eine Diskussion der relevanten Grundlagen und Begrifflichkeiten. In Kapitel 3 erfolgt gemäß der „Descriptive Study I“ die Analyse des Stands der Technik. Diese gliedert sich zum einen in die Untersuchung existierenden Modellierungs- und Simulationstechniken. Zum anderen werden Entwicklungsmethodiken aus den Einzeldomänen sowie aus dem Bereich der

Mechatronik im Hinblick auf die Umsetzung eines simulationsbasierten Entwicklungsprozesses untersucht. Die Erkenntnisse der Analysen werden in Kapitel 4 diskutiert und dienen dazu, den Handlungsbedarf abzuleiten. Durch die Detaillierung der initialen Forschungsfragen aus diesem Kapitel erfolgt der Übergang zur „Prescriptive Study“, welche hauptsächlich durch Kapitel 5 repräsentiert wird. Hierin werden aus den Erkenntnissen der vorangegangenen Kapitel Anforderungen an eine Entwicklungsmethodik abgeleitet und eine Gesamtmethodik strukturiert entwickelt. Die Validierung der Methodik erfolgt in Kapitel 6 anhand eines mechatronischen Entwicklungsprojektes und bildet somit die „Descriptive Study II“ ab. Abschließend erfolgt in Kapitel 7 eine Reflektion der gesamten Arbeit bezüglich der Erfüllung der ursprünglich gesetzten Ziele und der Beantwortung der Forschungsfragen. Darüber hinaus werden Möglichkeiten und Schnittstellen für zukünftige Forschungsarbeiten diskutiert.

2 Grundlagen

2.1 Mechatronik

Der Begriff Mechatronik wurde 1969 von Tetsura Mori, Ingenieur bei der Yaskawa Electric Corporation, geprägt [KyOh96], [WWW3]:

> *The word, mechatronics, is composed of "mecha" from mechanism and the "tronics" from electronics. In other words, technologies and developed products will be incorporating electronics more and more into mechanisms, intimately and organically, and making it impossible to tell where one ends and the other begins. [Bish08]*

Heute existieren in der Literatur zahlreiche weitere Definition für den Begriff und das Fachgebiet der Mechatronik:

> *[Mechatronics is] the synergetic integration of mechanical engineering with electronic and intelligent computer control in the design and manufacturing of industrial products and processes. [HaTF96]*

> *What is needed, as a solid basis for designing high-performance machines, is a synergetic cross-fertilization between the different engineering disciplines involved: mechanical engineering, control engineering, microelectronics, and computer science. This is exactly what mechatronics is aiming at; it is a concurrent-engineering view on machine design. [Brus96]*

> *Mechatronics is a methodology used for the optimal design of electromechanical products. [ShKo11]*

> *Mechatronik ist ein interdisziplinäres Gebiet der Ingenieurwissenschaften, das auf den klassischen Disziplinen Maschinenbau, Elektrotechnik und Informatik aufbaut. Ein typisches mechatronisches System nimmt Signale auf, verarbeitet sie und gibt Signale aus, die es z. B. in Kräfte und Bewegungen umsetzt. [Schw89]*

Mechatronische Systeme entstehen durch simultanes Entwerfen und die Integration von folgenden Komponenten oder Prozessen

- *Mechanische und mit ihr gekoppelte Komponenten/Prozesse*
- *Elektronische Komponenten/Prozesse*
- *Informationstechnik (einschließlich Automatisierungstechnik)*

Die Integration erfolgt durch die Komponenten (Hardware) und durch die informationsverarbeitenden Funktionen (Software). Ziel ist dabei, eine optimale Lösung zu finden zwischen der mechanischen Struktur, Sensor- und Aktor-Implementierung, automatischer digitaler Informationsverarbeitung und Regelung. Zusätzlich werden synergetische Effekte geschaffen, die erweiterte Funktionen und innovative Lösungen ergeben. [Iser08]

Die Auswahl der Definitionen zeigt die Vielseitigkeit der Mechatronik. Jeder Definition liegt ein unterschiedlicher Fokus zugrunde, wobei das Grundverständnis stets das gleiche ist. Dieses lässt sich durch folgende Aspekte beschreiben:

- Mechatronik besteht aus einer Kombination der klassischen Disziplinen[1] Maschinenbau, Elektrotechnik und Informationstechnik.
- Das Zusammenwirken dieser Disziplinen erzeugt Synergien und ermöglicht damit neue Funktionalitäten. Abbildung 2.1 verdeutlicht dies.
- Mechatronik bezieht sich nicht alleine auf das Endprodukt, sondern auch auf den gesamten damit verbundenen Entwicklungs- und Herstellungsprozess.

In der Richtlinie VDI 2206 [VDI2206] wird die Definition von Harashima et al. [HaTF96] übernommen. Diese entspricht auch dem Verständnis, welches dieser Arbeit zugrunde liegt. Daher sei hier auch die deutsche Übersetzung dieser Definition nach [VDI2206] erwähnt:

Mechatronik bezeichnet das synergetische Zusammenwirken der Fachdisziplinen Maschinenbau, Elektrotechnik und Informationstechnik beim Entwurf und der Herstellung industrieller Erzeugnisse sowie bei der Prozessgestaltung.

[1] Die Begriffe „Disziplin“ und „Domäne“ werden im Rahmen der Arbeit synonym verwendet und beschreiben im Wesentlichen die der Mechatronik zugrundeliegenden Teilgebiete Mechanik, Elektronik und Informatik.

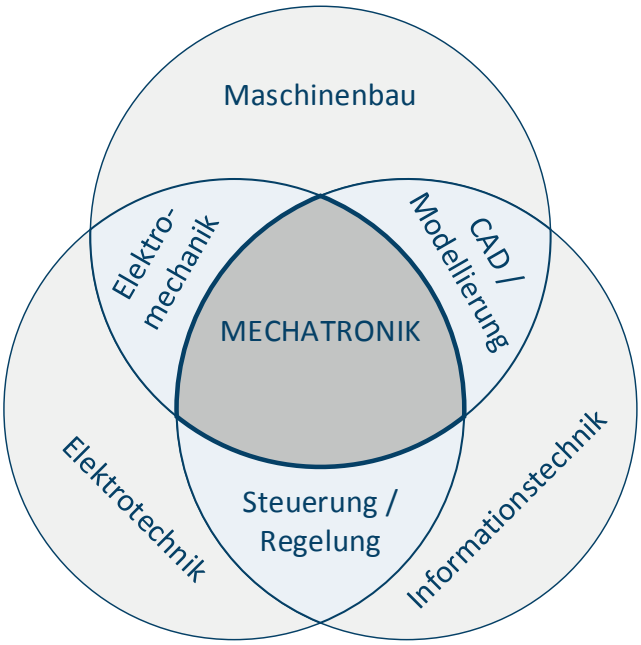


Abbildung 2.1: Synergetisches Zusammenwirken der Ingenieurdisziplinen in der Mechatronik, angelehnt an [Iser99]

Die Potentiale, die sich aus den Synergien ergeben, teilen sich nach [Möhr04] in:

- Funktions- und Verhaltensverbesserung technischer Systeme
- Reduzierung von Baugröße, Gewicht und Kosten
- Neue Funktionen und Anwendungen
- Intelligente, selbstoptimierende Systeme

Abbildung 2.2 zeigt den allgemeinen Aufbau eines mechatronischen Systems, bestehend aus einem – meist mechanischen [Schi97], [Wall95] – Grundsystem, einem informationsverarbeitenden Teil sowie Sensorik und Aktorik, welche als Schnittstelle zwischen beiden Teilsystemen fungieren. Das Grundsystem erfährt durch äußere Einflüsse der Umgebung Kräfte und Bewegungen, die im System Zustandsänderungen hervorrufen können. Diese werden über Sensoren erfasst und über (Mess-)Signale als Informationsträger an die Informationsverarbeitung übertragen. Nach der Verarbeitung der Messsignale – in der Regel durch Mikroprozessoren – werden (Stell-)Signale an die Aktorik übertragen, die wiederum über Kräfte, Momente und Bewegungen das Grundsystem beeinflusst. Durch diesen geschlossenen Kreis kann der Zustand des Grundsystems aktiv beeinflusst und in einen Sollzustand überführt werden.

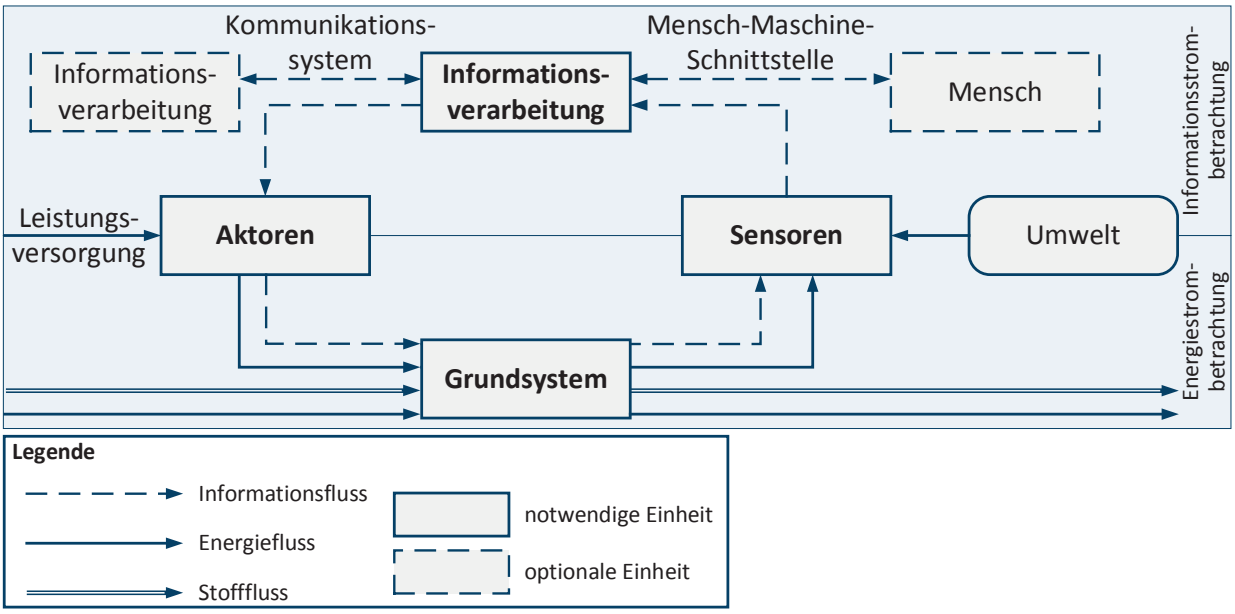


Abbildung 2.2: Allgemeiner Aufbau eines mechatronischen Systems nach [VDI2206] und [Wall95]

Ein typisches Beispiel für ein mechatronisches System ist das elektronische Stabilitätsprogramm (ESP) gemäß Abbildung 2.3, welches in modernen Kraftfahrzeugen verbaut ist. Die Funktion des ESP besteht darin, instabile Fahrzustände durch Regeleingriffe zu stabilisieren.

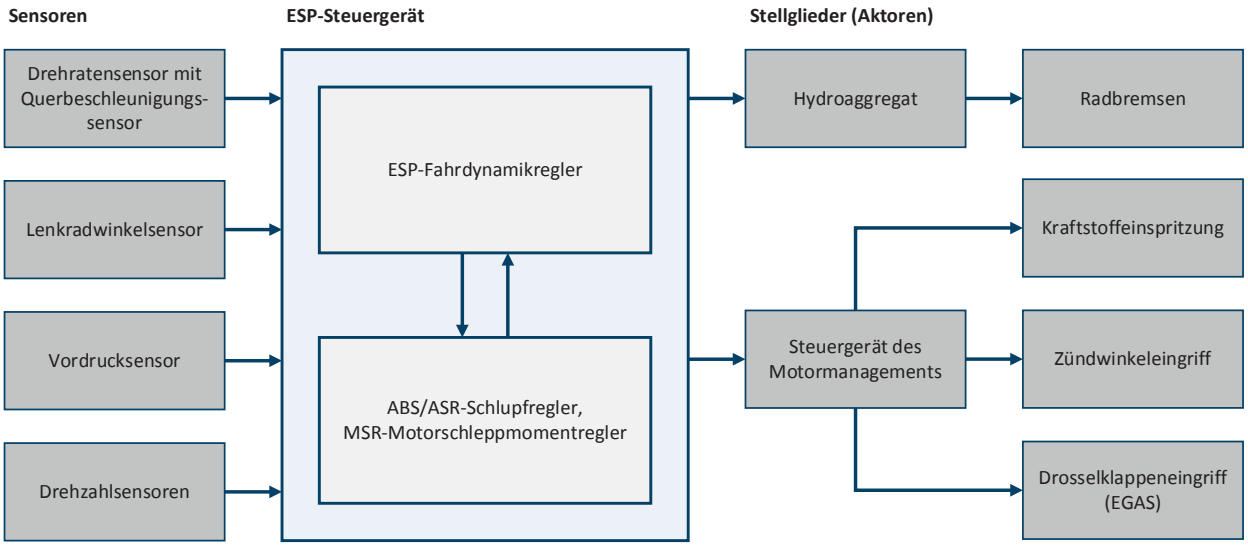


Abbildung 2.3: Beispiel für ein mechatronisches System: Elektronisches Stabilitätsprogramm (ESP), nach [Reif11]

Das mechanische Grundsystem entspricht in diesem Fall dem Gesamtfahrzeug, insbesondere dem Antriebsstrang und dem Bremssystem. Die Komponenten im linken Bereich von Abbildung 2.3 stellen den Sensorik-Anteil dar und geben Informationen über den aktuellen Fahrzustand

des Fahrzeugs. Der informationsverarbeitende Teil des Systems wird in diesem Fall durch das ESP-Steuergerät repräsentiert. Hierin werden die Sensorsignale ausgewertet und entsprechende Stellsignale errechnet, die an die Aktorik übertragen werden, welche in Abbildung 2.3 rechts dargestellt ist. Die Aktoren greifen wiederum in das mechanische Grundsystem ein. Im Falle des ESP bedeutet dies, dass einzelne Räder abgebremst werden und die Motorleistung beeinflusst wird, um einen stabilen Fahrzustand zu erzielen.

Das ESP stellt in seiner derzeitigen, komplexen und stark vernetzten Form ein modernes mechatronisches System dar. Der Grundstein für solche Systeme wurde mit der Entwicklung und Verbreitung der Mikroprozessoren und der Halbleitertechnik ab den 1960er Jahren gelegt. Hierdurch wurde es möglich, erste Systeme zu entwickeln, die dem Aufbau aus Abbildung 2.2 entsprechen. Beispiele hierfür sind Autofokuskameras und Industrieroboter. Mit dem Fortschritt in der Computer- und Kommunikationstechnik und dem damit verbundenen Fortschritt im Bereich der Steuer- und Regelungstechnik konnten leistungsfähigere Systeme entwickelt werden. Einen weiteren Schritt stellt die Entwicklung im Bereich der Mikromechanik und Mikroelektronik dar, wodurch immer höher integrierte und damit leichtere sowie kompaktere Systeme entwickelt werden konnten. Beispielhaft seien hier die Fortschritte im Bereich der Assistenzsysteme in Kraftfahrzeugen genannt, wie etwa das Antiblockiersystem (ABS) Ende der 1980er Jahre. Die immer weiter steigende Rechenleistung sowie die Miniaturisierung und Integration von Aktorik und Sensorik ermöglichen es heute, hochintegrierte, stark vernetzte und intelligente Systeme zu entwickeln. Auch hier seien beispielhaft moderne Fahrzeuge oder Unterhaltungselektronik, wie etwa Digitalkameras, erwähnt, die aus unzähligen miteinander vernetzen mechatronischen Systemen bestehen. Nur dadurch können moderne Funktionalitäten, wie adaptive Scheinwerfer oder eine automatische Bildstabilisierung, realisiert werden. Für weitere Erläuterungen zur historischen Entwicklung der Mechatronik sei auf die umfassende Literatur verwiesen, unter anderem [Brad10], [Huan02] und [Iser08].

Durch die rasante Weiterentwicklung der Computertechnologie zu Beginn des 21. Jahrhunderts ist die Rechenleistung nicht mehr der treibende Kostenfaktor in der Entwicklung mechatronischer Systeme. Vielmehr sind hierfür immer mehr die Integration und die Software selbst entscheidend [Brad10]. Dieser Trend wird sich voraussichtlich weiter fortsetzen, wodurch mechatronische Systeme immer komplexer und stärker vernetzt sein werden. Bereits im Jahr 2000 prognostizierte der Verband der deutschen Automobilindustrie, dass innerhalb von fünf Jahren

der Wertschöpfungsanteil der Mechatronik bei der Systementwicklung je nach Branche um das bis zu Vierfache wachsen wird [WWW4].

Für weitere Ausführungen bezüglich der zukünftigen Entwicklung mechatronischer Systeme, sowohl in Bezug auf die Produkttechnologie als auch auf den Entwicklungsprozess, sei beispielsweise auf Krause et al. [KrFG07] verwiesen.

2.2 Produktentwicklung

Der Begriff der (Produkt-)Entwicklung ist nach [VDI2221] definiert als das „zweckgerichtete Auswerten und Anwenden von Forschungsergebnissen und Erfahrungen“. In Abgrenzung dazu beschreibt der Begriff des Konstruierens die „Gesamtheit aller Tätigkeiten, mit denen – ausgehend von einer Aufgabenstellung – die zur Herstellung und Nutzung eines Produktes notwendigen Informationen erarbeitet werden und in der Festlegung der Produktdokumentation enden“ [VDI2221]. Somit beschreibt das Konstruieren eine Tätigkeit in der Produktentwicklung. Die Produktentstehung beschreibt dagegen alle Tätigkeiten im Lebenszyklus eines Produktes, die zu dessen Entstehung beitragen [Ehrl09] und ist somit ein wiederum umfassenderer Begriff, der die Produktentwicklung als Teilaspekt beinhaltet. Die Hierarchie der Begriffe ist in Abbildung 2.4 verdeutlicht.

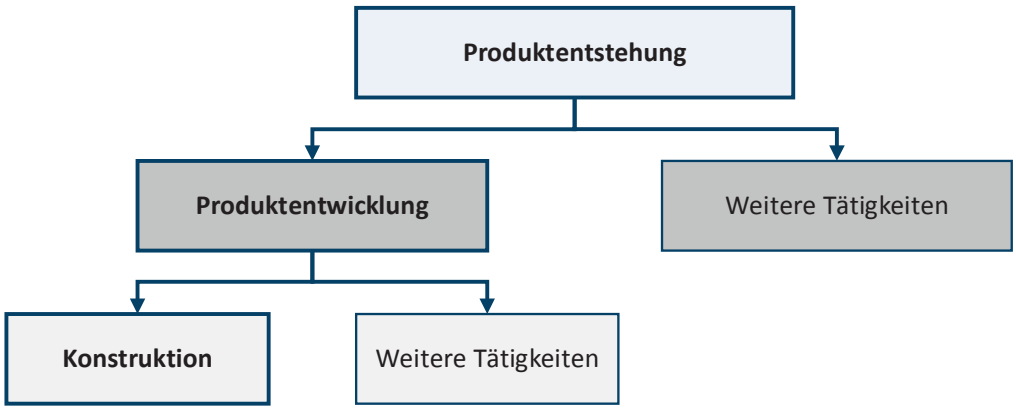


Abbildung 2.4: Abgrenzung von Produktentstehung, Produktentwicklung und Konstruktion

In dieser Arbeit wird durch die Entwicklung einer simulationsbasierten Produktentwicklungsmethodik insbesondere der Bereich der Produktentwicklung adressiert. Unter Produktentwicklungsmethodik oder Konstruktionsmethodik wird „ein geplantes Vorgehen mit konkreten Handlungsanweisungen zum Entwickeln und Konstruieren technischer Systeme“ [PBFG07] verstanden. Nach INCOSE (International Council on Systems Engineering) ist eine (Produktentwicklungs-)Methodik wie folgt definiert [Este08]:

- Eine **Methodik** ist eine Sammlung zusammengehöriger Prozesse, Methoden und Werkzeuge.
- Ein **Prozess** ist die logische Abfolge von Tätigkeiten, die zur Erreichung eines bestimmten Ziels ausgeführt werden. Prozesse beschreiben somit, was getan werden soll, ohne jedoch zu erläutern, wie dies geschieht.
- Eine **Methode** besteht aus Techniken, die eingesetzt werden, um Tätigkeiten auszuführen. Methoden beschreiben somit, wie Tätigkeiten ausgeführt werden.
- Ein **Werkzeug** ist ein Hilfsmittel, das bei Anwendung an einer bestimmten Methode die Effizienz einer Tätigkeit erhöhen kann. Werkzeuge ermöglichen somit die Ausführung von Methoden.

Dieses Begriffsverständnis findet im Rahmen der vorliegenden Arbeit Anwendung.

Ursprünglich sind Entwicklungsmethodiken stark vom Maschinenbau geprägt. Die ersten Ansätze für solche Methodiken sind nach Pahl et al. [PBFG07] im frühen 20. Jahrhundert in den Arbeiten von Erkens [Erke28] und Wögerbauer [Wöge43] zu finden. Doch auch davor sind bereits grundlegende Arbeiten zu finden, etwa von Reuleaux und Moll [ReMo54]. Seither wurden zahlreiche weitere Arbeiten auf diesem Gebiet verfasst, über die Pahl et al. [PBFG07] eine umfassende Übersicht geben. Durch die stetige Weiterentwicklung der Entwicklungswerkzeuge, beispielsweise der Rechnerunterstützung, verändern sich auch die Produktentwicklung und die damit verbundenen Methoden und Methodiken [Birk11]. Daher stellt die Produktentwicklung auch heute – und voraussichtlich auch in Zukunft – ein aktives und breites Forschungsfeld dar. Einige der Arbeiten auf diesem Gebiet werden im Rahmen dieser Arbeit in Kapitel 3.2 genauer analysiert.

2.3 Produkt und System

In den vorangegangenen Kapiteln wurden die Begriffe Produkt und System im Rahmen der Produktentwicklung und der Mechatronik verwendet. Der Unterschied dieser Begriffe soll anhand deren Definitionen verdeutlicht werden. So beschreibt der Begriff des (technischen) Systems:

> *[die] Gesamtheit von der Umgebung abgrenzbarer (Systemgrenzen), geordneter und verknüpfter Elemente, die mit dieser durch technische Eingangs- und Ausgangsgrößen in Verbindung stehen. [VDI2221]*

> *a set of interrelated components working together toward some common objective. [Koss11]*
>
> *a finite set of elements collected to form a whole under certain well-defined rules, whereby certain definite relationships exist between the elements, and to its environment. [HuEd88]*

Dagegen ist ein Produkt definiert als:

> *Erzeugnis, das als Ergebnis des Entwickelns und Konstruierens hergestellt oder angewendet wird. Das können materielle (z. B. Maschinen, Verfahren) oder auch immaterielle Erzeugnisse (z. B. Programme) sein. [VDI2221]*

Somit besteht ein System aus mehreren Komponenten oder Subsystemen, die im Verbund gewisse Eingangsgrößen in Ausgangsgrößen umwandeln. Dagegen kann ein Produkt auch aus nur einer Komponente bestehen und muss nicht zwingend einen technischen Zweck erfüllen. Im Rahmen dieser Arbeit wird die Definition von Hubka und Eder [HuEd88] verwendet, nach welcher ein Produkt mit einem technischen System gleichzusetzen ist. Obwohl dies nach obigen Definitionen nicht allgemeingültig ist, folgt die vorliegende Arbeit diesem Verständnis, da Produkte hier stets mechatronische Systeme darstellen.

2.4 Modell und Simulation

Wie bereits in Kapitel 2.2 erwähnt, ist die heutige Produktentwicklung geprägt durch die Entstehung und Weiterentwicklung von Entwicklungswerkzeugen. Hierzu zählt unter anderem die Simulation, die nach VDI-Richtlinie 3633 [VDI3633] wie folgt definiert ist:

> *Simulation ist ein Verfahren zur Nachbildung eines Systems mit seinen dynamischen Prozessen in einem experimentierfähigen Modell, um zu Erkenntnissen zu gelangen, die auf die Wirklichkeit übertragbar sind.*

Simulation beschreibt demnach im Umfeld der Produktentwicklung ein Verfahren, durch das die Eigenschaften eines Produktes analysiert werden können, ohne dass das Produkt physisch existiert. Im Rahmen dieser Arbeit bezieht sich der Begriff der Simulation vor allem auf computerbasierte Simulationen, welche einen Teilbereich der virtuellen Produktentstehung darstellen.

Der Begriff der virtuellen Produktentstehung umfasst alle rechnerunterstützten Prozesse, Methoden und Werkzeuge in der Produktentwicklung, beispielsweise CAD und PDM. Erste Arbeiten auf diesem Gebiet erfolgten bereits in den 40er Jahren des vergangenen Jahrhunderts. Seit den 80er Jahren entwickelt sich die virtuelle Produktentstehung zu einem festen Bestandteil heutiger Entwicklungsprozesse. Vajna et al. [VWBZ09] geben hierzu einen ausführlichen historischen Überblick.

Insbesondere durch den starken Anstieg der Rechenleistung heutiger Computersysteme steigen zum einen die Möglichkeiten der computerbasierten, meist numerischen Simulation, und zum anderen sinken die Simulationszeiten und damit auch die Kosten. Daher werden Simulationen immer häufiger im Entwicklungsprozess als Ersatz oder Ergänzung physischer Prototypen eingesetzt. Durch den Einsatz von Simulation ergeben sich diverse Vorteile, sowohl für den Entwicklungsprozess als auch für das zu entwickelte Produkt [Döbe08], [KLSL13], [Lang09], [SeRa08]:

- Verkürzung der Entwicklungszeit
- Verringerung von Entwicklungskosten
- Erhöhung der Produktqualität
- Steigerung der Innovationen durch Erweiterung des untersuchbaren Lösungsraum
- Beherrschung steigender Variantenvielfalt von Produkten
- Erhöhung des Reifegrades erster Prototypen
- Reduzierung physischer Prototypen und deren Änderungen
- Erweiterung von Erprobungsmöglichkeiten

Wie obige Definition zeigt, ist die Grundlage für eine Simulation immer ein experimentierfähiges Modell. Dieses beschreibt nach [VDI3633]

> *eine vereinfachte Nachbildung eines geplanten oder real existierenden Systems mit seinen Prozessen in einem anderen begrifflichen oder gegenständlichen System. Es unterscheidet sich hinsichtlich der untersuchungsrelevanten Eigenschaften nur innerhalb eines vom Untersuchungsziel abhängigen Toleranzrahmens vom Vorbild.*

Dabei existieren zahlreiche Arten von Modellen, die jeweils einem spezifischen Einsatzzweck zuzuordnen sind. Diese können zur reinen Visualisierung dienen – beispielsweise Tonmodelle im Bereich des Fahrzeugdesigns –, zur Beschreibung von Sachverhalten oder Prozessen – beispielsweise Vorgehensmodelle in der Produktentwicklung –, oder zur Simulation – beispielswei-

se mathematische Modelle zur Beschreibung des Systemverhaltens. Im Rahmen dieser Arbeit werden lediglich Simulationsmodelle betrachtet, also solche, die ausführbar sind und als Grundlage computerbasierter Simulationen dienen.

Prinzipiell lassen sich Modelle nach Wallaschek [Wall95] in vier Kategorien einteilen: Mentale Modelle stellen den Ausgangspunkt der Modellierung dar. Hiermit wird das reale System in einer abstrahierten Form abgebildet. Sie basieren meist auf graphischen, domänenspezifischen Beschreibungen. Aufbauend hierauf wird die mathematische Beschreibung des Systemverhaltens abgeleitet und damit ein mathematisches Modell erstellt. Durch eine Aufbereitung des mathematischen Modells für die rechnergestützte Lösung des Gleichungssystems entsteht ein algorithmisches Modell. Die Beschreibung dieses Modells durch eine Programmiersprache führt zu einem ausführbaren, rechnerinternen Modell.

Nach Vajna et al. [VWBZ09] teilt sich der Prozess der Modellierung und Simulation in drei Hauptschritte: Im „Preprocessing“ wird ein Modell für die Simulation erstellt und parametriert. Im „Processing“ erfolgt die eigentliche Simulation, das heißt das Modell wird ausgeführt und das Gleichungssystem gelöst. Der abschließende Schritt des „Postprocessing“ beinhaltet die Ergebnisdarstellung und -interpretation.

Eine detaillierte Betrachtung von Simulationsarten sowie von Modellen aus dem Umfeld der Entwicklung mechatronischer Systeme erfolgt in Kapitel 3.1.

2.5 Simulationsbasierte Entwicklung

Die zuvor beschriebenen Vorteile von Simulationstechniken legen nahe, diese in die Produktentwicklung zu integrieren. Entsprechend werden Simulationen, abhängig von Branche und Unternehmensstruktur, bereits intensiv genutzt [Aber06]. Über die vereinzelte Nutzung von Simulation im Entwicklungsprozess hinaus liegt der Fokus dieser Arbeit auf der simulationsbasierten Entwicklung. Die folgenden Definitionen spiegeln das Verständnis simulationsbasierter Entwicklung wider, wie es dieser Arbeit zugrunde liegt:

> *Simulation-driven design can be defined as a design process where decisions related to the behavior and the performance of the design in all major phases of the process are significantly supported by computer-based product modeling and simulation. [Sell99]*

> *Simulation-based design is a process in which simulation is the primary means of design evaluation and verification. [SBOW04]*

Hieraus lässt sich die in dieser Arbeit verwendete Definition ableiten:

> *Simulationsbasierte Entwicklung beschreibt einen Ansatz in der Produktentwicklung, bei dem die Überprüfung der Eigenschaften des zu entwickelnden Produktes in allen Phasen des Entwicklungsprozess hauptsächlich durch Simulation erfolgt.*

Einen Überblick über weitere Definitionen geben Karlberg et al. in [KLSL13].

Unter Berücksichtigung der immer weiter steigenden Anforderungen bezüglich Entwicklungszeit, Entwicklungskosten und Produktqualität stellt der Ansatz der simulationsbasierten Entwicklung durch intensive, durchgängige Eigenschaftsabsicherungen bereits von frühen Phasen an eine Möglichkeit zur Prozessoptimierung dar. Insbesondere für mechatronische Systeme, die aufgrund der Anzahl verkoppelter Elemente durch Komplexität und aufgrund der Anzahl verschiedener Domänen durch Heterogenität gekennzeichnet sind [Möhr04], bietet die simulationsbasierte Entwicklung großes Optimierungspotential. So ist die Komplexität mechatronischer Systeme mit vertretbarem Aufwand nur durch Simulation zu beherrschen [CWPH08], [MaBu11], beispielsweise durch den Vergleich verschiedener Konzepte mittels Simulationen [PDSK01].

Im folgenden Kapitel wird der Stand der Technik bezüglich der aktuellen Möglichkeiten der simulationsbasierten Entwicklung sowie deren Umsetzungsgrad in bestehenden Entwicklungsmethodiken analysiert. Daraus wird der Handlungsbedarf abgeleitet, welcher die Grundlage der in dieser Arbeit entwickelten Methodik darstellt.

3 Stand der Technik

Inhalt dieses Kapitels ist die Analyse relevanter Vorarbeiten, die in Verbindung zu simulationsbasierter Entwicklung stehen. Basierend zum einen auf einer initialen Literaturrecherche und zum anderen auf Diskussionen mit Experten aus relevanten Bereichen leitet sich folgende Eingrenzung der relevanten Forschungsbereiche ab:

- Modellierungstechniken (Kapitel 3.1)
- Simulationstechniken (Kapitel 3.1)
- Entwicklungsmethodiken (Kapitel 3.2)

Daher wird im Folgenden zu jedem dieser Einzelbereiche eine ausführliche Analyse des Stands der Technik durchgeführt. Der letzte Punkt basiert hauptsächlich auf einer Literaturrecherche und soll den aktuellen Stand bezüglich der Einbindung von Simulation in Entwicklungsprozessen bestimmen. Dagegen beinhalten die beiden ersten Punkte zudem Inhalte, die aus der Analyse aktuell am Markt verfügbarer (Software-)Werkzeuge stammen. Hierdurch sollen die Möglichkeiten und Schwachstellen, welche aktuell im Bereich der Modellierung und Simulation bestehen, identifiziert werden.

Im Hinblick auf die Zielsetzung dieser Arbeit werden jeweils nur die wesentlichen Merkmale der Techniken, Werkzeuge und Methodiken erläutert. Für tiefergehende Ausführungen sind jeweils Literaturverweise angegeben.

3.1 Modellierungs- und Simulationstechniken

Wie aus den Definitionen in Kapitel 2.4 erkenntlich wird, sind für Simulationen stets ausführbare Modelle als Grundlage erforderlich. Entsprechend werden in diesem Kapitel neben Simulations- auch Modellierungstechniken betrachtet. Die Betrachtungen erfolgen nach Möglichkeit unabhängig von spezifischen Softwarewerkzeugen. Sind Techniken jedoch direkt abhängig von spezifischen Softwarewerkzeugen, so werden diese mitbetrachtet.

Die der Mechatronik zugrundeliegenden Einzeldomänen haben über die Jahre individuelle Techniken entwickelt. Diese werden im Folgenden domänenspezifisch diskutiert. Darüber hinaus gibt es im Bereich der Forschung und seitens der Softwarehersteller Bemühungen, domänenübergreifende Aspekte in einem Modell darzustellen und auch ganzheitlich zu simulieren. Dies ist insbesondere für die Entwicklung mechatronischer System wichtig, weshalb diese Techniken nochmals gesondert betrachtet werden.

An dieser Stelle sei nochmals darauf hingewiesen, dass der Begriff des Modells sehr breit verstanden werden kann – siehe hierzu beispielsweise die Ausführungen von Stachowiak [Stac73]. Im Rahmen dieser Arbeit werden jedoch nur Modelle betrachtet, welche direkt oder indirekt für Simulationen relevant sind. Somit wird im Folgenden nur ein Ausschnitt der im Entwicklungsprozess eingesetzten Modelle beschrieben. Zudem werden hochspezialisierte Techniken nicht betrachtet. Diese sind oftmals anwendungsfall- oder sogar unternehmensspezifisch und daher für eine allgemeingültige Methodik wenig zielführend. Dennoch soll in der zu entwickelnden Methodik die Möglichkeit bestehen, auch solche Techniken und Werkzeuge individuell integrieren zu können.

Die im Folgenden präsentierte Analyse basiert neben praktischen Erfahrungen auch auf Veröffentlichungen des Autors. Diese fokussieren zum einen Techniken des frühen Systementwurfs [DoVi12b] und zum anderen Techniken zur Unterstützung des Übergangs von der Funktionsmodellierung zur Simulation des Gesamtsystems [EDGV13].

3.1.1 Allgemeine Techniken

3.1.1.1 Funktionsmodellierung

Funktionsmodellierung beschreibt nach Erden et al. [EKBA08] die Erstellung von Modellen von Geräten, Produkten, Objekten und Prozessen basierend auf deren Funktionalitäten und Komponenten. Diese abstrakte Form der Modellierung des Gesamtsystems fördert die Kommunikation und das Verständnis bei interdisziplinären Problemstellungen [EKBA08]. Basierend auf einer Aufteilung der Gesamtfunktion lässt sich das System in Teilfunktionen zerlegen. Die Verknüpfung dieser Teilfunktionen zur Gesamtfunktion bezeichnen Pahl et al. [PBFG07] als Funktionsstruktur, welche dazu dient, das System auf abstrakte Weise zu beschreiben und Lösungen für eine Problemstellung zu finden. Entsprechend der Einteilung nach Wallaschek aus Kapitel 2.4 lassen sich Funktionsmodelle als mentale Modelle einstufen. Sie sind interdisziplinär anwendbar und dienen als Vorstufe weiterer Modelle. Funktionsmodelle selbst sind nicht direkt simulierbar.

Einen Überblick über verschiedene Techniken der Funktionsmodellierung geben beispielsweise Eisenbart et al. [EiGB13] und Erden et al. [EKBA08].

3.1.1.2 Mathematische Beschreibung

Eine der allgemeinsten Beschreibungsformen technischer Systeme und deren Verhaltens sind mathematische Modelle, die unabhängig von domänenspezifischen Modellierungstechniken sind [KüHW98]. Basierend auf der mathematischen Beschreibung, welche im Allgemeinen über Differentialgleichungen erfolgt, kann durch das Lösen des Gleichungssystems das Verhalten eines Systems oder Produktes beschrieben werden. In einfachen Fällen lässt sich dieses System analytisch lösen, bei komplexeren Systemen oftmals nur numerisch. Hierfür existieren zahlreiche Werkzeuge, zu denen in den folgenden Kapiteln an den entsprechenden Stellen Beispiele genannt werden.

3.1.2 Mechanik

Im Folgenden werden Modellierungs- und Simulationstechniken aus dem Bereich der Mechanik diskutiert.

3.1.2.1 Skizzen, technische Zeichnungen und CAD

Der Ausgangspunkt in der Mechanik ist oftmals eine zwei- oder dreidimensionale Skizze, die als mentales Modell anzusehen ist. Obwohl Skizzen einen sehr geringen Formalisierungsgrad besitzen, haben sich dennoch im Bereich der Mechanik gewisse Symboliken entwickelt, welche das Verständnis für andere Domänen erschweren. Die stark formalisierte Version der Skizze stellt die technische Zeichnung dar, die durch eine Vielzahl von Normen standardisiert ist. Beide Modellformen sind nicht direkt für Simulationen geeignet.

Mittlerweile flächendeckend im Einsatz sind CAD-Systeme (Computer-aided Design). Diese bieten die Möglichkeit der virtuellen, meist dreidimensionalen Modellierung der Produktgeometrie. In der Regel geht dieser Modellierung die Erstellung einer Skizze als mentale Grundlage voraus. Da diese Modelle, abgesehen von der für den Nutzer sichtbaren Visualisierung des Produktes, eine mathematische Beschreibung der Produktgeometrie und meist auch weitergehende Informationen wie Materialkennwerte beinhalten, dienen CAD-Modelle häufig als Grundlage verschiedener Simulationen im Mechanikbereich. Entsprechend geht der Trend bei aktuellen CAD-Systemen dahin, dass auf Basis der eigentlichen CAD-Modellierung auch Simulationsfunktionalitäten im System integriert werden. Hierzu zählen beispielsweise rein geometrieabhängige Simulationen wie Toleranz-, Einbau- oder Montagesimulationen. Weiterhin werden auch aufwendigere Simulationen integriert, wie die Finite-Elemente-Methode (FEM) oder Mehrkörpersimulationen, die im Folgenden näher erläutert werden.

3.1.2.2 Finite-Elemente-Methode (FEM)

Die Finite-Elemente-Methode (FEM) stellt ein numerisches Näherungsverfahren zur Lösung partieller Differentialgleichungen dar und findet im gesamten Bereich der Ingenieurwissenschaften Anwendung [VWBZ09]. Der Begriff selbst wird jedoch in der Mechanik meist synonym für strukturmechanische, computerbasierte Simulationen verwendet. Hierbei sind beispielsweise mechanische Spannungen, Verformungen und Schwingungen von Interesse.

Wie der Name bereits ausdrückt, wird bei der Finite-Elemente-Methode ein Körper in eine endliche Anzahl an Elementen unterteilt, die sich in sogenannten Knoten berühren. Dieser Vorgang wird als Diskretisierung bezeichnet. Jedes dieser Elemente kann durch Differentialgleichungen beschrieben werden, wodurch unter Berücksichtigung der Übergangsbedingungen in den Knoten ein Gleichungssystem für den gesamten Körper aufgestellt werden kann. Dieses kann computergestützt gelöst werden. Die Ergebnisse lassen sich in diversen Formen darstellen, von Tabellen bis hin zu Niveauflächenbildern.

Ausgangspunkt für die FEM-Simulation ist die bekannte Geometrie des Bauteils. FEM-Simulationen sind mittlerweile in fast allen CAD-Systemen direkt integriert, beispielsweise in Creo von PTC [WWW5] oder CATIA von Dassault Systèmes [WWW6]. Hierdurch liegt die Produktgeometrie direkt bereit. Darüber hinaus existieren zahlreiche spezialisierte FEM-Softwarewerkzeuge, beispielhaft seien hier Abaqus von Dassault Systèmes [WWW7] oder Hyperworks von Altair Engineering [WWW8] genannt. Bei diesen spezialisierten Werkzeugen ist es erforderlich, die Geometrie entweder aus CAD-Modellen zu importieren oder direkt in der Software zu modellieren.

Für weitere Informationen über Verfahren und Möglichkeiten der Finite-Elemente-Methode im Bereich der Mechanik sei auf die umfangreiche Literatur hierzu verwiesen, beispielsweise [Bath02], [JuLa13] und [Klei07].

3.1.2.3 Mehrkörpersimulation

Bewegliche mechanische Systeme, die oftmals als Grundlage mechatronischer Systeme dienen [Rodd06], bestehen in der Regel aus mehreren Einzelkörpern. Gemäß der Definition von Woernle [Woer11], wonach ein Mehrkörpersystem (MKS) aus massebehafteten Körpern besteht, deren Bindungen geometrisch beschränkt sind und auf die Kräfte und Momente einwirken, stellen Mehrkörpersysteme daher oftmals die Grundlage mechatronischer Systeme dar. In der Regel werden starre Körper verwendet, wobei es mit entsprechend höherem Aufwand auch

möglich ist, deformierbare Körper zu berücksichtigen [RiSc10]. Die Modellierung erfolgt durch Diskretisierung des mechanischen Systems. Hierbei wird dieses in eine endliche Zahl massebehafteter Körper zerlegt, die untereinander über masselose Koppelelemente und mit der Umwelt durch Lagerungen verbunden sind [VWBZ09]. Neben den rein kinematischen Größen werden auch kinetische Aspekte betrachtet, also auch die bei der Bewegung auftretenden Kräfte und Momente. Für das modellierte System werden die Bewegungsgleichen erstellt und das entsprechende Gleichungssystem gelöst. Somit ergeben sich die relevanten kinematischen und kinetischen Größen. Diese können neben dem einfachen Tabellenexport sowie der Graphendarstellung meist auch in (gegebenenfalls dreidimensionalen) Animationen dargestellt werden.

Für Mehrkörpersimulationen sind Geometrie sowie Material oder Gewicht der Körper erforderlich. Ähnlich der FEM-Simulationen sind Mehrkörpersimulationen immer häufiger in CAD-Systemen integriert, etwa in PTC Creo [WWW5]. Hierdurch ergibt sich der Vorteil der Durchgängigkeit beim Geometrieexport und bei Geometrieänderungen. Darüber hinaus existieren spezialisierte Werkzeuge, die entweder CAD-Modelle importieren können oder die Möglichkeit bieten, Körper (vereinfacht) nachzumodellieren. Beispielhaft seien hier die Softwarelösungen Adams der Firma MSC Software [WWW9] und SIMPACK der SIMPACK AG [WWW10] genannt.

Zudem besitzen Softwarewerkzeuge aus dem Bereich der Mehrkörpersimulation in der Regel Funktionalitäten, die über die rein mechanischen Aspekte hinausgehen. So ist es in den meisten Werkzeugen möglich, Aktoren und Regler zu implementieren oder Co-Simulationen mit spezieller Regelungssoftware durchzuführen. Zudem können MKS-Werkzeuge mit FEM-Systemen gekoppelt werden. Hierdurch ist der Einsatz von MKS-Software wesentlich einfacher auf mechatronische Systeme ausweitbar.

Weitergehende Informationen zur Mehrkörpermodellierung und -simulation sind zum Beispiel in [RiSc10], [Rodd06], [VWBZ09] und [Woer11] zu finden.

3.1.2.4 Computational Fluid Dynamics (CFD)

Computational Fluid Dynamics (CFD, englisch für numerische Strömungsmechanik) bezeichnet ein numerisches Verfahren zur Modellierung und Simulation strömungsmechanischer Problemstellungen. Die strömungsmechanische Beschreibung erfolgt mittels partieller Differentialgleichungen, in Sonderfällen auch mittels Integralgleichungen [WeAn09]. Diese sind für die meisten Probleme nicht analytisch lösbar [Schw13]. Entsprechend erfolgen eine Diskretisierung des Strömungsgebietes und eine numerische Lösung des entstehenden Gleichungssystems.

Ähnlich der FEM und der Mehrkörpersimulation können die Simulationsergebnisse in Tabellen, Graphen, Niveauflächenbildern oder Animationen dargestellt werden.

Ausgangspunkt für CFD-Simulationen ist stets die Bauteilgeometrie. Derzeit existieren kaum integrierte Lösungen für die CFD-Simulationen in CAD-Systemen. Hauptsächlich werden spezialisierte Werkzeuge genutzt, beispielsweise OpenFoam der Firma OpenCFD [WWW11] oder ANSYS Fluent der Firma ANSYS [WWW12].

Für weitergehende Informationen zu den Grundlagen der numerischen Strömungssimulation sei beispielsweise auf [FePe08], [LaOe13], [Schw13] und [WeAn09] verwiesen.

3.1.2.5 Optimierungsverfahren

Optimierungsverfahren stellen im eigentlichen Sinne keine Simulationstechnik dar. Vielmehr basieren sie auf der Verwendung diverser Simulationen. Ziel ist es, bezüglich eines oder mehrerer Kriterien ein Optimum zu finden. Prinzipiell lassen sich analytische und numerische Optimierungsverfahren unterscheiden [VWBZ09]. Erstere sind im Rahmen dieser Arbeit von geringerem Interesse, da bei den hier besprochenen Simulationen meist keine analytisch lösbaren Modelle vorliegen. Numerische Verfahren basieren auf einer iterativen Annäherung durch mehrfaches Durchlaufen von Simulationen mit anschließender Anpassung der gewählten Kriterien, bis der gewünschte Zielwert erreicht ist. Optimierungsverfahren werden im Bereich der Mechanik häufig verwendet, um basierend auf FEM-Simulationen geometrische Strukturen zu optimieren. Sie sind aber auch in anderen Domänen anwendbar.

3.1.3 Fluidtechnik

Fluidtechnik umfasst die Teilgebiete Hydraulik und Pneumatik. Die Modellierung erfolgt in diesem Bereiche über Schaltpläne mit Symbolen, welche in [ISO1219] genormt sind. Diese Schaltpläne können rein manuell als Grafiken erzeugt werden (mentale Modelle) oder mit entsprechenden Softwarewerkzeugen. Bei diesen ist zu den einzelnen Elementen jeweils die Beschreibung des Verhaltens hinterlegt, wodurch bei der Erstellung des Schaltplans gleichzeitig das Verhaltensmodell erstellt wird. Somit ist bei entsprechender Parametrierung die Simulation des Systems möglich. Der Hauptvertreter diese Softwarewerkzeuge ist FluidSIM der Art Systems Software GmbH [WWW13].

3.1.4 Elektronik

Auch im Bereich der Elektronik haben sich zahlreiche Modellierungs- und Simulationstechniken etabliert, über die im Folgenden ein Überblick gegeben wird.

3.1.4.1 Schaltpläne

Schaltpläne dienen zur Modellierung der Struktur von Schaltungen (siehe Y-Modell nach Gajski und Kuhn [GaKu83] in Kapitel 3.2.3.1). Hier sind alle erforderlichen Bauteile, beispielsweise Widerstände und Kondensatoren, in Form standardisierter Symbole mit den entsprechenden Verbindungen dargestellt. Basierend auf dem Schaltplan wird das Schaltungslayout abgeleitet, das heißt die tatsächliche Platine inklusive der Anordnung der Bauteile. Dieses Layout dient als Grundlage für die Fertigung.

Schaltpläne und Schaltungslayouts können als rein graphische Darstellung prinzipiell von Hand erstellt werden und stellen mentale Modelle dar. Wesentliche Bedeutung gewinnt jedoch der Einsatz von Softwarewerkzeugen [Lien06], in denen der Schaltplan mithilfe von Bauteilbibliotheken erstellt werden kann. Zudem lassen sich hierdurch verschiedene Darstellungsformen und Informationen, wie beispielsweise das Schaltungslayout, direkt aus dem Schaltplan ableiten. Oftmals sind diese Werkzeuge in ECAD-Systemen (Electronic CAD) integriert. Verbreitete Softwarewerkzeuge sind beispielsweise Altium Designer der Firma Altium [WWW14] und EAGLE der CadSoft Computer GmbH [WWW15].

3.1.4.2 Zustandsautomaten

Insbesondere im Bereich der Digitalelektronik ist das Verhalten einer Schaltung stark davon abhängig, welche diskreten Zustände die Schaltung zu verschiedenen Zeitpunkten annehmen kann. Dieses Verhalten lässt sich mit endlichen Zustandsautomaten (englisch: Finite State Machine) beschreiben. Ein Zustandsautomat besteht nach [Bend05] aus einem Zustandsraum, Zustandsübergängen und (mindestens) einem initialen Zustand. Zur graphischen Darstellung eines Zustandsautomaten eignen sich beispielsweise „Statecharts“, die in [Hare87] beschrieben sind.

3.1.4.3 Hardwarebeschreibungssprachen

Neben der Struktur und dem Layout einer Schaltung ist das Verhalten ausschlaggebend (siehe Y-Modell in Kapitel 3.2.3.1). Hardwarebeschreibungssprachen (HDL – englisch: Hardware Description Language) ermöglichen die Beschreibung dieser drei Sichten. Im Bereich digitaler Schaltungen haben sich zwei Sprachen etabliert: VHDL (Very High Speed Integrated Circuit

Hardware Description Language) und Verilog. Diese ermöglichen die Modellierung zeitdiskreten Verhaltens, beispielsweise basierend auf Zustandsautomaten. Für weitere Informationen zu VHDL und Verilog sei stellvertretend für die umfangreiche Literatur auf [Ashe02] und [ThMo08] verwiesen.

Zusätzlich zu den Beschreibungssprachen für digitale Schaltungen haben sich Sprachen zur Beschreibung analoger und sogenannter Mixed-Signal-Schaltungen entwickelt, also Schaltung mit sowohl digitalen als auch analogen Schaltungsanteilen. Hauptvertreter sind hier VHDL-AMS (AMS: Analog and Mixed-signal) und Verilog-AMS. Nähere Informationen hierzu finden sich beispielsweise in [Ashe02] und [KuZi04].

Auf der Basis von VHDL(-AMS) und Verilog(-AMS) existieren diverse Simulatoren, welche den entsprechenden Code direkt in eine Simulation übertragen können. Somit lässt sich nach der Struktur- und Verhaltensmodellierung direkt das Verhalten einer Schaltung simulieren. Weiterhin sind Hardwarebeschreibungssprachen bereits in diversen ECAD-Systemen integriert.

3.1.4.4 Schaltungssimulation

Für die Schaltungssimulation hat sich die Software SPICE (Simulation Program with Integrated Circuit Emphasis) etabliert, die an der University of California in Berkley entwickelt wurde. SPICE dient als Grundlage zahlreicher kommerzieller Softwarewerkzeuge, wie etwa PSpice der Firma OrCAD [WWW16].

Mittels Bauteilbibliotheken, in denen das Verhalten der Bauteile durch mathematische Gleichungen beschrieben ist, wird ein Gleichungssystem für die gesamte Schaltung erstellt und gelöst. Somit lassen sich für analoge, digitale und Mixed-Signal-Schaltungen die gewünschten Ausgangsgrößen in Form von Tabellen oder Graphen darstellen. Die Schaltung kann entweder aus anderen Softwarewerkzeugen zur Schaltplanerstellung exportiert oder direkt in der entsprechenden SPICE-Software nachgebaut werden.

Detailliere Informationen zu SPICE und deren Nutzung können beispielsweise [SaHy06] entnommen werden.

3.1.4.5 Elektromagnetische Feldsimulation

Jeder stromdurchflossen Leiter erzeugt ein elektromagnetisches Feld. Diese Felder werden entweder bewusst genutzt – beispielsweise bei Funktechnologien – oder aber sie sind Störgrößen, deren Auswirkungen auf die restliche Schaltung – und damit beispielsweise auf die Signal-

übertragung – verringert werden sollen. Hierfür ist es erforderlich, die Ausbreitung und Ausprägung der Felder zu kennen. Zur Ermittlung dieser Kenngrößen werden elektromagnetische Feldsimulationen eingesetzt. Deren prinzipielle Funktionsweise basiert auf numerischen Verfahren, oftmals auf der Finite-Elemente-Methode: Der Untersuchungsraum wird diskretisiert, für jedes Element wird die mathematische Beschreibung erstellt und das entstehende Gleichungssystem wird numerisch gelöst. Die Simulationsergebnisse lassen sich, wie bei der mechanischen Anwendung der FEM, in Tabellen, Graphen oder (gegebenenfalls dreidimensionalen) Niveauflächenbildern darstellen. Auch für die elektromagnetische Feldsimulation ist die exakte Geometrie des Untersuchungsraums erforderlich. Maxwell der Firma ANSYS [WWW17] stellt beispielhaft ein kommerzielles Softwarewerkzeug dar.

Weitere Information über elektromagnetische Felder und deren Simulation finden sich unter anderem in [Henk07].

3.1.4.6 Sonstige Simulationen im Bereich der Elektronik

Auch wenn es sich nicht um mechatronische Produkte handelt, werden im Bereich der Elektronik je nach Anwendungsfall Simulationen aus der Mechanik angewandt. Zum einen sind alle elektrischen und elektronischen Bauteile verlustbehaftet, wodurch sich Wärmeverluste ergeben. Diese können dazu führen, dass die Umgebungstemperatur den zulässigen Betriebsbereich übersteigt. Entsprechend muss ein Kühlkonzept vorgesehen werden. Bei der Überprüfung der durch die Verlustwärme entstehenden Effekte sowie bei der Auslegung und Optimierung des Kühlsystems kommen CFD-Simulationen zum Einsatz. Weiterhin spielt oftmals auch die mechanische Belastbarkeit von Bauteilen, zum Beispiel des Gehäuses oder der Platine, eine wesentliche Rolle. Diese lässt sich mittels FEM-Simulationen absichern.

3.1.5 Software

Unter dem Begriff Software wird im ursprünglichen Sinne nur die reine Softwareentwicklung verstanden – wie beispielsweise die Entwicklung des Codes zum Auslesen von Sensoren und zur Ansteuerung von Aktoren oder die Erstellung einer Benutzeroberfläche (GUI: Graphical User Interface). Darüber hinaus wird im Rahmen dieser Arbeit unter diesem Punkt auch das gesamte Feld der Regelungstechnik integriert. Grund hierfür ist, dass Regelungen heute meist in Form von Softwarecode auf Mikrocontrollern ablaufen. Prinzipiell ist es auch möglich, Regelungen durch (digitale) elektronische Schaltungen abzubilden. Der höhere Aufwand und die geringere Flexibilität führen jedoch dazu, dass diese Variante nur in seltenen Fällen gewählt wird.

3.1.5.1 Petri-Netze, Zustandsdiagramme, Ablaufdiagramme

Software selbst sowie Regelungen und Steuerung besitzen oftmals (zeit-)diskretes Verhalten. Dies bedeutet, dass das System nur diskrete Zustände annehmen kann und der Übergang zwischen solchen Zuständen das Verhalten des Systems bestimmt. Zur Modellierung dieser Art von Systemverhalten haben sich verschiedene Techniken etabliert. Neben der Beschreibung mittels Zustandsautomaten, die bereits in Kapitel 3.1.4.2 erläutert wurden, sind Petri-Netze eine häufig verwendete Technik. Die Grundidee hierzu wurde von Adam Petri [Petr62] entwickelt und mit der Zeit zu den heute bekannten Petri-Netzen weiterentwickelt. Einer der Hauptvorteile von Petri-Netzen liegt darin, dass sie neben ihrer graphischen Darstellung auf einem mathematischen Formalismus basieren [GiVa01]. Grundsätzlich können bei Petri-Netzen zwei Elementtypen unterschieden werden [Reis13]: Plätze – eine passive Komponente, die diskrete Zustände annehmen kann – sowie Transitionen – eine aktiven Komponente. Kanten stellen die Beziehung zwischen Plätzen und Transitionen dar.

Ablaufdiagramme existieren in verschiedenen Ausführungen. Hier sei beispielhaft die DIN-Norm 66001 zur Informationsverarbeitung [DIN66001] erwähnt. Diese unterscheidet Datenflusspläne, welche den Fluss von Daten durch ein informationsverarbeitendes System beschreiben, sowie Programmablaufpläne, die den Ablauf von Operationen in einem informationsverarbeitenden System in Abhängigkeit der vorhandenen Daten beschreiben [DIN66001]. Aufgrund des Alters dieser DIN-Norm wird dort das Erstellen dieser Pläne mit händischen Mitteln empfohlen. Heute existieren für alle Modellierungstechniken Softwarewerkzeuge, welche die Modellierung entweder rein graphisch ermöglichen oder aber auch mit Logiken hinterlegen. Zudem bieten zahlreiche Werkzeuge auch die Möglichkeit der Simulation. Somit kann das Verhalten des diskreten Systems untersucht werden. Beispielhaft sei hier die Software Matlab von The MathWorks [WWW18] genannt, die mit entsprechenden Erweiterungen sowohl die Modellierung als auch die Simulation aller erwähnten Modellierungstechniken unterstützt.

3.1.5.2 Unified Modeling Language (UML)

UML (Unified Modeling Language) ist eine Modellierungssprache für die Softwareentwicklung, welche von der OMG (Object Management Group) [WWW19] entwickelt wurde. Diese Sprache dient dazu, Systemarchitekten, Softwareingenieuren und Softwareentwicklern Werkzeuge bereitzustellen, die bei der Analyse, Gestaltung und Implementierung softwarebasierter Systeme unterstützen [OMG11a]. UML vereint verschiedene Modellierungstechniken der Softwareent-

wicklung in einer standardisierten Sprache [Rump11]. So sind beispielsweise Zustands- und Ablaufdiagramme in UML direkt implementierbar.

Ein Hauptbestandteil von UML-Modellen sind Diagramme. Diese werden unterteilt in Strukturdiagramme, die den statischen Teil des Systems beschreiben, und Verhaltensdiagramme, die den dynamischen Teil beschreiben [OMG11b].

Aufgrund der weiten Verbreitung von UML existieren unzählige Softwarewerkzeuge, welche die Modellierung mittels UML unterstützen, so etwa Eclipse von The Eclipse Foundation [WWW20] oder Rational Rhapsody der Firma IBM [WWW21].

Weitere Informationen zu UML sowie deren Anwendung können der umfangreichen Literatur und der ebenso umfassenden Onlinedokumentation entnommen werden. Stellvertretend sei hier auf [MiHa06], [OMG11a], [OMG11b] und [Rump11] verwiesen.

3.1.5.3 Blockschaltbilder

Insbesondere in der Regelungstechnik hat sich die Modellierung mithilfe der Blockschaltbildmethode etabliert [Wall95]. Diese Modellierungstechnik bietet die Möglichkeit der abstrakten Modellierung technischer Systeme. Hierbei wird das zu modellierende System aus einzelnen Blöcken zusammengesetzt, die jeweils Eingangs- und Ausgangsgrößen besitzen und ein spezifisches Übertragungsverhalten aufweisen [Noll09]. Dieses Verhalten kann mathematisch über Übertragungsfunktionen beschrieben werden. Die Modellierung des Gesamtsystems erfolgt somit durch eine konsistente Verschaltung der Blöcke, wobei Ausgangssignale eines Blocks als Eingangssignale anderer Blöcke dienen. Meist kann damit ein System aus Standardblöcken aufgebaut werden, beispielsweise ein PID-Regler aus Blöcken für die Differentiation und Integration sowie aus Proportionalgliedern.

Neben der rein graphischen Erstellung von Blockschaltbildern, die auch manuell durchgeführt werden kann, bietet die Modellierung mittels spezieller Softwarewerkzeuge den Vorteil, dass das Gesamtübertragungsverhalten des Systems automatisch durch die Verknüpfung der Einzelblöcke – und damit auch des Übertragungsverhaltens – abgeleitet wird. Ein in Industrie und Forschung hierfür etabliertes Werkzeug ist Matlab/Simulink der Firma The MathWorks [WWW22], welches von der Modellierung bis hin zur Simulation von Blockschaltbildern Unterstützung bietet.

3.1.5.4 Reglersimulation

Die Simulation von Reglern und dem daraus resultierenden Gesamtverhalten des Systems erfolgt auf Basis der mathematischen Beschreibung des Systemverhaltens, im allgemeinen Fall ein System von Differentialgleichungen. Diese mathematische Beschreibung kann beispielsweise von geeigneten Softwarewerkzeugen aus dem Blockschaltbild generiert werden, wie in Kapitel 3.1.5.3 erläutert.

Mittels entsprechender Anfangs- und Randbedingungen kann somit das Gleichungssystem gelöst werden, wodurch das Ausgangsverhalten des Systems bestimmt wird. Bei komplexen Systemen ist das Lösen des Gleichungssystems oftmals analytisch nicht möglich, weshalb hier numerische Verfahren angewandt werden.

Wie bereits erwähnt, wird typischerweise Matlab/Simulink der Firma The MathWorks [WWW22] für die Reglersimulation eingesetzt. In dem Falle, dass analytische Verfahren ausreichen, können auch Computeralgebrasystem, wie etwa Maple der Firma Maplesoft [WWW23], zum Lösen des Gleichungssystems genutzt werden.

3.1.5.5 X-in-the-Loop (XIL)

Der Softwareanteil stellt in technischen Systemen mittlerweile einen bedeutenden Aspekt dar. Im Allgemeinen wird von einem eingebetteten System (englisch: Embedded System) gesprochen, wenn eine Software-/Hardware-Einheit Teil eines technischen Systems ist und darin Überwachungs-, Steuerungs- oder Regelungsaufgaben übernimmt [Bend05].

Im Entwicklungsprozess eingebetteter Systeme werden verschiedene Techniken eingesetzt, um das Verhalten des Systems zu überprüfen. Sehr früh im Prozess kommen Model-in-the-Loop-Tests (MIL) zum Einsatz. Hierbei wird das Modell der späteren Steuer- oder Regelungssoftware – beispielsweise ein Zustandsautomatenmodell – in Kombination mit einem Modell der Umgebung des eingebetteten Systems simuliert [Bend05]. Der nächste Schritt im Entwicklungsprozess ist die Übertragung des Softwaremodells in für das Zielsystem interpretierbaren Softwarecode. Die Software wird auf einem Rechner, jedoch nicht auf der Zielhardware, ausgeführt und die Umgebung simuliert. Dieses Vorgehen wird als Software-in-the-Loop (SIL) bezeichnet. Hierbei werden keine Echtzeitanforderungen gestellt [Iser08]. Ist neben der Software auch die Zielhardware vorhanden, das heißt das gesamte Steuergerät, welches das eingebettete System repräsentiert, kommen Hardware-in-the-Loop-Tests (HIL) zum Einsatz. Hierbei wird das reale Steuergerät in Verbindung mit einer Echtzeit-Umgebungssimulation getestet [Bend05]. Bei al-

len beschriebenen Verfahren stellt das Umgebungsmodell nur den für die Steuerung relevanten Teil dar und emuliert die Schnittstellen mit der Umgebung.

X-in-the-Loop-Techniken verbinden die Vorteile physischer Tests, beispielsweise die höhere Genauigkeit der Ergebnisse, mit denen von Simulationen, beispielsweise die schnellere Ausführung und Variantenbildung.

Insbesondere für Hardware-in-the-Loop-Prüfungen sind spezielle Softwarewerkzeuge und entsprechende echtzeitfähige Hardware erforderlich. Besonders im Automobilsektor haben sich Hardware- und Softwarelösungen der Firma dSpace etabliert. Im Umfeld mechatronischer Systeme kommt zudem häufig die Software CAMel-View mit der entsprechenden Hardware der iXtronics GmbH [WWW24] zum Einsatz.

Weitere Informationen zu X-in-the-Loop-Techniken können beispielsweise [AbBo06], [IsSS99] und [ScZu06] entnommen werden.

3.1.6 Mechatronik

In der Mechatronik-Entwicklung werden einerseits die eingesetzten Modellierungs- und Simulationstechniken aus den Einzeldomänen angewendet. Andererseits wird darüber hinaus eine gemeinsame Basis zur Kommunikation und Kooperation benötigt [MöGa02]. Daher existieren auch domänenübergreifende Techniken, welche diesen Aspekt adressieren sollen.

3.1.6.1 Bondgraphen

Bondgraphen sind eine graphische Beschreibungsform physikalischer Systeme, die unabhängig von domänenspezifischen Notationen arbeitet. Bondgraphen besteht aus drei Elementtypen [VWBZ09]: Knoten repräsentieren idealisierte Subsysteme oder Komponenten eines Systems. Kanten (englisch: Bonds) modellieren den idealisierten Energieaustausch zwischen Knoten, wobei Halbpfeile die Richtung des Energieflusses angeben. Die übertragene Leistung wird durch das Produkt der beiden Größen Effort und Flow beschrieben. Effort und Flow haben in den Einzeldomänen jeweilige Entsprechungen, beispielsweise in der Mechanik Kraft und Geschwindigkeit. Damit aus der graphischen Repräsentation auch die mathematische Beschreibung abgeleitet werden kann – und somit das System auch simuliert werden kann – erfolgt die Modellierung oftmals mit Standardelementen, die bereits mathematisch beschrieben sind [Rodd13].

Der Vorteil von Bondgraphen liegt in ihrer Anwendbarkeit auf alle Arten von physikalischen Systemen sowie in der Tatsache, dass sie gleichzeitig die topologische und die mathematische Struktur repräsentieren [Cell91]. Ein Nachteil von Bondgraphen ist ihr abstrakter Charakter, wodurch es nur schwer möglich ist, auf den ersten Blick die Systemstruktur zu durchblicken [Cell91], [PDSK01].

Für weitergehende Erläuterungen zur Modellierung mit Bondgraphen und der Ableitung von Simulationsmodellen sei auf [Boru00] und [Rodd13] verwiesen.

3.1.6.2 Systems Modeling Language (SysML)

SysML ist die Abkürzung für Systems Modeling Language der Object Management Group (OMG) [WWW25]. Diese graphische, diagrammbasierte Modellierungssprache basiert auf der bereits in Kapitel 3.1.5.2 beschriebenen Sprache UML, ergänzt diese jedoch um wichtige Aspekte des Systems Engineering, wie etwa die Möglichkeit zur Modellierung von Anforderungen sowie kontinuierlicher Systeme [Weil08]. SysML dient dazu, Struktur und Verhalten eines Systems zu beschreiben und hat sich in den letzten Jahren als eines der Standardwerkzeuge im Bereich des Modellbasierten Systems Engineering (siehe Kapitel 3.2.6) etabliert. Darüber hinaus gewinnt die Modellierungssprache auch im Bereich der Mechatronik immer weiter an Bedeutung.

Da SysML-Modelle nicht direkt simulierbar sind [HuRG07], kann hiermit das Systemverhalten nicht direkt analysiert werden. Jedoch gibt es im Bereich der Forschung Bestrebungen, SysML mit simulierbaren Modellen zu koppeln, beispielsweise [HuRG07] und [JKPB08].

Weitere Informationen zu SysML und deren Anwendung geben zum Beispiel Weilkiens [Weil08] und Follmer et al. [FHPZ10]. Eigner et al. zeigen darüber hinaus eine Integration von Systemmodellen auf Basis von SysML in PLM-Systemen [EGDF14].

3.1.6.3 VHDL-AMS, Verilog-AMS

Wie bereits in Kapitel 3.1.4.3 beschrieben, wurden die Hardwarebeschreibungssprachen VHDL-AMS und Verilog-AMS entwickelt, um neben diskretem Verhalten auch kontinuierliches Verhalten modellieren zu können. Mechatronische Systeme sind ebenfalls von einer Mischung diskreten und kontinuierlichen Verhaltens geprägt. Beispielsweise ist der mechanische Teil meist ein kontinuierliches System, der Softwareteil hingegen ein diskretes System. Da VHDL-AMS und Verilog-AMS in ihren Formulierungen weitgehend domänenneutral sind, eignen sie sich auch für die Beschreibung mechatronischer Systeme. Ein Vergleich der beiden Beschreibungssprachen für die Modellierung interdisziplinärer Systeme ist in [PeLV06] gegeben.

3.1.6.4 Modelica und ähnliche Ansätze

Modelica ist eine objektorientierte, gleichungsbasierte Sprache zur Modellierung komplexer, interdisziplinärer physikalischer Systeme und wird von der Modelica Association [WWW26] entwickelt und gepflegt. Der Benutzer erstellt ein Modell in der Regel über Schaltbilder, indem er einzelne Komponenten aus Modellbibliotheken – für das Bibliothekenkonzept ist der objektorientierte Ansatz von Vorteil – wählt und verknüpft. Die Darstellung dieser Schaltbilder ähnelt den domänenspezifischen Darstellungsformen. Somit eignet sich Modelica auch zur Beschreibung der Systemarchitektur [Frit04].

Das eigentliche Modelica-Modell ist eine textuelle Beschreibung, mit der die Komponenten des Systems definiert werden und mit der Komponentenbibliotheken erstellt werden können [Mode12]. Hierin ist das Verhalten jedes Elements mathematisch beschrieben. Modelica-basierte Simulationsumgebungen dienen dazu, die erwähnte graphische Benutzeroberfläche bereitzustellen sowie Simulationen durchzuführen. Der Nutzer ist zudem nicht auf die Benutzung vorhandener Bibliotheken angewiesen, sondern kann eigene Modelle mittels der textuellen Beschreibung erstellen.

Modelica hat sich in den letzten Jahren für die Modellierung und Simulation mechatronischer Systeme stark verbreitet. Entsprechend existiert eine große Menge an Softwarewerkzeugen, welche die Modellierung und Simulation unterstützen. Beispielhaft seien hier OpenModelica von Open Source Modelica Consortium [WWW27], Dymola der Firma Dassault Systèmes [WWW28] sowie SimulationX der ITI GmbH [WWW29] genannt.

Detailliertere Ausführungen zum Sprachkonzept von Modelica sowie dessen Anwendung sind in der umfangreichen Literatur zu finden, für die hier nur stellvertretend [ElMa97], [Frit04], [Mode12] und [Till01] genannt seien. Einen Vergleich von Modelica und der zuvor beschriebenen Beschreibungssprache VHDL-AMS im Kontext domänenübergreifender Systeme liefern Schwarz et al. [SCHS01].

Neben Modelica(-werkzeugen) wurden in der letzten Zeit diverse weitere Modellierungs- und Simulationswerkzeuge entwickelt. Diese basieren prinzipiell auf einem ähnlichen Grundkonzept, jedoch bauen sie nicht auf einer derart standardisierten Basis wie der Modelica-Sprache auf. Vielmehr nutzen sie proprietäre Sprachen. Beispiele sind das Zusatzmodul Simscape für Matlab von The MathWorks [WWW30], 20-sim der Firma Controllab Products [WWW31] oder CAMel-View der iXtronics GmbH [WWW24].

Auf Forschungsseite sei hier der Ansatz von Paredis et al. [PDSK01] erwähnt, der ebenfalls zur Modellierung und Simulation interdisziplinärer Systeme dient. Die portbasierte Beschreibung ähnelt der energiebasierten Betrachtung von Bondgraphen. Darüber hinaus unterscheiden Paredis et al. bei ihren Modellen zwischen Struktur und Verhalten. Beide sind unabhängig voneinander abgebildet, wodurch eine Anpassung des Modells an den steigenden Detaillierungsgrad im Entwicklungsprozess möglich ist. Zudem ähnelt der Ansatz den von Modelica oder VHDL-AMS bekannten Ansätzen, weshalb Paredis et al. auch auf diese Sprachen zurückgreifen. Eine Implementierung des Ansatzes von Paredis et al. in einem Softwarewerkzeug ist dem Autor nicht bekannt.

3.1.6.5 Integrierte Softwareumgebungen

Wie in den Kapitel 3.1.2 bis 3.1.5 bereits gezeigt wurde, hat jede der Einzeldomänen leistungsfähige, etablierte Techniken und Werkzeuge entwickelt. Insbesondere auf Systemebene ist es jedoch wenig sinnvoll, derartige Insellösungen zu verwenden, da sie den Prozess- und Datenfluss unterbrechen [PBFG07]. Daher existieren seit einigen Jahren Bemühungen von Softwareherstellern, diese Insellösungen in integrierten Softwareumgebungen zu vereinen. Meist dient hierbei ein CAD-System als Grundlage und das CAD-Modell als Modellbasis. Den prinzipiellen Vorteilen bezüglich der gemeinsamen Datenbasis, der Wiederverwendbarkeit von Modellen und der integrierten Benutzerumgebung stehen jedoch auch Nachteile gegenüber. So sind oftmals spezialisierte Werkzeuge leistungsfähiger. Zudem ist nicht für alle Simulationen ein vollständiges CAD-Modell erforderlich. Da dieses jedoch bei integrierten Softwareumgebungen meist erforderlich ist, ist die Modellierung in solchen Fällen mit Mehraufwand verbunden.

Die Integration der Standardsimulationen aus dem Bereich der Mechanik ist mittlerweile in den meisten CAD-Werkzeugen gegeben. Ebenso werden auch in der Elektronik integrierte Lösungen angeboten. Darüber hinaus entstehen derzeit erste Ansätze bezüglich der Integration mechatronischer Simulationen. Hier sei die Software CATIA Systems der Firma Dassault Systemès [WWW32] erwähnt. Diese soll mit dem RFLP-Ansatz (Requirements, Functional, Logical, Physical) [KlKr13] die Mechatronikentwicklung von der Anforderungsmodellierung bis zur CAD-Implementierung unterstützen. Für die Simulation nutzt CATIA Systems Modelica. Die aus der Anwendung dieser Software bekannten Probleme zeigen jedoch, dass eine derartige Integration noch nicht vollständig abgeschlossen ist.

3.1.6.6 Gekoppelte Simulationen

Die Probleme der integrierten Softwarewerkzeuge, insbesondere in Bezug auf die Leistungsfähigkeit, führen oftmals dazu, dass die Nutzung von Insellösungen unumgänglich ist. Jedoch bewirken komplexe, vernetzte mechatronische Systeme auch, dass Simulationsergebnisse voneinander abhängig sind. Ein manuelles, iteratives Wechseln zwischen Simulationswerkzeugen ist daher oftmals nicht effizient. Aus diesem Grunde kommen hier immer häufiger gekoppelte Simulationen [2] zum Einsatz: Die individuellen Simulationswerkzeuge besitzen gemeinsame Schnittstellen, über die simultan Ergebnisse ausgetauscht werden. So ist es beispielsweise bei der Simulation eines geregelten Mehrkörpersystems möglich, das mechanische System in einer spezialisierten MKS-Software zu simulieren. Dieses Modell dient in einer gekoppelten Simulation als Regelstrecke für den in einem spezialisierten Werkzeug simulierten Regler. Die simulierten Stellgrößen werden über die gemeinsame Schnittstelle an die MKS-Software zurückübermittelt, in der das geregelte System somit simuliert und visualisiert werden kann.

Nachteilig ist, dass gekoppelte Simulationen die Kompatibilität der individuellen Simulationswerkzeuge erfordern und darüber hinaus oftmals sehr rechen- und zeitintensiv sind.

3.2 Entwicklungsmethodiken

Wie bereits in Kapitel 2.2 beschrieben, liegen die Ursprünge von Entwicklungsmethodiken im Bereich des Maschinenbaus, weshalb in diesem Bereich zahlreiche Arbeiten existieren. Aufgrund der Fokussierung dieser Arbeit auf mechatronische Systeme sollen aber auch die anderen Teildisziplinen, Elektronik und Informationstechnik, in der Analyse berücksichtigt werden. Während im Bereich der Elektronik Entwicklungsmethodiken traditionell weniger Berücksichtigung erhalten – was sich in einer geringen Anzahl an Arbeiten auf diesem Gebiet widerspiegelt – haben sich im Bereich der Informationstechnik und Softwareentwicklung in den letzten Jahren eine Vielzahl von Methodiken etabliert. Neben den drei Einzeldomänen der Mechatronik finden sich zahlreiche Methodiken aus dem Bereich der Mechatronik selbst, die vorwiegend in jüngerer Zeit entstanden sind. Darüber hinaus wird im Folgenden kurz der Bereich des Systems Engineering betrachtet. Insbesondere dient diese Betrachtung zur Abgrenzung des Systems Engine-

[2] Neben dem Begriff der gekoppelten Simulation ist auch häufig der Begriff der Co-Simulation zu finden. Diese unterscheiden sich in technischen Details, wie in [GeKL06] erläutert. Im Rahmen dieser Arbeit werden diese Details jedoch nicht betrachtet, weshalb die Begriffe synonym verwendet werden können.

ering zur Mechatronikentwicklung, da diese oftmals als synonym angesehen werden, was jedoch nach den ursprünglichen Definitionen nur bedingt korrekt ist. Abschließend sollen die Entwicklungsmethodiken verglichen und bewertet werden. Dies erfolgt anhand vordefinierter Kriterien, die bei der folgenden Analyse besondere Berücksichtigung finden und im Folgenden erläutert werden.

Kapitel 3.2 basiert auf den Veröffentlichungen [DoVi12a] und [DoVi13] des Autors.

3.2.1 Analysekriterien

Für die Analyse der Entwicklungsmethodiken werden die untenstehenden Kriterien verwendet. Diese spiegeln wichtige Aspekte simulationsbasierter Entwicklung wider, welche sich aus den in Kapitel 2 beschriebenen Definitionen ableiten. Zudem fließen in diese Kriterien Erkenntnisse aus Literaturstudien sowie aus diversen Diskussionen mit Fachexperten ein.

3.2.1.1 Interdisziplinarität

Mechatronik als interdisziplinäres Fachgebiet beruht auf der Kombination verschiedener Teildisziplinen. Diese haben jeweils eigene Vorgehensweisen entwickelt. Für eine erfolgreiche, synergetische Entwicklung mechatronischer Systeme müssen diese Teildisziplinen daher sowohl kommunizieren als auch kooperieren [Möhr04], [PBFG07], [VDI2206]. Ein solches interdisziplinäres Zusammenwirken muss in einer Entwicklungsmethodik ermöglicht und gefördert werden.

3.2.1.2 Computerunterstützung

Die Entwicklung mechatronischer Systeme ist gekennzeichnet durch hohe Komplexität, sowohl bezüglich des Produktes selbst als auch hinsichtlich dessen Entwicklungsprozesses. Diese Komplexität entsteht durch eine große Anzahl an Komponenten und Subsystemen aus unterschiedlichen Disziplinen, die auch während der Entwicklung zusammenarbeiten müssen. Zu deren Beherrschung kann der Einsatz von rechnergestützten Methoden und Verfahren eingesetzt werden, weshalb nach Lückel [LüKS00] die Mechatronikentwicklung bereits von frühen Phasen an eine „ganzheitliche, fachübergreifende Arbeitsweise auf der Basis rechnergestützter Methoden“ berücksichtigen sollte. Daher wird mit diesem Kriterium berücksichtigt, ob und wie Computerunterstützung in den einzelnen Entwicklungsmethodiken integriert ist. Hierzu zählen alle rechnerunterstützten Methoden, von CAD über CAE bis hin zu PDM-Systemen.

3.2.1.3 Eigenschaftsanalyse

Prinzipiell lassen sich nach Hubka und Eder [HuEd88] drei Anwendungsfälle für Analysen[3] im Entwicklungsprozess unterscheiden:

- Überprüfung, wie gut ein realisiertes System ist
- Überprüfung, ob das System die durch das Problem formulierten Anforderungen erfüllt
- Vergleich verschiedener Systeme, die als Lösung für ein Problem dienen

In allen drei Fällen muss überprüft werden, ob die tatsächlichen Eigenschaften des Systems die Anforderungen erfüllen. Allgemein erfolgt dies in einem Schritt zur Eigenschaftsanalyse. Hierfür gibt es zahlreiche Methoden, die je nach Prozessphase und erforderlicher Genauigkeit eingesetzt werden. Zu diesen Methoden zählen beispielsweise Schätzungen, Vergleiche, Berechnungen, Simulationen und Versuche [Lind07]. Die Berücksichtigung von Eigenschaftsanalysen im Allgemeinen wird mittels dieses Kriteriums bewertet. Aufgrund ihrer Bedeutung in dieser Arbeit werden Simulationen jedoch in einem eigenen Kriterium betrachtet.

3.2.1.4 Einsatz von Simulationstechniken

Der Einsatz von Simulation verdrängt immer stärker klassische Versuche mit physischen Prototypen [Aber06]. Dies ist in den diversen Vorteilen begründet, die das unterschiedliche Vorgehen bei der Simulation im Vergleich zu realen Tests bietet. So ist die Erstellung von Simulationsmodellen oftmals weniger aufwendig als das Anfertigen physischer Prototypen. Insbesondere aber sind virtuelle Versuche an vorhandenen Modellen, also Simulationen, meist wesentlich einfacher und schneller durchzuführen als reale Tests. Dadurch ergibt sich die Möglichkeit, verhältnismäßig einfach verschiedene Varianten und Konzepte zu simulieren. Zudem können Simulationen bereits sehr früh im Entwicklungsprozess eingesetzt werden, da fehlende Informationen entweder durch vereinfachte Modelle oder durch Simulation verschiedener Parametervarianten ermöglicht werden. Die durchgängige Einbindung von Simulation in den Entwicklungsprozess ist daher bereits sehr früh und als durchgängige Methode zur Eigenschaftsanalyse möglich und sinnvoll. Hierdurch ergeben sich jedoch sowohl neue als auch veränderte Anforderungen an den Entwicklungsprozess im Vergleich zu klassischen Tests, was mit diesem Kriterium berücksichtigt werden soll.

[3] Hubka und Eder verwenden hierfür den Begriff der Evaluation, welcher im Rahmen der Arbeit synonym verwendet werden kann.

3.2.1.5 Iterationen

Iterationen beschreiben das wiederholte, zyklische Durchlaufen von Arbeitsschritten [Ehrl09]. Gemäß dieser Definition ist der reale Entwicklungsprozess von Natur aus iterativ [Ullm10]. Beim Einsatz später Eigenschaftsanalyse durch physische Prototypen ergeben sich bei Änderungen zwangsläufig Iterationen, die einen großen Bereich des Entwicklungsprozesses überspannen – im Folgenden Makroiterationen genannt. Ein solches Vorgehen ist bei Änderungen sowohl zeit- als auch kostenintensiv. Durch den durchgängigen Einsatz von Simulation parallel zu Entwurfs- und Gestaltungstätigkeiten lassen sich Eigenschaften sehr eng verzahnt und direkt analysieren. Somit decken Iterationen jeweils nur einen sehr kleinen Bereich des Entwicklungsprozesses ab – im Folgenden Mikroiterationen genannt – was eine wesentliche Möglichkeit zur Einsparung von Zeit und Kosten darstellt. Entsprechend können mehrere Mikroiterationen genutzt werden, um ein Konzept oder einen Entwurf schrittweise zu verbessern. Dies resultiert in ersten Prototypen oder Produkten, die bereits eine sehr hohe Reife besitzen. In Entwicklungsmethodiken sollte die Möglichkeit für Mikroiterationen vorgesehen sein. Zudem soll mit diesem Kriterium analysiert werden, ob die Methodik Hilfestellung gibt, wann eine Iteration erforderlich wird, wie diese gestaltet werden muss und welche Auswirkungen sie besitzt.

3.2.1.6 Umgang mit Analyseergebnissen

Das Ziel von Analysen – und von Simulation im Speziellen – ist die Erlangung von Erkenntnissen über das System. Auf diesen Erkenntnissen basierend kann das System optimiert werden, sodass die Anforderungen erfüllt werden. Hierzu müssen Informationen aus den Analyseergebnissen extrahiert werden, in der Regel durch Interpretationen von Entwicklern. Anschließend können diese Informationen genutzt werden, um das System zu optimieren, was unter anderem den Beginn einer Iteration bedeuten kann. Der Umgang mit Analyseergebnissen ist daher ein entscheidender Faktor und sollte in einer Methodik berücksichtigt werden, um Entwickler bei der Entwurfsoptimierung zu unterstützen.

Im Folgenden wird eine Auswahl von Entwicklungsmethodiken aus den Bereichen Mechanik, Elektronik, Software, Mechatronik und Systems Engineering bezüglich der definierten Kriterien analysiert. Diese Auswahl erhebt keinen Anspruch auf Vollständigkeit, repräsentiert jedoch nach Ansicht des Autors die gesamte Bandbreite der etablierten Methodiken.

3.2.2 Mechanik

3.2.2.1 Produktentwicklung nach Pahl und Beitz

Eine der etabliertesten Methodiken aus dem Bereich der Mechanik stammt aus dem Standardwerk „Konstruktionslehre“ von Pahl und Beitz, welches bereits 1977 in der 1. Auflage erschienen und aktuell in der 8. Auflage verfügbar ist [FeGr13][4]. Demnach gliedert sich der in Abbildung 3.1 dargestellte Entwicklungsprozess in vier Hauptphasen:

- Planen und Klären der Aufgabe

  Ausgangspunkt des Entwicklungsprozesses ist eine Aufgabenstellung, auf deren Basis Anforderungen abgeleitet werden, die an das zu entwickelnde System gestellt werden.

- Konzipieren

  Basierend auf den Anforderungen werden zunächst Funktionsstrukturen erstellt, die zur Abstraktion der Problemstellung dienen. Anschließend werden für die einzelnen Funktionen Wirkstrukturen erstellt, die dann zu prinzipiellen Lösungskonzepten weiterentwickelt werden.

- Entwerfen

  Die prinzipiellen Lösungskonzepte werden unter Berücksichtigung technischer und wirtschaftlicher Aspekte detailliert. Nach Auswahl eines Gesamtentwurfs wird dieser optimiert und ausgestaltet. Hierzu zählt auch die quantitative Definition von Merkmalen.

- Ausarbeiten

  Basierend auf der gestalterisch festgelegten Lösung wird die Lösung so ausdetailliert, dass alle herstellungstechnisch relevanten Aspekte definiert sind. Hierzu zählen beispielsweise die Festlegung von Werkstoffen und Herstellungsverfahren. In dieser Phase werden auch alle für die Realisierung des Produktes relevanten Dokumente erzeugt, wie etwa Stücklisten und Fertigungszeichnungen.

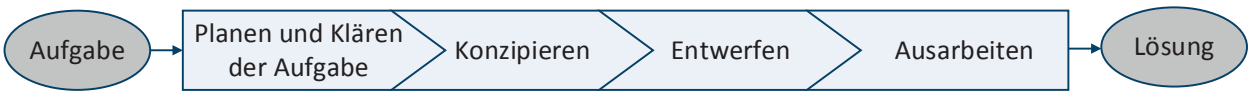


Abbildung 3.1: Prozessschritte nach Pahl und Beitz [PBFG07]

[4] Grundlage für die hier vorgenommene Analyse ist jedoch die 7. Auflage [PBFG07].

In den Phasen Konzipieren und Entwerfen erfolgt am Ende stets eine Analyse der Eigenschaften, entweder qualitativ oder aber auch quantitativ. Hierbei werden verschiedene geeignete Methoden erwähnt, die jedoch sehr stark auf physische Prototypen zielen. Simulationen werden zwar als eine der Methoden zur Eigenschaftsanalyse erwähnt, jedoch stammen diese nur aus dem Bereich der Mechanik und werden auch nicht weiter in den Entwicklungsprozess eingebunden.

Die Analysen dienen hauptsächlich dem Vergleich verschiedener Lösungen oder Konzepte. Im Gegensatz zu den Entwurfstätigkeiten in den einzelnen Phasen ist jedoch die Beschreibung der Analyseschritte nur sehr oberflächlich. Auch wird der Umgang mit Analyseergebnissen kaum angesprochen.

Das Thema Computerunterstützung wird ausführlicher behandelt. Aber auch hier zeigt sich der Mechanik-Fokus der gesamten Methodik. Insbesondere werden hier geometrieorientierte Methoden und Werkzeuge diskutiert.

Obwohl darauf hingewiesen wird, dass das Entwicklungsvorgehen stark iterativ ist, wird nicht erwähnt, wodurch Iterationen im Prozess entstehen und zu welchen Entwicklungsphasen sie zurückführen. Zudem beschränken sich Iterationen hauptsächlich auf Makroiterationen, Mikroiterationen werden nicht diskutiert.

Trotz der Ausrichtung der Gesamtmethodik auf Problemstellungen des Maschinenbaus ist die Methodik prinzipiell auch auf interdisziplinäre Entwicklungsprojekte anwendbar. Explizit wird dies für die Mechatronik erwähnt. Es erfolgt jedoch kein Hinweis darauf, wie sich das Entwicklungsvorgehen für solche Projekte ändert. Auch werden beispielsweise Kommunikation und Kooperation der involvierten Fachbereiche nicht diskutiert.

An dieser Stelle sei darauf hingewiesen, dass die Grundstruktur des Vorgehens nach Pahl und Beitz in den meisten Methodiken wiederzufinden ist [EGDF14].

3.2.2.2 VDI-Richtlinie 2221

Die VDI-Richtlinie 2221 „Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte“ [VDI2221] dient als Dachrichtlinie für die VDI-Richtlinien VDI 2222 [VDI2222], VDI 2223 [VDI2223]. Das Entwicklungsvorgehen, dargestellt in Abbildung 3.2, beginnt mit der Aufgabenstellung und gliedert sich danach in sieben Schritte. Inhaltlich orientiert sich die Richtlinie dabei sehr stark an der Methodik von Pahl und Beitz, weshalb an dieser Stelle nicht mehr weiter auf das Vorgehen eingegangen wird.

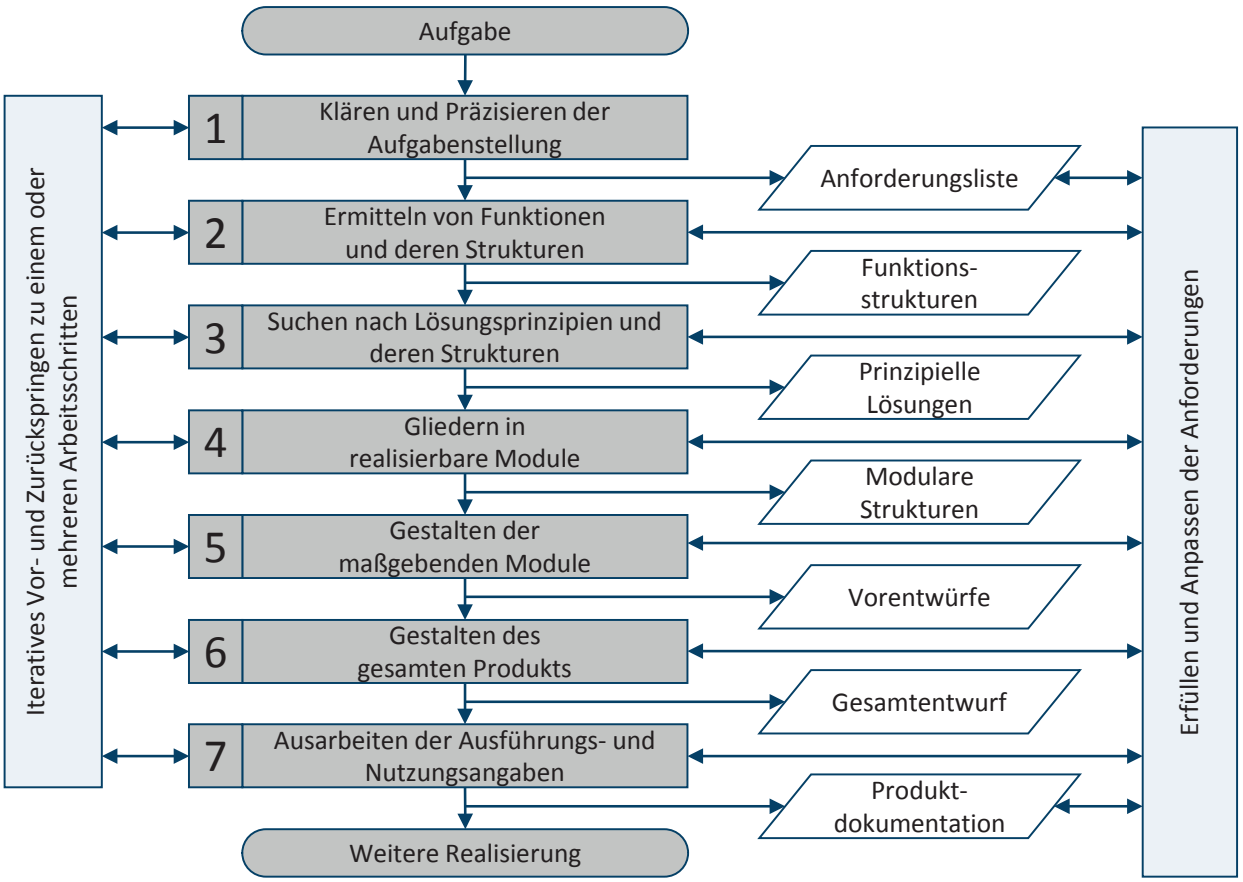


Abbildung 3.2: Vorgehensmodell nach VDI 2221 [VDI2221]

Eigenschaftsanalysen sind prinzipiell am Ende jeder Phase vorgesehen. Allerdings ist das Vorgehen stark an den klassischen Analysen mit physischen Prototypen orientiert. Konkrete Hilfestellungen bezüglich des Einsatzes von Eigenschaftsanalysen und den dafür möglichen Methoden oder Werkzeugen werden nicht gegeben. Dabei spielen Simulationen in der Richtlinie eine untergeordnete Rolle. Zwar wird an diversen Stellen erwähnt, dass Simulationen zur Eigenschaftsanalyse eingesetzt werden können, weiter geht die Richtlinie hier jedoch nicht ins Detail. Zudem wird auch hier, ähnlich wie bei Pahl und Beitz, die Beschränkung auf geometrieorientierte Techniken offensichtlich.

Der Umgang mit Ergebnissen aus Analysen wird in der Richtlinie nicht explizit thematisiert. Einzig wird erwähnt, dass Ergebnisse zur Entscheidung sowie zur Auswahl und Optimierung von Lösungen dienen können. Ein genauer Leitfaden, etwa wie durch die Ergebnisse Iterationen initiiert werden, wird nicht gegeben.

Computerunterstützung wird in der Richtlinie als eigenes Thema diskutiert. Entsprechend dem Alter der Richtlinien – sie stammt von 1993 – bilden die erwähnten Techniken jedoch nicht mehr den aktuellen Stand ab. Dabei bezieht sich die Diskussion hauptsächlich auf geometrieori-

entierte, also maschinenbaunahe Techniken. Eine Einbindung dieser Techniken erfolgt in der VDI-Richtlinie 2223 in Form einer tabellarischen Zuordnung der Techniken zu den Tätigkeiten aus den Phasen gemäß Abbildung 3.2. Eine weitergehende Diskussion oder konkretere Einbindung in den Entwicklungsprozess erfolgt nicht.

Die einzelnen Arbeitsschritte der Richtlinie werden explizit mehrmals iterativ durchlaufen. Zudem ist ein iteratives Springen zwischen Arbeitsschritten und damit zwischen Entwicklungsphasen vorgesehen, wie in Abbildung 3.2 dargestellt. Jedoch erfolgt keine weitere Detaillierung dieser Iterationen: Kriterien und die Auswirkungen von Iterationen auf einzelne Entwicklungsphasen werden nicht thematisiert.

Entsprechend der Methodik von Pahl und Beitz ist das Vorgehen der VDI-Richtlinie an den Methoden und Begrifflichkeiten des Maschinenbaus ausgerichtet. Zwar wird an vielen Stellen erwähnt, dass das Vorgehen auch für andere Disziplinen anwendbar ist, jedoch wird die tatsächliche Anwendung kaum erläutert. Das Zusammenwirken verschiedener Disziplinen bei der Entwicklung wird als „weitestgehend getrennt“ [VDI2221] angesehen, auch wenn erwähnt wird, dass das gemeinsame Entwickeln einer Prinziplösung sinnvoll sein kann. Eine weitergehende Diskussion hierzu erfolgt jedoch nicht.

3.2.2.3 Integrierte Produktentwicklung nach Ehrlenspiel

Ehrlenspiel [Ehrl09] beschreibt eine Gesamtmethodik zur integrierten Produktentwicklung. Unter dem Begriff der integrierten Produktentwicklungsmethodik wird dabei eine „Methodik zur Produkterstellung unter besonderer Berücksichtigung der Zielorientierung und Zusammenarbeit der beteiligten Menschen verstanden“ [Ehrl09]. Dabei ist diese über die reine Konstruktion hinaus auch in den Bereichen Produktion, Vertrieb, Materialwirtschaft und Controlling anwendbar. Kern der Methodik ist der Vorgehenszyklus aus Abbildung 3.3, der das Ziel verfolgt, durch eine Abfolge von Arbeitsschritten ausgehend von einer Problemstellung eine Lösung zu generieren. Der Zyklus basiert auf einer Abfolge von TOTE-Schemata (Test-Operate-Test-Exit) und ähnelt in seinen Grundzügen den bereits erläuterten Vorgehensmodellen nach Pahl und Beitz sowie nach VDI 2221.

Wie bereits erwähnt, ist die Eigenschaftsanalyse im Vorgehenszyklus als wichtiger Bestandteil abgebildet. Diese Analyse beschränkt sich jedoch auf einen einzelnen Zeitpunkt im Prozess, und zwar am Ende, wodurch keine kontinuierliche Analyse möglich ist. Ehrlenspiel beschreibt die Techniken und Methoden zur Eigenschaftsanalyse gesondert. Im Fokus stehen hierbei Berech-

nungs- sowie Versuchsmethoden, die auch anhand von Beispielen erläutert werden. Simulationen werden zwar als wichtig und hilfreich angesehen, jedoch erfolgt keinerlei Diskussion von Simulationstechniken und deren Anwendung im Entwicklungsprozess. Die kurzen Ausführungen beschränken sich zudem auf die Finite-Elemente-Methode, wodurch der Fokus auf Produkte des Maschinenbaus deutlich wird.

Im Vorgehenszyklus ist explizit erwähnt, dass Analyseergebnisse genutzt werden, um Eigenschaften der Lösungen zu ermitteln, und diese wiederum als Grundlage einer Bewertung verschiedener Lösungen dienen. Jedoch wird hier nur der Fall besprochen, dass aus verschiedenen Lösungen eine geeignete ausgewählt werden soll. Der Fall der Optimierung einer Lösung wird nicht behandelt.

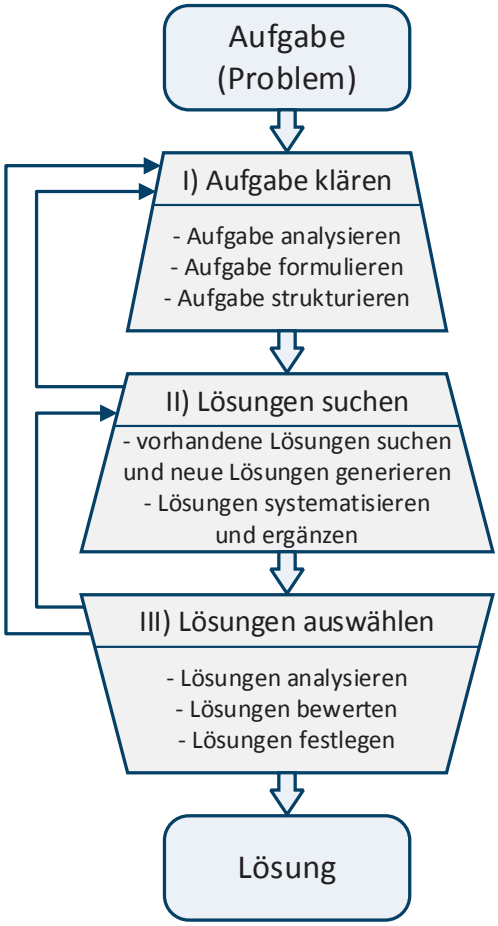


Abbildung 3.3: Vorgehenszyklus in der Integrierten Produktentwicklungsmethodik [Ehrl09]

Bezüglich Computerunterstützung werden nur vereinzelt an entsprechenden Stellen einzelne Techniken erwähnt. Auch hier wird durch die erwähnten Methoden, etwa CAD, der Fokus auf Produkte des Maschinenbaus deutlich. Eine detaillierte und umfassende Beschreibung der aktuellen Möglichkeiten der Computerunterstützung erfolgt jedoch nicht und entsprechend wird auch keine Einbindung in den Entwicklungsprozess diskutiert.

Iterationen werden von Ehrlenspiel als wichtiger Bestandteil des Entwicklungsprozess angesehen und sind daher auch im Vorgehenszyklus aus Abbildung 3.3 dargestellt. Hierbei können Iterationen sowohl einzelne als auch mehrere Ebenen des Entwicklungsprozesses überspringen. Prinzipiell ist durch die Ableitung aus dem TOTE-Schema auch eine Steuerungsmöglichkeit des Prozesses gegeben, welche jedoch nicht detaillierter ausgeführt wird. Iterationen im Bezug zu Optimierungsaufgaben werden nicht angesprochen.

Aufgrund der Fokussierung der Methodik auf die integrierte Produkterstellung, bei der alle Abteilungen und Spezialisten eng zusammenarbeiten müssen [Ehrl09], ist der Aspekt der Interdisziplinarität tief verankert. Hierzu werden diverse Methoden beschrieben. Interdisziplinarität bezieht sich hier jedoch auf Zusammenarbeit von Abteilungen, die in verschiedenen Phasen des Produktlebenszyklus tätig sind, beispielsweise Produktentwicklung, Produktion und Vertrieb. Die Entwicklung interdisziplinärer Produkte, bei der während der Entwicklung selbst verschiedene Disziplinen interagieren müssen, wird nicht besprochen.

3.2.2.4 Münchener Vorgehensmodell (MVM) nach Lindemann

Das in Abbildung 3.4 dargestellte Münchener Vorgehensmodell wurde von Lindemann [Lind07] entwickelt und basiert auf sieben Schritten, die in einer netzwerkartigen Struktur angeordnet sind. Diese Struktur dient dazu, die einzelnen Schritte in freier Reihenfolge durchlaufen zu können, wodurch sich das Modell von den meisten, linearen Vorgehensmodellen unterscheidet. Jedoch ist ein Standardvorgehen vorgeschlagen, welches in Abbildung 3.4 durch Pfeile dargestellt ist. Dieses ist wiederum den Vorgehen aus VDI 2221 und von Pahl und Beitz ähnlich.

Die Zielplanung befasst sich mit der Identifikation von Zielen und Maßnahmen zur Verbesserung eines Problems, die darauffolgende Zielanalyse beinhaltet die Formulierung der Anforderungen. Während der Problemstrukturierung wird das Problem abstrahiert und bei Bedarf in Teilprobleme aufgeteilt. Basierend darauf werden Lösungen ermittelt, für welche die Eigenschaften ermittelt werden, um diese mit den Anforderungen abzugleichen. Anhand der analysierten Eigenschaften wird eine Entscheidung für die beste Lösung getroffen. Die Absicherung der Zielerreichung dient dazu, Risiken während des Entwicklungsprozesses zu minimieren.

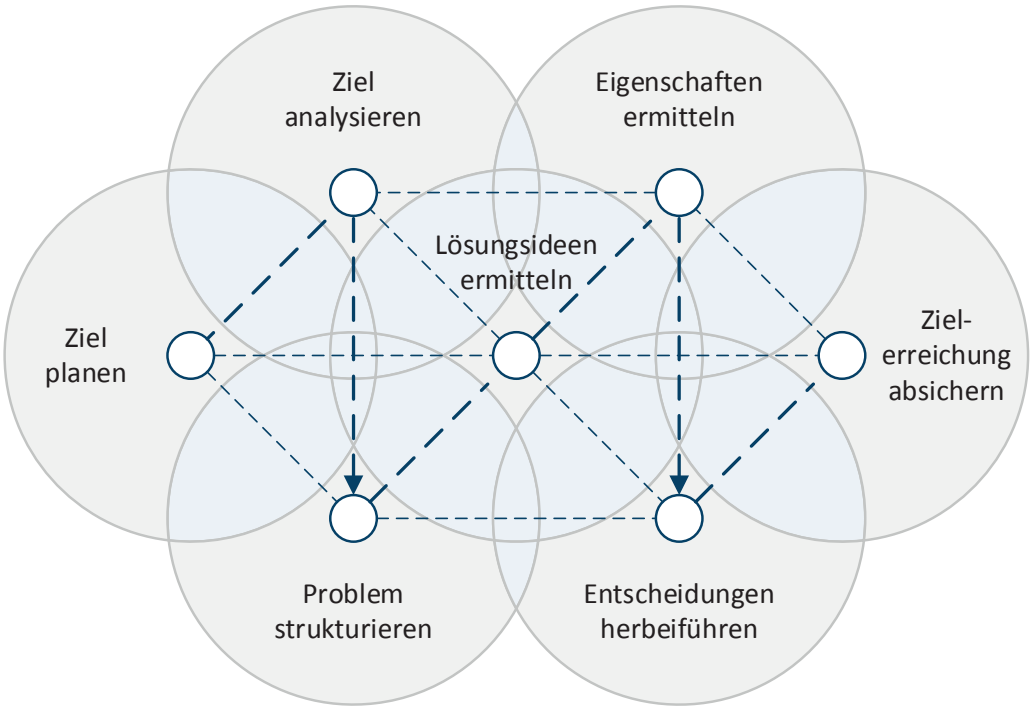


Abbildung 3.4: Münchener Vorgehensmodell (MVM), nach [Lind07]

Eigenschaftsanalysen sind als ein eigener Schritt in dem Netzwerk vorgesehen. Im Standardvorgehen wird dieser Schritt jedoch nur einmal durchlaufen, wodurch eine kontinuierliche Analyse im Entwicklungsprozess nicht möglich ist. Prinzipiell ist es jedoch denkbar, durch die Netzwerkstruktur mehrere Schritte zur Eigenschaftsanalyse in den Entwicklungsprozess einzubinden. Es werden verschiedene Methoden zur Eigenschaftsanalyse diskutiert, darunter auch Simulationen. Diese beziehen sich jedoch auf Finite-Elemente- sowie Mehrkörpersimulationen, und damit auf mechanikbasierte Techniken. Die prozessseitige Einbindung von Simulation wird nicht weiter diskutiert.

Prinzipiell werden die Ergebnisse der Eigenschaftsanalyse im Schritt „Entscheidungen herbeiführen“ genutzt, um verschiedene Lösungsalternativen miteinander zu vergleichen. Wie Ergebnisse im Detail in den Bewertungen genutzt werden, bleibt jedoch unerwähnt. Zudem werden Analysen nur zum Zweck des Vergleichs von Lösungen genutzt. Der Einsatz als Optimierungswerkzeug ist nicht direkt berücksichtigt.

Abgesehen von vereinzelten Hinweisen werden computerbasierte Techniken in der Methodik nicht weiter diskutiert. Entsprechend werden auch keine Hinweise gegeben, wie diese Techniken im Prozess nutzbar sind.

Das Hauptmerkmal des Vorgehensmodells ist die Netzwerkstruktur, die unter anderem Iterationen besser ermöglichen soll. Eine Konkretisierung der Iterationen, etwa wodurch sie initiiert

werden und auf welche Phasen oder Entscheidungen sie sich auswirken, wird jedoch nicht gegeben.

Prinzipiell ist das beschriebene Entwicklungsvorgehen nicht an eine Disziplin gebunden. Jedoch wird ein Fokus auf Produkte des Maschinenbaus deutlich. Eine konkrete Diskussion interdisziplinärer Zusammenarbeit erfolgt nicht.

3.2.2.5 Produktentwicklung nach Ulrich und Eppinger

Der in Abbildung 3.5 dargestellte Entwicklungsprozess von Ulrich und Eppinger [UlEp08] besteht aus einer sequentiellen Abfolge von Prozessschritten. Insbesondere die Konzeptphase wird im Detail als Subprozess erläutert, der wiederum selbst aus einer Abfolge sequentieller Prozessschritte besteht.

Sowohl im Gesamtmodell des Produktentwicklungsprozesses als auch in der Detailbetrachtung der Konzeptphase sind Prozessschritte zur Eigenschaftsanalyse vorgesehen. Da der Gesamtprozess sequentiell gestaltet ist und diese Analysephasen gegen Ende des Prozess eingebunden sind, ist eine frühe und kontinuierliche Analyse nur schwer möglich. In der Konzeptphase ist eine parallele, durchgängige Phase zur Erstellung von Prototypen integriert, die jedoch nicht weiter beschrieben wird.

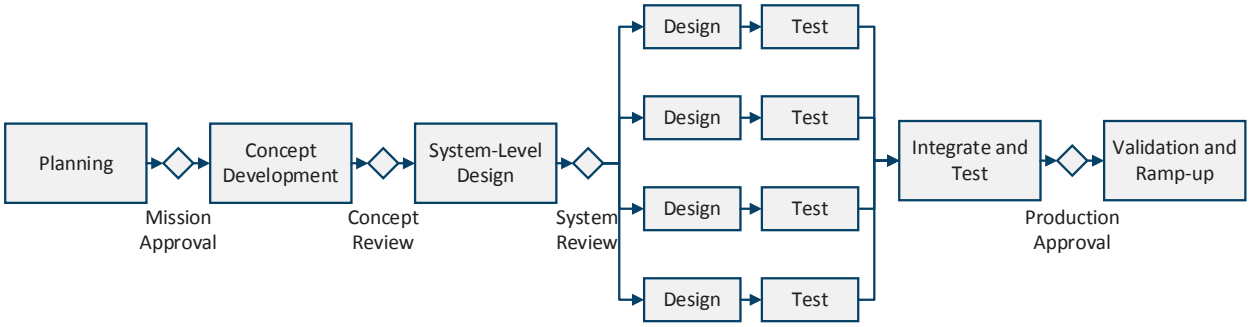


Abbildung 3.5: Prozessmodell für die Entwicklung komplexer Systeme [UlEp08]

Insgesamt sind Analysen sehr stark an physischen Prototypen orientiert. Während in der Konzeptphase Simulation nicht eingebunden ist, wird in der Analysephase des Gesamtprozesses die Möglichkeit der Simulation („analytical prototypes“) kurz diskutiert, wobei jedoch der Fokus insgesamt auf physischen Prototypen liegt. Eine detaillierte Auseinandersetzung mit Simulationstechniken sowie deren Möglichkeiten und Anforderungen erfolgt nicht. Simulationen werden nur als reine Alternative physischer Prototypen angesehen, wodurch das Potential, beispielsweise die frühere Nutzung im Prozess, nicht ausgeschöpft wird.

Für die Planung von Prototypen – sowohl physischer als auch analytischer – beschreiben Ulrich und Eppinger eine vorlagenbasierte Methode, welche die Entwicklung und den Einsatz von Prototypen strukturieren soll.

In der Konzeptphase beschreiben die Autoren, wie aus verschiedenen Konzepten geeignete ausgewählt werden. Hierfür sind Analysen erforderlich, die aber hauptsächlich qualitativ durchgeführt werden. Analysen nach der Konzeptauswahl dienen dazu, die Spezifikationen zu verfeinern. Detailliert wird aber nicht angegeben, wie genau Ergebnisse der Analysen hier genutzt werden. In der Analysephase auf oberster Prozessebene wird zwar beschrieben, wozu Prototypen genutzt werden, eine weitergehende Diskussion bezüglich der Nutzung von Ergebnissen erfolgt jedoch nicht.

Die Möglichkeiten zur Unterstützung von Entwicklungstätigkeiten durch den Einsatz computerbasierter Werkzeuge werden nicht erwähnt. Nur vereinzelt werden Hinweise auf derartige Werkzeuge gegeben, eine weitere Detaillierung bleibt jedoch aus.

Wie aus Abbildung 3.5 ersichtlich, werden Iterationen auf Gesamtprozessebene nicht dargestellt. Bei Betrachtung der detaillierten Konzeptphase wird deutlich, dass Iterationen nach jedem Schritt möglich sind, wobei Kriterien hierfür und deren Zweck nicht beschrieben werden.

Der Aspekt der Interdisziplinarität wird nur oberflächlich beschrieben. Für den Fall komplexer Systeme, bei denen interdisziplinäre Teams zusammenarbeiten müssen, wird der Gesamtprozess angepasst, indem die „Detail Design“ Phase in mehrere parallele Phasen unterteilt wird, wie in Abbildung 3.5 dargestellt. Eine weitergehende Diskussion interdisziplinärer Entwicklungsprojekte erfolgt nicht.

3.2.2.6 Mechanikentwicklung nach Ullmann

Ullmann [Ullm10] beschreibt den Entwicklungsprozess mechanischer Produkte als eine sequentielle Abfolge einzelner Prozessschritte, von der Produktidee über die Entwicklung eines Konzeptes und die eigentliche Produktentwicklung[5] bis hin zur Produktunterstützung. Jeder dieser einzelnen Prozessschritte ist wiederum weiter untergliedert und detailliert. Insbesondere von Interesse sind hier die Phasen Konzept- und Produktentwicklung. Beide sind prinzipiell ähnlich aufgebaut, wie aus Abbildung 3.6 ersichtlich.

[5] Ullman benutzt den Begriff der Produktentwicklung in einem wesentlich engeren Rahmen, als er in der vorliegenden Arbeit verstanden wird. Der Begriff beschreibt vielmehr Tätigkeiten, die denen des Entwerfens und Ausarbeiten nach Pahl und Beitz entsprechen.

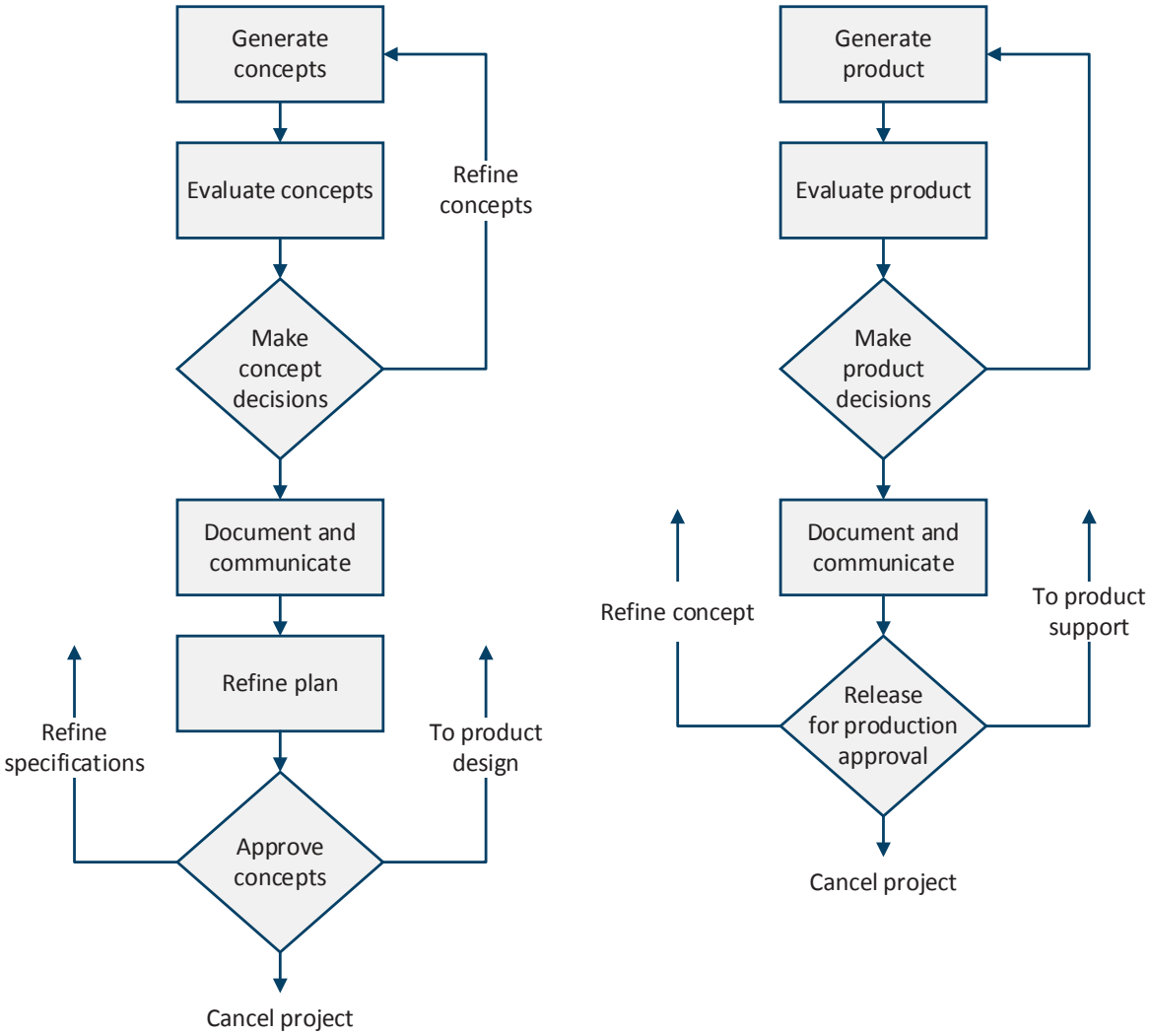


Abbildung 3.6: Konzeptphase (links) und Produktentwicklungsphase (rechts), nach [Ullm10]

Eigenschaftsanalysen sind sowohl in der Konzept- als auch in der Produktentwicklungsphase direkt nach der Konzept- beziehungsweise Produktentwicklung vorgesehen. Somit ist eine engmaschige Analyse im gesamten Entwicklungsprozess möglich. Es ist jedoch anzumerken, dass Eigenschaftsanalysen hauptsächlich zum Zweck des Vergleichs durchgeführt werden. Optimierungsaufgaben werden nicht ausführlich angesprochen. Die hierbei eingesetzten Methoden decken zum Großteil alle Möglichkeiten von Berechnungen über Simulationen bis zu physischen Prototypen ab. Insbesondere Simulationen werden jedoch nicht in der Tiefe behandelt – es erfolgt weder ein Überblick über aktuelle Techniken noch eine konkretere Einbindung in den Entwicklungsprozess. Im Allgemeinen werden Simulationen auch nur aus dem Bereich des Maschinenbaus erwähnt, wodurch wiederum der Fokus der Methodik deutlich wird.

Nach jeder Analysephase ist eine Entscheidung vorgesehen, bei der das geeignetste Konzept beziehungsweise Produkt ausgewählt wird. Mit dem in Abbildung 3.7 dargestellten P-Modell ist auch eine Methode vorgeschlagen, mittels derer diese Entscheidungen durchgeführt werden

können. Jedoch fehlt eine detailliertere Beschreibung, beispielsweise wie Kriterien für den Entscheidungspunkt definiert werden können oder woher die Informationen stammen.

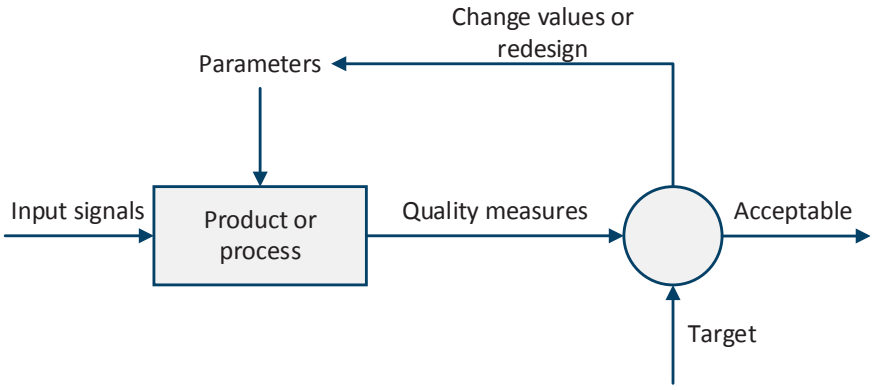


Abbildung 3.7: P-Modell (P: Produkt oder Prozess) nach [Ullm10]

Der Einsatz von Computermethoden wird von Ullman nicht explizit behandelt. Vereinzelt erfolgen kurze Hinweise auf Möglichkeiten der Computerunterstützung, etwa CAD oder FEM, ausführliche Diskussionen erfolgen jedoch nicht.

Iterationen sind in dem Vorgehen nach Ullmann sowohl in der Konzeptphase als auch in der Produktentwicklungsphase vorgesehen. Durch die Beschreibung mittels des P-Modells sind zudem prinzipiell auch Kriterien definierbar, welche Iterationen auslösen – jedoch wird dies nicht detailliert diskutiert. Iterationen, die phasenübergreifend erfolgen, etwa eine Konzeptüberarbeitung, sind nicht abgebildet.

Die Methodik nach Ullman richtet sich spezifisch an die Entwicklung mechanischer Produkte. Entsprechend erfolgt keine Diskussion interdisziplinärer Entwicklungsaspekte.

3.2.2.7 Zusammenfassung der Ergebnisse

Die Ergebnisse der Analyse der Entwicklungsmethodiken aus dem Bereich der Mechanik sind in Tabelle 3.1 zusammengefasst.

Tabelle 3.1: Zusammengefasste Analyse der Entwicklungsmethodiken aus der Mechanik (+: im Detail berücksichtigt; ○: berücksichtigt; -: nicht berücksichtigt)

Methodik	Eigenschafts-analyse	Analyse-ergebnisse	Einsatz von Simulation	Computer-unterstütz-ung	Iterationen	Interdiszi-plinarität
Pahl/Beitz	○	-	-	-	○	○
VDI 2221	○	-	-	-	○	○
Ehrlenspiel	○	○	-	-	○	○
MVM	○	-	-	-	+	-
Ulrich/ Eppinger	○	○	-	-	○	○
Ullman	+	+	-	-	○	-

3.2.3 Elektronik

3.2.3.1 Y-Diagramm nach Gajski und Kuhn

Das in Abbildung 3.8 dargestellte Y-Diagramm wurde in seiner ursprünglichen Version 1983 von Gajski und Kuhn [GaKu83] entwickelt. Es stellt ein fundamentales Modell dar, welches zur Beschreibung der Unterschiede verschiedener Entwurfswerkzeuge und Methodiken dient [GAGS09]. Entsprechend repräsentiert es weniger eine detaillierte Methodik, sondern dient auf abstrakter Ebene als übergeordnetes Bindeglied verschiedener Methodiken. Aus diesem Grunde wird das Y-Diagramm nicht in die Analyse mit einbezogen. Dennoch soll es hier erläutert werden, da alle Methodiken aus dem Bereich der Elektronikentwicklung zumindest Kernelemente davon aufweisen, so auch die im Folgenden besprochenen Methodiken. Zu diesem Zweck wird die Version von Lehmann et al. [LeWS94] herangezogen. Kernelement des Y-Diagramms stellen drei Sichten dar: Verhalten, Struktur und Geometrie. Mittels dieser Sichten lassen sich alle Tätigkeiten im (Elektronik-)Entwicklungsprozess beschreiben. Nach Gajski et al. [GAGS09] wird das Verhalten durch eine Black-Box mit dem Ein-/Ausgangsverhalten beschrieben. Der innere Aufbau dieser Black-Box wird über die Struktur bestimmt. Die physische Umsetzung der Struktur wird wiederum durch die Geometriesicht festgelegt. Über diese drei Sichten hinaus existieren verschiedene Abstraktionsgrade, in Abbildung 3.8 durch Kreise dargestellt: Systemebene, algorithmische Ebene, Register-Transfer-Ebene, Logikebene und Schaltkreisebene. Hierbei nimmt der Abstraktionsgrad nach außen hin zu.

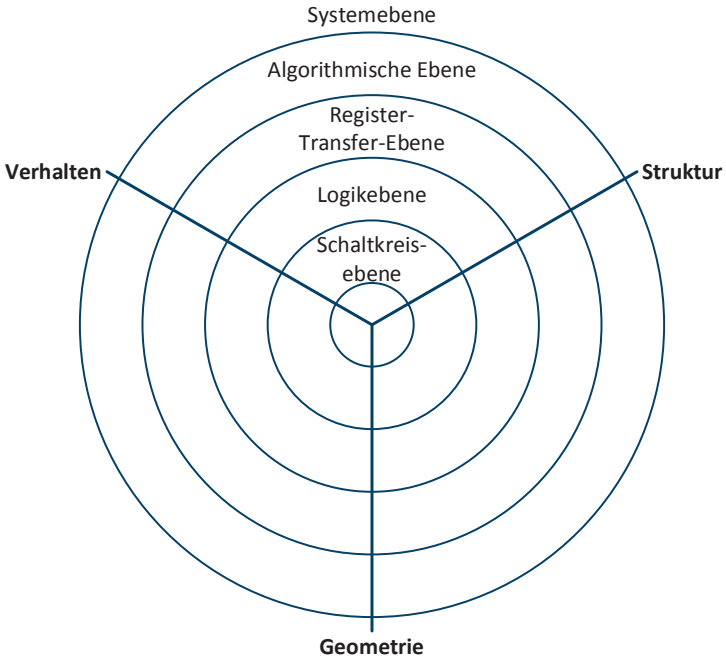


Abbildung 3.8: Y-Diagramm, nach [GaKu83] und [LeWS94]

3.2.3.2 VDI-Richtlinie 2422

Die VDI-Richtlinie 2422 [VDI2422] bietet einen Leitfaden für Entwickler von Geräten, deren Steuerung auf dem Einsatz von Mikroelektronik basiert. Ein Teilaspekt der Richtlinie bildet die Entwicklungsmethodik für elektronische Schaltungen, welche hier betrachtet werden soll. Diese Methodik basiert auf einem Phasenmodell, bestehend aus den Hauptphasen Schaltungsentwurf und Schaltungsausarbeitung. In der ersten Hauptphase werden die Aufgabenstellung präzisiert, eine Schaltungsstruktur entworfen und verfeinert, die Schaltung analysiert und abschließend Hardwaretests durchgeführt. Die zweite Phase besteht aus der Raumaufteilung, der Verdrahtung, der Layoutdefinition sowie der Erstellung der Fertigungs- und Prüfunterlagen.

Eigenschaftsanalysen sind im Prozessmodell an mehreren Stellen eingebunden. So wird bei der Schaltungsanalyse ein Vergleich von Lösungsalternativen durchgeführt. Eine genauere Beschreibung, welche Analysemethoden hier zum Einsatz kommen und welche Eigenschaften analysiert werden sollen, wird jedoch nicht gegeben. Anschließend erfolgen physische Tests der Hardware, beispielsweise zur Untersuchung des Störverhaltens und der Zuverlässigkeit. In der Phase der Erstellung von Fertigungs- und Prüfunterlagen werden Prototypen, Muster oder Vorserien erstellt, die zur Absicherung der seriennahen Eigenschaften dienen sollen. Jedoch erfolgt auch hier keine detaillierte Beschreibung.

Simulationen spielen im Vorgehen nach VDI 2422 keine bedeutende Rolle. Bis auf eine kurze Bemerkung, dass durch Logik- und Timingsimulation bereits früh Eigenschaften analysiert werden können, erfolgt keine Diskussion des Themas. Außer dieser Bemerkung zur Simulation erfolgt auch keine Betrachtung von computerunterstützten Werkzeugen und Methoden.

Eine konkrete Beschreibung, wie Analyseergebnisse verwendet werden, ist in der Richtlinie nicht gegeben. Jedoch wird erwähnt, dass Ergebnisse dazu genutzt werden sollen, um zu früheren Phasen zurückzuspringen und somit eine stufenweise Verbesserung der Schaltung zu erreichen. Abgesehen von diesem Hinweis auf ein mögliches Zurückspringen bleiben Iterationen unadressiert. Insbesondere die Gründe für Iterationen und deren konkretere Auswirkung auf Entwicklungsphasen bleiben offen.

Da die Richtlinie sich im Kern mit der Entwicklung von Geräten beschäftigt, die durch Mikroelektronik gesteuert werden, ist die Berücksichtigung eines interdisziplinären Vorgehens unumgänglich. So gliedert sich die Entwicklung der elektronischen Schaltung in einen übergeordneten Prozess der Geräteentwicklung ein. Hierbei wird zunächst ein Gerätekonzept entwickelt, welches dann in den Einzeldomänen detailliert wird. Abschließend werden die Einzelentwicklungen im gesamten Gerät erprobt. Die Steuerung und Koordination übernimmt ein Projektleiter. Eine Detaillierung der Zusammenarbeit der Domänen, insbesondere auch in den domänenspezifischen Entwicklungsphasen, ist nicht gegeben.

3.2.3.3 Elektroniksystementwicklung nach Siegl

Siegl [Sieg04] unterteilt den Entwicklungsprozess von Elektroniksystemen in drei Hauptphasen: Systementwurf, Feinentwurf und Layouterstellung. Im Systementwurf wird die Systemstruktur verschiedener Konzepte entwickelt, mit denen die Produktanforderungen erfüllt werden können. Im Feinentwurf wird für das Systemkonzept eine Schaltung entworfen und diese Schaltung verifiziert. Zudem wird das Testkonzept festgelegt, also wie die Eigenschaften der Schaltung später überprüft werden sollen. Die Layouterstellung befasst sich mit der physikalischen Umsetzung der Schaltung, das heißt den erforderlichen Bauteilen und deren Anordnung, was abschließend in der Erstellung der Fertigungsdaten endet. Mit diesen Daten können Prototypen gefertigt und anschließend getestet werden.

Allgemein spielen Eigenschaftsanalysen im Vorgehen nach Siegl eine wichtige Rolle. In jeder der Hauptphasen finden am Ende Analyseschritte statt. Zudem wird bereits im Vorfeld geplant, wann und wie Analysen durchgeführt werden sollen. Ein Großteil der Analysen erfolgt jedoch in

der dritten Phase durch physische Prototypen. Allerdings wird auch der frühe Einsatz von Systemsimulationen erwähnt, um die Machbarkeit von Systemkonzepten zu überprüfen und auch im Feinentwurf werden Simulationen eingesetzt, beispielsweise Schaltkreissimulationen oder Wärmeflusssimulationen. Für die Schaltkreissimulation erfolgt zudem eine detaillierte Beschreibung des für die Simulation erforderlichen Vorgehens. Neben der Simulation und der dafür erforderlichen digitalen Modellerstellung, beispielsweise von Schaltplänen, erfolgt keine weitere Einbindung von Computerunterstützung in den Entwicklungsprozess.

Der Einsatz von Analysen erfolgt zum einen, um verschiedene Systemkonzepte zu vergleichen. Zum anderen werden sie eingesetzt, um ein Schaltungskonzept zu optimieren. Jedoch werden Vergleichs- und Optimierungsvorgehen nicht weiter erläutert.

Die durch Analyseergebnisse initiierten Iterationen werden im Vorgehen kaum eingebunden. Einzig ein Hinweis auf die Entwurfsmodifikation ist gegeben. Hierbei handelt es sich um Makroiterationen, die mehrere Entwicklungsphasen umfassen.

Aufgrund der Fokussierung der Methodik auf die Entwicklung von Schaltungselektronik wird ein interdisziplinäres Vorgehen nicht diskutiert.

Andere Methodiken aus dem Bereich der Elektronik, wie beispielsweise das von Lienig [Lien06] vorgeschlagene Vorgehen, sind dem Vorgehen von Siegl sehr ähnlich und werden daher hier nicht weiter betrachtet.

3.2.3.4 Entwicklung digitaler Elektronik nach Gausemeier und Bigl

Gausemeier und Bigl [GaBi06] kombinieren die drei aus dem Y-Modell (siehe Kapitel 3.2.3.1) bekannten Sichten Verhalten, Struktur und Geometrie mit den drei Abstraktionsebenen Systemebene, algorithmische Ebene und Registertransferebene in einem Phasenmodell zur Entwicklung digitaler Elektronik. Das Prozessmodell besteht aus fünf Hauptphasen, in denen jeweils die verschiedenen Abstraktionsebenen durchlaufen werden. In der Spezifikation werden die Leistungsdaten der Schaltung festgelegt, welche in der Verhaltenssynthese genutzt werden, um mittels Funktionsblöcken eine abstrakte Beschreibung des gewünschten Verhaltens zu generieren. In der Struktursynthese werden zu den einzelnen Funktionsblöcken mittels Makros Strukturelemente gesucht. Die Layout-Synthese bildet die Struktur in ihrer tatsächlichen geometrischen Form ab. Abschließend erfolgt die Fertigung der Schaltung.

Eigenschaftsanalysen sind im Phasenmodell an verschiedenen Stellen vorgesehen. Auffällig ist beispielsweise im Vergleich zum Vorgehen nach Siegl, dass keine physischen Prototypen vorge-

sehen oder erwähnt sind. Vielmehr beschränken sich Analysen auf Simulationen, die am Ende der einzelnen Phasen durchgeführt werden, etwa Logiksimulationen und Simulationen des elektrischen Verhaltens bei der Struktursynthese oder Simulationen des elektromagnetischen Verhaltens bei der Layoutsynthese. Die Funktionsweise der Simulation sowie deren Einsatzweck werden allerdings nicht diskutiert. Über diese Simulationen hinaus wird nicht explizit erwähnt, wie weitere computerbasierte Methoden den Entwicklungsprozess beeinflussen können.

Im Ablauf des Prozesses wird deutlich, dass Simulationsergebnisse dazu genutzt werden, Optimierungen durchzuführen. Wie diese Ergebnisse im Detail genutzt werden und wie Optimierungen ablaufen, bleibt jedoch offen. Der Einsatz von Analysen, um verschiedene Konzepte zu vergleichen, ist in dem Vorgehen nicht integriert. Die erwähnten Optimierungen erfordern zwangsläufig Iterationen, welche im Prozessmodell jedoch weder abgebildet noch erwähnt werden.

Auch die interdisziplinäre Zusammenarbeit bleibt an dieser Stelle unbeachtet, was jedoch durch die Fokussierung auf digitale Elektronik begründet ist.

3.2.3.5 Zusammenfassung der Ergebnisse

Die Ergebnisse der Analyse der Entwicklungsmethodiken aus dem Bereich der Elektronik sind in Tabelle 3.2 zusammengefasst.

Tabelle 3.2: Zusammengefasste Analyse der Entwicklungsmethodiken aus der Elektronik (+: im Detail berücksichtigt; ○: berücksichtigt; -: nicht berücksichtigt)

Methodik	Eigenschafts-analyse	Analyse-ergebnisse	Einsatz von Simulation	Computer-unterstütz-ung	Iterationen	Interdiszi-plinarität
VDI 2422	○	○	-	-	-	○
Siegl	+	○	+	-	-	-
Gausemeier	○	○	+	-	-	-

3.2.4 Software

3.2.4.1 Wasserfallmodell

Das Wasserfallmodell ist eines der ersten und bekanntesten Prozessmodelle der Softwareentwicklung. Es existiert in zahlreichen Abwandlungen, wobei die ursprüngliche Version von Royce

vorgeschlagen wurde [Royc70]. Es handelt sich hierbei um ein planungsbasiertes Vorgehensmodell, das heißt im Vorfeld werden die einzelnen Schritte vorgeplant. Zudem kann eine Phase erst begonnen werden, wenn die vorangegangene beendet ist. Eine angepasste und weit verbreitete Version wurde 1981 von Boehm vorgestellt [Boeh81], welche in Abbildung 3.9 dargestellt ist. Kennzeichnend ist hierbei die Integration von Analysen in jeder Phase, um große Änderungen durch späte Testphasen, wie von Royce ursprünglich vorgesehen, zu vermeiden.

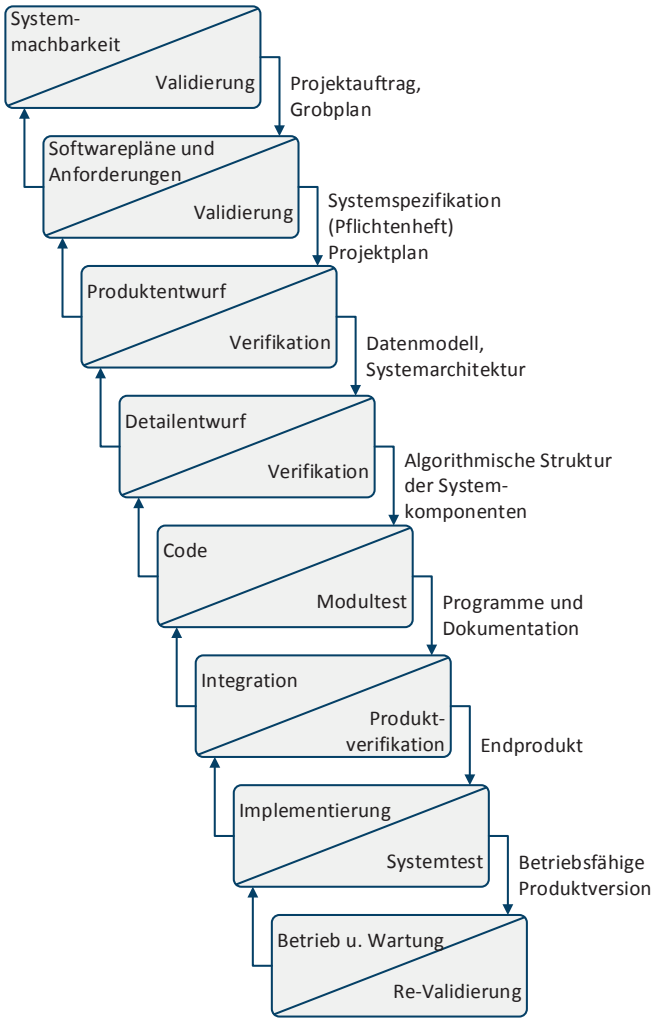


Abbildung 3.9: Wasserfallmodell nach [Boeh81] und [PoPr04]

In der ursprünglichen, von Royce vorgeschlagenen Variante ist ein Analyseschritt erst zum Ende des Prozesses vorgesehen, das bedeutet nach der Code-Erstellung. In dem von Boehm angepassten Modell werden Analysen über alle Phasen verteilt, wodurch eine fast kontinuierliche Überprüfung der Entwicklungstätigkeiten ermöglicht wird.

Softwareentwicklung erfolgt naturgemäß rechnerbasiert. Software allgemein ist zu großen Teilen unabhängig von physischen Komponenten – abgesehen von der Zielhardware, auf der die Software eingesetzt wird und möglichen Schnittstellen zu angesteuerter Hardware. Diese Tat-

sache macht es wesentlich einfacher, Tests der Software durchzuführen, da zu deren Ausführung keine physischen Komponenten benötigt werden oder diese emuliert werden können. Daher wird im Folgenden kein Unterschied zwischen Eigenschaftsanalysen im Allgemeinen und Simulation gemacht. Zudem ist der Aspekt der Computerunterstützung hier auch nicht einzeln zu bewerten, da diese bei der Softwareentwicklung unumgänglich ist.

Außer der Darstellung von Iterationen wird nicht weiter darauf eingegangen, wie und zu welchem Zweck Analysen im Prozess verwendet werden und wie Analyseergebnisse genutzt werden.

Viele Darstellungen des Wasserfallmodells verzichten vollständig auf Iterationen. In dieser Arbeit werden jedoch nur die Varianten von Royce und Boehm zugrunde gelegt. Die in Abbildung 3.9 dargestellten Iterationen sollen das rein sequentielle Vorgehen anderer Prozessmodelle verhindern [PoPr04]. Demnach sind Iterationen zwischen aufeinanderfolgenden Phasen möglich und deren Wichtigkeit wird auch betont. Größere Iterationen, die mehrere Phasen umfassen, werden nicht berücksichtigt. Diese sollen bewusst vermieden werden, um die damit verbundenen großen Aufwände für Änderungen zu vermeiden [Royc70].

Aufgrund der Fokussierung des Wasserfallmodells auf die Entwicklung von Software bleiben interdisziplinäre Ansätze unbeachtet. Dies bedeutet auch, dass die Zusammenarbeit unterschiedlicher Entwicklungsteams nicht berücksichtigt wird.

3.2.4.2 Spiralmodell nach Boehm

Boehm entwickelte das Wasserfallmodell zum Spiralmodell weiter [Boeh86], welches ein risikobasiertes Vorgehen ermöglichen soll. Der in Abbildung 3.10 dargestellte Entwicklungsablauf durchläuft spiralförmig vier Schritte, wobei diese Spirale mehrfach durchlaufen wird, jeweils mit höherem Reifegrad. Im vierten Quadranten können dann detailliertere Prozessmodelle integriert werden, wie hier beispielsweise das Wasserfallmodell. Somit ist das Spiralmodell adaptierbar.

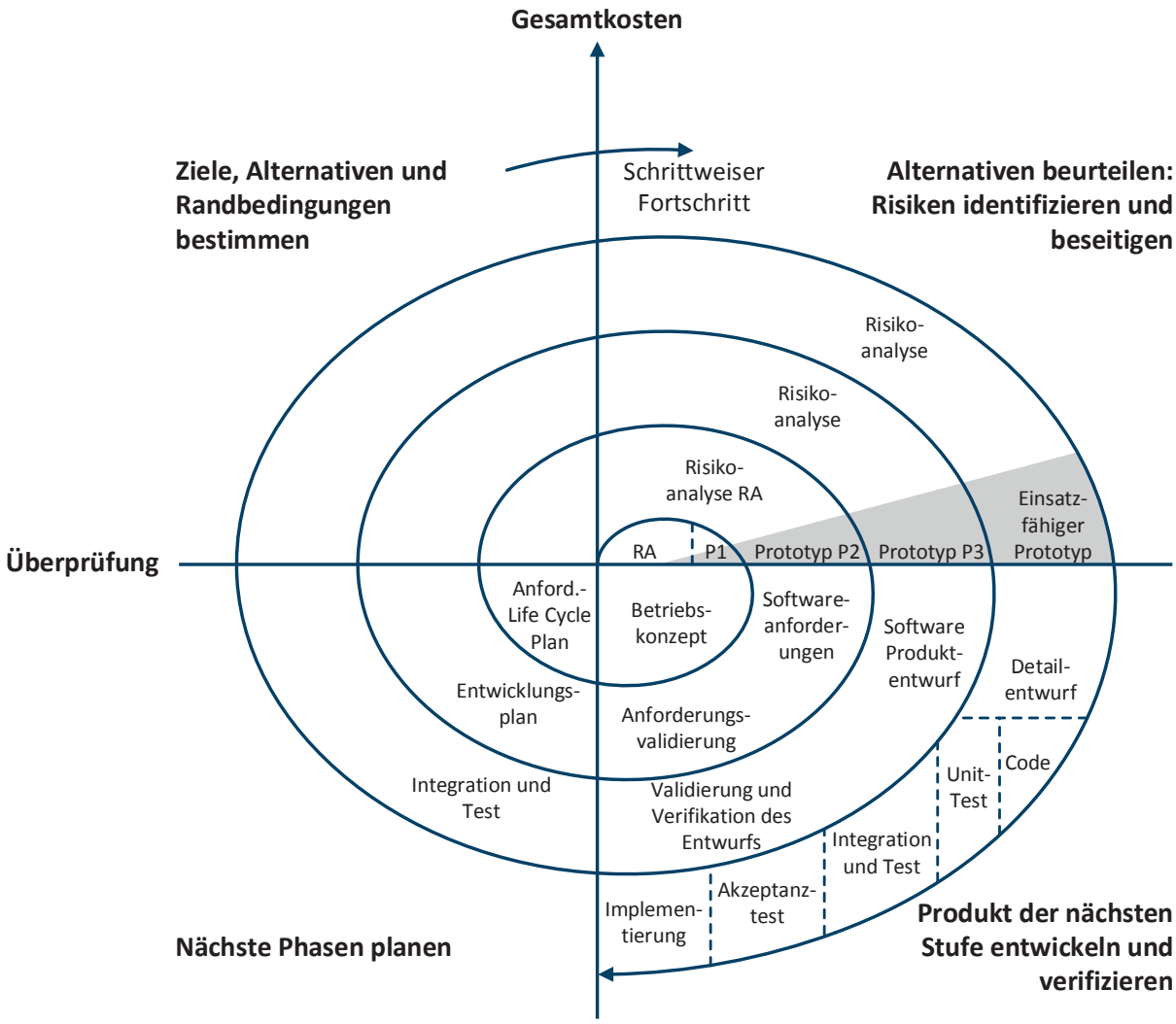


Abbildung 3.10: Spiralmodell nach Boehm [Boeh86]

Ähnlich wie in Boehms Variante des Wasserfallmodells spielen Eigenschaftsanalysen eine wesentliche Rolle. Jeder Durchlauf einer Spirale beinhaltet am Ende eine Testphase. Somit werden in sehr geringen Abständen Analysen durchgeführt. Der Umgang mit den aus Analysen gewonnenen Erkenntnissen wird jedoch nicht diskutiert.

Ein wesentlicher Aspekt des Spiralmodells ist der iterative Charakter. Zum einen werden die einzelnen Phasen durch mehrere Spiralen mehrfach durchlaufen. Zum anderen ermöglicht die Struktur stets ein Zurückspringen in frühere Phasen. Dies ist jedoch im Hinblick auf die Übersichtlichkeit nicht explizit dargestellt. Entsprechend der fehlenden Beschreibung des Umgangs mit Analyseergebnissen werden auch keine Kriterien für Iterationen geliefert.

Zwar wird das Zusammenspiel verschiedener Domänen im mechatronischen Sinne nicht behandelt, jedoch erwähnt Boehm die Wichtigkeit der Abstimmung verschiedener am Entwick-

lungsprozess beteiligter Gruppen. Dies gilt insbesondere für den vierten und dritten Quadranten.

3.2.4.3 Extreme Programming

Extreme Programming [Beck00] zählt zu den bekanntesten Methoden der agilen – oder leichtgewichtigen – Softwareentwicklung und soll hier stellvertretend für diese betrachtet werden. Im Gegensatz zu den schwergewichtigen Methodiken, wie etwa dem Wasserfallmodell, besitzen agile Methodiken einen geringeren Formalisierungsgrad, einen geringeren Dokumentationsgrad und weniger fest definierte Zwischenergebnisse [JaSa05], [PoPr04]. Im Extreme Programming wird Software – oder ein Teil der Software (auch Release genannt) – in einzelnen Inkrementen (auch Tasks genannt) nach dem in Abbildung 3.11 dargestellten Prozess entwickelt. Das Vorgehen beruht auf vier Grundwerten, die den Prozess und das Handeln bestimmen: Kommunikation, Einfachheit, Rückmeldung und Courage. Daraus leiten sich vier Basisaktivitäten ab, zu denen unter anderem Testen gehört. Der Ablauf aus Abbildung 3.12 ordnet sich im Block „Entwickeln / Integrieren / Testen der Software“ aus Abbildung 3.11 ein. Wie dort dargestellt, wird, bevor Softwarecode geschrieben wird, ein Test für jeden Task geschrieben. Diese Tests müssen erfolgreich absolviert werden, bevor der generierte Code in das Restsystem integriert wird. Andernfalls muss der Code in Iterationen angepasst werden. Dieses Vorgehen hat sich unter dem Begriff testgetriebene Entwicklung auch auf andere Vorgehensweise der Softwareentwicklung ausgebreitet.

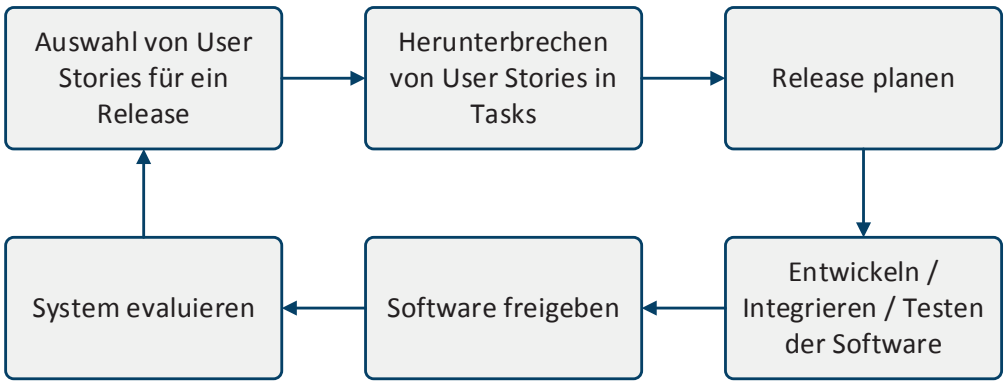


Abbildung 3.11: Prozessablauf im Extreme Programming für ein Release nach [Somm11]

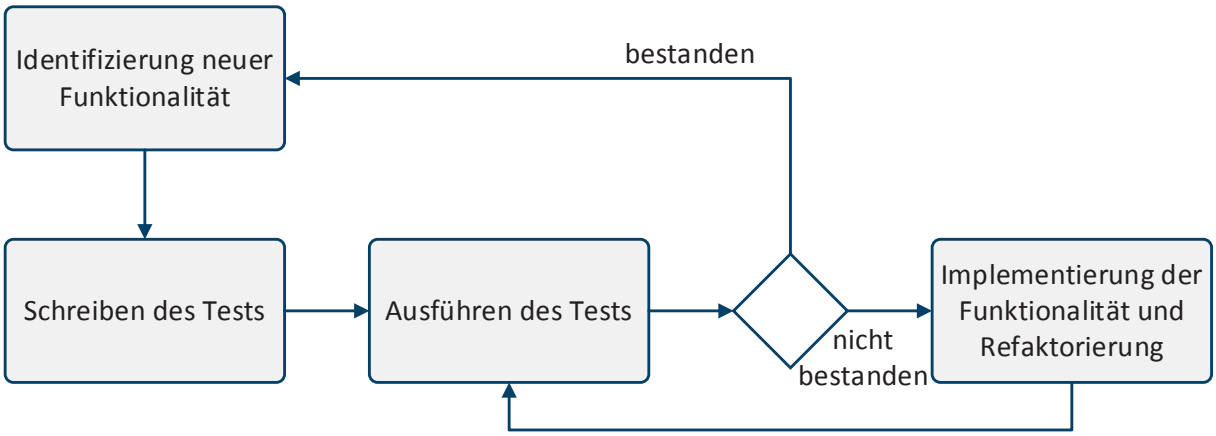


Abbildung 3.12: Testgetriebene Entwicklung nach [Somm11], Ablauf für ein Task

Eigenschaftsanalysen spielen eine zentrale Rolle im Extreme Programming und in der testgetriebenen Entwicklung. Dies ist bereits daran zu erkennen, dass Testen als eine von vier Basisaktivitäten angesehen wird. Zudem wird noch vor der Code-Erstellung der entsprechende Test für jeden Task geplant und geschrieben. Entsprechend ist eine kontinuierliche Eigenschaftsanalyse in den Prozess integriert.

Das in Abbildung 3.12 dargestellte Vorgehen der testgetriebenen Entwicklung, in Kombination mit der Anwendung auf einzelne Tasks, macht deutlich, wann und wie getestet werden muss und wie Ergebnisse in die eigentliche Code-Design-Aktivitäten zurückfließen. Entsprechend diesem Vorgehen wird auch deutlich, dass Extreme Programming ein hochiteratives Vorgehen beschreibt und diese Iterationen deutlich im Prozessmodell abgebildet sind.

Einen der vier Grundwerte stellt die Kommunikation dar. Auch wenn hiermit nicht explizit die Kommunikation interdisziplinärer Teams gemeint ist, sondern allgemein die Kommunikation aller am Entwicklungsprozess beteiligter Personen oder Institutionen, wird deutlich, dass der Aspekt der Zusammenarbeit als sehr wichtig angesehen wird. Jedoch werden keine genauen Hinweise zum Ablauf dieser Zusammenarbeit gemacht.

3.2.4.4 Zusammenfassung der Ergebnisse

Die Ergebnisse der Analyse der Entwicklungsmethodiken aus dem Bereich der Softwareentwicklung sind in Tabelle 3.3 zusammengefasst.

Tabelle 3.3: Zusammengefasste Analyse der Entwicklungsmethodiken aus der Softwareentwicklung

(+: im Detail berücksichtigt; ○: berücksichtigt; -: nicht berücksichtigt)

Methodik	Eigenschafts-analyse	Analyse-ergebnisse	Einsatz von Simulation	Computer-unterstütz-ung	Iterationen	Interdiszi-plinarität
Wasserfall-modell	+	-			+	-
Spiralmodell	+	-			+	○
Extreme Program-ming	+	+			+	o

3.2.5 Mechatronik

Die Entwicklungsmethodiken aus dem Bereich der Mechatronik stellen vergleichsweise junge, voneinander unabhängige Arbeiten dar. Aus diesem Grunde erfolgt deren Diskussion im Folgenden in chronologischer Reihenfolge.

3.2.5.1 Spiralmodell nach Chan und Leung

Chan und Leung [ChLe96] stellen ein Ansatz vor, der auf einem Spiralmodell basiert, ähnlich dem der Softwareentwicklung in Abbildung 3.10. Die Spirale ist in fünf Phasen (Erforschung, Untersuchung, Prototyp, Prozessentwicklung, Pilotanlauf) unterteilt, die jeweils einem Spiralumlauf entsprechen. Jede Phase ist zusätzlich in vier Aktivitäten (Analyse, Gestaltung, Herstellung, Test) unterteilt. Diese Spiralform soll den hochiterativen Entwicklungsprozess mechatronischer Systeme besser abbilden können.

Jede der fünf Phasen des Spiralmodells wird mit einer Testphase beendet. Die durchgeführten Tests müssen bestanden werden, bevor die nächste Phase begonnen werden kann. Hierdurch wird eine enge Absicherung innerhalb des Prozesses erreicht. Zudem wird für jede Phase beschrieben, welche Tests durchgeführt werden sollen. Die in den Testaktivitäten durchgeführten Analysen beruhen weitestgehend auf physischen Prototypen. Einzig für die Erforschungsphase wird der Einsatz von Simulationstechniken angedeutet. Eine detaillierte Beschreibung der anwendbaren Techniken sowie deren Möglichkeiten und Anforderungen wird nicht gegeben.

Ebenso wenig werden auch weitere computerbasierte Techniken im Prozessverlauf eingeordnet.

Für jede der Phasen wird beschrieben, zu welchem Zweck die einzelnen Tests durchgeführt werden. Jedoch wird nicht erläutert, wie verfahren werden soll, wenn Tests nicht bestanden werden. Zwar ist, wie bereits erwähnt, die Struktur des Prozessmodells darauf ausgerichtet, Iterationen besser abzubilden, jedoch wird der Iterationsprozess selbst nicht beschrieben. Wann Iterationen notwendig sind und in welche Phase diese zurückführen, bleibt offen.

Das Thema der Interdisziplinarität betrachten Chan und Leung eher aus dem Blickpunt des gesamten Produktentstehungsprozesses. Hierbei wird betrachtet, wie Teams aus verschiedenen Phasen des Entstehungsprozesses, beispielsweise Entwicklungs- und Fertigungsingenieure, zusammenarbeiten können. Hierzu wird ein Vorgehen zur Unterstützung des „Concurrent Engineering“ vorgeschlagen. Den Aspekt der domänenübergreifenden Zusammenarbeit – also zwischen Mechanik-, Elektronik- und Softwareentwicklung – berücksichtigt das Vorgehensmodell allerdings nicht.

3.2.5.2 Entwurf mechatronischer Systeme nach Kallenbach

Kallenbach et al. [KBSS97] schlagen das in Abbildung 3.13 dargestellte Phasenmodell vor, das auf einem iterativen Top-Down-Ansatz basiert. Einen wesentlichen Aspekt bilden die Betrachtung und Strukturierung von zu erfüllenden Funktionen sowie die Suche nach Gestaltungselementen, welche diese Funktionen erfüllen.

Die Analyse von Eigenschaften wird nur am Rande thematisiert. Am Ende des gesamten Prozesses ist eine einzelne Phase für die Erprobung mit physischen Prototypen vorgesehen. Die dort angewendeten Techniken sowie das Vorgehen sind jedoch nicht weiter beschrieben. Während der Gestaltung sollen durch Simulationen auf Systemebene die Gestaltungstätigkeiten abgesichert werden. Jedoch erfolgt auch hier keine weitergehende Auseinandersetzung mit den dafür erforderlichen Tätigkeiten und Techniken.

Die Autoren sind der Überzeugung, dass der mechatronische Entwicklungsprozess nur rechnerunterstützt ablaufen kann, worauf mehrfach hingewiesen wird. So wird etwa eine beispielhafte Übersicht von Softwarewerkzeugen gezeigt. Jedoch erfolgt keine Einordung in den Entwicklungsprozess, also wann und zu welchem Zweck diese Werkzeuge eingesetzt werden können und sollen.

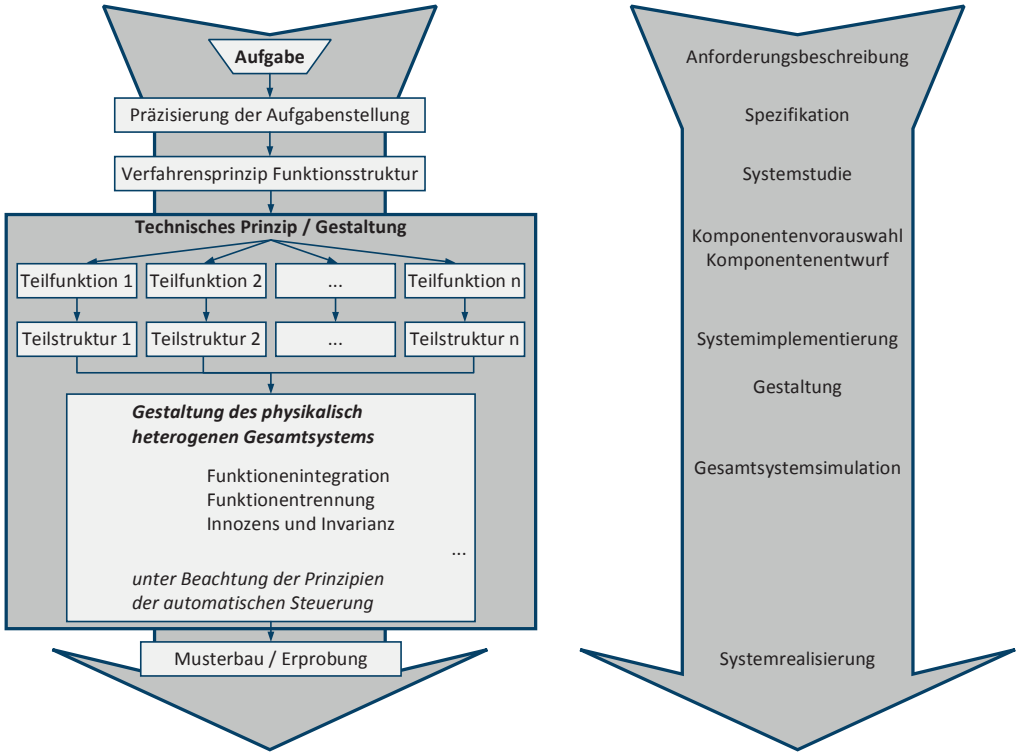


Abbildung 3.13 Phasenmodell nach Kallenbach [KBSS97]

Entsprechend der wenig tiefgehenden Auseinandersetzung mit Analyseaktivitäten wird auch nicht weiter darauf eingegangen, wie Analyseergebnisse im Prozess genutzt werden. Obwohl erwähnt wird, dass der Entwicklungsprozess iterativ ist, wird nicht beschrieben, wo Iterationen auftreten oder wodurch diese ausgelöst werden. Im Phasenmodell sind zudem keine Iterationen abgebildet.

Den interdisziplinären Charakter mechatronischer Entwicklungsprojekte berücksichtigen Kallenbach et al. durch die Aufteilung und Hierarchisierung der Gesamtfunktion. Zudem stellen Funktionentrennung und -integration einen wichtigen Aspekt dar. Jedoch wird nicht diskutiert, wie die einzelnen Disziplinen interagieren.

3.2.5.3 Entwurfsmethodik nach Isermann

Isermann [Iser99][6] stellt einen Entwicklungsansatz auf Basis des V-Modells vor, der in drei Hauptphasen eingeteilt ist: Systementwurf, Systemintegration und Systemtest. Diese werden wiederum in kleineren Schritten detailliert. Die V-Form des Prozessmodells symbolisiert das Top-Down-Vorgehen im Entwurf und das Bottom-Up-Vorgehen in der Systemintegration. Wei-

[6] Die hier durchgeführte Analyse basiert auf der 2. Auflage des Buches [Iser08].

terhin bildet der linke Arm des V-Modells hauptsächlich Entwurfstätigkeiten ab, während der rechte Arm alle Testaktivitäten beinhaltet.

Das Prozessmodell sieht mehrere Schritte vor, in denen Eigenschaften analysiert werden. Insbesondere im rechten Arm des V-Modells sind verschiedene physische Tests vorgesehen (Komponententests, Systemtests, Feldtests). Allerdings werden diese nicht detailliert beschrieben. Zwischen System- und Komponentenentwurf sieht Isermann einen eigenen Schritt für Modellierung und Simulation vor. Hierzu wird auch ein kurzer Überblick gegeben, was und wozu simuliert wird sowie welche Softwarewerkzeuge eingesetzt werden. Allerdings erfolgt diese Beschreibung recht oberflächlich, weshalb eine detaillierte Einbindung nicht ermöglicht wird. Dadurch, dass die Analysen in vereinzelten Schritten durchgeführt werden, ist eine durchgehende, begleitende Analyse nicht möglich. Zudem sind Analysen sehr stark auf physische Tests am Ende des Prozesses ausgerichtet.

Neben Simulationen erwähnt Isermann, dass Computerunterstützung für die Entwicklung mechatronischer Produkte erforderlich ist. Hierzu gibt er, neben den Simulationswerkzeugen, einen kurzen Überblick der einsetzbaren Werkzeuge, insbesondere für den Komponentenentwurf. Wozu diese im Detail genutzt werden und wie sie den Prozess beeinflussen, erläutert Isermann nicht.

Für die einzelnen Analyseschritte gibt Isermann jeweils deren Zweck an. Jedoch wird nicht detailliert erläutert, wie diese Analyse genutzt werden, um den Entwurf zu unterstützen und das Konzept oder Produkt zu verbessern. Entsprechend bleiben Iterationen im Detail unbeachtet: Es erfolgt nur der Hinweis, dass die einzelnen Schritte um Iterationen zu ergänzen sind, jedoch ohne genauere Empfehlungen.

Der interdisziplinäre Charakter mechatronischer Systeme wird berücksichtigt, indem zu Beginn der Entwurf auf Systemebene stattfindet. Basierend hierauf werden Verantwortlichkeiten an die einzelnen Disziplinen verteilt, welche dann die einzelnen Komponenten entwerfen. Abschließend werden die Komponenten wieder zum Gesamtsystem integriert. Dieses Top-Down-Bottom-Up-Vorgehen wird durch die V-Struktur symbolisiert. Im Detail wird jedoch die Zusammenarbeit der Disziplinen während des Komponentenentwurfs nicht beschrieben.

3.2.5.4 Integrierter Entwicklungsprozess nach Schön

Schön [Schö99] präsentiert ein Vorgehensmodell, welches einen integrierten Entwicklungsprozess beschreibt. Besondere Aufmerksamkeit wurde bei der Entwicklung des Modells auf die

Interdisziplinarität und die Verifikation von Entscheidungen in frühen Phasen gelegt [Schö99]. Das Modell basiert auf einer Integration etablierter Vorgehensmodelle aus den einzelnen Domänen. Dem Gesamtprozess liegt das prinzipielle Vorgehen nach VDI 2221 [VDI2221] zugrunde. Demnach wird in der Konzeptphase eine domänenübergreifende Wirkstruktur erstellt, woraufhin Entwicklungsaufgaben an die einzelnen Domänen verteilt werden, die wiederum ihre etablierten Vorgehensweisen nutzen.

Der Aspekt der Eigenschaftsanalyse findet in der integrierten Vorgehensweise keine direkte Berücksichtigung. Die individuellen Vorgehensweisen der Domänen – siehe hierzu auch die vorangegangenen Kapitel – bieten hier zwar durchaus Ansätze, diese sind aber im Modell nach Schön nicht weiter integriert.

Schön entwickelt neben dem Prozessmodell auch ein Assistenzsystem, welches Entwickler bei der Synthese und Analyse in frühen Phasen rechnerbasiert unterstützen soll. Hierin ist auch eine Simulationsfunktionalität integriert. Der Zweck der Simulation und der Umgang damit im Prozess werden jedoch nicht erläutert. Zudem beschränkt sich diese Simulation nur auf die Konzeptphase, die späteren Phasen werden nicht berücksichtigt. Weiterhin wird auch nicht diskutiert, wie sich Erkenntnisse aus Simulationen auf den Prozessablauf und das Produkt auswirken. Entsprechend werden auch Iterationen von Schön nicht angesprochen.

Die starke Ausrichtung des gesamten Prozessmodells auf die interdisziplinäre Entwicklung wird deutlich in der Aufteilung in eine domänenübergreifende Konzeptphase und die darauffolgende domänenspezifische Entwicklung. Eine abschließende Integration des Systems, wie zum Beispiel bei Isermann, ist nicht explizit vorgesehen.

3.2.5.5 Mechatronischer Entwurf nach Lückel

Lückel [LüKS00] stellt das in Abbildung 3.14 dargestellte Prozessmodell vor. Es basiert auf den Ideen, bereits von Beginn an domänenübergreifend und rechnerunterstützt zu denken und zu arbeiten. Wie bereits viele der vorher beschriebenen Vorgehensmodelle, sieht auch das von Lückel entwickelte Modell zunächst die Funktionsmodellierung und darauf basierend eine Lösungssuche vor.

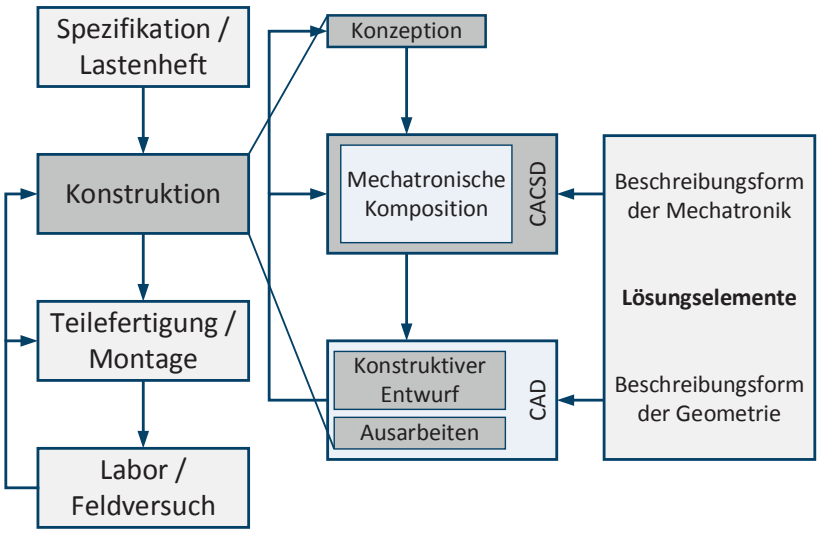


Abbildung 3.14: Prozessmodell für den Entwurf mechatronischer Systeme nach Lückel [LüKS00]

Im Prozessmodell selbst sieht Lückel lediglich Labor- und Feldversuche am Ende des Prozesses vor. Deren Zweck und Durchführung werden jedoch nicht weiter erläutert. In dem in [LüKS00] beschriebenen Anwendungsbeispiel werden zusätzlich Simulationen genutzt, um Konzeptstudien durchzuführen. Die hierfür erforderliche mathematische Modellierung des Verhaltens erfolgt hauptsächlich manuell. Mögliche Softwarewerkzeuge werden nicht erwähnt. Der Einsatz von Simulationen über die Konzeptphase hinaus wird nicht diskutiert.

Obwohl Lückel erwähnt, dass die Entwicklung mechatronischer Systeme rechnerunterstützt erfolgen soll, geht die Beschreibung des Vorgehensmodells nicht derart in die Tiefe, dass die entsprechenden Techniken und Methoden erläutert werden. Neben Simulationen wird im Prozessmodell nur am Rande der Einsatz von CAD (Computer-aided Design) erwähnt.

Entsprechend der wenig tiefgehenden Erläuterungen der Tätigkeiten zur Eigenschaftsanalyse erfolgt auch keine Beschreibung, wie daraus entstehende Ergebnisse im Prozess genutzt werden. Einzig der kurze Hinweis, dass Arbeitsschritte so lange durchlaufen werden, bis die Anforderungen erfüllt sind, stellt einen Zusammenhang zwischen Analysen und Iterationen her. Ansonsten bleiben Iterationen weitgehend unbeachtet.

Ebenso wie in den bereits diskutierten Vorgehensmodellen betont auch Lückel die Notwendigkeit einer domänenübergreifenden Zusammenarbeit in der Mechatronikentwicklung. Er berücksichtigt dies insbesondere in frühen Phasen durch eine Beschreibungsform, welche die mechatronische Konzeptfindung unterstützen soll. Interdisziplinäre Aspekte, die nach der Konzeptphase auftreten, berücksichtigt er nicht.

3.2.5.6 Entwicklungsvorgehen nach Gausemeier

Gausemeier et al. [GFDK09] entwickeln eine Spezifikationstechnik, die insbesondere bei der Konzepterstellung mechatronischer Systeme unterstützen soll und daher den interdisziplinären Charakter besonders berücksichtigt. Dabei wird diese Spezifikationstechnik um ein methodisches Vorgehen ergänzt, jedoch beschränkt sich dieses auf die Anwendung der Technik und damit auf die Konzeptphase. Auch wenn die Wichtigkeit iterativen Vorgehens erwähnt wird, werden Iterationen nicht weiter diskutiert. Eigenschaftsanalysen werden aufgrund der Konzentration auf die Spezifikationstechnik nicht behandelt, wodurch der Umgang mit Analyseergebnissen ebenfalls nicht erläutert wird. Entsprechend beschränkt sich auch die Verwendung von Computerunterstützung auf den Einsatz der Spezifikationstechnik. Virtuelle Techniken inklusive Eigenschaftsanalysen werden jedoch in anderen Arbeiten von Gausemeier et al. beschrieben, beispielsweise in [GMPB04] und [BAGG11]. Eine ganzheitliche Integration erfolgt jedoch nicht.

3.2.5.7 VDI-Richtlinie 2206

Die VDI-Richtlinie 2206 [VDI2206] beschäftigt sich mit der domänenübergreifenden Entwicklung mechatronischer Systeme, wobei insbesondere Vorgehensweisen, Methoden und Werkzeuge für den Systementwurf im Vordergrund stehen sollen. Das Vorgehen beruht auf drei Elementen: einem allgemeinen Problemlösungszyklus auf Mikroebene, dem V-Modell auf Makroebene sowie vordefinierten Prozessbausteinen für wiederkehrende Arbeitsschritte. Der Problemlösungszyklus beschreibt den allgemeinen Ablauf bei der Lösung von Problemen, von der Situationsanalyse über die Generierung und Auswahl geeigneter Lösungen bis hin zur Planung des weiteren Vorgehens. Dieser Zyklus tritt in jeder der Entwicklungsphasen auf. Das in Abbildung 3.15 dargestellte V-Modell beschreibt den Makrozyklus der Richtlinie, also das gesamte Vorgehen inklusive Entwicklungsphasen. Im Systementwurf wird ein Konzept entwickelt, welches dann in den einzelnen Domänen detailliert wird. Abschließend werden diese Einzelentwicklungen wieder zu einem Gesamtsystem integriert und dessen Eigenschaften analysiert. Somit verfolgt die VDI-Richtlinie, wie auch Isermann, ein Top-Down-Bottom-Up-Vorgehen. Für jede der Phasen des V-Modells werden in der Richtlinie Prozessbausteine beschrieben, die den Ablauf innerhalb dieser Phasen detaillieren.

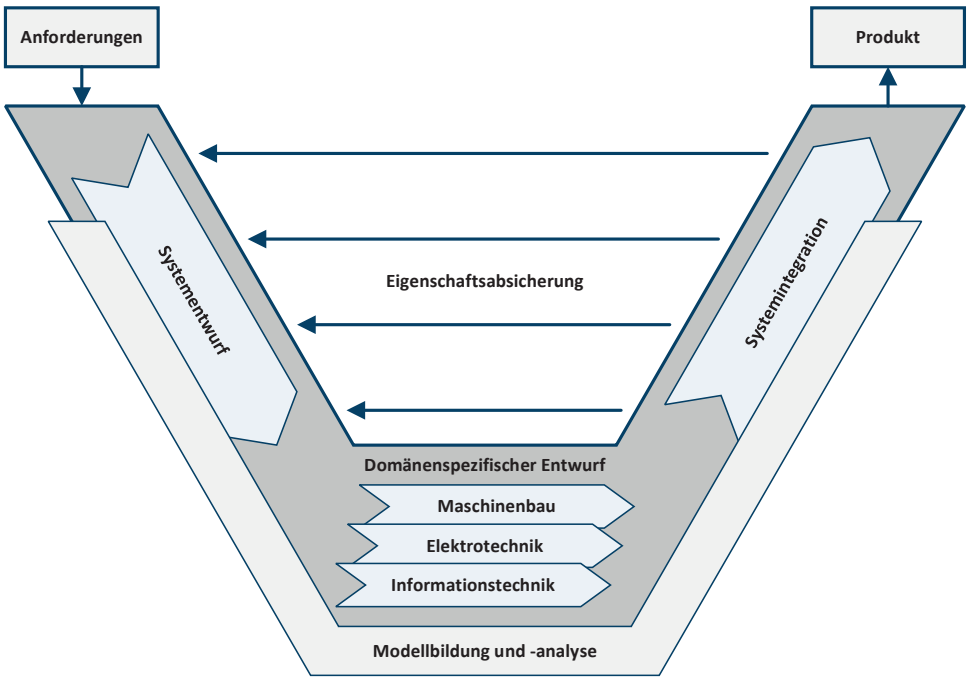


Abbildung 3.15: V-Modell als Makrozyklus nach [VDI2206]

Im Prozessmodell selbst sind Eigenschaftsabsicherungen am Ende eines Makrozyklus-Durchlaufs vorgesehen, vorwiegend über verschiedene Arten von Mustern und Prototypen. Auf dieser Prozessebene wird jedoch nicht weiter detailliert, wozu diese Prototypen eingesetzt werden und welche Voraussetzungen erfüllt sein müssen. Im Problemlösungszyklus, also im Mikrozyklus, erfolgt auch eine Analyse der Eigenschaften. Die konkrete Einbindung in Entwicklungsphasen sowie der Ablauf und die eingesetzten Techniken bei diesen Analysen werden nicht ausgeführt.

Im Makrozyklus sieht die Richtlinie den durchgängigen, begleitenden Einsatz von Modellbildung und -analyse, das heißt Simulation, vor. Zusätzlich erfolgt eine detaillierte Beschreibung des Ablaufs der Modellierung und Simulation im Allgemeinen. Weiterhin wird eine Übersicht zu den Modellierungs- und Simulationstechniken gegeben. Die Einordnung, wo genau im Prozess welche Simulationen zu welchem Zweck genutzt werden, wird jedoch nicht erläutert.

Auch in der VDI-Richtlinie wird der Einsatz von Computerunterstützung als unumgänglich für die Entwicklung mechatronischer System angesehen. Neben den bereits erwähnten Aspekten der Simulation werden auch weitere Werkzeuge der computerbasierten Entwicklung allgemein beschrieben, beispielsweise CAD- und CASE- (Computer-aided Software Engineering) Werkzeuge. Stellenweise fehlen hier jedoch eine genauere Detaillierung sowie die konkrete Einordnung im Entwicklungsprozess.

Allgemein wird beschrieben, dass Analysen einerseits zum Vergleich verschiedener Konzepte eingesetzt werden können, andererseits aber auch, um das Verhalten einer neuen oder vorhandenen Lösung zu analysieren und zu optimieren. Jedoch erfolgt die Beschreibung, wie der Umgang mit Analyseergebnissen, nicht im Detail. Bezüglich der meist durch Analyseerkenntnisse ausgelösten Iterationen werden entsprechend auch keine detaillierteren Anweisungen gegeben. Die Richtlinie sieht Iterationen hauptsächlich im Makrozyklus vor. Innerhalb des Problemlösungszyklus sind wiederum Iterationen vorgesehen, die, übertragen auf den Makrozyklus, als Mikroiterationen angesehen werden können. Auslöser und Auswirkungen dieser Iterationen werden allerdings nicht diskutiert.

Die Ausrichtung der Richtlinie auf die domänenübergreifende Zusammenarbeit wird im gesamten Verlauf deutlich. Neben der Strukturierung des Gesamtprozesses in Systementwurf, domänenspezifischen Entwurf und Systemintegration werden auch organisatorische Aspekte, wie beispielsweise die Zusammenstellung interdisziplinärer Projektteams behandelt.

Das V-Modell existiert in zahlreichen Abwandlungen. Hierzu zählt auch das W-Modell von Anderl und Nattermann [NaAn13]. Dieses unterscheidet sich hauptsächlich durch den Zwischenschritt der „Virtuellen Systemintegration“, welcher die W-Form verantwortet. Hierbei wird auf Modellebene bereits eine Systemintegration durchgeführt. Da das W-Modell sich in den Grundzügen nicht maßgeblich vom V-Modell der VDI-Richtlinie 2206 unterscheidet, wird es hier nicht eingehend betrachtet.

3.2.5.8 3-Ebenen-Vorgehensmodell nach Bender

Bender [Bend05] entwickelt ein Vorgehensmodell für die Entwicklung mechatronischer Systeme, wobei der Fokus auf Embedded Systems liegt. Die Grundstruktur bildet, wie bereits von VDI 2206 und Isermann bekannt, das V-Modell. Während der linke Ast die Entwicklungstätigkeiten repräsentiert und hierarchisch in System-, Subsystem und Komponentenebene unterteilt ist, bildet der rechte Ast die dazugehörigen Testaktivitäten ab. Das Modell soll dazu dienen, die den Domänen eigenen Vorgehensmodelle in einer gemeinsamen Vorgehensweise zusammenzuführen.

Eigenschaften werden im Modell nach Bender erst nach den Entwicklungstätigkeiten im rechten Ast durchgeführt. Hierbei werden Tests von Komponenten- bis Systemebene durchgeführt, die sich hauptsächlich auf physische Prototypen stützen. Diese Tests werden nicht weiter erläutert. Unter dem Begriff des virtuellen Funktionstest – welcher ähnlich der Simulation ist – be-

schreibt Bender zudem allgemein, dass über den gesamten Prozess verteilt Test mit virtuellen Modellen durchgeführt werden sollen. Hierbei liegt der Fokus entsprechend der Embedded-Systems-Ausrichtung auf Software und Elektronik. Weiterhin fehlt eine detaillierte Beschreibung geeigneter Techniken und Werkzeuge. Abgesehen von den virtuellen Funktionstests und vereinzelten Hinweisen auf den Einsatz anderer computerbasierter Methoden, zum Beispiel für das Anforderungsmanagement, erfolgt keine weitere Einbindung der Computerunterstützung in den Gesamtprozessverlauf.

Entsprechend der eher oberflächlichen Beschreibung der Analysetätigkeiten erfolgt auch keine Beschreibung, wie Analyseergebnisse im Prozess genutzt werden. Ebenso spielen Iterationen eine untergeordnete Rolle. Es erfolgt einzig der Hinweis darauf, dass der gesamte Prozessablauf mehrfach durchlaufen werden muss, mit jeweils steigendem Reifegrad. Diese Makroiterationen werden einer simulationsbasierten Entwicklung nicht gerecht.

Ebenso wie die Richtlinie VDI 2206 legt auch Bender ein besonderes Augenmerk auf die Interdisziplinarität bei der Entwicklung mechatronischer Systeme. Jedoch geht die Beschreibung der Zusammenarbeit weniger in die Tiefe als beispielsweise in [VDI2206]. Zudem wird hier auch der Fokus auf Embedded Systems wieder deutlich.

3.2.5.9 Entwicklungsvorgehen nach Eigner

Eigner et al. [EiGZ12] präsentieren einen Ansatz, der prinzipiell auf dem V-Modell der VDI-Richtlinie 2206 basiert. Dieses wird jedoch insbesondere im linken Ast um Aspekte des Systems Engineering und der virtuellen Eigenschaftsabsicherung ergänzt. Hierbei stützen sich Eigner et al. auf das in Abbildung 3.16 dargestellte und von ihnen entwickelte X-Modell [EGDF14], welches das Y-Diagramm der Elektronikentwicklung (siehe Kapitel 3.2.3.1) um die Ebene der Funktion erweitert und somit alle im Entwicklungsprozess relevanten Sichten beinhaltet. Grundidee dieses Modells ist es, dass je nach Anwendungsfall die entsprechende Sicht – Funktion, Verhalten, Struktur oder Geometrie – berücksichtigt werden kann und somit ein für alle Domänen und die Mechatronik allgemeingültiges Modell zur Verfügung steht.

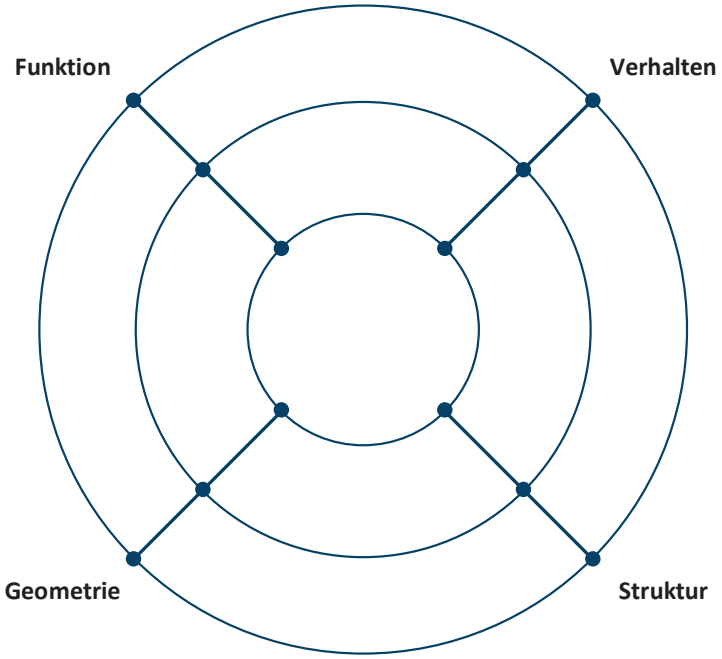


Abbildung 3.16: X-Modell nach Eigner et al. [EGDF14]

Das erweiterte V-Modell ist nach dem RFLP-Ansatz strukturiert, wie er beispielsweise auch in CATIA Systems umgesetzt ist (siehe Kapitel 3.1.6.5). Somit beginnt der Entwicklungsprozess mit der Anforderungsbeschreibung, gefolgt von der Funktionsmodellierung. Basierend darauf wird ein Lösungskonzept durch die Definition logischer Komponenten erstellt, was abschließend im domänenspezifischen Entwurf mit einer physischen Umsetzung endet. Insbesondere im linken Ast binden Eigner et al. Modellierung und Simulation wesentlich detaillierter ein als beispielsweise die VDI-Richtlinie. Somit sind bereits sehr früh kontinuierliche Eigenschaftsanalysen vorgesehen. Spezifische Simulationstechniken und konkretere Hinweise, welche Eigenschaften analysiert werden müssen, stehen jedoch nicht im Fokus.

Über die Simulation hinaus werden auch weitere Techniken der Computerunterstützung integriert, wie beispielsweise die Systemmodellierung mit SysML und UML. Zudem bestehen Verknüpfungen zu PLM-Systemen. Der Umgang mit Analyseergebnissen und Iterationen steht allerdings nicht im Vordergrund und wird von den Autoren daher nicht im Detail diskutiert. Jedoch wird erwähnt, dass Iterationen über mehrfache Durchläufe des V-Modells möglich und erforderlich sind.

Der interdisziplinäre Charakter steht, ähnlich wie bei der VDI-Richtlinie 2206, im Vordergrund und ist entsprechend durchgängig berücksichtigt.

3.2.5.10 Zusammenfassung der Ergebnisse

Die Ergebnisse der Analyse der Entwicklungsmethodiken aus dem Bereich der Mechatronik sind in Tabelle 3.4 zusammengefasst.

Tabelle 3.4: Zusammengefasste Analyse der Entwicklungsmethodiken aus der Mechatronik (+: im Detail berücksichtigt; ○: berücksichtigt; -: nicht berücksichtigt)

Methodik	Eigenschafts-analyse	Analyse-ergebnisse	Einsatz von Simulation	Computer-unterstütz-ung	Iterationen	Interdiszi-plinarität
Spiralmodell	+	○	-	-	○	○
Kallenbach	○	-	-	○	-	○
Isermann	○	-	○	○	-	+
Schön	-	-	○	○	-	○
Lückel	○	-	○	○	○	○
Gausemeier	○	-	○	○	-	+
VDI 2206	○	-	○	+	○	+
Bender	○	-	-	○	-	○
Eigner	+	-	+	+	○	+

3.2.6 Systems Engineering

Ein in den letzten Jahren immer häufiger in der Literatur auftauchender Ansatz ist der des Systems Engineering (SE). Dieser hat sich aufgrund der rasanten technischen Entwicklungen während des Zweiten Weltkriegs und danach entwickelt und ist auch heute noch von militärischen Einrichtung geprägt, wie etwa an den diversen Systems-Engineering-Leitfäden des United States Department of Defense (DoD) oder der NASA (National Aeronautics and Space Administration) ersichtlich wird.

Häufig wird Systems Engineering fälschlicherweise mit den rein technischen Disziplinen, wie beispielsweise der Mechatronik, gleichgesetzt. Zur Verdeutlichung des eigentlichen Fokus sollen die beiden folgenden Definitionen dienen:

> *SE is a discipline that concentrates on the design and application of the whole (system) as distinct from the parts. It involves looking at a problem in its entirety, taking into ac-*

> *count all the facets and all the variables and relating the social to the technical aspects. [FAA06]*
>
> *Systems Engineering consists of two significant disciplines: the technical knowledge domain in which the systems engineer operates, and systems engineering management. [DoD01]*

Demnach werden im Systems Engineering nicht nur technische Aspekte betrachtet, sondern zudem auch noch Management-Aspekte [MöSt10]. Weiterhin liegt der Fokus nicht rein auf der Produktentstehung, sondern auf dem gesamten Lebenszyklus eines Produktes. Dies bedeutet, dass das System von der ersten Idee über die Entstehung und Nutzung bis hin zum Recycling betrachtet und gemanagt wird. Einen wichtigen Aspekt stellt zudem die Integration des Menschen als Teil des Systems dar. Dies muss nicht bedeuten, dass der Mensch aktiv mit dem System interagiert, sondern kann ebenso bedeuten, dass Auswirkungen eines Systems auf den Menschen und dessen Umfeld berücksichtigt werden.

Entsprechend kann Systems Engineering als eine Art Rahmenwerk gesehen werden, welches unter anderem die in den vorangegangen Kapiteln diskutierten Vorgehensmodelle der einzelnen Domänen als Teilaspekte integriert. Eine Gleichsetzung von Systems Engineering im ursprünglichen Sinne mit Mechatronik wäre – nach Meinung des Autors – somit nicht korrekt.

Jedoch haben viele Vorgehensmodelle der Einzeldomänen Techniken und Methoden des Systems Engineering adaptiert und in ihrem Entwicklungsvorgehen integriert. So nutzen beispielsweise mehrere Mechatronik-Entwicklungsmethodiken das Top-Down-Prinzip. Insbesondere die VDI-Richtlinie 2206 nutzt zudem auch den Problemlösungszyklus, der in dieser Form seinen Ursprung im Systems Engineering hat.

Ein weiterer Trend liegt im Modellbasierten Systems Engineering (MBSE). Dabei wird angestrebt, alle Dokumente aus dem klassischen Vorgehen, beispielsweise Anforderungslisten, durch modellbasierte Ansätze zu ersetzen. Hierdurch sollen höhere Effizienz, bessere Rückverfolgbarkeit und weniger Dokumentationsaufwand ermöglicht werden. Der Einsatz der Modellierungssprache SysML (siehe Kapitel 3.1.6.2) ist im Modellbasierten Systems Engineering weit verbreitet.

Trotz des Einsatzes von Modellen ist es im Modellbasierten Systems Engineering nicht obligatorisch, dass diese Modelle ausführbar und damit simulierbar sind, viele sind rein statisch. Daher ist Modellbasiertes Systems Engineering klar zu trennen von simulationsbasierter Entwicklung.

Entsprechend der Abgrenzung zwischen Systems Engineering und (simulationsbasierter) Mechatronikentwicklung werden hier keine Vorgehen des Systems Engineering analysiert. Nichtsdestotrotz bleiben die Ansätze und Methoden auch im weiteren Verlauf dieser Arbeit relevant.

Für weitergehende Informationen über Systems Engineering und die dort verbreiteten Ansätze und Prozesse sei auf die umfangreiche Literatur verwiesen, beispielsweise [DoD01], [Hask11], [HWFV12], [Koss11] und [SHAC07].

4 Erkenntnisse aus den Analysen und Handlungsbedarf

4.1 Erkenntnisse aus den Analysen

4.1.1 Analyse von Modellierungs- und Simulationstechniken

Aus der Analyse des Stands der Technik bezüglich Modellierungs- und Simulationstechniken in Kapitel 3.1 wird deutlich, dass dieser Bereich bereits sehr weit fortgeschritten ist. Sowohl domänenspezifisch als auch domänenübergreifend existiert eine Vielzahl von Techniken und Werkzeugen, wodurch sehr vielfältige Einsatzmöglichkeiten für Simulationen entstanden sind. Darüber hinaus bestehen intensive Forschungsaktivitäten, um diese Techniken zu verbessern und neue Techniken zu entwickeln. Über die analysierten Techniken hinaus existieren zudem anwendungsfallspezifische Techniken und Werkzeuge, welche diese Tatsache noch weiter verdeutlichen. Entsprechend werden in diesen technologischen Grundlagen keine bedeutenden Hindernisse für eine simulationsbasierte Entwicklungsmethodik gesehen. Jedoch stellen viele dieser Werkzeuge Insellösungen dar, weshalb eine Prozessintegration oftmals nicht vorhanden ist.

4.1.2 Analyse von Entwicklungsmethodiken

Die Ergebnisse der in Kapitel 3.2 durchgeführten Analyse von Entwicklungsmethodiken sind in Tabelle 4.1 zusammengefasst.

In den Analysen werden teilweise signifikante Unterschiede zwischen den Domänen deutlich. So sind Eigenschaftsanalysen im Bereich der Mechanik hauptsächlich erst spät im Prozess vorgesehen, zudem basieren diese meist auf physischen Prototypen. Im Gegensatz dazu sind die Bereiche Elektronik- und Softwareentwicklung wesentlich stärker auf den Bereich der Eigenschaftsanalysen ausgerichtet: Hier werden bereits sehr früh im Prozess Eigenschaften abgesichert. Stellenweise werden bereits vor den ersten Entwürfen Testszenarien definiert, etwa im Extreme Programming oder in der Elektronikentwicklung nach Siegl. Die Entwicklungsmethodiken der Mechatronik lassen in diesem Bereich erkennen, dass sie oftmals auf Methodiken der Mechanik gründen, wobei jedoch Ansätze für eine bessere Integration der Eigenschaftsanalysen erkennbar sind. Ein ähnliches Bild zeigt sich bei der Integration von Simulationen in den Prozessverlauf: Im Bereich der Mechanik erfolgt diese kaum, wohingegen die Elektronikentwicklung stark simulationsorientiert arbeitet. Auch hier zeigen die Mechatronikmethodiken eine

stärkere Integration als die aus der Mechanik, haben jedoch im Vergleich zur Elektronik deutliche Defizite.

Hinsichtlich des Umgangs mit aus Analysen gewonnenen Ergebnissen ist insgesamt zu erkennen, dass dieser Umgang kaum im Entwicklungsvorgehen behandelt wird. Vor allem die Entwicklungsmethodiken aus dem Bereich der Mechatronik weisen hier deutliche Lücken auf.

Die Abbildung und Beschreibung von Iterationen zeigt, dass in der Softwareentwicklung teilweise hochiterative Prozesse genutzt werden. Die restlichen Domänen setzen sich mit Iterationen sowie deren Ursprüngen und Auswirkungen deutlich weniger intensiv auseinander, sowohl auf Makro- als auch auf Mikroebene.

Auffällig im Bereich der Mechatronikentwicklung ist die Übereinstimmung über die Notwendigkeit der Computerunterstützung. Jedoch ist deren Integration noch nicht derart fortgeschritten, dass sie durchgängig und konsistent im gesamten Prozess genutzt wird.

Sehr deutlich im Vergleich zu den Einzeldomänen ist ebenso, dass die meisten Vorgehensmodelle aus der Mechatronik die interdisziplinäre Zusammenarbeit stark in das Entwicklungsvorgehen integrieren. Meist geschieht dies durch ein Top-Down-Vorgehen mit anschließender Bottom-Up-Integration. Dieser Grundgedanke entstammt dem Systems Engineering.

Zusammenfassend kann festgestellt werden, dass die Entwicklungsmethodiken der Mechatronik derzeit noch Optimierungspotential im Bereich der Eigenschaftsanalysen besitzen. Hierzu zählen die Integration von Analysen allgemein, aber insbesondere auch Simulationen. Die Domänen Elektronik und Informatik sind in diesem Bereich als Vorreiter anzusehen. Darüber hinaus wird derzeit kaum Unterstützung beim Umgang mit Analyseergebnissen und den damit verbundenen Iterationen geboten. Im Hinblick auf eine simulationsbasierte Entwicklung stellen diese Aspekte daher Kerneigenschaften einer Entwicklungsmethodik dar.

Tabelle 4.1: Zusammengefasste Analyse der Entwicklungsmethodiken aus Mechanik, Elektronik, Software und Mechatronik

(+: im Detail berücksichtigt; ○: berücksichtigt; -: nicht berücksichtigt)

Methodik	Eigenschafts-analyse	Analyse-ergebnisse	Einsatz von Simulation	Computer-unterstütz-ung	Iterationen	Interdiszi-plinarität
Mechanik						
Pahl/Beitz	○	-	-	-	○	○
VDI 2221	○	-	-	-	○	○
Ehrlenspiel	○	○	-	-	○	○
MVM	○	-	-	-	+	-
Ulrich/ Eppinger	○	○	-	-	○	○
Ullman	+	+	-	-	○	-
Elektronik						
VDI 2422	○	○	-	-	-	○
Siegl	+	○	+	-	-	-
Gausemeier	○	○	+	-	-	-
Software						
Wasserfall-modell	+	-			+	-
Spiralmodell	+	-			+	○
Extreme Program-ming	+	+			+	o
Mechatronik						
Spiralmodell	+	○	-	-	○	○
Kallenbach	○	-	-	○	-	○
Isermann	○	-	○	○	-	+
Schön	-	-	○	○	-	○
Lückel	○	-	○	○	○	○
Gausemeier	○	-	○	○	-	+
VDI 2206	○	-	○	+	○	+
Bender	○	-	-	○	-	○
Eigner	+	-	+	+	○	+

4.2 Ableitung des Handlungsbedarfes

In Kapitel 3.1 wurde ein Überblick über die Möglichkeiten der Modellierung und Simulation im Umfeld mechatronischer Systeme gegeben. Hieraus wird ersichtlich, dass insbesondere die Einzeldomänen sehr umfangreiche und ausgereifte Möglichkeiten besitzen. Domänenübergreifende Techniken haben in den vergangenen Jahren ebenfalls ein signifikantes Wachstum erfahren und können in den meisten Fällen als ausgereift beschrieben werden. Darüber hinaus bestehen weiterhin Forschungsaktivitäten, die Modellierungstechniken und Simulationswerkzeuge weiter verbessern sollen.

In Kapitel 3.2 wurde analysiert, inwieweit die identifizierten Möglichkeiten der Modellierung und Simulation bereits in Entwicklungsmethodiken integriert sind. Zusätzlich zu den Einzeldomänen wurde vor allem der Bereich der Entwicklungsmethodiken für mechatronische Produkte analysiert. Ergebnis dieser Analyse ist, dass der Aspekt der simulationsbasierten Entwicklung in keiner der Methodiken durchgehend berücksichtigt wird. Besonders die Methodiken aus der Mechanik haben, im Gegensatz zum Bereich der Elektronik und Software, nur eine sehr geringe Integrationstiefe von Simulation. Der insbesondere für diese Arbeit relevante Bereich der Mechatronik zeigt im Vergleich zur Mechanik bereits deutliche Fortschritte, jedoch wird auch hier simulationsbasierte Entwicklung nicht ermöglicht.

Zwar existiert ein Vielzahl von Arbeiten, die Teilaspekte simulationsbasierter Entwicklung adressieren, insbesondere im Bereich der Modellierungstechniken und Simulationswerkzeuge, etwa Paredis et al. [PDSK01] (siehe Kapitel 3.1.6.4). Forschungsarbeiten bezüglich einer umfassenden Methodik mit dem Ziel der simulationsbasierten Entwicklung sind dem Autor jedoch nicht bekannt.

Entsprechend wird in der Verbindung der beiden Säulen simulationsbasierter Entwicklung – Modellierungs- und Simulationstechniken sowie Entwicklungsmethodiken – bedeutendes Potential zur Optimierung gesehen. Hieraus leitet sich das Hauptziel dieser Arbeit ab, die Entwicklung einer umfassenden Methodik, welche die simulationsbasierte Entwicklung mechatronischer Systeme ermöglichen und Entwickler dabei unterstützen soll.

Basierend auf den Erkenntnissen der Analysen können die initialen Forschungsfragen aus Kapitel 1 weiter detailliert werden:

F1. Wie können Analysen, und insbesondere Simulationen, sinnvoll in den Prozessverlauf eingebunden werden?
F2. Wie können Iterationen organisiert und dargestellt werden?
F3. Wie wirken sich Ergebnisse aus der Simulation auf vorangegangene und nachfolgende Entwurfsaktivitäten aus?
F4. Zu welchem Zeitpunkt können und müssen welche Eigenschaften analysiert werden?
F5. Wie können Entwickler dabei unterstützt werden, hierfür die passende Simulation zu wählen?
F6. Wie kann mit Unsicherheiten hinsichtlich Entwurfs- und Modellparametern, insbesondere in frühen Phasen, umgegangen werden?
F7. Wie können Modelle, Simulationen und Simulationsergebnisse entlang des Prozesses und über die Domänen organisiert und verknüpft werden?
F8. Welche Softwarewerkzeuge können von Entwicklern für die einzelnen Simulationen genutzt werden?
F9. Wie können die verschiedenen Domänen und deren Simulationen verknüpft werden?

Diese detaillierten Forschungsfragen sollen im Rahmen der vorliegenden Arbeit beantwortet werden. Entsprechend wird im folgenden Kapitel die Entwicklung einer simulationsbasierten Entwicklungsmethodik beschrieben. Anschließend wird in Kapitel 6 das entwickelte Konzept an einem mechatronischen Entwicklungsprojekt validiert und veranschaulicht.

5 Methodik für die simulationsbasierte Entwicklung mechatronischer Systeme

In diesem Kapitel wird eine Entwicklungsmethodik für die simulationsbasierte Entwicklung mechatronischer Systeme beschrieben, welche die zuvor identifizierten Defizite adressiert und die Forschungsfragen aus Kapitel 4.2 beantworten soll. Hierzu wird in Kapitel 5.1 zunächst ein Gesamtrahmen für die Methodik definiert. Aus diesem Rahmen betrachtet die vorliegende Arbeit einen Kernausschnitt, der die Grundlage für eine langfristige Weiterentwicklung der Gesamtmethodik bildet. Basierend auf den Forschungsfragen aus Kapitel 4.2 sowie den Erkenntnissen der Analyse des Stands der Technik aus den Kapiteln 3.1 und 3.2 werden in Kapitel 5.2 zunächst Anforderungen an die zu entwickelnde Methodik abgeleitet. Darauf basierend erfolgt in den Kapiteln 5.3 bis 5.7 die Herleitung der Methodik. Diese wird im Rahmen von Kapitel 6 validiert.

5.1 Gesamtkonzept für eine Methodik und Eingrenzung der Arbeit

Nach Ulrich und Eppinger [UlEp08] lässt sich der Produktentwicklungsprozess in sechs Hauptphasen einteilen, beginnend mit der Planungsphase über insgesamt vier Entwicklungsphasen bis hin zum Produktionsanlauf (siehe Kapitel 3.2.2.5). In Abbildung 5.1 ist dargestellt, wie sich die hier präsentierte Methodik in diesen Gesamtprozess einordnet: Ausgangspunkt ist eine abgeschlossene Produktplanung, die Produktanforderungen als Eingangsgröße liefert. Als Ausgangsgröße liefert die Methodik eine fertige Produktdokumentation, die wiederum als Eingang für den Produktionsanlauf dient. Kern der Arbeit sind somit die vier Phasen zwischen Produktplanung und Produktionsanlauf, die durch die Methodik im Gesamten abgebildet werden sollen. Es sei jedoch darauf hingewiesen, dass im Sinne des Simultaneous oder Concurrent Engineering (siehe hierzu beispielsweise [Ehrl09] und [PaSu93]) die Berücksichtigung von Fertigungsaspekten bereits in den eigentlichen Entwicklungsphasen vorteilhaft sein kann. Insbesondere die simulationsbasierte Entwicklung kann hier ein adäquates Hilfsmittel darstellen, da bereits parallel zur Entwicklung auch Fertigungsprozesse simuliert werden können.

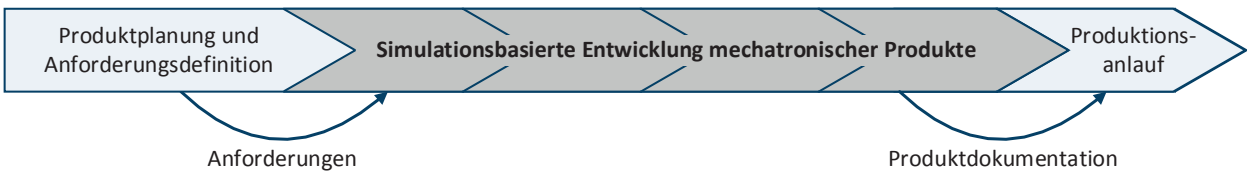


Abbildung 5.1: Einordnung der Methodik im Entwicklungsprozess, basierend auf [UlEp08]

In [DoVi12a] wurde vom Autor der grundlegende Gesamtrahmen der Methodik präsentiert, der in seinen Grundzügen auf der Darstellung von Vielhaber et al. [ViBC10] basiert. Die in dieser Arbeit überarbeitete Darstellung sieht den Entwicklungsprozess und ein entsprechendes Prozessmodell als den Kern der Methodik an. Hierum ordnen sich verschiedene Sichten, die wichtige Aspekte der simulationsbasierten Entwicklung einbinden sollen. Diese Aufteilung leitet sich aus der Definition des Begriffs der Methodik aus Kapitel 2.2 ab und greift ebenso die Sichtenaufteilung nach Burr et al. auf [BuMV07]. Zudem fließen Erkenntnissen aus den in Kapitel 3 durchgeführten Analysen, aus Diskussionen mit Experten aus Industrie und Forschung sowie aus Erfahrungen in der Anwendung entsprechender Simulationswerkzeuge ein. Das Gesamtkonzept ist in Abbildung 5.2 dargestellt.

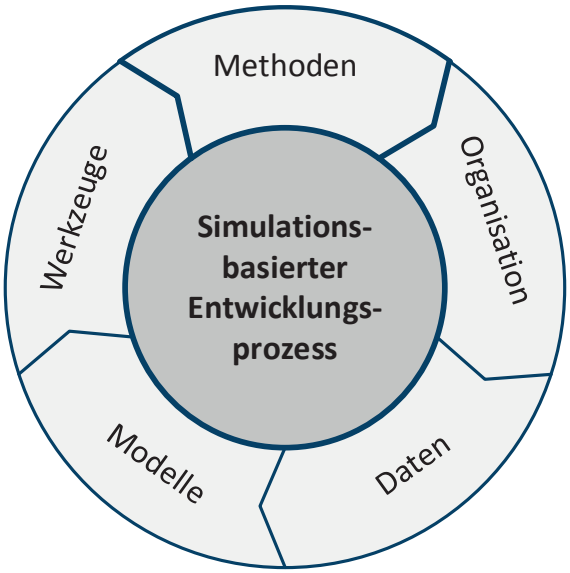


Abbildung 5.2: Gesamtkonzept der Entwicklungsmethodik

Demnach gliedert sich das Gesamtkonzept in insgesamt sechs Sichten:

- **Prozess:** Die Prozesssicht wird als Grundlage der Methodik angesehen. Diese Sicht beschreibt die einzelnen Tätigkeiten in der Entwicklung sowie deren Abfolge in Form eines Prozesses. Hierbei ist die Erreichung eines Ziels – in diesem Fall die Erfüllung der Anforderungen – das bestimmende Element. Der Prozess bildet gemeinsam mit den anderen Sichten die Methodik.
- **Methoden:** Entwickler müssen entscheiden, wann im Prozess welche Simulation zu welchem Zweck eingesetzt werden soll. Darüber hinaus müssen Ergebnisse interpretiert und entsprechende Entscheidungen getroffen werden. Hierfür müssen Methoden bereitgestellt werden, die Entwickler in unterschiedlichen Entscheidungen im Prozessver-

lauf unterstützen. Methoden müssen dabei oftmals werkzeug- oder modellspezifische Aspekte berücksichtigen.

- **Modelle:** Grundlage von Simulationen bilden stets Modelle (siehe Kapitel 2.4). Im Entwicklungsprozess kommt eine Vielzahl unterschiedlicher, gegebenenfalls abhängiger Modelle zum Einsatz, im Allgemeinen mit steigendem Detaillierungsgrad während des Prozessverlaufs [RVPK01]. Zudem sind in den verschiedenen Modellen oftmals Informationen redundant abgebildet, wodurch Ressourcen verschwendet werden. Entsprechend ist die Berücksichtigung der Modellsicht essentiell.
- **Daten:** Die Grundlage für Modelle selbst stellen Daten dar. Dabei beeinflusst die Qualität dieser Daten auch direkt die Qualität des Modells. Merkt und Krastel [MeKr06] zeigen anhand einer Studie, dass etwa die Hälfte der Zeit bei der Neuerstellung eines Simulationsmodells auf die Beschaffung von Daten entfällt. Somit wird unter Berücksichtigung der Modellvernetzung und der Informationsredundanz, die oftmals in Modellen vorzufinden ist, deutlich, dass Daten eine entscheidende Rolle in der simulationsbasierten Entwicklung spielen.
- **Werkzeuge:** Letztendlich werden Modelle und Daten in (Simulations-)Werkzeugen vereint und stellen über Benutzeroberflächen die direkte Verbindung zum Entwickler her. Die große Anzahl verschiedener Werkzeuge kann Entwickler jedoch schnell überfordern. Es ist daher notwendig, diese bei der Auswahl passender Werkzeuge zu unterstützen. Zudem ist entscheidend, dass Entwickler Unterstützung in der Benutzung sowie im Umgang mit benötigen Informationen erhalten.
- **Organisation:** Das Zusammenspiel der obigen Sichten und des Entwicklungsprozesses wird über die Organisation koordiniert. Hier ist festgelegt, wer welche Verantwortlichkeiten besitzt und wer welche Tätigkeiten ausführt. Insbesondere im Bereich der Mechatronik spielt die Kommunikation und Kooperation der Domänen eine wichtige Rolle [Brad10].

Eine alle Sichten gleichermaßen umfassende Bearbeitung und Integration dieses Gesamtkonzeptes zu einer Entwicklungsmethodik stellt eine langfristige Aufgabe dar, für die im Rahmen dieser Arbeit die Grundlagen gelegt werden sollen. Daher erfolgt an dieser Stelle eine Konzentration auf die grundlegenden Elemente, welche später als Basis einer weiteren Detaillierung der Methodik dienen sollen. Hierbei wird, wie bereits beschrieben und auch in Abbildung 5.2 hervorgehoben, ein Prozessmodell als zentrales Element angesehen. Damit eng verzahnt sind die

eingesetzten Methoden, weshalb diese hier neben dem Prozessmodell im Fokus stehen werden. Eine isolierte Betrachtung dieser beiden Sichten ist jedoch wenig sinnvoll und meist auch nicht möglich. So ist die Berücksichtigung der Organisationssicht beispielsweise bei der Prozessdefinition unerlässlich. Zudem beeinflussen Daten, Modelle und Werkzeuge, wie oben beschrieben, auch die eingesetzten Methoden, weshalb bei deren Beschreibung auch Teilaspekte dieser Sichten einfließen.

5.2 Anforderungen an eine Methodik

Als Grundlage der Entwicklung einer simulationsbasierten Methodik wurden in [DoVi12a] und [DoVi13] grundlegende Anforderungen an eine solche definiert und im Rahmen dieser Arbeit ergänzt:

A1. Einbindung kontinuierlicher Eigenschaftsanalyse durch Simulation
A2. Frühe Einbindung von Simulationen
A3. Übergang von Makro- zu Mikroiterationen
A4. Bereitstellung von Methoden zur Bestimmung zu analysierender Eigenschaften
A5. Bereitstellung von Methoden zur Auswahl von Werkzeugen zur Simulation
A6. Bereitstellung von Methoden zur Iterationssteuerung
A7. Berücksichtigung des interdisziplinären Charakters der Mechatronik
A8. Möglichkeit zur Bereitstellung von Modellen und der inhärenten Informationen entlang des Entwicklungsprozesses und über Domänen hinweg
A9. Einbindung der Methodik in organisatorische Strukturen
A10. Unterstützung der Methodik durch Daten und Modelle

Auf Prozessebene erfordert die Möglichkeiten der Simulation ein verändertes Vorgehen, um diese effektiv zu integrieren. So sind Simulationen im Gegensatz zu physischen Tests bereits sehr früh im Entwicklungsprozess einsetzbar [DoVi12b], [Lang09], [PDSK01] und im Bereich der Mechatronik oftmals sogar erforderlich [FoHZ12], [WiMo98]. Diese Bedeutung zeigen etwa Eigner et al. [EiGZ12] durch die frühe Einbindung virtueller Eigenschaftsabsicherungen in ihrer Methodik (siehe Kapitel 3.2.5.9). Gleichzeitig wird hierdurch das Konzept des Front-Loading unterstützt, welches die Entwicklungsleistung durch eine Verschiebung der Problemidentifikation und -lösung hin zu früheren Phasen des Entwicklungsprozesses fördert [ThFu00]. Darüber hinaus sind Simulationen und die entsprechenden Ergebnisse wesentlich schneller verfügbar, wodurch die Analysephasen beträchtlich beschleunigt werden können [BeSZ05]. Damit werden

Iterationen allgemein gefördert [DeSi05] und der Übergang von wenigen Makroiterationen zu vielen Mikroiterationen ermöglicht [DoVi12a], [DoVi13], wodurch die Produktreife während des Entwicklungsprozesses erhöht wird. Thomke und Fujimoto zeigen den Vorteil der kürzeren Iterationen am Beispiel der Crashsimulationen im Vergleich zu physischen Versuchen [ThFu00]. Daher muss ein simulationsbasierter Prozess zum einen bereits von Beginn an Eigenschaftsanalysen durch Simulation vorsehen. Zum anderen soll über den gesamten Prozessverlauf kontinuierlich und parallel zu Entwurfstätigkeiten die Analyse über Simulationen ermöglicht werden.

Wie aus der Analyse in Kapitel 3.1 ersichtlich, existieren zahlreiche Möglichkeiten der Simulation. Entsprechend ist es wichtig, Entwicklern Methoden bereitzustellen, die bei der Bestimmung von zu analysierenden Eigenschaften sowie bei der Auswahl der richtigen Analysemethode und dem dazugehörigen (Software-)Werkzeug Unterstützung bieten. Darüber hinaus ist der Umgang mit Simulationsergebnissen essentiell. Hierzu zählen die Steuerung von Iterationen, die Identifikation von Handlungsmaßnahmen sowie die Festlegung, wie solche zurück in den Entwurf fließen. Auch in diesen Aspekten soll die Methodik Unterstützung bieten.

Insbesondere in frühen Phasen ist die interdisziplinäre Zusammenarbeit unerlässlich. Aus diesem Grunde ist die Nutzung interdisziplinärer Modelle und Simulationen ausschlaggebend [RVPK01] und soll in der Methodik berücksichtigt werden. Die Modelle und Simulationsergebnisse, die während des Entwicklungsprozesses generiert werden, stellen gleichzeitig einen großen Wissensspeicher dar [MaBu11]. Darüber hinaus ändert sich die Modellierungstiefe im Prozessverlauf [PDSK01], [SBOW04], weshalb die Wiederverwendung und Detaillierung von Modellen eine große Rolle spielt. Modelle und die damit verbundenen Simulationen sollten daher im gesamten Entwicklungsprozess allen Beteiligten zur Verfügung gestellt werden, um die Wiederverwendung von Modellen über Entwicklungsphasen, Domänen und IT-Werkzeuge hinweg zu ermöglichen. Hierdurch lassen sich Modellredundanzen verringern, welche die Entwicklungseffizienz reduzieren.

Im Rahmen dieser Arbeit stehen insbesondere die Anforderungen A1 bis A8 im Vordergrund. Organisatorische Aspekte sollen jedoch am Rande mitbetrachtet werden. Diese umfassen etwa die Zuordnung von Verantwortlichkeiten zwischen Domänen sowie bei der Modellierung und Simulation. Ebenso werden die Daten-, Modell- und Werkzeugsichten mitberücksichtigt, wobei im Ausblick der Arbeit (siehe Kapitel 7) Möglichkeiten zu deren weiterer Integration in die Methodik diskutiert werden.

Im Folgenden wird eine Methodik beschrieben, welche die diskutierten Anforderungen adressiert und dem beschriebenen Aufbau entspricht. Hierzu wird zunächst die Prozesssicht betrachtet: Ausgehend von einer vereinfachter Beschreibung des Prozessmodells und dessen Kernelementen in den Kapiteln 5.3 und 5.4 erfolgt eine schrittweise Detaillierung in Kapitel 5.5. Hierzu zählen sowohl die detaillierte Beschreibung der einzelnen Phasen des Prozessmodells als auch die allgemeine Beschreibung von Iterationszyklen und Simulationsphasen, die bereits Methodenaspekte beinhalten. Ergänzend zu den Prozessaspekten wird in Kapitel 5.6 eine Methode zur Auswahl von Analyseverfahren beschrieben, die ebenso Aspekte der Werkzeug-Sicht berücksichtigt. Abschließend erfolgt in Kapitel 5.7 die Einbindung von Methoden zum Umgang mit Unsicherheiten bei Simulationen, um Simulationen möglichst früh einsetzen zu können. Diese Methode greift Aspekte der Daten-Sicht auf.

5.3 Überblick Prozessmodell

Abbildung 5.2 verdeutlicht die zentrale Bedeutung des Prozessmodells in der zu entwickelnden Methodik, weshalb hier mit dessen Beschreibung begonnen werden soll. Im Sinne der Übersichtlichkeit und des besseren Verständnisses wird hier zunächst eine vereinfachte Version gemäß Abbildung 5.3 erläutert, welche die Grundelemente des Prozessmodells verdeutlicht. Im Detail wird das Prozessmodell in Kapitel 5.5 beschrieben. Die Vorarbeiten, auf denen die hier vorgestellte Version des Prozessmodells basiert, wurden vom Autor in [DoVi12a], [DoVi13], [DoVi14a] vorgestellt.

Startpunkt des in Abbildung 5.3 dargestellten Prozessmodells ist die Definition von Anforderungen. Der Prozess der Anforderungsdefinition selbst sowie alle vorangehenden Tätigkeiten der Produktplanung werden im Rahmen dieser Arbeit nicht näher betrachtet. Detaillierte Ausführungen hierzu sind in der umfassenden Literatur zu finden, beispielsweise [Ehrl09], [PBFG07], [UlEp08] und [Ullm10].

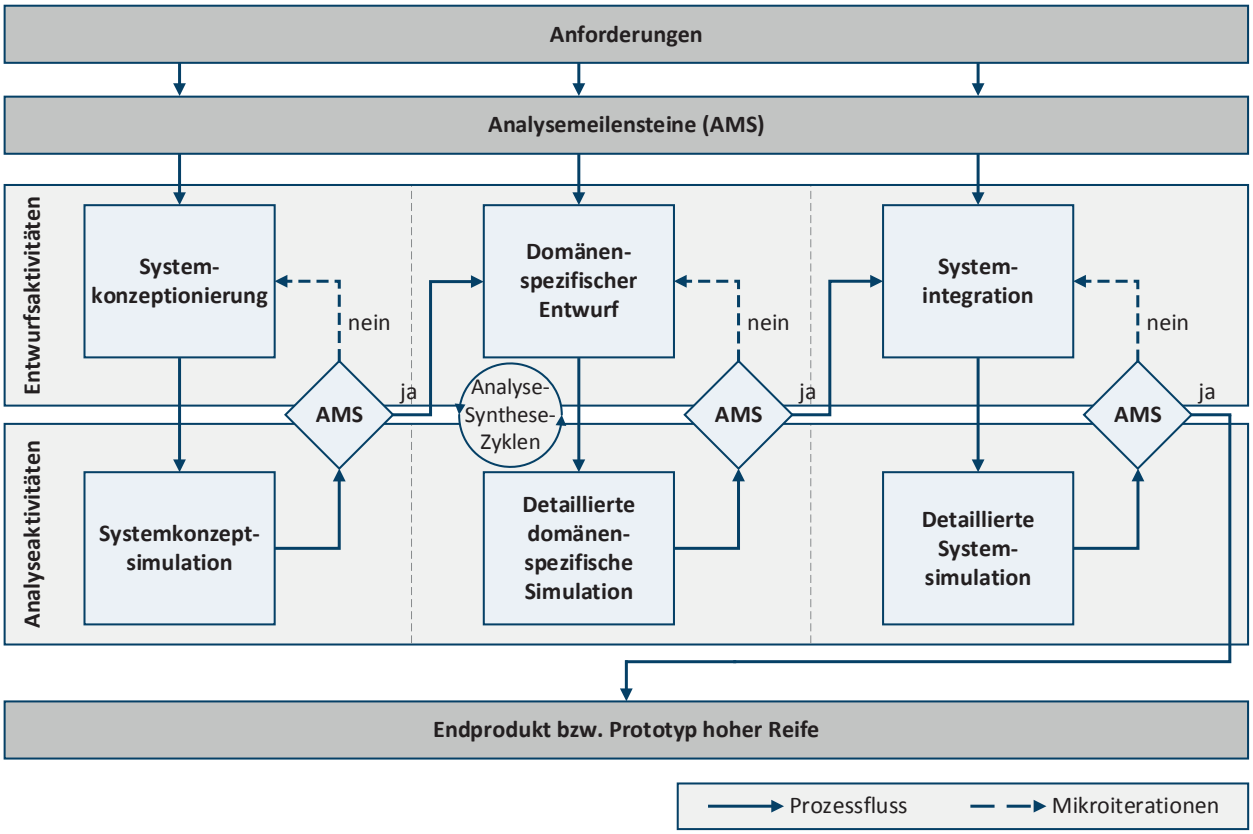


Abbildung 5.3: Vereinfachte Darstellung des Prozessmodells

Das Prozessmodell basiert auf drei grundlegenden Elementen:

- Zwei parallele Aktivitätsstränge
- Drei Hauptphasen
- Analysemeilensteine

Aus der Analyse verschiedener Entwicklungsmethodiken in Kapitel 3.2 wird deutlich, dass die Richtlinie VDI 2206 den Entwurf mechatronischer Systeme bereits sehr gut abbildet und insbesondere auch den Aspekt der Interdisziplinarität sehr gut einbindet. Weiterhin stellt sie die bisher etablierteste Entwicklungsmethodik für Mechatronik dar. Daher erfolgt die Phasenaufteilung der Entwurfsaktivitäten des hier präsentierten Prozessmodells in Anlehnung an diese VDI-Richtlinie, wodurch sich der Gesamtprozess in drei Hauptphasen aufteilt:

- Systemkonzeptionierung

  In dieser Phase werden domänenübergreifende Systemkonzepte entwickelt. Zudem erfolgt eine Zuweisung von Modul- und Komponentenverantwortlichkeiten zu den Domänen inklusive der Schnittstellendefinition.

- Domänenspezifischer Entwurf

  Auf Basis des Systemkonzeptes erfolgt die detaillierte Entwicklung und Ausgestaltung der domänenspezifischen Module und Komponenten bis hin zur Fertigungsdokumentation. Hierbei muss die Verträglichkeit mit anderen Domänen berücksichtigt werden.
- Systemintegration

  Die zuvor entwickelten Module und Komponenten werden zu einem Gesamtsystem integriert.

Entsprechend hierzu gliedern sich die Analyseaktivitäten in drei korrespondierende Phasen:

- Systemkonzeptsimulation

  Verschiedene Konzepte werden analysiert und verglichen. Zusätzlich erfolgt eine Optimierung bezüglich der Erfüllung der Produkteigenschaften durch vereinfachte domänenübergreifende Simulationen auf Systemebene.
- Detaillierte domänenspezifische Simulation

  Mittels detaillierter domänenspezifischer Simulationen erfolgt die Analyse und Optimierung spezifischer Modul- und Komponenteneigenschaften.
- Detaillierte Systemsimulation

  Auf Basis der Integration der domänenspezifischen Lösungen wird durch detaillierte domänenübergreifende Simulationen auf Systemebene das Systemverhalten analysiert. Hierbei können gegebenenfalls auch physische Prototypen oder Mischformen (beispielsweise HIL, siehe Kapitel 3.1.5.5) zum Einsatz kommen.

Prinzipiell sind diese Phasen auch adaptier- und erweiterbar, beispielsweise bei firmenspezifischen Prozessen. Im Rahmen dieser Arbeit erfolgt die Prozessaufteilung in beiden Aktivitätssträngen jedoch mittels der drei generischen Phasen.

Zudem sei an dieser Stelle darauf hingewiesen, dass die Struktur des V-Modells bewusst nicht übernommen wurde. Im ursprünglichen Sinne des V-Modells aus der Softwareentwicklung beschreibt der rechte Arm alle Analyseaktivitäten des Prozesses (siehe [Boeh79]). Dies ist nach Meinung des Autors in der VDI-Richtlinie 2206 nicht entsprechend umgesetzt, da hier im rechten Arm lediglich die Systemintegration stattfindet. Zudem lässt sich das Konzept der beiden parallelen Aktivitätsstränge nicht sinnvoll mit der V-Struktur vereinbaren. Zwar geht die visuelle Verdeutlichung des wechselnden Top-Down-Bottom-Up-Vorgehens durch den Verzicht auf die V-Darstellung verloren, dieses findet sich jedoch in den einzelnen Phasen wieder, weshalb dies hier akzeptiert wird.

Die detaillierten Entwurfsaktivitäten der einzelnen Phasen, welche in Kapitel 5.5 beschrieben werden, orientieren sich zudem an der Richtlinie VDI 2221 und dem Vorgehen nach Pahl und Beitz [PBFG07], welche auch über den deutschsprachigen Raum hinaus anerkannte Entwicklungsmethodiken darstellen.

Die beiden parallelen Stränge „Entwurfsaktivitäten“ und „Analyseaktivitäten“ bilden das zentrale Element zur Darstellung der kontinuierlichen Analyseaktivitäten, die als Anforderung definiert wurden. Ausgehend von den im oberen Strang erstellten Entwürfen werden parallel dazu bereits Simulationsmodelle erstellt und damit Analysen durchgeführt. Die gewonnenen Erkenntnisse können somit direkt wieder in Entwurfsaktivitäten zurückgeführt werden, wodurch das Konzept der Mikroiterationen ermöglicht wird. Entsprechend werden parallel zur Festlegung von Struktur und Geometrie auch Funktion und Verhalten überprüft und berücksichtigt, wie von Eigner et al. in ihrem X-Modell vorgeschlagen [EGDF14] (siehe Kapitel 3.2.5.9). Hierdurch wird eine isolierte Betrachtung dieser Sichten verhindert, was in der Mechatronikentwicklung einen wichtigen Aspekt darstellt [KüHW98], [Paet06]. Das eng verzahnte Zusammenspiel der beiden Aktivitätsstränge stellt Analyse-Synthese-Zyklen dar, welche in Abbildung 5.3 beispielhaft verdeutlicht sind. Diese Zyklen können beispielsweise durch den „FBS-Framework“ (F: Function, B: Behavior, S: Structure) von Gero in [Gero90] sowie von Gero und Kannengiesser in [GeKa04] beschrieben werden – siehe hierzu Abbildung 5.4. Demnach wird aus den Funktionen (F), welche die Produktanforderungen widerspiegeln, ein gewünschtes Verhalten (Be) abgeleitet, woraus wiederum durch Synthese eine Lösungsstruktur (S) generiert wird. Analysen dienen zur Bestimmung des tatsächlichen Verhaltens der Lösungsstruktur (Bs), welches in der Evaluation mit dem gewünschten Verhalten verglichen wird. Auch in diesem Analyse-Synthese-Zyklus ist die im Prozessmodell verankerte Zweiteilung zu erkennen, die in Abbildung 5.4 durch eine horizontale Linie verdeutlicht ist.

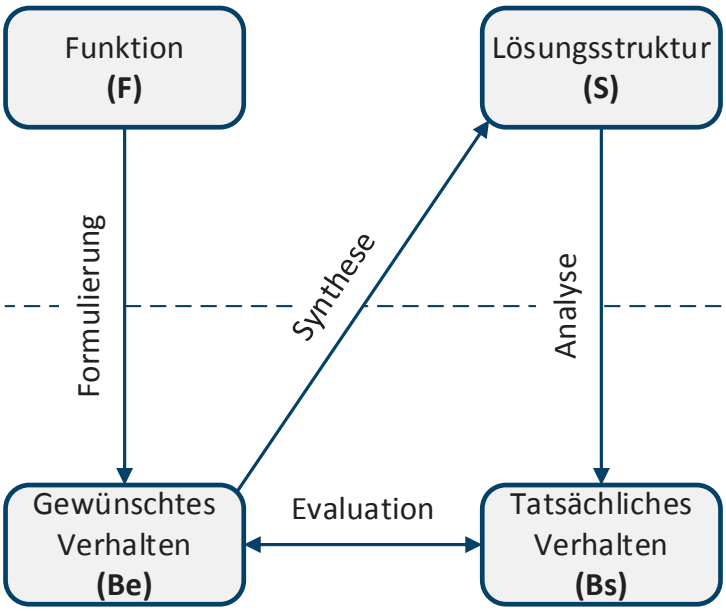


Abbildung 5.4: Beschreibung von Analyse-Synthese-Zyklen mithilfe des FBS-Frameworks, vereinfacht nach [GeKa04]

Die Steuerung der Analyse-Synthese-Zyklen, und damit der Mikroiterationen, erfolgt im Prozessmodell durch Analysemeilensteine (AMS), welche in Kapitel 5.4 im Detail erläutert werden. Diese Meilensteine regeln den Prozessablauf, ähnlich dem Vorgehen im TOTE-Schema (Test-Operate-Test-Exit), welches von Miller et al. [MiGP60] ursprünglich für die Verhaltensbeschreibung in der Psychologie entwickelt wurde und auch von Ehrlenspiel verwendet wird (siehe Kapitel 3.2.2.3). Nur wenn durch Simulationsergebnisse nachgewiesen wird, dass das tatsächliche mit dem gewünschten Verhalten übereinstimmt – zumindest in einem definierten Toleranzbereich – kann zu der nächsten Entwurfsphase übergegangen werden. Ansonsten muss in Mikroiterationen das Verhalten durch entsprechende Änderungen im Entwurf angepasst werden. Darüber hinaus dienen Analysemeilensteine auch allgemein zur Strukturierung und Organisation von Analysetätigkeiten. Im folgenden Kapitel werden das Konzept und der Aufbau von Analysemeilensteinen detailliert erläutert.

5.4 Analysemeilensteine (AMS)

Das Konzept der Analysemeilensteine nimmt eine zentrale Rolle im Prozessmodell ein und wurde in [DoVi12a] vorgestellt. Zum einen werden hierdurch das Zusammenspiel der beiden Aktivitätsstränge und damit auch Iterationen geregelt. Zum anderen dienen Analysemeilensteine zur Strukturierung der Analyseaktivitäten. Inspiriert ist dieses Konzept von dem Vorgehen in den Bereichen Elektronik- und Softwareentwicklung. Wie in den Analysen der Entwicklungsmetho-

diken aus diesen Bereichen (siehe Kapitel 3.1.4 und 3.1.5) erläutert, ist der Einsatz von Simulationen hier bereits stärker integriert. Insbesondere die testgetriebene Entwicklung stellt mit der durchgängigen Integration von Tests und der Definition von Testfällen im Vorfeld zu Entwurfstätigkeiten die Grundlage für das Konzept der Analysemeilensteine dar. Entsprechend werden diese Meilensteine direkt nach der Definition der Anforderungen aus diesen initial abgeleitet und während des Entwicklungsprozesses angepasst und erweitert. Die Struktur und Inhalte von Analysemeilensteinen sind in Abbildung 5.5 dargestellt.

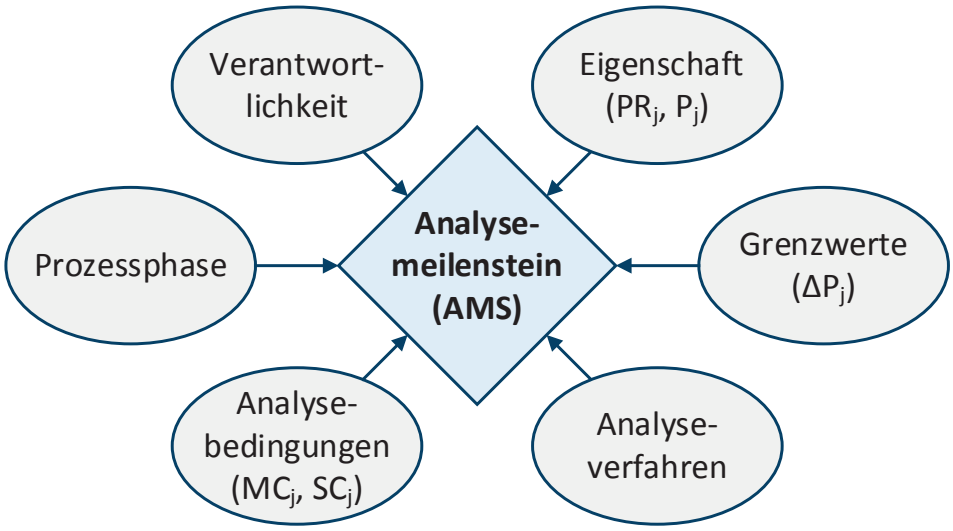


Abbildung 5.5: Struktur und Inhalt von Analysemeilensteinen

Der wichtigste Aspekt bei der Definition von Analysemeilensteinen besteht in der Festlegung, welche **Eigenschaft** des Produktes analysiert werden soll. Im Rahmen dieser Arbeit wird der Eigenschaftsbegriff gemäß der Definition von Weber et al. verwendet, die im Rahmen der CPM-Theorie (Characteristics-Properties-Modeling) [Webe05], [Webe07], [WeWD03], [WeWe00] verwendet wird. Die Begrifflichkeiten dieser Theorie haben sich während der Entwicklung der hier vorgestellten Methodik als hilfreich erwiesen, insbesondere bei der detaillierten Beschreibung von Iterationszyklen und Simulationsphasen (siehe Kapitel 5.5 und 5.6). Kernbestandteil der CPM-Theorie ist die Unterscheidung zwischen Merkmalen C_i (englisch: Characteristics) sowie Eigenschaften P_j (englisch: Properties) von Produkten. Merkmale beschreiben nach Weber die produktbestimmenden Parameter, welche von Entwicklern oder Konstrukteuren direkt beeinflusst werden können. Dagegen beschreiben Eigenschaften das Verhalten des Produktes, welches nur indirekt, über die Merkmale, beeinflusst werden kann. Nach Weber et al. [WeWD03] ist der verwendete Eigenschaftsbegriff ähnlich zu den „äußeren Eigenschaften“ nach Hubka und Eder [HuEd92] sowie den „Funktionsanforderungen“ nach Suh [Suh90]. Entsprechend korrespondieren die Merkmale mit den „inneren Eigenschaften“ von Hubka und

Eder sowie den „Gestaltungsparametern“ nach Suh. Über die reine Begrifflichkeit hinaus beschreibt Weber auf Basis der CPM-Theorie auch Modelle des Entwicklungsprozesses und darin enthaltener Tätigkeiten, beispielsweise die in Abbildung 5.6 dargestellten Prozesse der Analyse und Synthese. Da diese Modelle sich im Rahmen der vorliegenden Arbeit für die formalisierte Beschreibung von Methoden und Prozessen als hilfreich erwiesen haben, wird hier durchgängig die Begrifflichkeit nach Weber verwandt. Zudem lässt sich auch der Zusammenhang zu Geros FBS-Modell aus Abbildung 5.4 herstellen: Die Merkmale C_i entsprechen den in der Lösungsstruktur S definierten Parametern und das tatsächliche Verhalten Bs entspricht den Eigenschaften P_j. Die Funktion F sowie das gewünschte Verhalten Be entspricht Webers geforderten Eigenschaften PR_j (englisch: Required Properties) [Webe05].

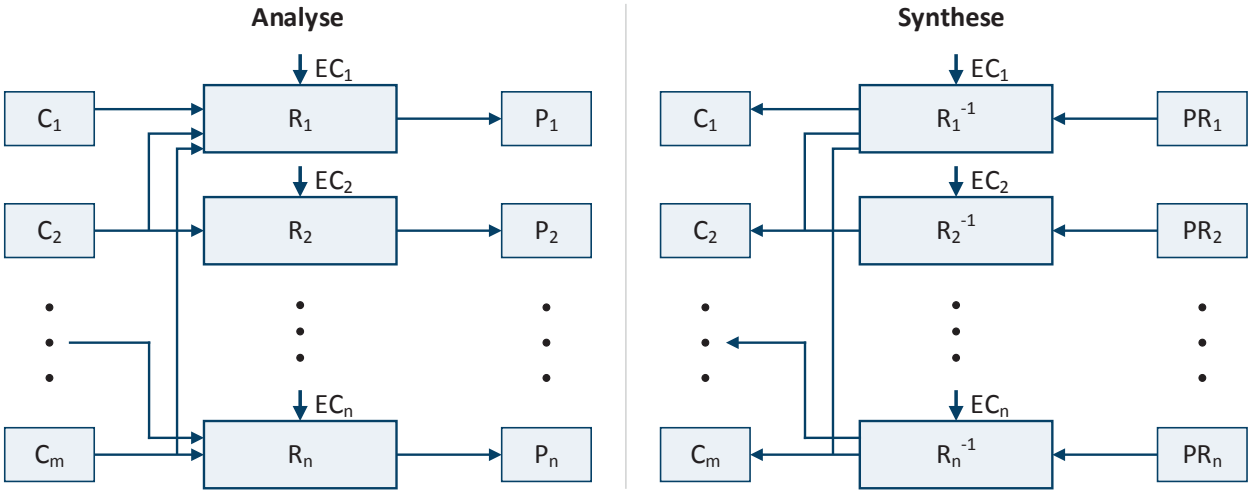


Abbildung 5.6: Beschreibung von Analyse und Synthese mittels CPM-Notation, nach [Webe05]

Bei der Festlegung der zu analysierenden Eigenschaft lassen sich in den Meilensteinen prinzipiell zwei Anwendungsfälle unterscheiden: Entweder wird durch Analysen überprüft, ob der Entwurf die gewünschten Eigenschaften erfüllt, wodurch die Eigenschaft P_j bestimmt wird. Anderenfalls dienen Analysen dazu, neue Anforderungen zu generieren oder zu präzisieren. In diesem Fall wird durch Simulationsstudien die Eigenschaft PR_j bestimmt. Die Unterscheidung beider Anwendungsfälle erfolgt darüber, ob P_j oder PR_j im Meilenstein angegeben wird. Die Definition der zu analysierenden Eigenschaft beinhaltet in der Regel neben der reinen Benennung auch die Quantifizierung des jeweiligen Eigenschaftswertes.

Simulationen sind hauptsächlich bei der Analyse quantifizierbarer Eigenschaften einsetzbar, wobei diese Quantifizierung meist einen Toleranzbereich enthält, innerhalb dessen der tatsächliche Wert liegen soll. Dieser Toleranzrahmen wird durch den **Grenzwert** ΔP_j festgelegt, wodurch der Wert der Eigenschaft innerhalb des folgenden Bereichs liegen muss:

$$P_j \pm \Delta P_j \tag{5.1}$$

Diese Eigenschaften können mit verschiedenen **Analyseverfahren** bestimmt werden. Nach Lindemann [Lind07] lassen sich diese in Schätzungen, Vergleiche, Berechnungen, numerische Simulationen sowie Versuche einteilen. Obwohl der Fokus dieser Arbeit auf dem Einsatz numerischer Simulationen liegt, können in Analysemeilensteinen auch die anderen Analyseverfahren berücksichtigt werden. Die Definition des Analyseverfahrens sollte möglichst detailliert erfolgen, beispielsweise durch Festlegung des Simulationsverfahrens und des einzusetzenden Softwarewerkzeugs. Dabei ist die Wahl des Analyseverfahrens abhängig von verschiedenen Einflussparametern, etwa finanziellen oder zeitlichen Einschränkungen, der gewünschten Genauigkeit der Ergebnisse oder dem Wissen innerhalb eines Unternehmens bezüglich der verschiedenen Analyseverfahren.

Eng an die Analyseverfahren gebunden sind **Analysebedingungen**. Modelle bilden die Realität nur in einem gewissen Toleranzbereich ausreichend ab (siehe Kapitel 2.4). So ist es möglich, stark vereinfachte Modelle mit dem Vorteil der schnellen Modellgenerierung zu erstellen. Sind jedoch genaue Ergebnisse erforderlich, kommen genaue und damit aufwendigere Modelle zum Einsatz. Insbesondere bei der Interpretation der Simulationsergebnisse ist es wichtig, diesen Aspekt zu beachten. Weber et al. [WeWD03] berücksichtigen diese Tatsache in der CPM-Theorie durch die Modellierungsbedingungen MC_j. Darüber hinaus besitzt jedoch auch das Simulationsverfahren einen Einfluss auf das Ergebnis, beispielsweise durch Wahl des Gleichungslösers (auch Solver genannt). Aus diesem Grunde werden hier zusätzlich Simulationsbedingungen SC_j eingeführt, die gemeinsam mit den Modellierungsbedingungen die Analysebedingungen bilden.

Unter **Prozessphase** wird festgelegt, welcher Phase des Prozesses der Meilenstein zuzuordnen ist, das heißt wann die Analyse durchgeführt werden soll. Diese Einteilung erfolgt auf Basis der Phasenaufteilung des Gesamtprozesses aus Abbildung 5.3. Darüber hinaus erfolgt eine Einordnung bezüglich der zeitlichen Abfolge der Meilensteine. Dargestellt wird dieses gesamte Einteilung durch eine Indizierung der Meilensteine: $AMS_{P\_n.m}$. Hierbei beschreibt der Index „P“ die Prozessphase, in welcher der Meilenstein gesetzt ist. Der Index „n.m“ stellt einen chronologischen Zählindex dar, der die sequentielle Abfolge der Meilensteine beschreibt, wobei der Index m dazu dient, parallele Meilensteine zu strukturieren.

Zu jedem Meilenstein muss zudem die **Verantwortlichkeit** festgelegt werden. Hierzu gehört die Überwachung der planmäßigen Durchführung der Analyse sowie der Verbreitung der Analy-

seergebnisse an die entsprechenden Abteilungen und Personen. Auf oberster Ebene erfolgt die Zuordnung der Verantwortlichkeiten auf Basis der Domänen, oder auf Basis der Abteilungen des Unternehmens[7]. Dies kann aber auch bis auf Personenebene verteilt werden.

Die Definition und Dokumentation von Analysemeilensteinen kann in einer dokumentenbasierten Form erfolgen. Eine Vorlage hierfür ist in Anhang A.1 beigefügt. Jedoch erschwert diese Form der Dokumentation die Definition und insbesondere die Änderung und Nachverfolgung der Meilensteine. Aus diesem Grunde wurde im Rahmen dieser Arbeit ein Softwarewerkzeug erstellt, welches diese Nachteile durch einen modellbasierten Ansatz umgeht. Die Anwendung des Programms wird in Kapitel 6 beschrieben. Zudem ist in Anhang A.2 die Benutzeroberfläche des Programms in Form von Bildschirmfotos dargestellt.

5.5 Detailbetrachtung des Gesamtprozesses

In diesem Kapitel erfolgt die detaillierte Beschreibung des in Abbildung 5.3 vereinfacht dargestellten Prozessmodells. Hierzu erfolgt in den Kapiteln 5.5.2 bis 5.5.4 eine detaillierte Beschreibung der einzelnen Phasen des Prozessmodells. In den Kapiteln 5.5.5 und 5.5.6 wird anschließend der allgemeine Ablauf von Iterationszyklen beziehungsweise von Simulationsphasen beschrieben. Vorab soll jedoch eine Betrachtung der Systemhierarchie erfolgen, da diese die Detailstruktur des Prozesses beeinflusst.

5.5.1 Systemhierarchie

Jedes System lässt sich nach Haberfellner et al. [HWFV12] hierarchisch gliedern. Die VDI-Richtlinie 2206 sieht eine derartige Dekomposition durch die Aufteilung in Systemebene und domänenspezifische Ebene vor. Diese Einteilung ist jedoch sehr grob und berücksichtigt nicht, dass mechatronische Systeme meist wiederum aus verschiedenen Subsystemen aufgebaut sind. In Abbildung 5.7 ist die hierarchische Dekomposition eines Systems in Form einer Product Breakdown Structure nach [SHAC07] dargestellt. Demnach kann jedes System in verschiedene Hierarchieebenen aufgeteilt werden, in denen Subsysteme und Komponenten anzusiedeln sind, wobei jedes Subsystem wiederum aus Subsystemen oder Komponenten bestehen kann. In der Regel enden die verschiedenen Hierarchieäste nicht auf einer Ebene. Auf unterster Hierarchieebene, in Abbildung 5.7 mit „Level s = i“ gekennzeichnet, existieren nur noch Komponenten

[7] In Unternehmen sind Abteilungen nicht zwingend strikt nach Domänen geordnet. Es kann daher sinnvoller sein, die Verantwortlichkeiten nach Abteilungen zu verteilen. Im Rahmen dieser Arbeit wird der Einfachheit halber jedoch durchgängig von Domänen gesprochen.

oder Subsysteme, die eindeutig einer Domäne zugeordnet werden können, weshalb keine weitere Dekomposition nötig ist.

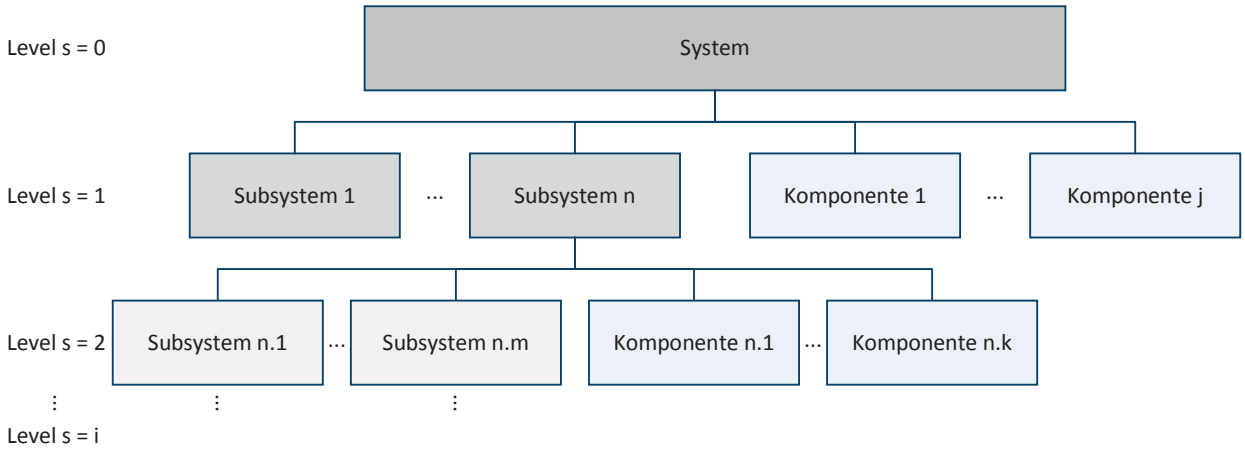


Abbildung 5.7: Systemhierarchie

Diese hierarchische Systemaufteilung wird im Prozessmodell verwendet, um auch den Prozess gemäß dieser Systemlevel zu strukturieren. Somit soll in der Systemkonzeptionierung ein stufenweiser Top-Down-Prozess von Level 0 bis Level i mit anschließender stufenweiser Bottom-Up-Integration von Level i bis Level 0 in der Systemintegration ermöglicht werden.

Diese systemhierarchische Gliederung der einzelnen Prozessphasen wird auch in der Strukturierung und Bezeichnung der Analysemeilensteine übernommen. In diesem Fall wird der Index um den Systemlevel s ergänzt: $AMS_{P\_s\_n.m}$.

Das Prozessmodell ist in detaillierter Form in Abbildung 5.8 dargestellt. In den folgenden Teilkapiteln wird jede der einzelnen Entwurfs- und Analysephasen gesondert beschrieben.

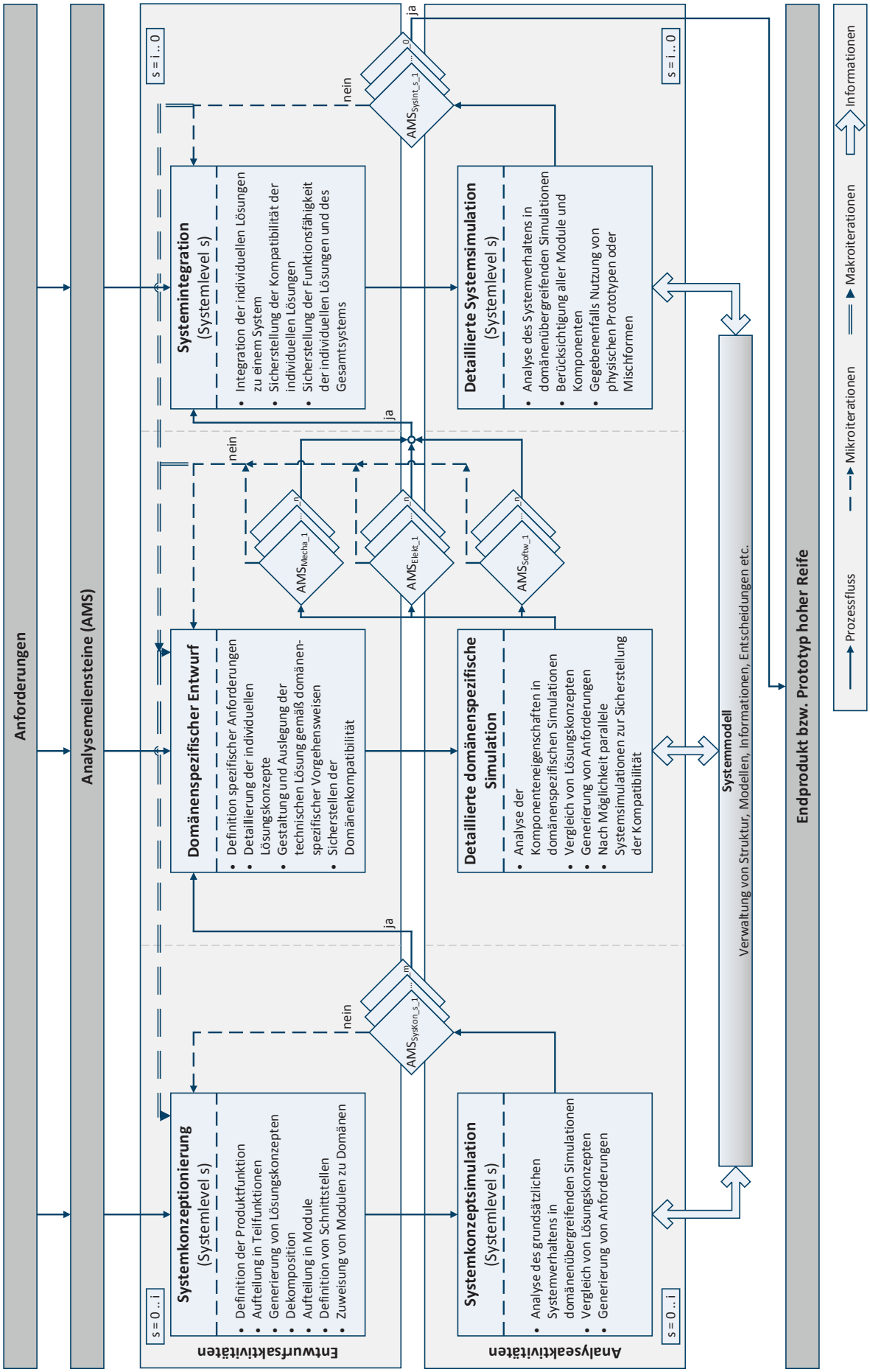


Abbildung 5.8: Detaillierte Darstellung des Prozessmodells

5.5.2 Systemkonzeptionierung und Systemkonzeptsimulation

In Abbildung 5.9 ist die Phase der Systemkonzeptionierung zusammen mit der Systemkonzeptsimulation als Ausschnitt des Gesamtprozesses gesondert dargestellt.

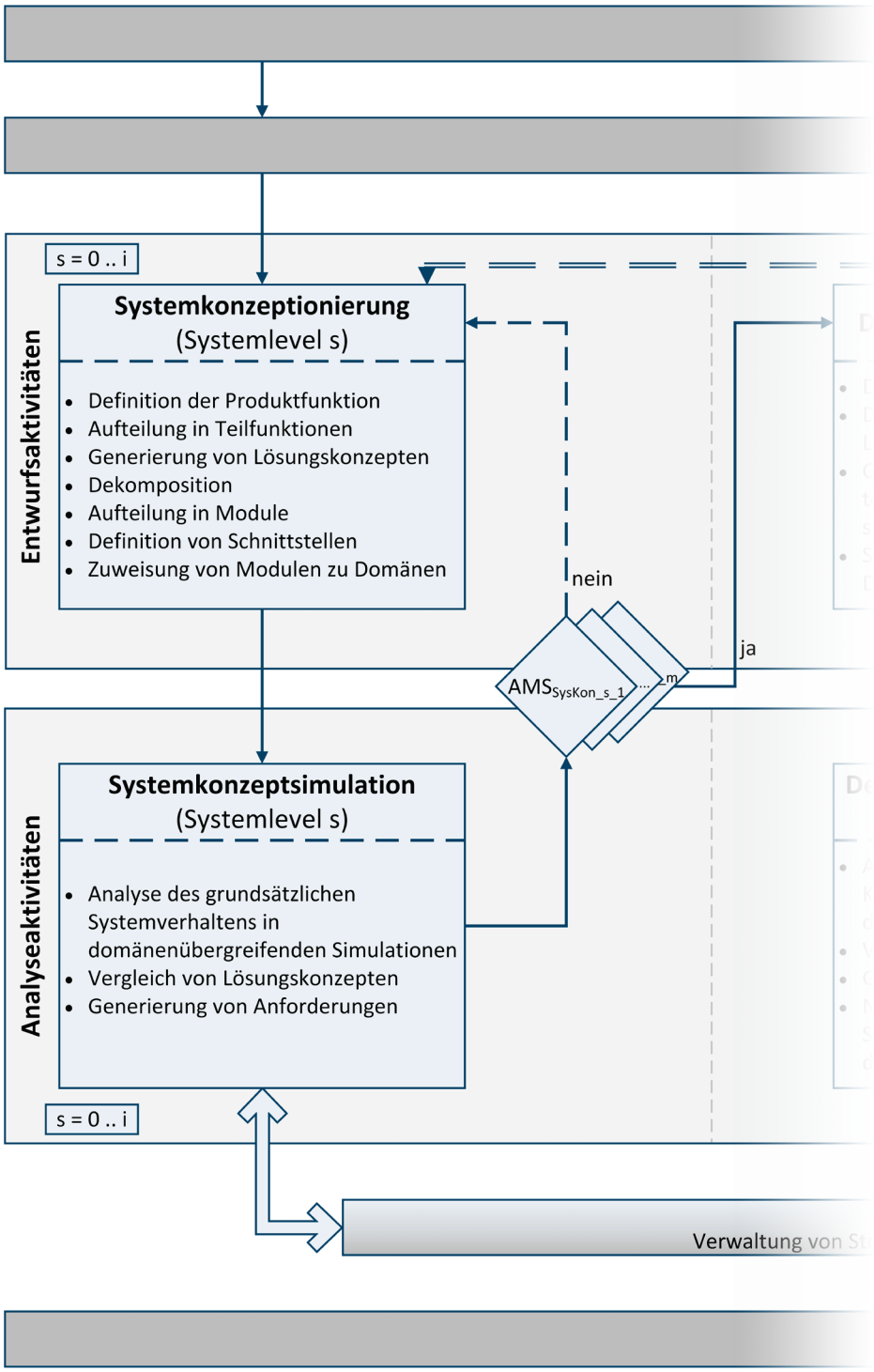


Abbildung 5.9: Detailliertes Vorgehen in der Systemkonzeptionierung und Systemkonzeptsimulation

Während der Systemkonzeptionierung wird ein Konzept für das Gesamtsystem erarbeitet und parallel dazu analysiert. Im Gegensatz zu den späteren Phasen ist der Detaillierungsgrad der

Lösungen hier geringer, da nur eine prinzipielle Lösung auf Systemebene generiert wird. Abhängig vom Neuigkeitsgrad der Entwicklung[8] variiert der Detaillierungsgrad jedoch, da in der Regel bei geringerem Neuigkeitsgrad bereits Vorwissen und Erfahrungen vorhanden sind. Bei der Lösungsgenerierung müssen Experten aller beteiligten Domänen zusammenarbeiten [Möhr04], um das synergetische Potential auszuschöpfen. Darüber hinaus ist es sinnvoll, Entwickler zu involvieren, die ein ausgeprägtes Gesamtsystemverständnis besitzen und damit zwischen den Domänenexperten vermitteln können. Die zunehmende Verbreitung des Berufsbildes des „Systemingenieurs“ adressiert diesen Bedarf.

Im Sinne des Front-Loading gewinnt die Systemkonzeptionierung einen immer höheren Stellenwert, da in diesen frühen Phasen die Produktbeeinflussung und damit die Möglichkeiten der Fehlerausbesserung besonders groß sind [VWBZ09]. Dieses Konzept wird insbesondere durch den intensiven Einsatz von Simulation gestützt, da sich hiermit in frühen Phasen die Erkennungswahrscheinlichkeit von Fehlern erhöhen lässt.

Die Aktivitäten in der Phase der Systemkonzeptionierung lehnen sich an die Richtlinie VDI 2206 [VDI2206] an. Da die Beschreibung hierin jedoch wenig detailliert erfolgt, wird das Vorgehen um die einzelnen Schritte des Konzipierens nach Pahl und Beitz ergänzt [PBFG07]. Demnach beginnt diese Phase mit der Ableitung der gewünschten Produktfunktion aus den Anforderungen. Aufgrund der Komplexität mechatronischer Systeme ist die Gliederung der Produktfunktion in Teilfunktionen sinnvoll. Pahl und Beitz verwenden hierfür eine Funktionsstruktur, die diese Teilfunktionen zudem über Stoff-, Energie- und Signalflüsse koppelt. Darüber hinaus existieren jedoch auch zahlreiche weitere Ansätze zur Funktionsmodellierung, die insbesondere im Bereich der Modellierung interdisziplinärer Systeme Vorteile besitzen. Eisenbart et al. [EiGB13] sowie Erden et al. [EKBA08] geben hierüber jeweils einen umfangreichen Überblick. Durch die Zuweisung von Wirkprinzipien zu den einzelnen Teilfunktionen können Lösungskonzepte generiert werden [PBFG07]. Gegebenenfalls sind hier weitere Detaillierungen notwendig, um die verschiedenen Konzepte vergleichen zu können. Im Allgemeinen ist der Lösungsraum bei der Entwicklung mechatronischer Produkte größer [CWPH08], [MaBu11], da Lösungsprinzipien der verschiedenen Domänen miteinander konkurrieren. Entsprechend ist ein objektiver und schneller Vergleich der Konzepte bezüglich der Anforderungserfüllung unumgänglich, was mit ver-

[8] Pahl und Beitz [PBFG07] unterscheiden hierbei beispielsweise drei Konstruktionsarten: Anpassungs-, Änderungs- und Neukonstruktion.

tretbarem Aufwand aufgrund des abstrakten und wenig detaillierten Entwurfsstatus nur durch Simulationen möglich ist [CWPH08], [MaBu11].

Entwickler benötigen in der Systemkonzeptionierung eine gemeinsame Basis zur Beschreibung der Lösungen [MöGa02]. Übertragen auf die Simulation bedeutet dies, dass die Modellierung des Systemverhaltens für alle Beteiligten verständlich erfolgen muss. In Kapitel 3.1 wurden hierzu diverse Techniken und Werkzeuge erläutert. Insbesondere Modelica-basierte Simulationswerkzeuge bieten bei der Konzeptionierung mechatronischer Systeme Vorteile. Aufgrund der graphischen Oberfläche und des Bibliothekenkonzeptes lassen sich sehr schnell und einfach systemübergreifende Simulationsmodelle erstellen. Diese repräsentieren zudem die einzelnen Systemelemente in der spezifischen Darstellungsform der Domänen, der sie entstammen. Somit ist für jeden beteiligten Entwickler die Struktur des Systems direkt ersichtlich. Darüber hinaus bieten die meisten Simulationsumgebungen auch die Möglichkeit zu domänenspezifischen Simulationen, beispielsweise vereinfachte Mehrkörpersimulationen oder Ablaufsimulationen mittels Statecharts. Ein weiterer Vorteil des Bibliothekenkonzeptes wurde vom Autor bereits in [DoVi13] und [DEHB14] beschrieben: Ähnlich dem Konzept der Konstruktionskataloge nach Roth [Roth00] lassen sich Modellbibliotheken als Speicher für Wissen und Lösungselemente nutzen. Somit können diese Bibliotheken beim Übergang von der Funktionsstruktur zu Lösungskonzepten unterstützend eingesetzt werden. Gleichzeitig wird hierbei ein Simulationsmodell erstellt, welches anschließend parametriert werden kann. Aufgrund der vordefinierten Ein- und Ausgangsports in den einzelnen Modellelementen ist zudem eine erste Kompatibilitätsprüfung der verschiedenen Lösungselemente möglich. Im Gegensatz zu späteren Phasen, in denen detaillierte Modelle und Simulationen erstellt werden, werden in der Systemkonzeptionierung vereinfachte Modelle und Simulationen benötigt, die prinzipielle Aussagen über das System zulassen [RVPK01]. Jedoch hat sich bei der Evaluation verschiedener Softwarewerkzeuge für die Analyse in Kapitel 3.1 gezeigt, dass integrierte Modellbibliotheken meist nur sehr detaillierte Modelle beinhalten, die aufgrund der Vielzahl erforderlicher Parameter für frühe Phasen ungeeignet sind. Sinnvoll ist daher die Erstellung von Bibliotheken mit vereinfachten Modellen speziell für die Systemkonzeptionierung. Da Produkte meist auf bereits bekannten und eingesetzten Komponenten aufbauen [PDSK01], ist es empfehlenswert, dass Unternehmen derartige Modellbibliotheken für ihren Anwendungsbereich selbst aufbauen und pflegen.

Auch bei der Modellierungstiefe und -detaillierung ist der Neuigkeitsgrad einer Entwicklung entscheidend. Mit entsprechendem Vorwissen und Erfahrungen aus Vorgängerprodukten las-

sen sich Modelle wesentlich einfacher parametrieren. Bei Neukonstruktionen entfällt diese Erfahrung jedoch in der Regel, weshalb Entwickler mit einer Vielzahl unscharfer Parameter konfrontiert sind. Diese Problematik wird in Kapitel 5.7 aufgegriffen, indem Methoden für den Umgang mit derartigen Unsicherheiten in die Methodik eingebunden werden.

Neben den beschriebenen Modelica-basierten Werkzeugen können je nach Struktur des zu entwickelnden Systems auch andere Simulationswerkzeuge, gegebenenfalls in Kombination, eingesetzt werden. Hierbei ist jedoch zu beachten, dass die Kopplung zwischen den Werkzeugen meist manuell über den Nutzer durch Informationstransfer erfolgt und damit aufwändiger und fehleranfälliger ist. Auch in diesem Fall müssen in der Regel vereinfachte Modelle eingesetzt werden. Der detaillierte Ablauf von Simulationsphasen wird in Kapitel 5.5.6 unabhängig von der Art des Simulationswerkzeugs weiter erläutert.

Die durch Simulationen ermittelten Eigenschaften werden mit den in den Analysemeilensteinen definierten Zielwerten abgeglichen. Werden diese nicht erreicht, muss durch Mikroiterationen der Entwurf überarbeitet werden, was im Prozessmodell durch gestrichelte Linien dargestellt ist. Diese Iterationszyklen werden in Kapitel 5.5.5 näher erläutert.

Bei der Systemkonzeptionierung werden nach Möglichkeit verschiedene Konzepte betrachtet und das beste ausgewählt. Dies erfolgt stufenweise gemäß der Systemhierarchie aus Abbildung 5.7, beginnend auf oberster Ebene (Level s = 0). Bei der Auswahl des Konzeptes dienen die Ergebnisse aus den für die Systemkonzeptionierung definierten Analysemeilensteinen – im Prozessmodell mit $AMS_{SysKon\_s\_1}$ bis $AMS_{SysKon\_s\_m}$ gekennzeichnet – als Grundlage der Bewertung. Sobald das Konzept auf der obersten Ebene der Systemhierarchie alle Meilensteine durchlaufen hat und damit das beste Konzept gewählt wurde, kann die Konzeptionierung im Sinne eines Top-Down-Prozesses auf der nächsten Hierarchiestufe fortfahren. Diese Strukturierung des Prozesses nach Systemleveln wird auch in der Indizierung der Analysemeilensteine über den Index s mitgeführt. An dieser Stelle sei darauf hingewiesen, dass nicht auf allen Hierarchieebenen Simulationen möglich und sinnvoll sind. Insbesondere auf oberster Systemebene können Simulationen schnell zu umfangreich werden, sodass diese auf untere Systemebenen verlagert oder durch andere Analyseverfahren ersetzt werden, siehe hierzu Kapitel 5.4.

Sobald die unterste Hierarchiestufe erreicht ist (Level s = i), kann der Übergang zur nächsten Phase, dem domänenspezifischen Entwurf, erfolgen. Hierfür müssen die Verantwortlichkeiten der einzelnen Subsysteme und Komponenten den individuellen Domänen zugordnet werden. Die Definition von Schnittstellen zwischen den einzelnen Subsystemen oder Komponenten er-

leichtert dabei die Einhaltung der Kompatibilität der im weiteren Verlauf entwickelten Lösungen.

Derartige Informationen bezüglich der Systemstruktur sind für den gesamten weiteren Entwicklungsprozess relevant und sollten daher allen Beteiligten zur Verfügung stehen. Aus diesem Grunde ist parallel zum Gesamtprozess in Abbildung 5.8 ein Systemmodell vorgesehen, welches als Metamodell dazu dient, sowohl vertikal – über verschiedene Domänen – als auch horizontal – über den gesamten Prozessverlauf – Informationen und Daten zu speichern und zur Verfügung zu stellen. Hierzu zählen beispielsweise die Funktionen des Systems, seine Struktur inklusive der Schnittstellen und Verantwortlichkeiten. Aber auch Modelle und Simulationsergebnisse können hier hinterlegt werden, um Modellredundanzen zu vermeiden. Somit steht Entwicklern eine zentrale Wissensbasis zur Verfügung, in der sie im Gesamtverbund ihre verantworteten Subsysteme und Komponenten sehen und so Auswirkungen auf weitere Teilsysteme einschätzen können. Neben dem Zweck der Dokumentation soll dadurch auch die interdisziplinäre Zusammenarbeit vereinfacht und gefördert werden. Ansätze zu solchen Modellen existieren bereits, insbesondere im Bereich des Modellbasierten Systems Engineerings auf Basis der Modellierungssprache SysML. Daher werden Entwicklung und Aufbau eines derartigen Modells im Rahmen dieser Arbeit nicht weiter detailliert. Weitergehende Informationen zu derartigen Ansätzen sind beispielsweise in den Veröffentlichungen von Eigner et al. [EiGZ12], Follmer et al. [FHPZ10], [FHPR11], Stetter et al. [SSCV11] sowie von Weilkiens [Weil08] zu finden.

5.5.3 Domänenspezifischer Entwurf und domänenspezifische Simulation

Sobald auf unterster Systemhierarchieebene alle Meilensteine erfüllt sind und die Zuordnung der Subsysteme und Komponenten zu den Domänen erfolgt ist, beginnt die Phase des in Abbildung 5.10 dargestellten domänenspezifischen Entwurfs. Hierbei werden die Konzepte für die Teillösungen bis zur Produktionsreife ausgearbeitet. Entsprechend dem Vorgehen aus der Richtlinie VDI 2206 wird diese Phase in die einzelnen Domänen aufgeteilt. Jede der Domänen hat spezialisierte Entwickler und spezielle, etablierte Entwicklungsmethodiken, weshalb die Entwicklung der Domänen in der Regel parallel, aber getrennt erfolgt. Dies steht jedoch im Widerspruch zum synergetischen Charakter der Mechatronik und gefährdet die Kompatibilität der Lösungen, weshalb zwar ein separates, aber kein isoliertes Vorgehen ermöglicht werden soll. Daher ist auch in dieser Phase Kommunikation und Kooperation unerlässlich, um spätere Probleme bei der Systemintegration und damit Makroiterationen zu vermeiden. Hierbei soll das bereits erwähnte Systemmodell unterstützen.

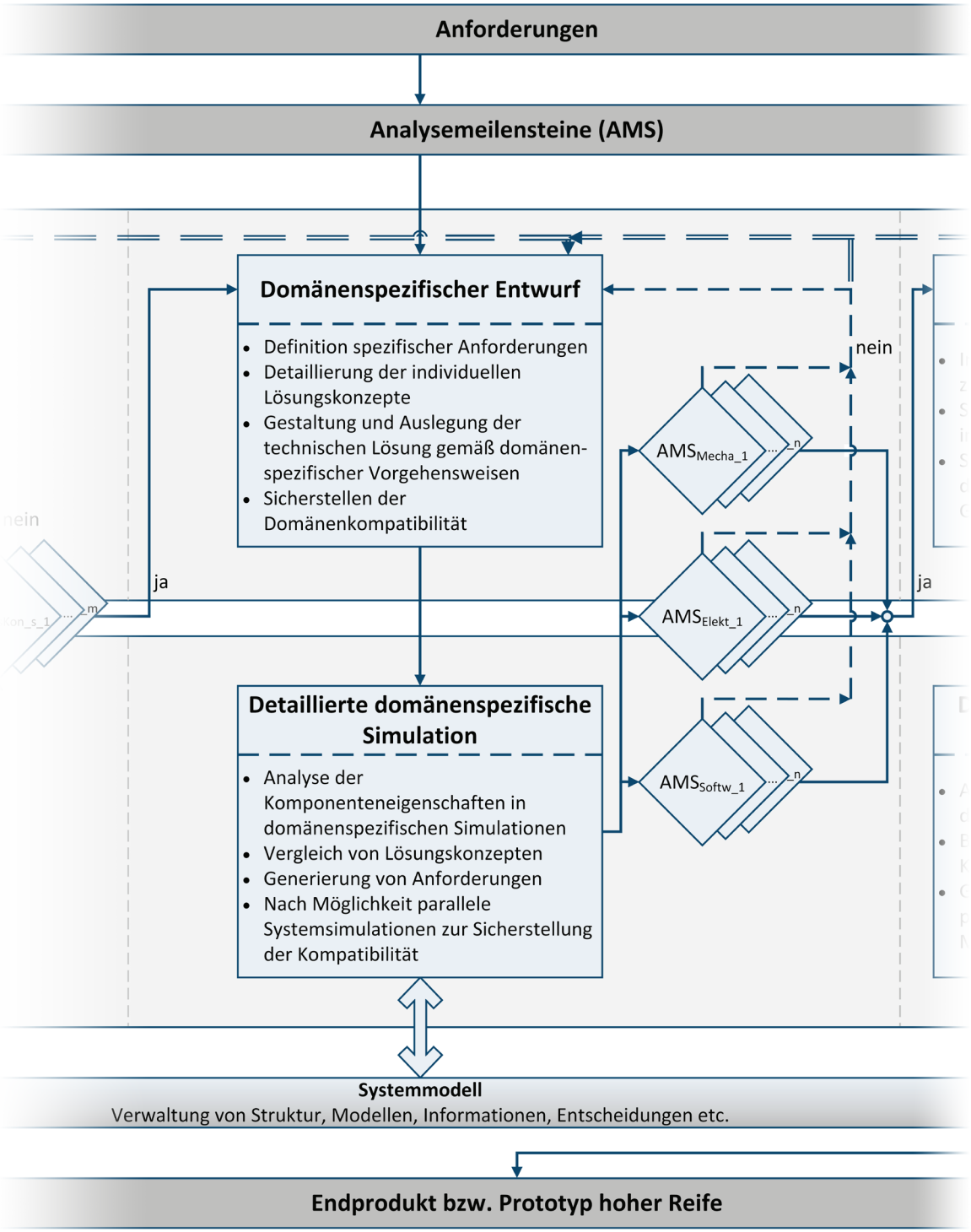


Abbildung 5.10: Detailliertes Vorgehen im domänenspezifischen Entwurf und bei der domänenspezifischen Simulation

Im ersten Schritt müssen die Anforderungen an die einzelnen Teillösungen detailliert werden. Hierbei dienen die Ergebnisse der Systemkonzeptionierung als Grundlage. Darüber hinaus können Simulationen dazu genutzt werden, Anforderungen zu generieren und zu präzisieren. Dieser Anwendungsfall kann ebenfalls durch Analysemeilensteine abgebildet werden, wie bereits

in Kapitel 5.4 beschrieben. Ebenso müssen die Analysemeilensteine detailliert und ergänzt werden.

Basierend auf diesen Anforderungen werden die individuellen Lösungskonzepte detailliert, wobei an dieser Stelle der Einsatz der individuellen Entwicklungsmethodiken beginnt. Die in dieser Arbeit vorgestellte Methodik soll hierfür ein Rahmenwerk bieten, weshalb auf eine weitere Detailbeschreibung der domänenspezifischen Vorgehensweisen verzichtet wird. Hierfür sei auf die Quellen zu den in Kapitel 3.2 vorgestellten Methodiken verwiesen. Ebenso werden in dieser Phase die domänenspezifischen Gestaltungs- und Auslegungsregeln angewandt. Unterstützt wird das gesamte Vorgehen durch die in Kapitel 3.1 vorgestellten Modellierungstechniken.

Entsprechend dem Konzept der beiden parallelen Aktivitätsstränge erfolgt auch in dieser Phase eine enge Verzahnung von Synthese und Analyse, die über Analysemeilensteine – diese enthalten hier als Indizierung die Domänenbezeichnungen – geregelt ist. Ein wesentlicher Unterschied zur Systemkonzeptionierung besteht jedoch im höheren Detaillierungsgrad der Entwicklung, wodurch bereits mehr Merkmale definiert sind. Entsprechend können die Analysen auch mit höherem Detailgrad erfolgen. Insbesondere der intensive Einsatz domänenspezifischer Simulationstechniken, wie in Kapitel 3.1 beschrieben, ist in dieser Phase möglich. Hierdurch können verschiedene Lösungskonzepte verglichen werden oder Bauteilauslegungen unterstützt werden. Speziell in Parameterstudien, bei denen der Einfluss der Variation eines oder mehrerer Parameter auf das Verhalten bestimmt wird, ist der Einsatz von Simulationen in dieser Phase aufgrund der verhältnismäßig einfachen Parametrierung sinnvoll.

Im Sinne der Sicherung der Kompatibilität der individuellen Lösungen sollen Entwickler der einzelnen Domänen ihre Lösungen über die Systemstruktur und die Schnittstellen im Systemmodell auf Verträglichkeit überprüfen. Sollten Änderungen erforderlich sein, können zudem direkt die betroffenen Bereiche und Personen mittels der im Modell hinterlegten Informationen identifiziert werden. Darüber hinaus stellt die parallele Simulation des Gesamtsystems, angepasst an den aktuellen Entwicklungsstand, ein probates Mittel zur Absicherung der Kompatibilität und des Gesamtsystemverhaltens dar, da zahlreiche verhaltensbestimmende Aspekte nur auf Systemebene zu überprüfen sind. Eine derartige Simulation erfordert jedoch zusätzlichen Aufwand und setzt zudem eine Person oder Abteilung voraus, die den Überblick über alle Domänenaktivitäten besitzt. Im Umkehrschluss bedeutet diese Simulation jedoch auch das Vorziehen von Tätigkeiten aus der Systemintegration, wodurch wiederum in dieser späten Phase der Auf-

wand reduziert werden kann. Die Entscheidung über den Einsatz paralleler Systemsimulation muss letztlich individuell abgewogen werden.

Sollten Meilensteine nicht erfüllt werden, müssen Iterationen eingeleitet werden. Im Gegensatz zur Phase der Systemkonzeptionierung bestehen im domänenspezifischen Entwurf jedoch zwei Möglichkeiten: Zum einen kann der domänenspezifische Entwurf durch Mikroiterationen angepasst und optimiert werden. Zum anderen kann es vorkommen, dass auf dieser Entwicklungsstufe festgestellt wird, dass das Systemkonzept selbst die Anforderungen nicht erfüllen kann. In einem solchen Fall muss über Makroiterationen, die im Prozessmodell ebenfalls dargestellt sind, in der Systemkonzeptionierung das Systemkonzept angepasst werden. Idealerweise tritt der zweite Fall aufgrund der fundierten Konzeptauswahl und der Absicherung durch Simulationen jedoch nicht ein. Allerdings existieren insbesondere bei Neukonstruktionen zu viele Unwägbarkeiten, um dies völlig auszuschließen.

Sobald alle Meilensteine der Domänen erfüllt sind, kann die Phase des domänenspezifischen Entwurfs beendet werden. Sollten einige Subsysteme oder Komponenten bereits vor anderen fertiggestellt sein, können diese gegebenenfalls bereits in der Systemintegration genutzt werden.

5.5.4 Systemintegration und Systemsimulation

Die in Abbildung 5.11 dargestellte Systemintegration beinhaltet nach VDI-Richtlinie 2206 die Integration der Teillösungen zu einem Gesamtsystem, um deren Zusammenwirken zu analysieren. In einem ideal simulationsbasierten Prozess entfällt diese Phase, da bereits in den vorangegangenen Phasen das System vollständig analysiert und abgesichert wurde. Jedoch ist diese Idealvorstellung kaum mit der Realität vereinbar, sei es aufgrund fehlender Simulationsmöglichkeiten, aus Kosten-/Nutzengründen oder wegen unvorhersehbarer Phänomene. Dennoch ist auch bei realen simulationsbasierten Prozessen zu erwarten, dass die Reife der Teillösungen durch die kontinuierliche Absicherung wesentlich höher ist als bei traditionellen Entwicklungsvorgehen. Somit besteht die Hauptaufgabe in der Überprüfung der Kompatibilität der einzelnen Domänenlösungen sowie in der detaillierten Analyse des Gesamtverhaltens des Systems. Hierbei wird in umgekehrter Reihenfolge zur Systemkonzeptionierung die Systemhierarchie in einem Bottom-Up-Prozess von der untersten bis zur obersten Ebene (von Level s = i bis s = 0) durchlaufen. Die Analyseaktivitäten sind wiederum durch Analysemeilensteine geregelt – bezeichnet mit $AMS_{SysInt\_s\_1}$ bis $AMS_{SysInt\_s\_0}$ – die nun auf den finalen Anforderungen basieren. Ähnlich der vorangegangenen Phasen sollten in der Systemintegration idealerweise nur Mikroi-

terationen auftreten, in denen beispielsweise kleinere Änderung der Schnittstellen oder Anpassungen von Softwareparametern durchgeführt werden. Wird in der Systemintegration jedoch erkannt, dass Fehler nur in früheren Phasen behoben werden können, müssen Makroiterationen zum domänenspezifischen Entwurf oder sogar zur Systemkonzeptionierung initiiert werden. Genau diese Makroiteration sollten jedoch durch kontinuierliche Analysen mittels Simulation verhindert werden.

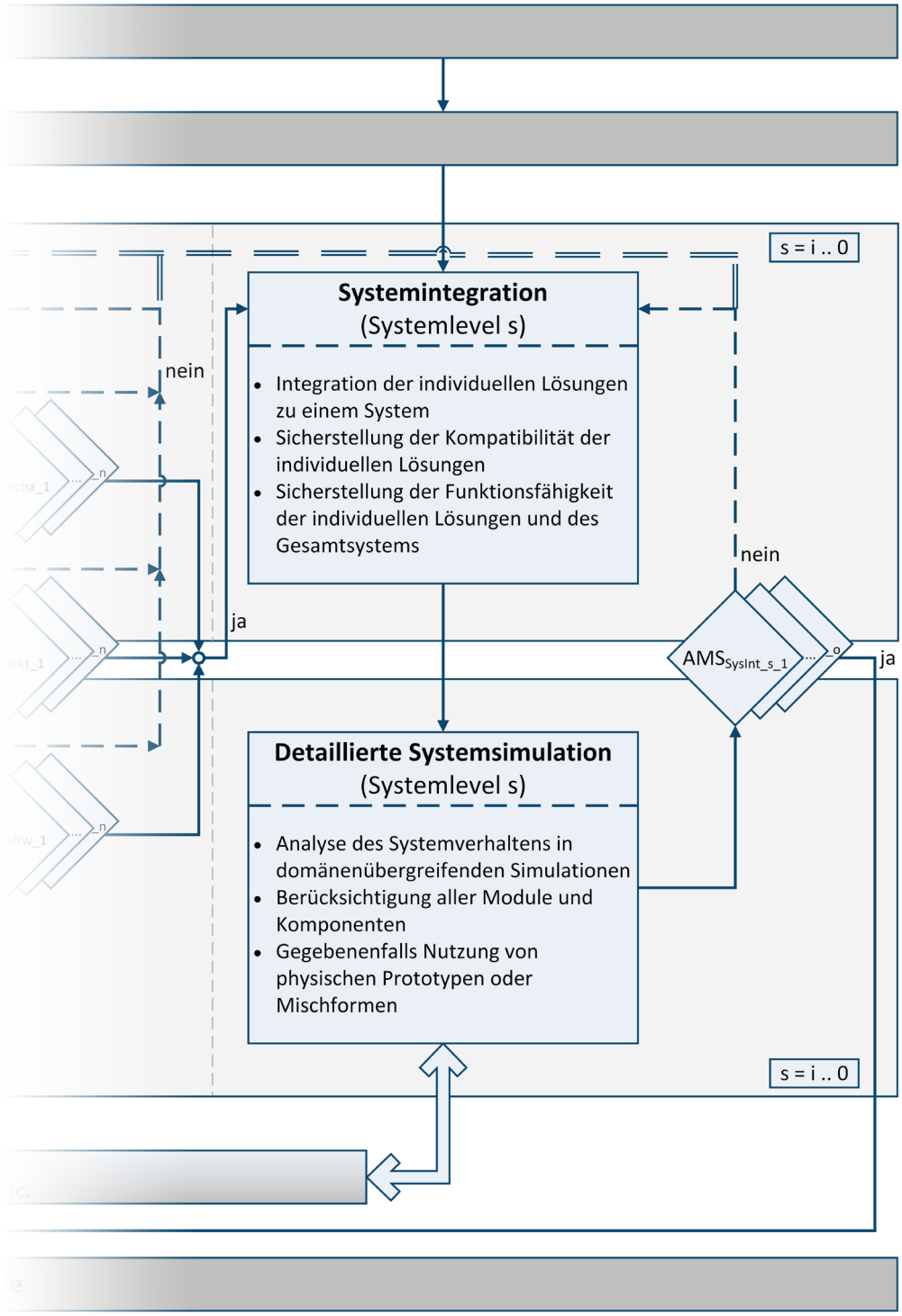


Abbildung 5.11: Detailliertes Vorgehen in der Systemintegration und Systemsimulation

Auch wenn das Ziel der virtuellen Produktentwicklung – und damit der simulationsbasierten Entwicklung – die Entwicklung ohne physische Prototypen ist [SKKR10], werden diese in realen Prozessen auch weiterhin eingesetzt werden. Die bereits angesprochene hohe Produktreife durch den intensiven Einsatz von Simulationen führt allerdings dazu, dass bereits der erste gefertigte Prototyp eine hohe Reife besitzt. Die Kombination physischer und virtueller Techniken – speziell die in Kapitel 3.1.5.5 erläuterten X-in-the-Loop-Techniken (XIL) – verbindet die Vorteile der einzelnen Ansätze. Derartige Techniken gewinnen in der Entwicklung immer größere Bedeutung, insbesondere in der Systemintegration.

Durch den Einsatz des Metamodells kann in dieser Phase zusätzlicher Aufwand eingespart werden, etwa durch die Wiederverwendung von Simulationsmodellen. So kann beispielsweise ein Mehrkörpermodell, welches im domänenspezifischen Entwurf erstellt wurde, als Umgebungsmodell für Hardware-in-the-Loop-Untersuchungen des Steuergerätes dienen.

Das Ergebnis der Systemintegration ist somit nach Erfüllung aller Analysemeilensteine idealerweise das fertigungsbereite Produkt oder aber ein Prototyp, der bereits in weiten Bereichen dem Serienstand entspricht.

5.5.5 Beschreibung von Iterationszyklen

Im Rahmen der simulationsbasierten Entwicklung kommt Iterationen und deren Auswirkungen auf den Entwicklungsprozess eine entscheidende Rolle zu. Daher wird an dieser Stelle der Iterationszyklus, basierend auf der bereits veröffentlichten Vorarbeit des Autors zu diesem Thema [DoVi14a], detailliert betrachtet. Hierzu zählt neben der Beschreibung von Kriterien, die bei der Initiierung von Iterationen wichtig sind [Paet06], auch der Iterationsprozess selbst, der Ergebnisse aus den Analyseaktivitäten zur Optimierung zurück in den Entwurf führt. Diese Betrachtung erfolgt unter der Verwendung der CPM-Notation von Weber, welche eine formalisierte Beschreibung der Teilprozesse ermöglicht. In Abbildung 5.12 ist der detaillierte Ablauf eines Iterationszyklus zwischen Entwurfs- und Analyseaktivitäten dargestellt. Hierfür werden generische Entwurfs- und Simulationsphasen als Ausschnitt des Gesamtprozesses aus Abbildung 5.3 herangezogen. Diese stehen stellvertretend für die zuvor beschriebenen Phasen, in denen Iterationen in dieser Form auftreten.

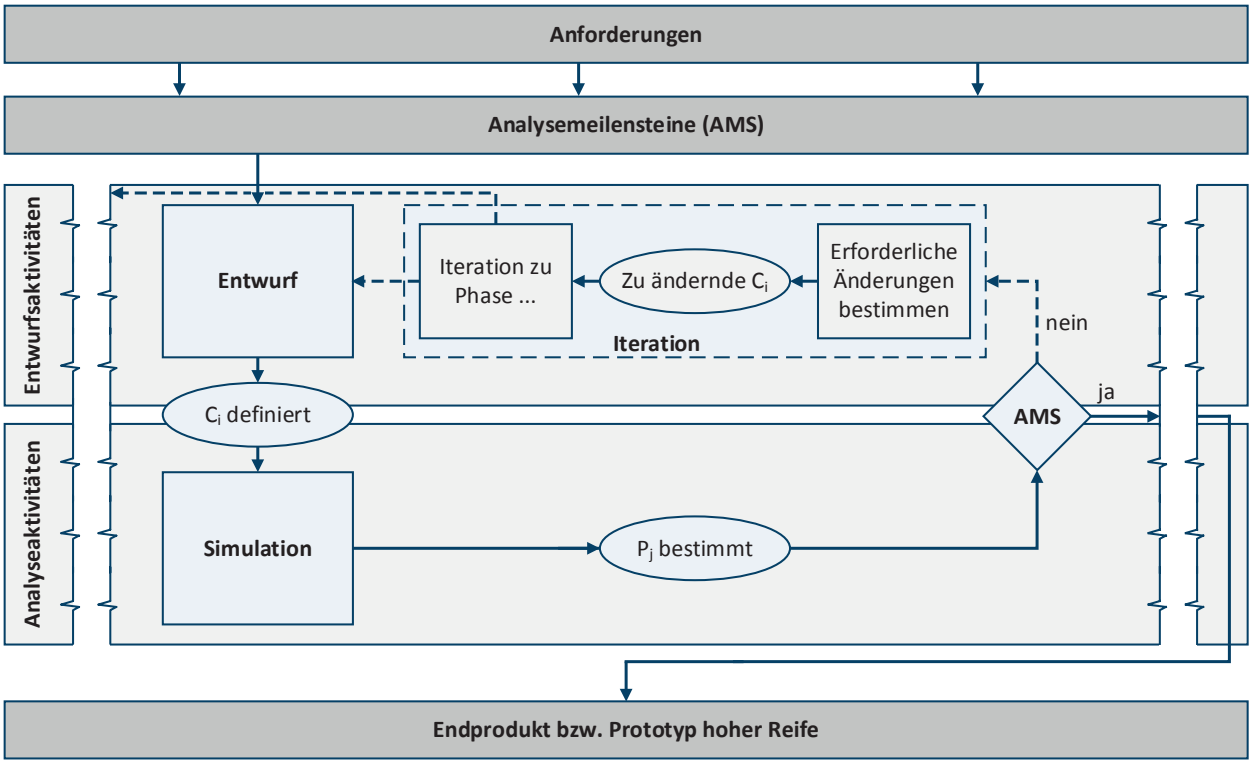


Abbildung 5.12: Darstellung eines Iterationszyklus als Ausschnitt des Gesamtprozesses

In der Entwurfsphase wird ein – mehr oder weniger detaillierter – Entwurf erarbeitet, der durch einen Satz definierter Merkmale C_i charakterisiert ist. Im Sinne der kontinuierlichen Analyse erfolgt im Anschluss an die Definition dieser Merkmale der Wechsel zu den Analyseaktivitäten, um zu überprüfen, ob der Entwurf die in der Anforderungsliste definierten Eigenschaften PR_j erfüllt. Obwohl die Analyse mit unterschiedlichen Verfahren durchgeführt werden kann, werden hier aufgrund des Fokus der Arbeit nur Simulationen betrachtet. Der prinzipielle Ablauf ist jedoch bei allen Verfahren identisch. In der Simulationsphase – diese wird in Kapitel 5.5.6 ausführlich beschrieben – werden durch Modellierung der Zusammenhang R_j zwischen Merkmalen und Eigenschaften dargestellt [WaCK07] (siehe Abbildung 5.6) und durch Simulation des Modells die Eigenschaften P_j des Entwurfs bestimmt. Zu diesem Zweck werden die entsprechenden Informationen bezüglich Analysezweck, -verfahren und -bedingungen aus den zugehörigen Analysemeilensteinen entnommen. Nachdem die Eigenschaften des Entwurfs bestimmt sind, erfolgt mittels der in dem zugehörigen Analysemeilenstein festgelegten Sollwerte die Entscheidung, ob der Entwurf den Anforderungen genügt. Damit kann ein objektives Kriterium definiert werden:

$$\left(PR_j - \Delta P_j\right) \leq P_j \leq \left(PR_j + \Delta P_j\right) \tag{5.2}$$

Wenn Gleichung (5.2) erfüllt ist, kann zu der nächsten Entwurfsphase übergegangen werden. Anderenfalls muss eine Iteration initiiert werden.

Dieses Kriterium stellt jedoch eine Vereinfachung der Entscheidung dar, da die Analysebedingungen nicht eingehen und somit die Zuverlässigkeit und Genauigkeit des Modells sowie der Simulationstechnik oder des Simulationswerkzeug nicht berücksichtigt werden. Für die Berücksichtigung dieser Aspekte führt Weber einen Konfidenzfaktor ε_j ein [Webe07], der die eingesetzten Methoden und Werkzeuge bezüglich ihrer Zuverlässigkeit und Genauigkeit bewertet. Demnach erhalten Methoden, die in frühen Phasen eingesetzt werden, einen niedrigen Wert und Methoden aus späten Phasen einen hohen Wert. Unter Berücksichtigung der in Kapitel 5.4 eingeführten Modellierungs- und Simulationsbedingungen kann dieser Konfidenzfaktor durch folgende Gleichung ausgedrückt werden:

$$\varepsilon_j = k \cdot MC_j \cdot SC_j \tag{5.3}$$

Der Faktor k in Gleichung (5.3) dient als Kalibrierungskonstante des Konfidenzfaktors, damit dieser direkt bei der Eigenschaftsberechnung übernommen werden kann und gleichzeitig auf den Maximalwert von 1 normiert wird. Modellierungs- und Simulationsbedingungen lassen sich prinzipiell auf zwei Arten ausdrücken: Existieren quantifizierbare Werte bezüglich der Genauigkeit und Verlässlichkeit der Modelle und Simulationstechniken – etwa basierend auf Erfahrungen oder Studien – so können diese Werte genutzt werden, um MC_j und SC_j objektiv zu quantifizieren. Existieren diese quantifizierten Werte nicht, so können MC_j und SC_j qualitativ, auf Basis von Entwicklerwissen und -erfahrung, bewertet werden. In Tabelle 5.1 ist dies beispielhaft für eine fiktive Bewertung aufgeführt: Auf einer Skala von sehr ungenau (1, stark vereinfachte Modelle) bis sehr genau (4, physische Tests) werden die Analysebedingungen bewertet. Hierbei ist anzumerken, dass Modellierungs- und Simulationsbedingungen nicht zwingend, wie in Tabelle 5.1 dargestellt, identische Bewertungen erhalten müssen.

Tabelle 5.1: Konfidenzfaktor ε_j, basierend auf qualitativer Bewertung für MC_j und SC_j

	Stark vereinfacht / sehr ungenau	Vereinfacht / ungenau	Kaum vereinfacht / genau	Nicht vereinfacht / sehr genau
MC_j	1	2	3	4
SC_j	1	2	3	4
ε_j	1·k	2·k	3·k	4·k

Anschaulich gesehen definiert der Konfidenzfaktor damit, wie in Abbildung 5.13 dargestellt, einen Vertrauensbereich um den simulierten Eigenschaftswert, innerhalb dessen der tatsächliche Eigenschaftswerts des realen Produkts zu finden sein wird.

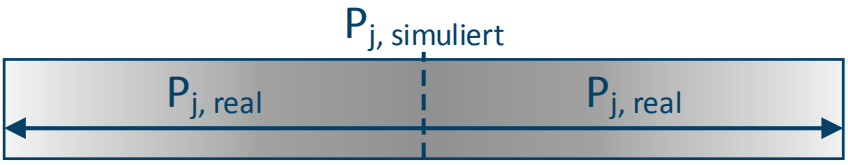


Abbildung 5.13: Vertrauensbereich des simulierten Eigenschaftswertes

Der Konfidenzfaktor verhält sich damit gemäß Gleichung (5.4) umgekehrt proportional zur Standardabweichung des Simulationsergebnis: Je kleiner der Konfidenzfaktor – das bedeutet je ungenauer die Modellierungs- und Simulationstechniken – desto größer die Standardabweichung. Umgekehrt bedeutet dies, dass bei hohen Konfidenzfaktoren – also bei sehr genauen Techniken – die Standardabweichung gering ist und somit der Bereich um die simulierte Eigenschaft, innerhalb dessen der reale Eigenschaftswert liegen wird, klein ist.

$$\varepsilon_j \sim \frac{1}{\sigma} \tag{5.4}$$

Mittels dieses Konfidenzfaktors kann die Iterationsbedingung aus Gleichung (5.2) entsprechend erweitert werden:

$$\frac{1}{\varepsilon_j}\left(PR_j - \Delta P_j\right) \leq P_j \leq \left(PR_j + \Delta P_j\right)\frac{1}{\varepsilon_j} \tag{5.5}$$

Entsprechend vergrößert sich in frühen Phasen der Bereich, in dem die simulierte Eigenschaft liegen muss. Allerdings verringern sich damit die Zuverlässigkeit der Simulationsergebnisse und somit auch deren Aussagefähigkeit.

Gemäß der Darstellung in Abbildung 5.12 muss in dem Fall, dass das Kriterium aus Gleichung (5.5) beziehungsweise (5.4) nicht erfüllt ist, eine Iteration eingeleitet werden. Im ersten Schritt

müssen, basierend auf dem Unterschied zwischen der simulierten Eigenschaft P_j und der geforderten Eigenschaft PR_j, die erforderlichen Änderungen vom Entwickler identifiziert werden. Dies kann, über die Erfahrung des Entwicklers hinaus, unter Zuhilfenahme des Simulationsmodells erfolgen, welches den Zusammenhang zwischen Merkmalen und Eigenschaften ausdrückt. Entsprechend können hiermit die relevanten Merkmale identifiziert werden, wobei gegebenenfalls auch Parameterstudien mit dem Simulationsmodell unterstützen können.

Nach der Identifikation der zu ändernden Merkmale ist die Bestimmung der Phase erforderlich, zu der die Iteration zurückführt. Zu diesem Zweck wird die Definition nach Weber [WeWD03] genutzt, wonach Phasen durch die Anzahl der definierten Merkmale bestimmt sind. Unter der Voraussetzung, dass die Merkmale nach dem Zeitpunkt ihrer Definition chronologisch nummeriert werden, können somit die Phasen eindeutig bestimmt werden:

- $C_1 \ldots C_m$: Systemkonzeptionierung
- $C_{m+1} \ldots C_n$: Domänenspezifischer Entwurf
- $C_{n+1} \ldots C_o$: Systemintegration

Somit kann bei Kenntnis der zu ändernden Merkmale direkt bestimmt werden, in welche Phase die Iteration zurückführen muss. Gleichzeitig kann auch die verantwortliche Person oder Abteilung identifiziert werden, vorausgesetzt, die Verantwortung der Merkmale wurde dokumentiert, etwa im Systemmodell. Ein weiterer Vorteil dieser Zuordnung ist, dass die erläuterten Kriterien sowohl für Mikro- als auch für Makroiterationen gelten. Bei Mikroiterationen entstammen die zu ändernden Merkmale der aktuellen Phase, bei Makroiterationen vorangegangenen.

5.5.6 Beschreibung von Simulationsphasen

Im Konzept der kontinuierlichen Analysen spielen Simulationsphasen eine entscheidende Rolle, da hier der Zusammenhang zwischen den Merkmalen C_i aus den Entwurfsaktivitäten und den Eigenschaften P_j hergestellt wird, die wiederum, wie zuvor beschrieben, als Bewertungskriterium dienen. Entsprechend sollen die Simulationsphasen hier auf Basis von Abbildung 5.14 erläutert werden. Hierfür wird eine generische Simulationsphasen als Ausschnitt des Gesamtprozesses aus Abbildung 5.3 betrachtet.

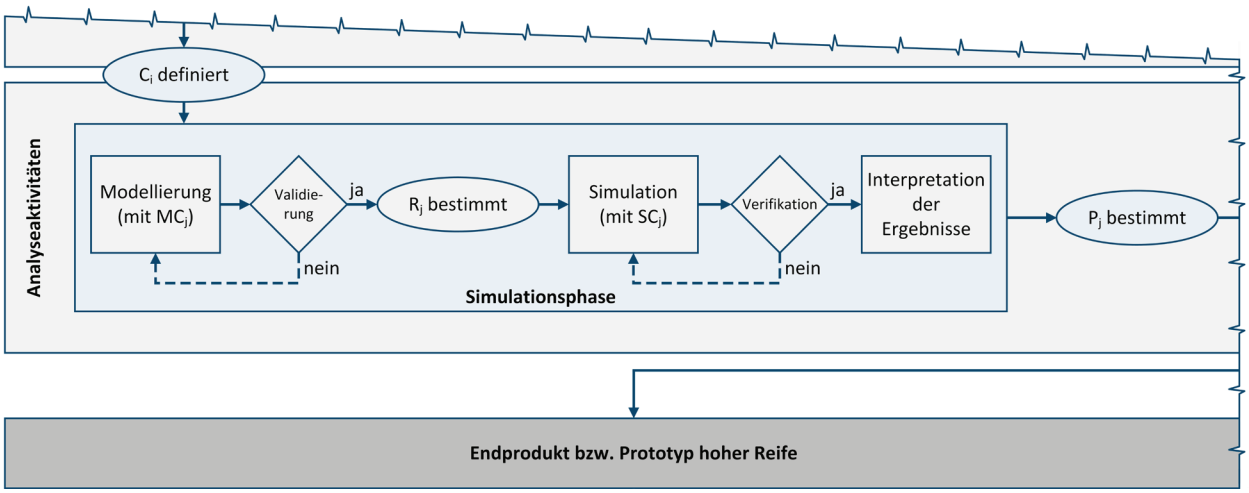


Abbildung 5.14: Darstellung einer Simulationsphase als Ausschnitt des Gesamtprozesses

Basierend auf den definierten Merkmalen wird das Verhalten des Systems oder der Komponente modelliert. Dies bedeutet, dass unter Berücksichtigung der Analysebedingungen, des Analyseverfahrens und der zu untersuchenden Eigenschaft, welche in dem zugehörigen Analysemeilenstein definiert sind, ein Modell erstellt und parametriert wird und somit die Beziehungen R_j zwischen Merkmalen und Eigenschaften bestimmt werden. Dieses Modell muss anschließend validiert werden, wobei Validierung nach [IEEE97] wie folgt definiert ist:

> *The process of determining the degree to which a distributed simulation is an accurate representation of the real world from the perspective of the intended use(s) as defined by the requirements.*

Nach Janschek [Jans10] sowie nach VDI-Richtlinie 2206 [VDI2206] bedeutet dies vereinfacht, ob unter Berücksichtigung des Einsatzzwecks das richtige Modell erstellt wurde.

Ist das Modell nicht valide, muss eine Überarbeitung erfolgen. Im Anschluss kann die Simulation erfolgen. Hierfür muss das Modell in einer entsprechenden Umgebung implementiert werden. Der Prozess der Modellierung und Simulation kann hierbei entweder vollständig manuell in einer allgemeinen Programmiersprache erfolgen – beispielsweise bei der Erstellung eines mathematischen Modells – oder kann durch eine graphische Oberfläche eines Softwarewerkzeugs zu großen Teilen automatisiert und damit vereinfacht werden – bespielweise bei der Nutzung von Modellbibliotheken in Modelica-basierten Werkzeugen. Basierend auf der Simulation muss das erstellte Modell anschließend verifiziert werden, was nach [IEEE97] wie folgt definiert ist:

> *The process of determining that an implementation of a distributed simulation accurately represents the developer's conceptual description and specifications.*

Vereinfacht ausgedrückt bedeutet diese Definition, wiederum nach Janschek sowie VDI-Richtlinie 2206, ob das Modell richtig implementiert wurde.

Janschek [Jans10] weist darauf hin, dass die Definitionen von Validierung und Verifikation oftmals in verschiedenen Fachbereichen unterschiedlich belegt sind, was sich auch in den Erfahrungen des Autors widerspiegelt. Entsprechend werden in dieser Arbeit obige Definitionen verwendet, die insbesondere im Bereich der Mechatronik sowie der Modellierung und Simulation verbreitet sind, siehe beispielsweise [Sarg99] und [Wall07].

Techniken und Methoden zur Validierung und Verifikation von Simulationsmodellen stellen in sich ein umfangreiches Forschungsgebiet dar, weshalb deren Diskussion im Rahmen dieser Arbeit nicht in der Tiefe erfolgen kann. Daher wird an dieser Stelle auf die Arbeiten von Pelz [Pelz03] und Sargent [Sarg99] verwiesen, die eine umfassende Zusammenfassung des Themas geben.

Der letzte – und oftmals wichtigste Schritt – in der Simulationsphase ist die Interpretation der Simulationsergebnisse durch den Entwickler. Dieser muss unter Berücksichtigung des gesamten Modellierungs- und Simulationsprozesses aus den Ergebnissen die relevanten Informationen extrahieren. Hierdurch wird die Eigenschaft P_j des Entwurfs schlussendlich bestimmt, welche dann wiederum in den zuvor beschriebenen Iterationszyklen verwendet wird.

5.6 Auswahl von Analyseverfahren

In den vorangegangenen Teilkapiteln wurde ein detaillierter Prozess beschrieben, der die simulationsbasierte Entwicklung mechatronischer Systeme ermöglicht. Das Konzept der Analysemeilensteine dient dabei zur Strukturierung und Organisation aller Analyseaktivitäten. Die Auswahl eines geeigneten Simulationswerkzeugs wurde hierbei jedoch nicht erläutert. Wie die in Kapitel 3.1 erläuterten Modellierungs- und Simulationstechniken zeigen, ist die Auswahl an Werkzeugen und Techniken äußerst umfangreich, wobei die hochspezialisierten, anwendungsfallspezifischen Werkzeuge noch nicht berücksichtigt sind. Diese Werkzeuge sowie deren Möglichkeiten sind von Entwicklern kaum zu überblicken, weshalb in diesem Kapitel eine Methode beschrieben wird, die bei der Auswahl geeigneter Analyseverfahren und -werkzeuge unterstützen kann.

Prinzipiell bestehen, wie in Kapitel 5.4 beschrieben, diverse Analysemöglichkeiten, von Schätzungen über Simulationen bis hin zu physischen Tests, wobei jede ihre Vorteile und Berechtigung besitzt. In dieser Arbeit wird die simulationsbasierte Entwicklung adressiert, weshalb im Folgenden auch nur Simulationstechniken betrachtet werden. Grundsätzlich ist die in diesem Kapitel beschriebene Methode jedoch auch auf andere Analyseverfahren anwendbar. Sie verwendet die CPM-Notation nach Weber [WeWD03], die bereits in Kapitel 5.4 erläutert wurde.

Zunächst soll jedoch die Klassifizierung von Eigenschaften bezüglich deren Analysierbarkeit – und insbesondere Simulierbarkeit – diskutiert werden, die als Vorauswahl für die eigentliche Auswahl von Analyseverfahren dient.

5.6.1 Einteilung von zu analysierenden Eigenschaften

Eigenschaften lassen sich generell gemäß Abbildung 5.15 nach ihrer Simulierbarkeit einteilen.

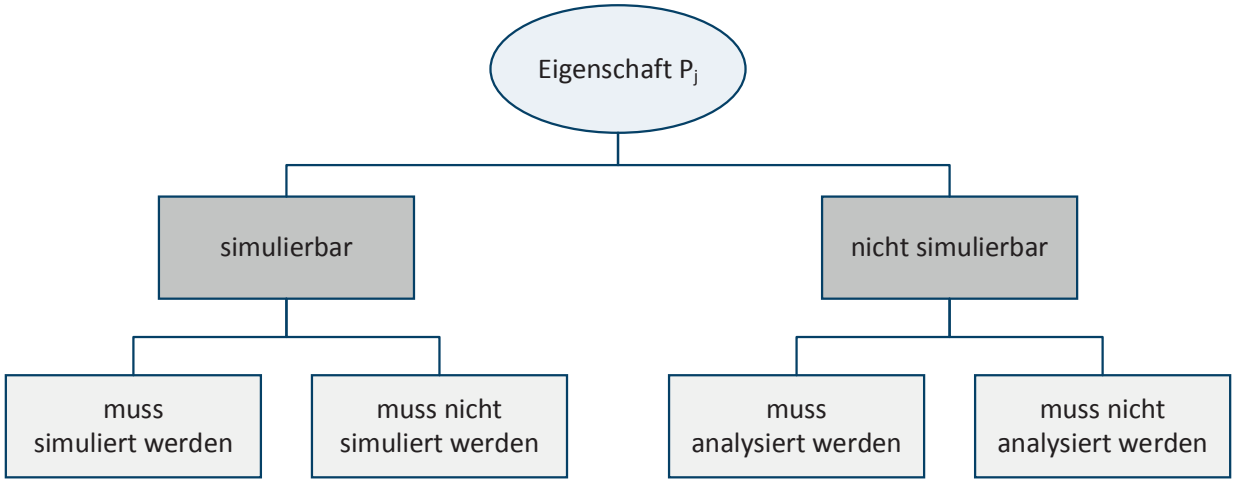


Abbildung 5.15: Einteilung von Eigenschaften nach ihrer Simulierbarkeit

Trotz der Vielzahl an ausgereiften Simulationstechniken existieren noch immer Eigenschaften, die nicht oder nur mit unzureichender Genauigkeit und Verlässlichkeit simuliert werden können. Diese werden durch den rechten Zweig in Abbildung 5.15 repräsentiert. Hierzu zählen beispielsweise Verhalten sowie Empfinden von Menschen[9]. Diese nicht simulierbaren Eigenschaften lassen sich nach ihrer Beschaffenheit und Wichtigkeit nochmals unterteilen: Eigenschaften, die direkt ersichtlich sind, müssen nicht mehr weiter analysiert werden. Hierzu zählt beispielsweise die Farbe eines Produktes, die mit Festlegung der Merkmale in der Regel bereits bestimmt ist. Die zweite Kategorie umfasst Eigenschaften, die nicht direkt bestimmt werden kön-

[9] Zwar liefern beispielsweise Bernard et al. [BGHX07] sowie Krüger et al. [KrMW12] hierzu erste Ansätze, diese sind jedoch noch nicht ausgereift und etabliert.

nen, wie etwa Komforteigenschaften. Zu deren Analyse werden beispielsweise Untersuchungen an physischen Prototypen notwendig.

Der linke Zweig in Abbildung 5.15 stellt alle Eigenschaften dar, die mit existierenden Simulationstechniken analysiert werden können. Auch wenn der Stand der Technik, wie bereits gezeigt, einen sehr umfangreichen Einsatz von Simulationen erlaubt, so ist dies nicht immer sinnvoll. Jede Simulation erfordert Zeit- und Ressourcenaufwendungen, weshalb eine Abwägung erfolgen sollte, ob Simulationen erforderlich sind oder nicht. So ist in der Regel bei der Auslegung des mechanischen Grundsystems eines mechatronisches Systems die strukturmechanische Analyse mittels FEM-Simulation unerlässlich. Dagegen existieren beispielsweise für Zukaufteile oder -systeme bereits Information über deren Eigenschaften, die durch Analysen des Zulieferers bestimmt wurden. Eine erneute Analyse ist hier oftmals nicht sinnvoll.

Entsprechend dieser Einteilung sollte die Eigenschaften eines Produktes bereits bei der Definition von Analysemeilensteinen klassifiziert werden, um die Analysen möglichst sinnvoll und effizient zu strukturieren.

5.6.2 Methode zur Auswahl von Analyseverfahren

Die im Folgenden beschriebene Methode ist insbesondere für jene Eigenschaften geeignet, die simulierbar sind und auch simuliert werden müssen. Das Konzept ist jedoch auch auf andere Analyseverfahren übertragbar. Die Methode soll Entwickler dabei unterstützen, für einen speziellen Anwendungsfall – definiert durch eine zu analysierende Eigenschaft, die Prozessphase sowie die geforderte Genauigkeit – ein geeignetes Simulationsverfahren zu finden.

Die Methode basiert auf zwei Matrizen, ähnlich der Konzepte der Design Structure Matrix (DSM) und Domain Mapping Matrix (DMM), siehe hierzu beispielsweise [LiMB09]. Grundlage stellt die im Analysemeilenstein festgelegte Eigenschaft P_j beziehungsweise PR_j dar, die analysiert werden soll. Die erste Matrix, beispielhaft dargestellt in Tabelle 5.2, ordnet zu analysierenden Eigenschaften geeignete Simulationstechniken oder -werkzeuge (ST_k) zu. Diese Zuordnung kann generell erfolgen und stellt somit einen Katalog an Simulationsverfahren zur Verfügung. Hierbei ist darauf zu achten, dass die Benennung der Eigenschaften einen ausreichend Abstraktionsgrad besitzt, um eine allgemeingültige Zuordnung zu ermöglichen. Die Erstellung dieser Zuordnung erfordert die umfassende Analyse von Simulationstechniken bezüglich ihrer Analysemöglichkeiten sowie idealerweise Erfahrungen mit diesen Techniken. In Anbetracht der Vielzahl an Simulationstechniken ist die Erstellung der Matrix daher nur für spezifische Anwen-

dungsfälle oder Produktkategorien sinnvoll. Alternativ kann die Erstellung auch abteilungs- oder unternehmensintern erfolgen, um den Erstellungs- und Pflegeaufwand zu reduzieren sowie die Übersichtlichkeit zu gewährleisten. Darüber hinaus kann eine Klassifizierung von Eigenschaften und Simulationstechniken die Anwendbarkeit fördern. Dies wird in Kapitel 5.6.3 weiter erläutert.

Tabelle 5.2: Matrix 1 – Zuordnung von Simulationstechniken zu Eigenschaften (beispielhaft)

Eigenschaft P_j	Simulationstechnik ST_k				
	1	2	3	…	m
1	X				X
2			X		
3		X			X
…					
n		X			

Die in Tabelle 5.3 beispielhaft dargestellte zweite Matrix dient Entwicklern dazu, die für die ausgewählte Simulationstechnik erforderlichen Merkmale C_i zu bestimmen. Somit lässt sich auf Basis der bereits definierten Merkmale feststellen, ob die Simulationstechnik anwendbar ist oder welche Informationen zur Anwendung noch fehlen. Die Erstellung dieser Matrix sollte parallel zur ersten Matrix erfolgen. Hierbei können neben eigenen Erfahrungen und Analysen Informationen der jeweiligen Softwarehersteller dienlich sein.

Tabelle 5.3: Matrix 2 – Zuordnung von Merkmalen zu Simulationstechniken (beispielhaft)

Simulations-technik ST_k	Merkmal C_i				
	1	2	3	…	o
1		X			X
2			X		
3	X	X			
…					
m	X		X		X

Gemeinsam ermöglichen die beiden Matrizen die Bestimmung geeigneter Simulationstechniken für eine zu untersuchende Eigenschaft unter Berücksichtigung der aktuell definierten Merkma-

le. Abbildung 5.16 verdeutlicht diese dreidimensionale Zuordnung graphisch[10], wobei die Simulationstechniken durch Punkte im Raum dargestellt sind. Die markierte Frontebene repräsentiert die Zuordnung von Simulationstechniken zu Eigenschaften aus Matrix 1. Die markierte Rückebene repräsentiert die Zuordnung von Merkmalen zu den Simulationstechniken aus Matrix 2.

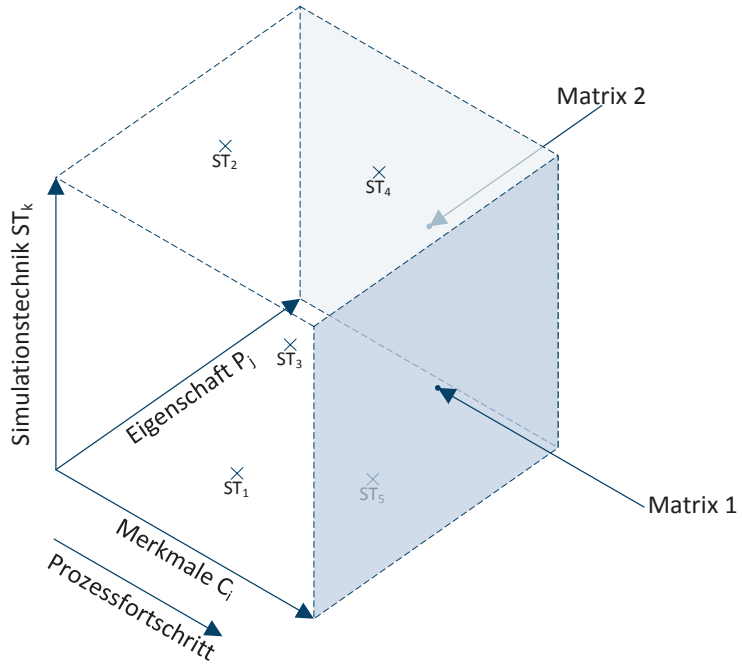


Abbildung 5.16: Dreidimensionale Zuordnung von Eigenschaften, Merkmalen und Simulationstechniken (beispielhaft)

Mit der Definition der Prozessphase nach Anzahl definierter Merkmale, die bereits in Kapitel 5.5.5 genutzt wurde, wird ein weiterer Vorteil dieser matrixbasierten Zuordnung deutlich: Unabhängig von der aktuellen Prozessphase, in der die Analyse durchgeführt werden soll, können die gleichen Matrizen genutzt werden. Über die zweite Matrix und die Anzahl der bereits definierten Merkmale werden alle Simulationstechniken herausgefiltert, die zum aktuellen Zeitpunkt nicht anwendbar sind. Demnach repräsentiert die Merkmalsachse in Abbildung 5.16 gleichzeitig auch den Prozessfortschritt. Diese Achse lässt sich zudem mittels der definierten Merkmale und der Phaseneinteilung aus Kapitel 5.5.5 in die einzelnen Phasen unterteilen. Im Umkehrschluss bedeutet dies auch, dass die Zuordnung zur Untersuchung geeignet ist, in wel-

[10] Diese Darstellung soll keinen direkten Zusammenhang zwischen Merkmalen und Eigenschaften herstellen. Sie dient einzig der Zuordnung von Simulationstechniken auf Basis gewünschter Eigenschaften und definierter Merkmale sowie der Suche nach noch zu definierenden Merkmalen.

cher Phase eine spezielle Eigenschaft analysiert und eine bestimmte Simulationstechnik angewandt werden kann oder aber welche Merkmale noch definiert werden müssen.

Die Bereitstellung einer allgemeingültig ausgefüllten Zuordnung ist aufgrund der Anzahl an Simulationstechniken weder durchführbar noch sinnvoll. Wie bereits erwähnt, muss dies anwendungs- und unternehmens- oder abteilungsspezifisch erfolgen. Daher wird in dieser Arbeit auf eine detaillierte Zuordnung verzichtet. Dennoch sind in Anhang A.5 die beiden Matrizen aus Tabelle 5.2 und Tabelle 5.3 beispielhaft für eine Auswahl der wichtigsten Simulationswerkzeuge zusammengefasst.

5.6.3 Möglichkeiten der Klassifizierung

Aufgrund der großen Anzahl an Simulationstechniken und Eigenschaften können die Matrizen aus Tabelle 5.2 und Tabelle 5.3 sehr schnell unübersichtlich werden, weshalb eine Klassifizierung sinnvoll ist. Neben der im vorangegangenen Teilkapitel beschriebenen Einteilung nach Prozessphasen kann eine solche Klassifizierung auch nach Domänen, Genauigkeit des Simulationsverfahrens oder nach dessen Ressourcenverbrauch erfolgen. Somit lassen sich über Filter Simulationstechniken oder Eigenschaften auf die relevanten einschränken. Hierfür muss jeder Simulationstechnik beziehungsweise jeder Eigenschaft ein entsprechendes Attribut zugeordnet werden, beispielsweise die Domänenzugehörigkeit.

Insbesondere die Berücksichtigung der Analysebedingungen ist bei der Auswahl des Simulationsverfahrens entscheidend, da hierdurch die Ergebnisgenauigkeit und -vertrauenswürdigkeit sowie der Ressourcenverbrauch maßgeblich beeinflusst werden. Diese Analysebedingungen lassen sich ebenfalls über Attribute mit anschließender Filterung einbinden. Hierbei bietet sich der in Kapitel 5.5.5 eingeführte Konfidenzfaktor ε_j an, der Analysebedingungen sowohl qualitativ als auch quantitativ berücksichtigen kann.

In der in Anhang A.5 dargestellten dokumentenbasierten Form der Matrizen ist die Klassifizierung und Filterung über Attribute nur schwer realisierbar. Darüber hinaus können diese Matrizen bei vielen Simulationstechniken, Eigenschaften und Merkmalen sehr schnell unübersichtlich werden. Aus diesem Grunde wurde ein Softwarewerkzeug entwickelt, welches das Auswahlverfahren unterstützt. Dessen Anwendung wird in Kapitel 6 beschrieben. Darüber hinaus finden sich in Anhang A.6 Bildschirmfotos des Programms, die den Aufbau und die Benutzeroberfläche verdeutlichen.

5.7 Umgang mit Unsicherheiten im Entwicklungsprozess

Eines der Hauptziele der hier entwickelten Methodik ist die frühe Einbindung von Simulationen im Entwicklungsprozess (siehe Anforderung A2 in Kapitel 5.2). Das während des Entwicklungsprozess stetig steigende Wissen über das zu entwickelnde System [Ullm10] führt jedoch zwangsläufig dazu, dass in früheren Phasen des Entwicklungsprozesses das Wissen noch nicht vollständig ist. Somit ist hier die Anzahl definierter Merkmale deutlich geringer [WeWD03]. Entwickler müssen daher mit den Unsicherheiten umgehen, die aus dieser Wissenslücke entstehen. Insbesondere bei der Simulation in frühen Phasen stellt dies eine Herausforderung dar, da die für die Parametrierung der Simulationsmodelle erforderlichen Parameter und Daten oftmals nicht oder nur sehr vage bekannt sind. Aus diesem Grunde sollen in diesem Kapitel Methoden zum Umgang mit Unsicherheiten in die Methodik eingebunden werden. Zunächst wird hierzu in Kapitel 5.7.1 der Begriff der Unsicherheit näher erläutert. In Kapitel 5.7.2 erfolgt die Einbindung geeigneter Methoden in die Entwicklungsmethodik. Deren Anwendung wird im Rahmen der Validierung in Kapitel 6 näher erläutert.

Dieses Kapitel basiert auf einer Vorarbeit des Autors zu diesem Thema [DoVi14c].

5.7.1 Arten von Unsicherheiten

Die Ursachen von Unsicherheiten sind vielfältig. In der Literatur lassen sich basierend auf ihren Ursache jedoch zwei Arten von Unsicherheiten unterscheiden [Thun03]: Aleatorische und epistemische. Nach Oberkampf et al. [ODRD02] werden für aleatorische Unsicherheiten oftmals auch die Begriffe der nicht-reduzierbaren, inhärenten oder stochastischen Unsicherheit sowie der Begriff der Varianz genutzt. Entsprechend sind sie zufallsbasiert [AnTa06] und werden daher gewöhnlich über Wahrscheinlichkeitsverteilungen berücksichtigt [ODRD02]. Übertragen auf technische Systeme beschreiben aleatorische Unsicherheiten die dem System oder der betrachteten Umwelt inhärenten Abweichungen [ODRD02]. Hierzu zählen etwa Toleranzen bei technischen Systemen. Für den Umgang mit dieser Art der Unsicherheit hat sich die Anwendung der Wahrscheinlichkeitstheorie etabliert [OHJW04]. Epistemische Unsicherheiten sind häufig auch unter den Begriffen der reduzierbaren, subjektiven oder kognitiven Unsicherheiten zu finden [ODRD02]. Oberkampf et al. [ODRD02] definieren diese Unsicherheiten als mögliche Ungenauigkeit bei der Modellierung aufgrund von Wissenslücken oder unvollständigen Informationen. Letztere können durch Unstimmigkeiten, Unbestimmtheit und Unschärfe verursacht werden [ODRD99].

Die in dieser Arbeit betrachteten Simulationsmodelle basieren in der Regel auf mathematischen und physikalischen Gesetzmäßigkeiten und können daher als deterministisch angesehen werden [OHJW04]. Entsprechend sind aleatorische Unsicherheiten bei der Erstellung der hier relevanten Modelle vernachlässigbar und werden somit im Rahmen dieser Arbeit nicht weiter diskutiert. Vielmehr stehen epistemische Unsicherheiten im Vordergrund. Diese werden gemäß der obigen Beschreibung durch fehlendes Wissen oder fehlende Informationen hervorgerufen, wodurch typischerweise der Entwicklungs- und Modellierungsprozess, insbesondere in frühen Phasen, kennzeichnet ist. Diese Wissens- und Informationsdefizite entstehen durch nicht definierte oder nicht konkretisierte Merkmale sowie allgemein durch fehlende Systemkenntnis.

Ursprünge von Unsicherheiten bei der Modellierung beschreiben Kiureghian und Ditlevsen [KiDi09]. Diese reichen von Unsicherheiten der Eingangsvariablen oder -parameter über Unsicherheiten probabilistischer sowie physikalischer Submodelle bis hin zu Unsicherheiten aufgrund von Rechenfehlern oder numerischen Approximationen. Oberkampf et al. [OHJW04] beziehen die Ursachen explizit auf die Modellierung physikalischer Systeme. Hierbei unterscheiden sie zwischen Parameterunsicherheiten – diese entsprechen den Unsicherheiten der Eingangsvariablen oder -parametern nach [KiDi09] – und Modellunsicherheiten – diese entsprechen restlichen Unsicherheiten nach [KiDi09]. Parameterunsicherheiten können sowohl aleatorisch als auch epistemisch sein, während Modellunsicherheiten stets epistemisch sind [OHJW04].

Unsicherheiten bei der Modellierung und Simulation führen zwangsläufig zu Fehlern. Diese beschreiben nach Oberkampf et al. [ODRD99] einen erkennbaren Mangel bei der Modellierung und Simulation, der nicht durch fehlendes Wissen hervorgerufen wird. Darüber hinaus lassen sich bewusste (englisch: acknowledged) und unbewusste (englisch: unacknowledged) Fehler unterscheiden [ODRD99]. Erstere können beispielsweise bewusste Vereinfachungen von Modellen sein, die oftmals eingesetzt werden, um Zeit und Aufwand bei der Modellerstellung und -parametrisierung sowie bei der Simulation einzusparen. Unbewusste Fehler werden typischerweise durch menschliche Fehler verursacht.

5.7.2 Methoden zum Umgang mit Unsicherheiten

Unsicherheiten treten aufgrund des geringeren Wissensgrades insbesondere in frühen Phasen auf. In der Regel wird versucht, das fehlende Wissen durch den Einsatz vereinfachter Modelle zu kompensieren, die weniger Systemparameter benötigen, jedoch auch ungenauer sind. Diese Vereinfachungen sind bewusste Fehler nach der Definition im vorangegangenen Teilkapitel.

Oftmals sind jedoch auch die bekannten Parameter mit Unsicherheiten behaftet, das heißt, dass kein definierter Wert bekannt ist, sondern vielmehr ein Parameterbereich. Dieser Anwendungsfall wird in der Übersicht von Methoden in Abbildung 5.17 durch den rechten Ast repräsentiert.

Neben dem Einsatz der Simulation als Mittel zur Überprüfung von Eigenschaften können Simulationen auch zur Reduzierung von Unsicherheiten genutzt werden, indem – basierend auf gewünschten Eigenschaften – Anforderungen und Modellparameter generiert und spezifiziert werden (siehe auch Kapitel 5.4). Da Simulationen allerdings nicht einfach invertierbar sind, muss ein iteratives Vorgehen gewählt werden, bei dem ausgehend von einer Startparametrierung der Sollwert schrittweise angenähert wird. Da diese Vorgehensweise jedoch zeitaufwendig ist, muss hier ein strukturiertes Vorgehen angewendet werden. In Abbildung 5.17 wird dieser Anwendungsfall durch den linken Ast repräsentiert.

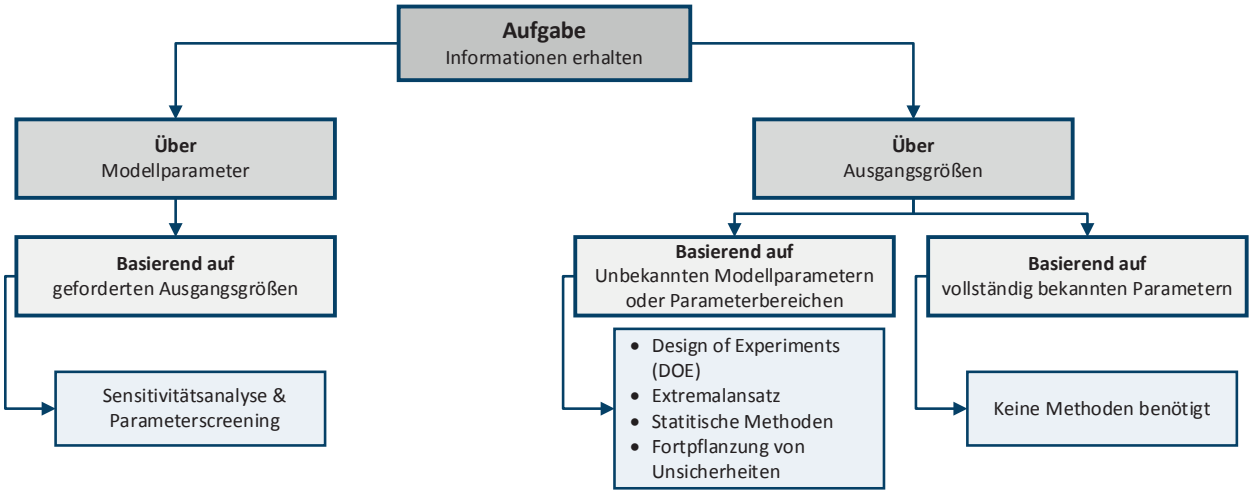


Abbildung 5.17: Übersicht über Methoden zum Umgang mit Unsicherheiten

Die in Abbildung 5.17 dargestellten Methoden zum Umgang mit epistemischen Unsicherheiten werden im Folgenden in die Entwicklungsmethodik integriert.

Anzumerken ist, dass die Übersicht aus Abbildung 5.17 nur einen Teil der existierenden Methoden widerspiegelt, da eine vollständige Bearbeitung des Themengebietes hier nicht möglich ist. Mit den ausgewählten Methoden wird Entwicklern allerdings eine Grundauswahl für wichtige Anwendungsfälle im Rahmen der simulationsbasierten Entwicklung gegeben. Tiefergehende Informationen können den jeweils angegebenen Literaturreferenzen entnommen werden.

5.7.2.1 Design of Experiments (DOE)

Design of Experiments (DOE, deutsch: statistische Versuchsplanung) ist eine statistische Methode, die mit einem Minimum an Experimenten versucht, den Zusammenhang zwischen Eingang und Ausgang eines Systems zu bestimmen. Ursprünglich wurde die Methode für die Anwendung bei physischen Experimenten entwickelt, jedoch ist die Anwendung auf Simulationen ohne Einschränkungen übertragbar. Bei klassischen Ansätzen werden die Eingangsparameter stets einzeln und nacheinander geändert, was für jeden geänderten Parameter ein zusätzliches Experiment bedeutet. Entsprechend hoch ist der Aufwand für dieses Verfahren, zudem sind gegenseitige Beeinflussungen von Parametern nicht vollständig zu bestimmen. Design of Experiments adressiert diese Probleme durch die Planung, Gestaltung und Analyse von Experimenten, sodass effizient und effektiv fundierte Schlüsse gezogen werden können [Anto00].

Prinzipiell lassen sich vollfaktorielle und teilfaktorielle Versuchspläne unterscheiden, die entweder zwei- oder dreistufig ausgelegt sind [Anto00]. Zwei- und dreistufig bezieht sich auf die Anzahl an Experimenten, mit denen der Parameterbereich abgedeckt wird. Bei zweistufigen Experimenten werden nur die Randwerte des Bereichs genutzt, bei dreistufigen ein zusätzlicher Wert in der Mitte. Teilfaktoriell bedeutet, dass im Gegensatz zu vollfaktoriellen Versuchsplänen nicht alle Eingangsparameter variiert werden, weniger relevante Parameter werden konstant gehalten. Hierbei lässt sich die Anzahl an Experimenten signifikant reduzieren – bei vollfaktoriellen, zweistufigen Versuchsplänen für k Parameter sind 2^k Experimente notwendig. Jedoch werden Beeinflussung und Abhängigkeiten höherer Ordnung zwischen den Parametern nicht berücksichtigt [Anto00], was wiederum einem bewussten Fehler entspricht. Die Entscheidung für voll- oder teilfaktorielle Versuchspläne muss situationsabhängig durch den Entwickler erfolgen.

Design of Experiments kann in der simulationsbasierten Entwicklung bei allen Simulationsstudien eingesetzt werden. Insbesondere bei Parameterstudien, die in der Regel eine große Anzahl an Experimenten erfordern, lässt sich damit der Ressourcenaufwand deutlich reduzieren. So wenden Seppäla et al. [SeBC11] diese Technik beispielsweise an, um bei der Konzeptauswahl die Anzahl an Simulation zu reduzieren.

Für ausführliche Informationen zur Planung, Gestaltung und Analyse von Experimenten mittels Design of Experiments sei auf die Literatur zu diesem Thema verwiesen, beispielsweise [Anto00], [Klep11] und [SiVH10].

5.7.2.2 Extremalansatz

Der Extremalansatz dient dazu, auf Basis gegebener Eingangsparameter den Bereich zu bestimmen, in dem die simulierten Ergebniswerte liegen werden [DuCh00]. Durch die reine Betrachtung der Randwerte der Eingangsparameter ist der Simulationsaufwand verhältnismäßig gering. Jedoch werden nur die maximal und minimal möglichen Ausgangswerte bestimmt. Das Verhalten zwischen diesen Extremwerten sowie die gegenseitigen Beeinflussungen von Parametern können nicht bestimmt werden.

Der Extremalansatz ist mit dem vollfaktoriellen, zweistufigen Versuchsplan gemäß Design of Experiments vergleichbar.

Auf diese Weise lässt sich sehr einfach feststellen, in welchem Bereich sich das Verhalten oder eine Eigenschaft eines zu simulierenden Systems befinden wird, was beispielsweise bei Konzeptentscheidungen eine fundierte Grundlage bietet.

5.7.2.3 Statistische Methoden

Neben dem Extremalansatz sind statistische Methoden weit verbreitet. Beispielhaft hierfür sei die Monte-Carlo-Simulation (MCS) genannt. Hierbei wird der Wertebereich der Eingangsparameter durch eine statistische Verteilung, zum Beispiel eine Gauß-Verteilung, abgebildet, die somit auch Rückschlüsse über die Wahrscheinlichkeit des Auftretens eines bestimmten Werts des Eingangsparameters gibt. Aus dieser Verteilung wird zufällig ein Wert (englisch: Sample) bestimmt, mit dem die Simulation durchgeführt wird – entsprechend werden diese Methoden als Sampling-basiert bezeichnet. Dies wird mehrfach wiederholt, bis die Standardabweichung der resultierenden Verteilung der Simulationsergebnisse abgeschätzt werden kann [CoSt09]. Da diese Art der Simulation je nach Modellkomplexität und Verteilung der Eingangsparameter aufgrund der Vielzahl erforderlicher Simulationsläufe einen enormen Aufwand darstellen kann, existieren weitere Sampling-basierte Methoden, welche die Anzahl an Simulationen reduzieren können. Bei komplexen Simulationen hat sich insbesondere das Latin Hypercube Sampling (LHS) etabliert [HJSS06].

Die Anwendung statistischer Methoden bietet sich an, wenn genauere Kenntnis über die Verteilung des Wertebereichs von Modellparametern besteht. In diesem Fall lässt sich eine Verteilung abschätzen, die für eine Monte-Carlo-Simulation herangezogen werden kann. Entsprechend liefert die Simulation auch eine Verteilung der Ergebnisse, die wesentlich detailliertere Schlüsse

zulässt als beispielsweise der Extremalansatz. Dabei ist die Monte-Carlo-Simulation auch für die Simulation aleatorischer Unsicherheiten geeignet, wie etwa von Toleranzeinflüssen.

Eine weiterführende Diskussion diese Methoden würde die eigentliche Zielsetzung der vorliegenden Arbeit übersteigen, weshalb auf die umfangreiche Literatur verwiesen wird, in der auch die mathematische Formulierung diskutiert wird. Beispielhaft seien hier [CoSt09], [HJSS06] und [McBC00] angeführt.

Statistische Methoden haben den Nachteil, dass eine Wahrscheinlichkeitsverteilung für die Modellparameter bekannt oder abgeschätzt werden muss. Insbesondere in frühe Phasen, etwa der Systemkonzeptionierung, ist dies jedoch nur schwer möglich. In diesem Fall können Methoden der Informationstheorie herangezogen werden, zu der unter anderem die Fuzzy-Set-Theorie zählt. Fuzzy-Sets berücksichtigen graduelle und flexible Spezifikationen und ermöglichen den Umgang mit unvollständigen Informationen [DuPr93]. Sind die Grenzen des Bereichs für Modellparameter nicht vollständig bekannt, so wird diese Unsicherheit mittels einer charakteristischen Funktion ausgedrückt, die Werte im Intervall von null bis eins annimmt [DuPr93]. Die Definition dieser charakteristischen Funktion kann wesentlich einfacher sein als die Definition einer Wahrscheinlichkeitsverteilung für die Monte-Carlo-Simulation, weshalb die Fuzzy-Set-Theorie als alternative Methode angesehen werden kann. Weiterführende Informationen hierzu sind beispielsweise in [KuSc05], [Ross10] und zu finden.

5.7.2.4 Fortpflanzung von Wahrscheinlichkeiten

Die bisher beschriebenen Methoden sind hauptsächlich für einzelne Modelle gedacht. Die Entwicklung mechatronischer Systeme erfordert jedoch die Nutzung von systemübergreifenden Simulationsmodellen, die in der Regel aus mehreren Submodellen bestehen. Jedes dieser Submodelle wird dabei von Unsicherheiten beeinflusst. Im Gesamtsystemmodell führt dies zu einem kumulierten Effekt der einzelnen Unsicherheiten [DuCh00], da sich diese im Gesamtmodell fortpflanzen. Du und Chen [DuCh00] drücken dies in einem Fortpflanzungsmodell gemäß Abbildung 5.18 aus.

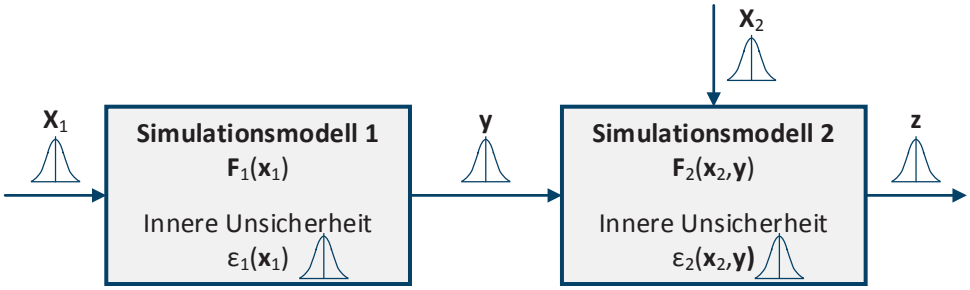


Abbildung 5.18: Fortpflanzung von Unsicherheiten, nach [DuCh00]

Demnach ist der Vektor x der Eingangsparameter eines Modells mit Unsicherheiten behaftet, die durch Wahrscheinlichkeitsverteilungen repräsentiert sind. Darüber hinaus besitzt jedes Submodell eine interne Unsicherheit ε. Somit ergibt sich nach [DuCh00] als Ausgangsergebnis von Modell 1:

$$y = F_1(x_1) + \varepsilon_1(x_1) \tag{5.6}$$

Dieses Ausgangsergebnis y stellt wiederum den Eingangswert für Modell 2 dar, wodurch sich für das Ausgangsergebnis von Modell 2 ergibt:

$$z = F_2(x_2, y) + \varepsilon_2(x_2, y) \tag{5.7}$$

Dieses Vorgehen lässt sich für beliebig viele Submodelle erweitern und ist mit den bisher beschriebenen Methoden kombinierbar.

Insbesondere bei der Entwicklung mechatronischer Systeme ist es wichtig, dass Entwickler diese Fortpflanzung von Unsicherheiten berücksichtigen, da diese das Simulationsergebnis am Ende der Fortpflanzungskette signifikant beeinflussen können.

5.7.2.5 Sensitivitätsanalyse und Parameterscreening

Sensitivitätsanalysen und Parameterscreening dienen dazu, auf Basis von gewünschten Ausgangsgrößen Informationen über Modellparameter zu erhalten. Sie decken somit den linken Ast aus Abbildung 5.17 ab. Sensitivitätsanalysen werden dazu genutzt, den Einfluss einzelner Parameter und deren Unsicherheit auf das Gesamtverhalten, inklusive der Gesamtunsicherheit, zu bestimmen [HJSS06]. Dadurch werden die einzelnen Modellparameter bezüglich ihres Einflusses auf das Simulationsergebnis untersucht und bewertet. Basierend auf dieser Bewertung können die einflussreichsten Parameter identifiziert werden. In einer Simulationsstudie müssen somit nur diese über den gesamten Parameterbereich variiert werden. Die restlichen Modellparameter können auf einen oder wenige Werte festgesetzt werden, wodurch die Anzahl an Pa-

rametervariationen und damit ebenso die Anzahl an Simulationen deutlich reduziert werden kann. Entsprechend lässt sich wesentlich schneller eine optimale Parameterkombination identifizieren.

Typischerweise werden bei der Sensitivitätsanalyse und beim Parameterscreening die bereits zuvor erwähnten Techniken angewandt. So wird beispielsweise mittels Design of Experiments ein teilfaktorieller, zweistufiger Versuchsplan erstellt, der mit einem Minimum an Experimenten die Bestimmung der Einflüsse einzelner Parameter ermöglicht. Für das Paramterscreening werden häufig sogenannte Plackett-Burman-Versuchspläne genutzt, die einen Sonderfall der teilfaktoriellen, zweistufigen Pläne darstellen [AnTa06]. Die Ergebnisse dieser Analysen lassen sich in Pareto-Diagrammen oder Einflussgraphen veranschaulichen. Zu beachten ist jedoch, dass mit diesen Ansätzen gegenseitige Beeinflussungen von Parametern kaum bestimmt werden können.

Darüber hinaus sind auch die statistischen Methoden für die Sensitivitätsanalyse anwendbar, wobei typischerweise Methoden gewählt werden, die weniger Experimente benötigen, wie zum Beispiel das Latin Hypercube Sampling. Weitere Information zur Nutzung statistischer Methoden bei der Sensitivitätsanalyse geben Helton et al. [HJSS06].

Die Anwendung der hier beschriebenen Methoden und Techniken wird in Kapitel 6 anhand eines Validierungsbeispiels näher erläutert.

6 Validierung der Entwicklungsmethodik

6.1 Hintergrund und Ziele

In Kapitel 5 wurde eine Methodik für die simulationsbasierte Entwicklung mechatronischer Systeme beschrieben. Ziel dieses Kapitels ist nun die Validierung der Methodik anhand eines mechatronischen Entwicklungsprojektes. Hierbei soll insbesondere die Anwendbarkeit der Methodik und deren Eignung für den zu Beginn von Kapitel 5 definierten Anwendungsbereich im Vordergrund stehen. Darüber hinaus soll dieses Validierungsbeispiel die Anwendung der in Kapitel 5 beschriebenen theoretischen Aspekte der Methodik verdeutlichen.

Zunächst wird in Kapitel 6.2 das Entwicklungsprojekt beschrieben, anhand dessen die Validierung durchgeführt wird. Anschließend erfolgt in Kapitel 6.3 die Beschreibung der Anwendung der Entwicklungsmethodik. Die Erkenntnisse der Validierung werden in Kapitel 6.4 diskutiert.

Die Ergebnisse dieses Kapitels wurden in Teilen bereits vom Autor in [DoVi14b] und [DoVi14c] veröffentlicht.

6.2 Beschreibung des Validierungsprojektes

Die Validierung der Methodik wird an der Entwicklung eines aktiven Fahrwerks für Fahrräder durchgeführt. Hierbei handelt es sich um ein typisches mechatronisches System gemäß dem in Kapitel 2.1 beschriebenen Aufbau. Da in diesem Kapitel die Anwendung der Entwicklungsmethodik im Fokus stehen soll, stellt das aktive Fahrwerk aufgrund seines mäßigen Komplexitätsgrades ein geeignetes Projekt dar. Die Entwicklung wird an einem dreirädrigen Fahrrad vom Typ „Scorpion“ der Firma HP Velotechnik durchgeführt, welches in Abbildung 6.1 dargestellt ist.

Abbildung 6.1: HP Velotechnik Scorpion [WWW33]

Dieses Fahrzeug verfügt über einen passiven Öldämpfer mit Luft- oder Spiralfeder, welcher in Abbildung 6.1 hervorgehoben ist. Das Feder-Dämpfer-Element dient zur Reduzierung von Stößen im Sitzbereich, die durch Bodenunebenheiten induziert werden. Prinzipbedingt ergeben sich jedoch verschiedene Nachteile: Vertikalbewegungen des Sitzes können aufgrund der passiven Arbeitsweise nicht vollständig ausgeglichen werden. Weiterhin lässt sich der Dämpfer nicht ohne weiteres an verschiedene Bodenbedingungen anpassen, etwa Asphalt oder Waldwege. Durch den Einsatz eines aktiven Fahrwerks, welches auf die aktuellen Bodenbedingungen reagiert, sollen Fahrkomfort und -stabilität verbessert werden. Insbesondere die Vertikalbewegungen des Sitzes sollen hierdurch reduziert werden. Zudem soll die Möglichkeit gegeben sein, das System an verschiedene Bodenbedingungen anzupassen, entweder autonom oder über eine Benutzerschnittstelle. Aus dem Kraftfahrzeugbereich sind aktive Fahrwerke bereits bekannt und prinzipiell übertragbar. Beispielsweise beschreiben Plitt et al. [PlTT89] die Entwicklung eines derartigen Fahrwerks sowie dessen Vorteile gegenüber passiven Systemen.

Aufgrund der Fokussierung auf die Validierung und Verdeutlichung der Entwicklungsmethodik wird die Entwicklung des aktiven Fahrwerks teilweise vereinfacht dargestellt und erhebt nicht den Anspruch auf Vollständigkeit. Tiefergehende Informationen bezüglich der Funktionalität sowie der Entwicklung aktiver Fahrwerke können beispielsweise [Iser06] und [WuER11] entnommen werden.

6.3 Anwendung der Entwicklungsmethodik

Im Folgenden wird die Entwicklung des aktiven Fahrwerks unter Benutzung der in Kapitel 5 vorgeschlagenen Methodik beschrieben. Der Ablauf richtet sich dabei nach dem Prozessmodell aus Abbildung 5.8: Zunächst wird in Kapitel 6.3.1 die Definition von Analysemeilensteinen und darauf basierend in Kapitel 6.3.2 das Vorgehen in der Systemkonzeptionierung und -simulation beschrieben. In dieser Phase wird ebenso erläutert, wie Unsicherheiten methodisch behandelt werden können. Die Erläuterung der Phase des domänenspezifischen Entwurfs und der detaillierten domänenspezifischen Simulation erfolgt in Kapitel 6.3.3 und enthält neben der Beschreibung des methodischen Vorgehens auch Erläuterungen zur Auswahl von Analysewerkzeugen sowie zu Iterationszyklen. Abschließend erfolgt in Kapitel 6.3.4 die Betrachtung der Systemintegration und der detaillierten Systemsimulation.

6.3.1 Anforderungen und Analysemeilensteine

Startpunkt des Prozesses stellen die (vorläufig) definierten Anforderungen dar. Auf deren Basis beginnt der eigentliche Entwicklungsprozess im Rahmen der hier vorgestellten Methodik.

6.3.1.1 Initiale Definition der Analysemeilensteine

Basierend auf den zu Beginn bekannten Anforderungen wird ein erster Satz an Analysemeilensteinen definiert, um den Entwicklungsprozess zu strukturieren und die Analysetätigkeiten zu planen. Zu diesem frühen Zeitpunkt des Entwicklungsprozesses bereits bekannte Anforderungen sind beispielsweise die geometrischen Einbaubedingungen sowie die grundsätzlichen funktionalen Anforderungen, etwa die gewünschte Reaktionszeit und -genauigkeit des Systems. Die initialen Analysemeilensteine für die Systemkonzeptionierung und Systemkonzeptsimulation sind in Tabelle 6.1 dargestellt, welche sich prinzipiell nach dem Aufbau der Vorlage für Analysemeilensteine aus Anhang A.1 richtet.

Tabelle 6.1: Definition von Analysemeilensteinen für Systemkonzeptionierung und Systemkonzeptsimulation

Analyse-meilenstein	Eigenschaft P_j	Wert für P_j	Grenz-wert ΔP_j	Analyse-verfahren	Analyse-bedingungen
$AMS_{Syskon\_1.1}$	Energieverbrauch	Vergleichs gleichs-größe	-	Systemlevel-Simulation, SimulationX (Modelica)	Vereinfachte Simulations-modelle
$AMS_{Syskon\_1.2}$	Reaktionszeit	0,5 s	Max		
$AMS_{Syskon\_1.3}$	Abweichung der vertikalen Position	Sollwert	+/- 10 %		
$AMS_{Syskon\_2.1}$	Gesamt-mehrgewicht	5 kg	Max	Schätzung	Basierend auf Erfahrung o. Ä.
$AMS_{Syskon\_2.2}$	Kosten	3500 €	Max		
Nach erfolgter Konzeptauswahl ($AMS_{Syskon\_1.1}$ bis $AMS_{Syskon\_2.2}$ erfüllt)					
$AMS_{Syskon\_3.1}$	Erforderliche Aktorkraft	Abgeleitet aus Simu-lation (PR_j)	-	Systemlevel-Simulation, SimulationX (Modelica)	Vereinfachte 2D-MKS-Simulation, inkl. Aktor und vereinfachter Regelung
$AMS_{Syskon\_3.2}$	Erforderliche Ak-torgeschwindigkeit		-		

6.3.1.2 Softwarewerkzeug zur Definition und Organisation von Analysemeilensteinen

Wie bereits in Kapitel 5.4 erläutert, besitzt die Definition und Dokumentation von Analysemeilensteinen in dokumentenbasierter Form einige Nachteile: Die Übersichtlichkeit von Tabellen leidet bei einer großen Anzahl an Meilensteinen. Zudem können Änderungen von Meilensteinen nicht zurückverfolgt werden und das parallele Arbeiten mehrerer Personen an einem Dokument ist nicht möglich. Darüber hinaus ist keine automatisch generierte graphische Darstellung der Meilensteine möglich, beispielsweise in einer Zeitleiste. Aus diesem Grunde wurde im Rahmen dieser Arbeit ein webbasiertes Softwarewerkzeug entwickelt, welches die Definition und Dokumentation von Analysemeilensteinen modellbasiert ermöglicht. Dabei fördert die webbasierte Implementierung die Zugänglichkeit und ermöglicht das kollaborative Arbeiten verschiedener Teams. Die Vorteile dieses Übergangs von dokumenten- zu modellbasierten Ansätzen sind bereits aus dem Modellbasierten Systems Engineering bekannt und in Kapitel 3.2.6 erläutert. An das Softwarewerkzeug werden folgende Anforderungen gestellt:

- Möglichkeit zur Definition von Analysemeilensteinen
- Möglichkeit der Definition ohne festgelegte Reihenfolge
- Möglichkeit zur Änderungen inklusive Rückverfolgbarkeit
- Zuordnung der Analysemeilensteine zu Phasen und Systemleveln
- Anordnung der Analysemeilensteine in einer Zeitleiste mit Detaildarstellung von Phasen und Systemleveln
- Unterscheidung von parallelen und sequentiellen Analysemeilensteinen
- Markierung erfüllter Meilensteine

In Abbildung 6.2 ist für das Beispiel des aktiven Fahrwerks die Prozessübersicht aus dem Softwarewerkzeug inklusive aller Analysemeilensteine dargestellt, die zu diesem frühen Zeitpunkt definiert werden können. Zur Verdeutlichung sind die Meilensteine aus Tabelle 6.1 in der Darstellung zusätzlich beschriftet. Bereits erfüllte Analysemeilensteine werden grau dargestellt. Zudem ist es möglich, durch Anwählen einzelner Phasen oder Analysemeilensteine diese in einer Detailansicht aufzurufen. Hierin können die Analysemeilensteine nach Systemleveln geordnet dargestellt und auch bearbeitet werden. Weitere Darstellungen der Benutzeroberfläche des Programms, unter anderem auch von der Definition und Bearbeitung von Analysemeilensteinen, sind in Anhang A.2 zu finden.

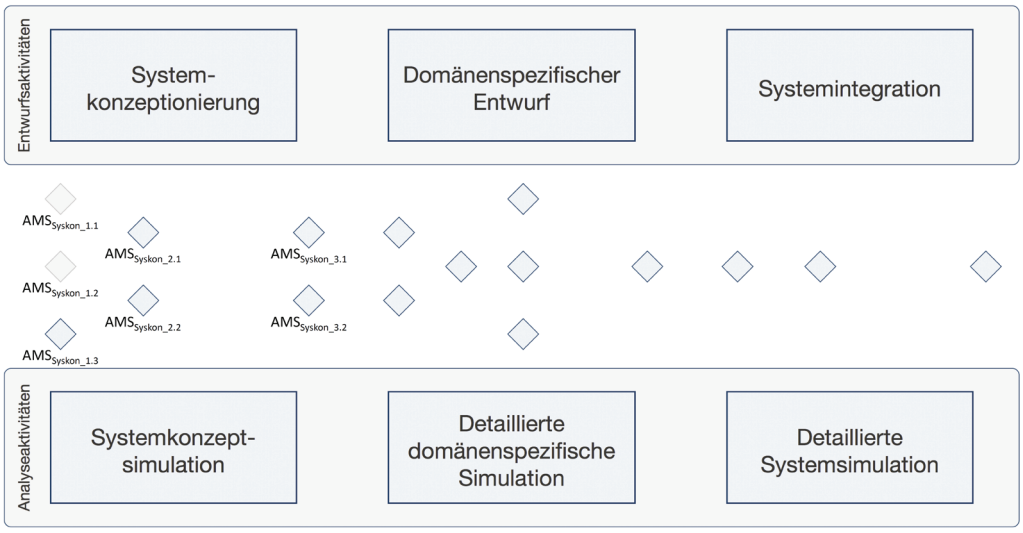


Abbildung 6.2: Gesamtprozessübersicht mit Analysemeilensteinen im entwickelten Softwarewerkzeug

6.3.2 Systemkonzeptionierung und Systemkonzeptsimulation

Die in diesem Kapitel verwendeten Modelle und Simulationen basieren teilweise auf der Arbeit von Huwig [Huwi13].

6.3.2.1 Methodische Konzeptgenerierung und -absicherung

Der Ablauf der Entwicklung in dieser Phase ist in Abbildung 6.3 als Ausschnitt des vereinfachten Prozessmodells erneut dargestellt.

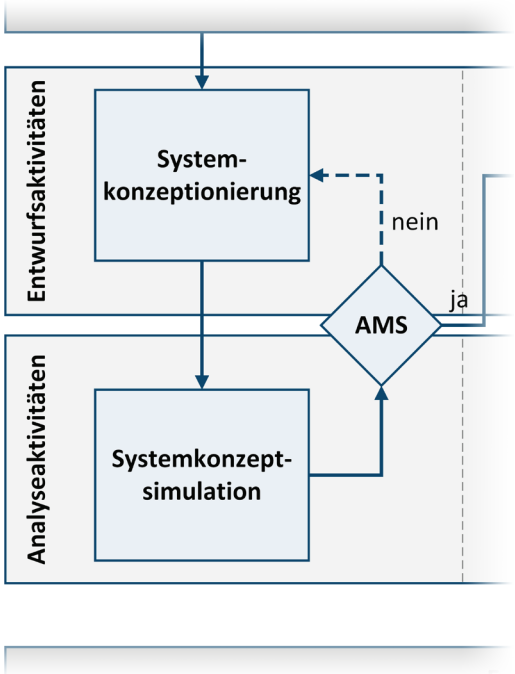


Abbildung 6.3: Prozessablauf in der Systemkonzeptionierung und Systemkonzeptsimulation

Demnach beginnt der Prozess – und gleichzeitig der erste Analyse-Synthese-Zyklus – in der Systemkonzeptionierung, die einen Teil der Entwurfsaktivitäten darstellt. Die Generierung von Systemkonzepten beginnt mit der Funktionsmodellierung des Systems. Für das aktive Fahrwerk ist diese in Abbildung 6.4 nach dem Vorgehen von Pahl und Beitz [PBFG07] dargestellt. Prinzipiell besitzt diese Art der Funktionsmodellierung gewisse Beschränkungen bei der Modellierung mechatronischer Systeme. Hier soll jedoch aufgrund der Einfachheit des Beispiels und der Verbreitung der Vorgehensweise nach Pahl und Beitz auf zusätzliche Ausführungen verzichtet werden. Weitere Funktionsmodellierungsansätze sind in [EiGB13] und [EKBA08] zusammengefasst. Weiterhin wäre eine Ergänzung dieses Funktionsmodell etwa durch eine Statechart-Modellierung der prinzipiellen Software-Funktionalität denkbar. Diese Modellierung lässt sich ebenso mit SysML durchführen, wodurch eine konsistente Modellierung ermöglicht wird. Ein derartiges SysML-Modell des aktiven Fahrwerks kann im gesamten Entwicklungsprozess als Basis eines übergeordneten Systemmodells verwendet werden. Da hier jedoch die Methodik im Vordergrund steht, wird auf eine weitergehende Erläuterung verzichtet. Der Vollständigkeit halber ist die SysML-Modellierung des aktiven Fahrwerks in Anhang A.3 ausschnittsweise dargestellt.

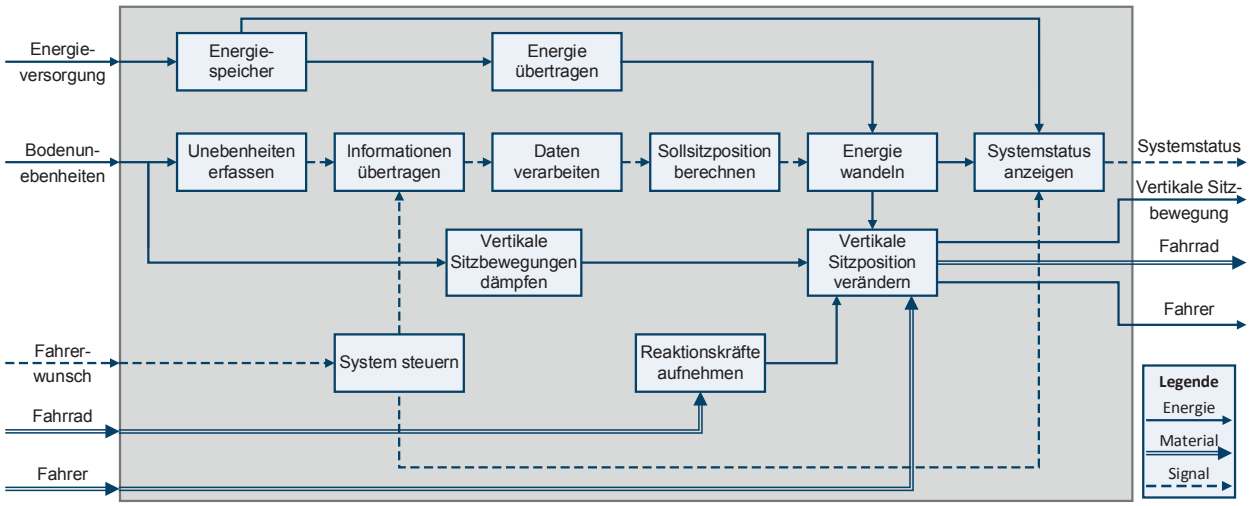


Abbildung 6.4: Funktionsmodell für das aktive Fahrwerk

Basierend auf dem Funktionsmodell werden Lösungskonzepte erarbeitet, beispielsweise nach dem Vorgehen von Pahl und Beitz. Zu jeder Teilfunktion werden Wirkprinzipien gesucht, die in Kombination eine Gesamtlösung darstellen. Hierbei können Konstruktionskataloge angewendet werden, wie etwa von Roth [Roth00] beschrieben. Die Generierung erster Konzepte ist hierdurch beendet, weshalb direkt in den zweiten Aktivitätsstrang, die Analysetätigkeiten, gewechselt wird. In der Methodik stellt die Einteilung des Prozessmodells in zwei parallele Aktivitätsstränge ein zentrales Element für die kontinuierliche Eigenschaftsabsicherung dar. Im Hinblick

auf diese parallele Simulation der entwickelten Konzepte wurden die Vorteile von Modelica-basierten Simulationswerkzeugen bereits in Kapitel 5.5.2 erläutert: Modellbibliotheken können bereits während der Entwurfstätigkeiten als Konstruktionskataloge genutzt werden und somit bei der Konzeptgenerierung unterstützen. Somit wird die Parallelität der beiden Aktivitätsstränge weiter gefördert. In der Systemkonzeptsimulation wird auf die Analysemeilensteine zurückgegriffen, in denen festgelegt ist, welche Eigenschaften wie analysiert werden sollen. Im vorliegenden Beispiel stellen „Vertikale Sitzbewegung dämpfen“ und „Vertikale Sitzposition verändern“ die wichtigsten Funktionen dar. Im Hinblick auf den Simulationsaufwand werden daher zunächst nur diese Funktionen betrachtet und die entsprechenden Konzepte in vereinfachten Simulationen verglichen. Übertragen auf die Systemhierarchie bedeutet dies, dass die Entwurfs- und Simulationstätigkeiten zu diesem Zeitpunkt nicht auf oberstem Systemlevel (s = 0, siehe Kapitel 5.5.1) erfolgen, sondern auf einem darunterliegenden Level (s = 1, je nach Aufbau der Systemhierarchie). Auf dieser Systemebene lassen sich hauptsächlich drei Wirkprinzipen für die beiden genannten Funktionen unterscheiden:

- Elektromechanischer Linearaktor
- Pneumatischer Linearaktor
- Hydraulischer Linearaktor

Aus der Definition der Analysemeilensteine in Tabelle 6.1 – insbesondere $AMS_{Syskon\_1.1}$ bis $AMS_{Syskon\_1.3}$ – wird ersichtlich, dass aufgrund des geringen Wissens über das System zu diesem Zeitpunkt nur stark vereinfachte Simulationen auf Systemebene durchgeführt werden können. Im vorliegenden Beispiel wird dabei das Modelica-basierte Softwarewerkzeug SimulationX der ITI GmbH verwendet[11]. In den Modellen kommt nur ein vereinfachter PID-Regler zum Einsatz. Ebenso ist die Kinematik des Fahrrades nicht abgebildet, der Aktor wird stattdessen über ein vereinfachtes Modell direkt angeregt, welches aus dem Kraftfahrzeugbereich als Viertelfahrzeugmodell bekannt ist [HeEr07]. Hierbei werden der Boden-Reifen-Kontakt sowie das Fahrwerk über Feder-Dämpfer-Systeme vereinfacht modelliert. Beispielhaft sind die Modelle für das elektromechanische Prinzip in Abbildung 6.5 sowie für das hydraulische Prinzip in Abbildung 6.6 dargestellt.

[11] Prinzipiell sind auch andere Softwarewerkzeuge, welche zur Simulation mechatronischer Systeme geeignet sind, einsetzbar (siehe hierzu Kapitel 3.1.6).

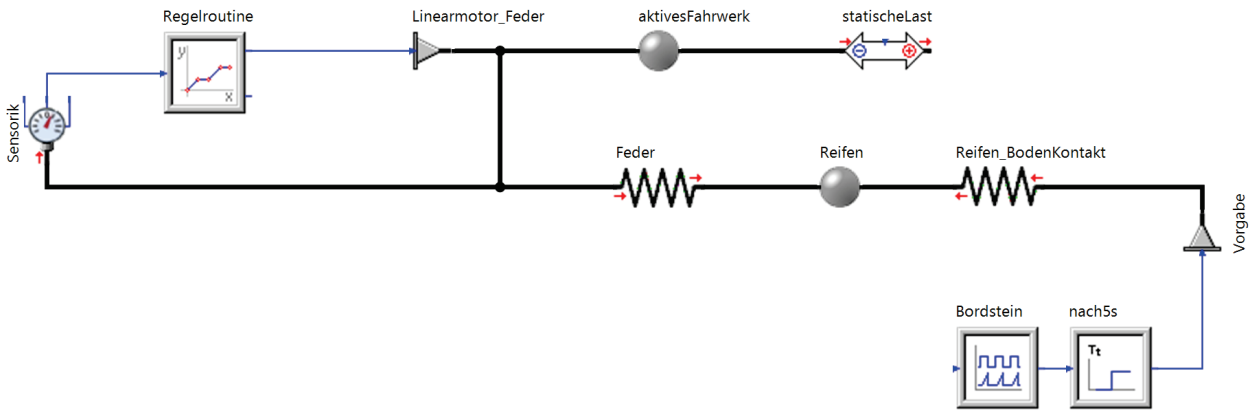


Abbildung 6.5: Vereinfachtes Simulationsmodell des elektromechanischen Konzeptes [Huwi13]

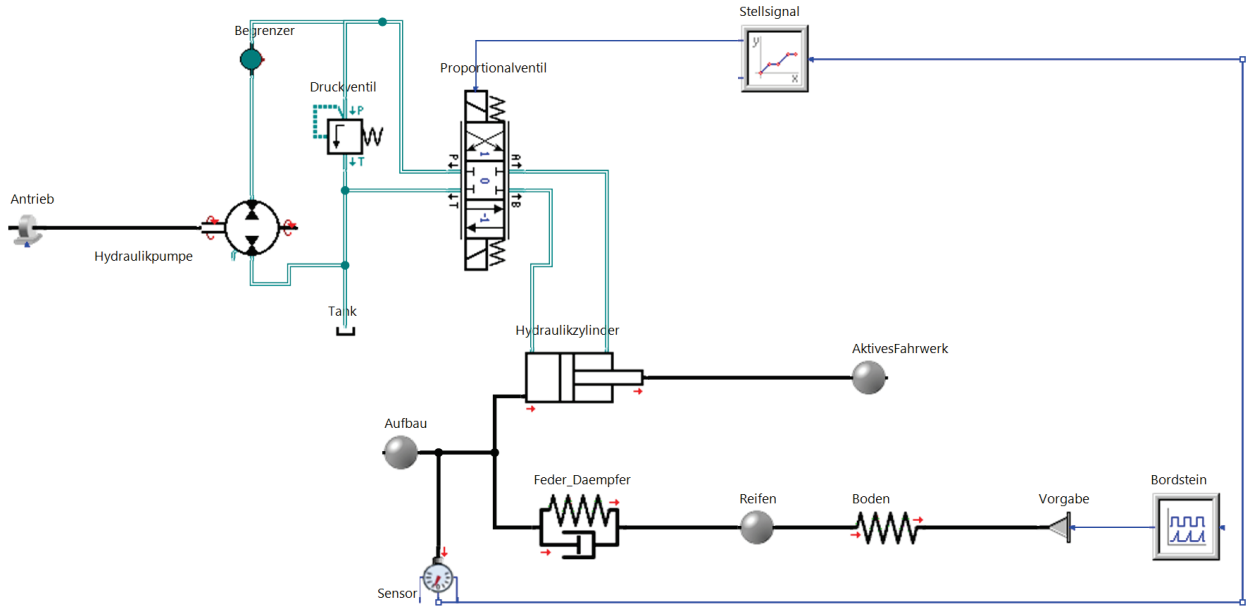


Abbildung 6.6: Vereinfachtes Simulationsmodell des hydraulischen Konzeptes [Huwi13]

Für die Simulationen werden zwei Lastfälle (LF) definiert:

- Bordsteinüberfahrt (LF1): Bordsteinhöhe 100 mm, Überfahrzeit 0,5 s (15 km/h)
- Unebener Boden (LF2): Sinusförmige Anregung mit einer Amplitude von 50 mm und einer Frequenz von 0,5 Hz

Diese Lastfälle stellen eine Änderungen der Anforderungen dar und wirken sich entsprechend direkt auf die Analysemeilensteine aus. Daher muss in diesem Fall der Meilenstein $AMS_{Syskon\_1.2}$ gemäß Tabelle 6.2 angepasst werden. Derartige Anpassungen sind im entwickelten Softwarewerkzeug über Revisionen möglich, die zudem auch rückverfolgbar sind.

Tabelle 6.2: Beispiel für die Anpassung von Analysemeilensteinen

Analyse-meilenstein	Eigenschaft P_j	Wert für P_j	Grenz-wert ΔP_j	Analyse-verfahren	Analyse-bedingungen
$AMS_{Syskon\_1.2}$	Reaktionszeit	0,3 s für LF1	Max	Systemlevel-Simulation, SimulationX (Modelica)	Nur Aktor mit vereinfachter Regelung
		0,5 s für LF1	Max		

Basierend auf den parallel zur Systemkonzeptionierung erstellten Modellen werden die in $AMS_{Syskon\_1.1}$ bis $AMS_{Syskon\_1.3}$ geforderten Simulationen zum Zweck des Konzeptvergleichs durchgeführt. Hierbei wird prinzipiell, wie bei allen Simulationen im Entwicklungsprozess, das in Kapitel 5.5.6 beschriebene Vorgehen innerhalb von Simulationsphasen angewendet. Über die simulierbaren Eigenschaften hinaus existieren in dieser frühen Phase Eigenschaften, die nicht oder nur mit sehr hohem Aufwand simuliert werden können. Hierzu zählen das Systemgewicht sowie die Kosten inklusive des allgemeinen konstruktiven Aufwands (siehe $AMS_{Syskon\_2.1}$ und $AMS_{Syskon\_2.2}$). Diese Analysen erfolgen in der Systemkonzeptionierung und Systemkonzeptsimulation über Schätzungen, die auf Erfahrungen oder ersten Recherchen basieren. Prinzipiell lassen sich Simulationsergebnisse auch in Bewertungsverfahren integrieren, etwa nach VDI-Richtlinie 2223 [VDI2223], indem diese als objektive Kriterien für eine Einstufung herangezogen werden. Auf diese Art der Bewertung wird hier jedoch nicht weiter eingegangen.

Mittels des im Prozessmodell verankerten TOTE-Schemas erfolgt der Vergleich der ermittelten mit den in Analysemeilensteinen geforderten Eigenschaften. Hierbei wird deutlich, dass das elektromechanische Prinzip die geforderten Eigenschaften am besten erfüllt, weshalb dieses als Systemkonzept gewählt wird. Dieses Konzept ist ähnlich dem Fahrwerkssystem, welches von der Bose GmbH [WWW34] für Kraftfahrzeuge entwickelt wurde. Ein Nichterfüllen der in Analysemeilensteinen geforderten Eigenschaften würde an dieser Stelle eine Iteration auslösen. Da dies jedoch hier nicht zutrifft, ist der erste Analyse-Synthese-Zyklus zwischen Systemkonzeptionierung und der parallelen Systemkonzeptsimulation beendet.

Durch die Wahl des elektromechanischen Konzeptes ist die Reife des Systemkonzeptes gestiegen, weshalb im nächsten Schritt eine weitere Detaillierung erfolgen muss, um die Anforderungserfüllung und der Realisierbarkeit abzusichern. Hierbei erfolgt ein Wechsel des Systemlevels: Durch Integration des gesamten Fahrrads als vereinfachtes, zweidimensionales Mehrkörpermodell inklusive Aktorik und Regelung erfolgen die weiteren Untersuchungen in beiden Ak-

tivitätssträngen auf oberstem Systemlevel (s = 0). Auch hier erfolgt wiederum eine enge Kopplung der beiden Aktivitätsstränge in Form von Analyse-Synthese-Zyklen. Die entsprechenden geometrischen Größen sind durch die Festlegung auf das Fahrrad vom Typ „Scorpion“ bekannt. Das in Abbildung 6.7 dargestellte Mehrkörpermodell kann in dem vorhandenen Simulationsmodell aus Abbildung 6.5 in SimulationX ergänzt werden, wodurch der Modellierungsaufwand reduziert wird.

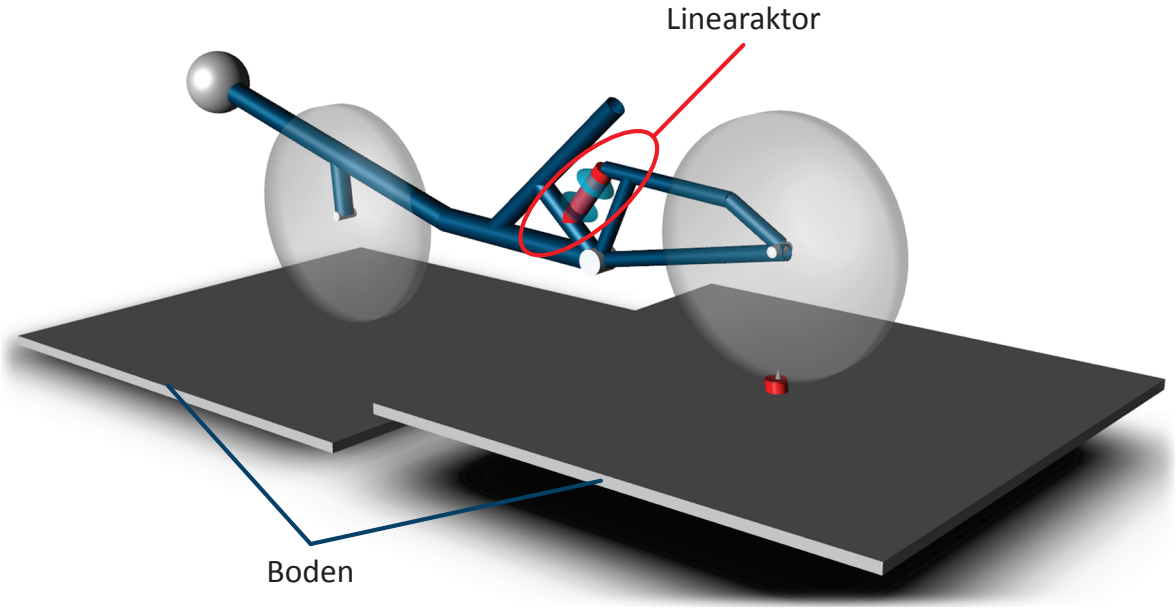


Abbildung 6.7: Vereinfachtes Mehrkörpermodell des Systemkonzeptes

Gleichzeitig wird das Regelkonzept verfeinert, indem ein Skyhook-Regler nach Karnopp et al. [KaCH74] im Modell implementiert wird. Mittels dieses Modells können – neben der nochmaligen Überprüfung der Analysemeilensteine $AMS_{Syskon\_1.1}$ bis $AMS_{Syskon\_2.2}$ – die Analysemeilensteine $AMS_{Syskon\_3.1}$ und $AMS_{Syskon\_3.2}$ überprüft werden. Diese dienen dazu, neue Anforderungen bezüglich Aktorkraft und -geschwindigkeit zu generieren, welche bei der Auswahl eines geeigneten Aktors als Zukaufteil relevant sind. Die Simulationsergebnisse für diese beiden Meilensteine sind in Abbildung 6.8 dargestellt. Neben den direkt ablesbaren Kraftgrößen können aus den Simulationen ebenso die erforderlichen Verfahrgeschwindigkeiten des Aktors bestimmt werden.

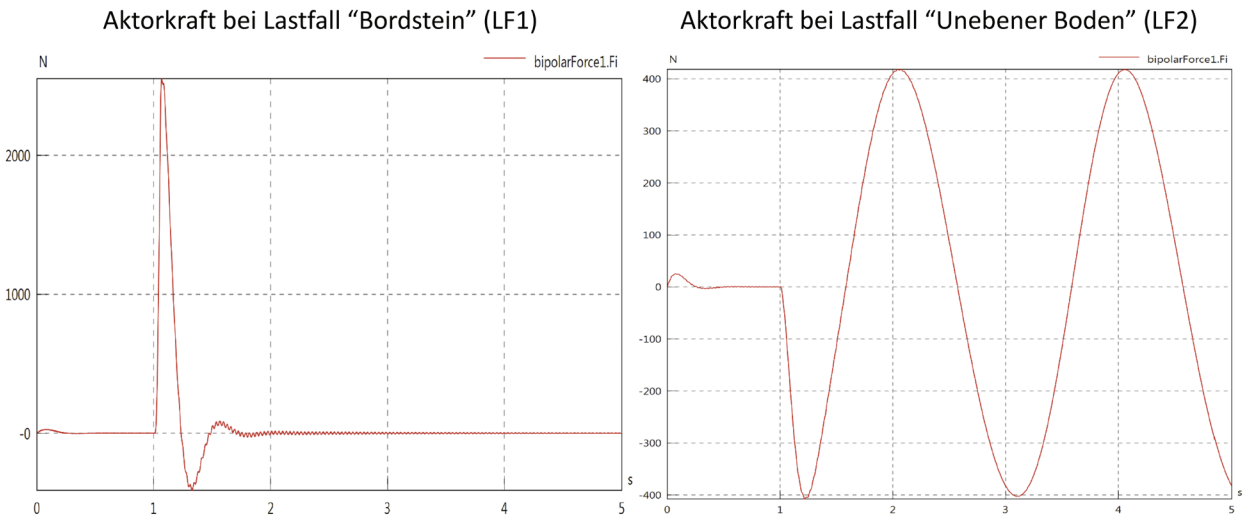


Abbildung 6.8: Ergebnisse der Aktorkraftsimulationen für beide Lastfälle

Mittels dieser neuen Anforderungen an einen Aktor bezüglich Kraft und Geschwindigkeit kann in einer Marktrecherche überprüft werden, ob geeignete Aktoren verfügbar sind. Ist dies nicht der Fall, muss in Mikroiterationen das Systemkonzept angepasst werden. Diese Entscheidung erfolgt wiederum mittels des TOTE-Schemas und der in den Analysemeilensteinen definierten Eigenschaften. In diesem Fall sind entsprechende Aktoren verfügbar und alle Analysemeilensteine erfüllt. Somit ist der zweite Analyse-Synthese-Zyklus abgeschlossen und das Systemkonzept so weit finalisiert, dass die einzelnen Module und Komponenten den spezifischen Domänen, Abteilungen oder Teams zugeordnet werden können. Bei diesem Übergang zur nächsten Prozessphase unterstützt die in Abbildung 6.9 dargestellte Systemstruktur, welche alle Module des Systems, deren Schnittstellen sowie die Verantwortlichkeiten beinhaltet. Darüber hinaus dient diese Systemstruktur auch im weiteren Prozessverlauf als Orientierung für die einzelnen Domänen, um Auswirkungen auf andere Subsysteme und Komponenten einschätzen zu können. Prinzipiell kann eine solche Darstellung auch in SysML mit Blockdefinitionsdiagrammen (BDD) und internen Blockdiagrammen (IBD) modellbasiert erstellt werden (siehe Anhang A.3).

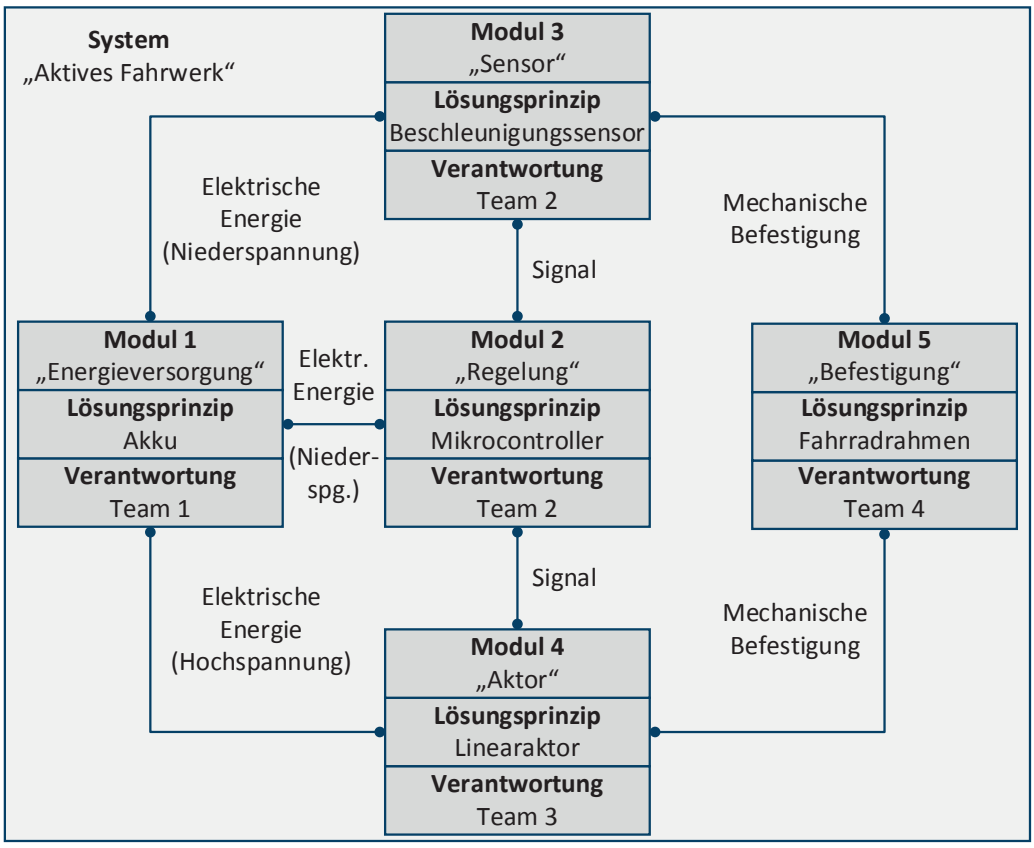


Abbildung 6.9: Systemstruktur mit Modulen sowie deren Schnittstellen

6.3.2.2 Anwendung von Methoden zum Umgang mit Unsicherheiten

In der Systemkonzeptionierung und Systemkonzeptsimulation wird deutlich, dass das System noch einen geringen Reifegrad besitzt und entsprechend viele Merkmale und Parameter noch unbekannt sind. Dieses Phänomen ist typisch für frühe Phasen des Entwicklungsprozesses. Zum Umgang mit solchen Unsicherheiten wurden in Kapitel 5.7 diverse Methoden beschrieben. Grundsätzlich sind diese Methoden im gesamten Entwicklungsprozess relevant, jedoch ist deren Bedeutung aufgrund der angesprochenen Wissenslücken in frühen Phasen besonders ausgeprägt. Aus diesem Grunde wird die Anwendung der Methoden stellvertretend in den frühen Phasen veranschaulicht.

Insbesondere bei Parameterstudien bieten strukturierte Versuchsmethoden, etwa Design of Experiments, im gesamten Entwicklungsprozess die Möglichkeit zur Effizienzsteigerung des Simulationsprozesses. Als Sonderfall kann dabei der Extremalansatz dazu genutzt werden, die Grenzbereiche des Verhaltens zu simulieren. So ist es etwa möglich, den Ladezustand des Akkus in die Simulation einzubinden: In zwei Simulationsläufen wird überprüft, welches Systemverhalten bei entleertem sowie bei geladenem Akku zu erwarten ist. Die mit dem Ladezustand verbundene elektrische Spannung besitzt direkte Auswirkungen auf die Kraft- und Geschwindig-

keitsgrößen des Aktors und damit auf das Gesamtsystem. Somit sind Rückschlüsse auf den zu wählenden Akkutyp und dessen Dimensionierung möglich. Ähnlich lässt sich der Extremalansatz auch bei der Überprüfung der Realisierbarkeit des Systemkonzeptes einsetzen: Basierend auf den Erkenntnissen einer ersten Marktrecherche bezüglich Aktoren lässt sich mittels deren minimalen sowie maximalen Kräften und Geschwindigkeiten das Systemverhalten in seinen Grenzen bezüglich der Erfüllung der Gesamtanforderungen beurteilen.

Große Unsicherheiten treten bei der Simulation von Lastfall LF2 (siehe $AMS_{Syskon\_1.2}$) im Bodenmodell auf: Die Beschreibung von Bodenunebenheiten mittels einer Sinusfunktion stellt eine Vereinfachung dar, wobei insbesondere die Amplitude nur abgeschätzt werden kann. Mittels einer Abschätzung der statistischen Verteilung der Bodenamplitude lässt sich über Monte-Carlo-Simulationen die Verteilung der Aktorkraft simulieren. Damit lässt sich bewerten, für welchen Kraftbereich der Aktor tatsächlich ausgelegt werden muss. Das Vorgehen ist in Abbildung 6.10 für eine abgeschätzte Verteilung der Bodenamplitude verdeutlicht.

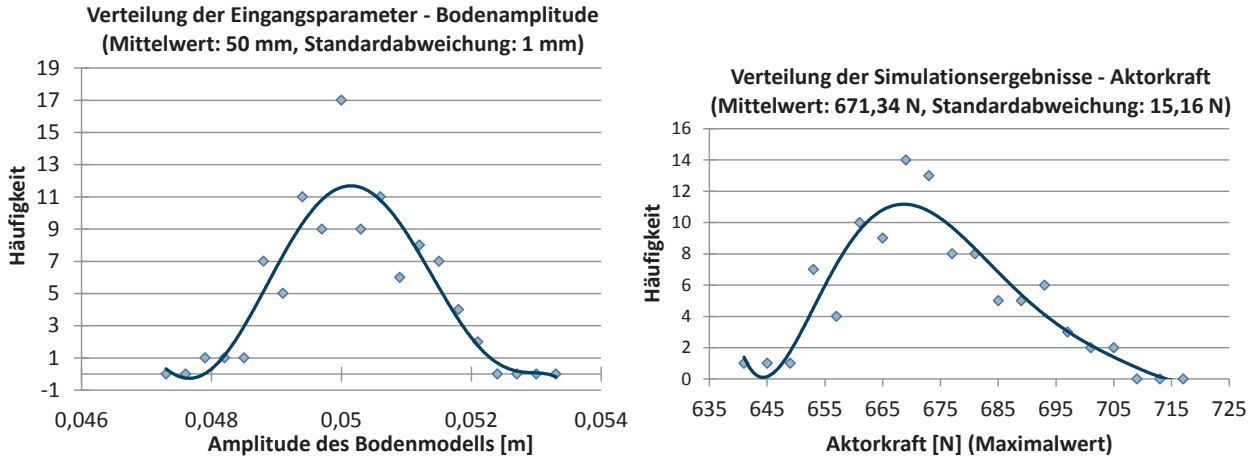


Abbildung 6.10: Beispiel für Monte-Carlo-Simulation – Aktorkraft in Abhängigkeit der Verteilung der Bodenamplitude

Wie bereits in Kapitel 5.7 erläutert, können Simulationen auch bei der Sensitivitätsanalyse eingesetzt werden. Insbesondere bei Systemlevelmodellen, wie in der Systemkonzeptsimulation verwendet, besitzen Modelle zahlreiche Eingangsparameter. Eine vollständige Variation aller Parameter erfordert eine Vielzahl von Simulationen und ist daher oftmals nicht sinnvoll. Am Beispiel des aktiven Fahrwerks kann gezeigt werden, wie mittels Sensitivitätsanalysen die Anzahl zu variierender Parameter reduziert werden kann. Mithilfe eines Plackett-Burmann-Versuchsplans (siehe Anhang A.4) kann der Einfluss von sieben Modellparametern auf die Aktorkraft mit nur acht Simulationsläufen bestimmt werden. Die Ergebnisse der Sensitivitätsana-

lyse sind in Abbildung 6.11 in Form von Einflussgraphen dargestellt. Jeder der dargestellten Graphen stellt eine lineare Interpolation zwischen den Extremwerten der einzelnen Modellparameter dar. Eine geringe Steigung der Graden deutet daher auf einen geringen Einfluss des Parameters auf die Gesamtsimulation, also auf die Aktorkraft, hin. Diese Parameter können für erste Untersuchungen konstant gehalten werden, wodurch sich die erforderliche Anzahl an Simulationen bei Parameterstudien signifikant reduziert.

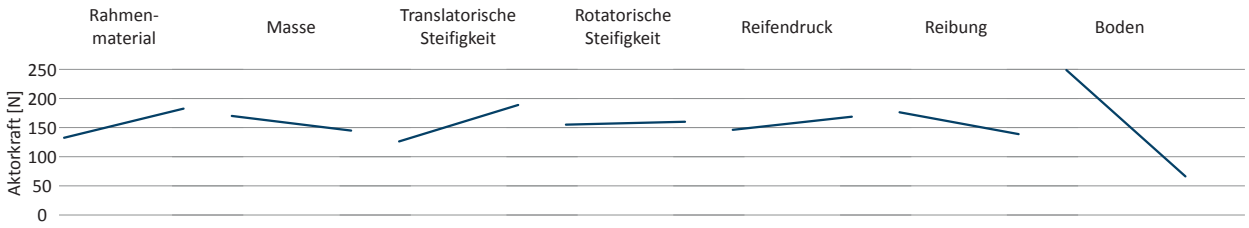


Abbildung 6.11: Ergebnisse der Sensitivitätsanalyse bezüglich der Aktorkraft

6.3.3 Domänenspezifischer Entwurf und detaillierte domänenspezifische Simulation

6.3.3.1 Methodisches Entwerfen und Absichern der domänenspezifischen Lösungen

Gemäß dem Prozessablauf aus Kapitel 5.5 erfolgt nach Auswahl und Detaillierung der Übergang zur domänenspezifischen Entwicklung. Der Prozessablauf in dieser Phase ist in Abbildung 6.12 als Ausschnitt des Gesamtprozesses dargestellt.

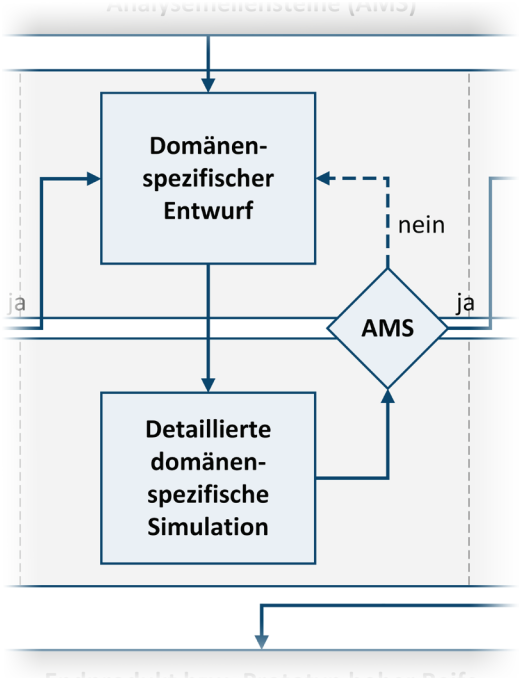


Abbildung 6.12: Prozessablauf im domänenspezifischen Entwurf und der domänenspezifischen Simulation

Basierend auf der Systemstruktur aus Abbildung 6.9 erfolgt eine Aufteilung und Zuweisung der Entwicklungsaufgaben zu den einzelnen Teams, die weitestgehend domänenorientiert aufgeteilt sind. Zudem erfolgt auf Basis der verbesserten Systemkenntnis eine Anpassung und Ergänzung der Analysemeilensteine. Somit ergeben sich folgende Aufgaben für die einzelnen Teams:

- Team 1: Entwurf und Absicherung eines geeigneten Akkus
- Team 2: Detaillierung und Simulation des Reglers
- Team 3: Auswahl und Absicherung eines geeigneten Aktors
- Team 4: Aufbau und Simulation eines detaillierten Mehrkörpermodells sowie Anpassung und strukturmechanische Absicherung der Rahmengeometrie an den gewählten Aktor

Basis dieses Entwicklungsschrittes ist die Auswahl eines geeigneten Aktors durch Team 3. Als Zukaufteil erfordert dieser keine gesonderte Simulation des Aktorverhaltens, vielmehr wird auf die Verhaltensbeschreibung des Herstellers zurückgegriffen. Auch in dieser Phase der Entwicklung erfolgt durch die beiden Aktivitätsstränge die parallele Eigenschaftsabsicherung mittels Simulation. Parallel zu den Tätigkeiten von Team 3 wird von Team 4 ein Mehrkörpermodell in einem spezialisierten Softwarewerkzeug – in diesem Fall Adams der Firma MSC Software – aufgebaut und die kinematische Einbindung des Aktors in diversen Mikroiterationen mittels Simulationen optimiert. Gleichzeitig wird von Team 2 der Entwurf des Skyhook-Reglers ebenfalls in zahlreichen Mikroiterationen in einem domänentypischen Werkzeug detailliert und durch Simulationen abgesichert und optimiert – in diesem Beispiel wird Matlab/Simulink von der Firma The Mathworks eingesetzt. Hierbei muss unter anderem ein überlagerter Regelkreis implementiert werden, der ein Abdriften des Dämpfers aus der Mittellage verhindert. Grundsätzlich ist es bei diesem Vorgehen möglich, dass die beiden Teams unabhängig voneinander arbeiten. In einem solchen Fall wird in dem Mehrkörpermodell ein einfacher Regelkreis implementiert und im Umkehrschluss erfolgt im Reglermodell eine vereinfachte mathematische Beschreibung der Fahrwerkskinematik. Dieses Vorgehen hat bezüglich des Detailgrades der Simulationen und insbesondere bezüglich der Kooperation der Teams Nachteile. Aus diesem Grunde werden an dieser Stelle Co-Simulationen eingesetzt: Über softwareseitig vorhandene Schnittstellen werden die individuellen Modelle der Teams untereinander verknüpft. Somit kann der hohe Detaillierungsgrad beider Modelle genutzt werden und gleichzeitig findet eine enge Kooperation der Teams statt, wodurch der Systemgedanke auch in dieser Phase Anwendung findet. In zahlreichen Mikroiterationen können somit Kinematik und Regler gemeinsam optimiert werden, wo-

bei die Kompatibilität der beiden Teillösungen gewährleistet ist. Die beiden Teilmodelle für die Co-Simulation sind in Abbildung 6.13 dargestellt.

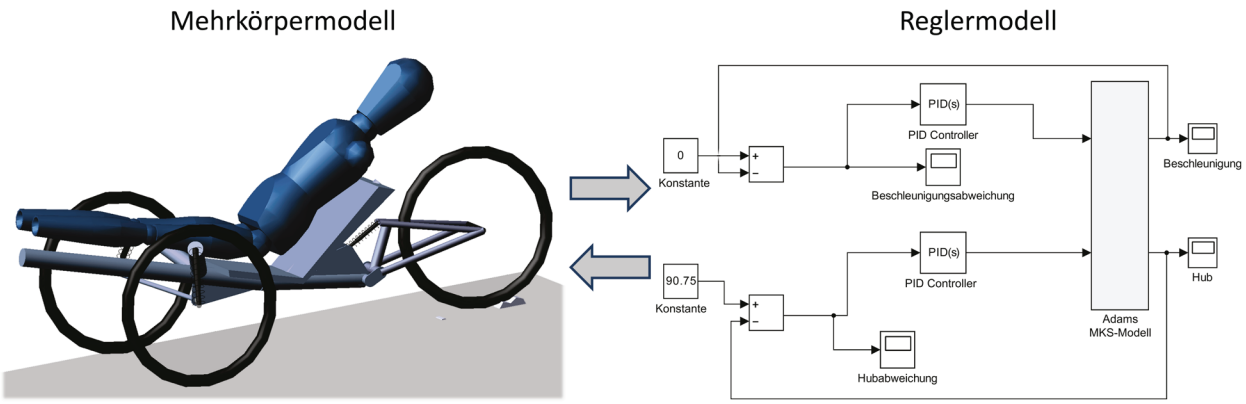


Abbildung 6.13: Co-Simulation von Mehrkörpermodell (links) und Reglermodell (rechts)

Ebenso eng vernetzt erfolgt die Entwicklung mit Team 3, welches für den Aktor verantwortlich ist. Durch die Auswahl eines Aktors werden Anforderungen festgelegt, welche sich auf die Regelung sowie die Rahmengeometrie – und damit auf das Mehrkörpermodell – auswirken. Dies betrifft insbesondere die Einbaumaße sowie die maximale Stellkraft und -zeit des Aktors. Neue Anforderungen werden direkt in Analysenmeilensteinen sowie im Entwurf der anderen Teams eingebaut und durch Simulation auf ihre Realisierbarkeit und Konformität bezüglich der restlichen Anforderungen überprüft. Die Absicherungen erfolgen auch in dieser Phase in Analyse-Synthese-Zyklen und über die entsprechenden Analysemeilensteine. In den detaillierten Simulationen wird deutlich, dass die Anforderungen der zu Beginn definierten Lastfälle mit den verfügbaren Aktoren nicht voll erfüllt werden können. Entsprechend muss in Absprache aller beteiligten Teams eine Anpassung der Anforderungen und der Analysemeilensteine erfolgen. Hierbei werden Simulationen eingesetzt, um die entsprechenden Anforderungen zu definieren. Insbesondere der Bordsteinlastfall, welcher bezüglich der Aktorstellkraft und -geschwindigkeit die höchsten Anforderungen stellt, muss angepasst werden: Die Bordsteinhöhe sowie die Überfahrgeschwindigkeit werden schrittweise reduziert, bis die Simulationsergebnisse realisierbare Kräfte und Geschwindigkeiten erreichen. Diese Anpassung erfolgt somit ebenso in Mikroiterationen. Nachteilig an dieser Lösung ist, dass bei höheren Bordsteinen und höheren Geschwindigkeiten das System die entstehenden Stöße nicht mehr vollständig ausgleichen kann. Sollte dies nicht gewünscht sein, wäre eine Anpassung des gesamten Konzeptes nötig. Eine derartige Änderung wäre nur in der Systemkonzeptionierung möglich, was eine Makroiteration mit entsprechendem Aufwand erfordern würde. In diesem Beispiel werden eine Bordsteinhöhe von 50 mm sowie eine Überfahrgeschwindigkeit von 5 km/h akzeptiert, welche aus der Anpassung der An-

forderungen in Mikroiterationen resultiert. Entsprechend sind keine Makroiterationen erforderlich.

Die Auswahl eines Akkus durch Team 1 erfolgt auf Basis von Anforderungen, die von den anderen Teams generiert werden. In den Analysemeilensteinen sind Simulationen vorgesehen, mit denen der Energieverbrauch über eine definierte Strecke ermittelt wird. Dieser bestimmt, welche Kapazität der Akku besitzen muss. Die Berechnung der Energiemenge pro Strecke kann aus den oben beschriebenen Co-Simulationen abgeleitet werden. Darüber hinaus stellt die Wahl des Aktors die Anforderungen an Spannung und maximale Stromstärke des Akkus. In diesem Fall ist der Akku als Zukaufteil vorgesehen und kann entsprechend dieser drei Größen ausgewählt werden.

Neben der Reglerauslegung verantwortet Team 2 zudem die Sensorauswahl sowie die Programmierung der Regelungshardware und der Benutzerschnittstelle. Sowohl die elektronischen Komponenten als auch die Software sind in diesem Beispiel sehr einfach gehalten, da diese hauptsächlich auf einem Mikrocontroller aufbauen. Daher wird dieser Aspekt hier nicht weiter diskutiert. In Anhang A.3 ist jedoch die Zustands- und Aktivitätsmodellierung des Systems dargestellt.

In der Phase des domänenspezifischen Entwurfs und der detaillierten domänenspezifischen Simulation finden zahlreiche Analyse-Synthese-Zyklen zwischen den beiden Aktivitätssträngen statt, welche mittels Analysemeilensteinen gesteuert und organisiert werden. Die Aufteilung in verschiedene Domänen bedeutet hierbei jedoch nicht, dass die Zyklen der einzelnen Domänen unabhängig sind. Vielmehr müssen die Abhängigkeiten der individuellen Entscheidungen auch in den domänenspezifischen Phasen berücksichtigt werden, wie obige Erläuterungen zeigen. Durch die Berücksichtigung der Kompatibilität der individuellen Lösungen bereits in dieser Phase können Probleme bei der Systemintegration vermieden werden.

Die domänenspezifische Entwicklung ist beendet, sobald alle Module und Komponenten aus Abbildung 6.9 ausgestaltet sind und die Erfüllung der Anforderungen nachgewiesen ist, das heißt wenn alle Analysemeilensteine aus dieser Phase erfüllt sind.

6.3.3.2 Softwarewerkzeug zur Auswahl von Analyseverfahren

Insbesondere in der domänenspezifischen Entwicklung steht eine Vielzahl von Simulationstechniken und -werkzeugen zur Verfügung, weshalb hier die Anwendung der Methode zur Auswahl geeigneter Analyseverfahren aus Kapitel 5.6.2 besonders sinnvoll ist. Entsprechend wird die

Anwendung in diesem Kapitel anhand der strukturmechanischen Auslegung der Aktoraufnahme beschrieben. Wie bereits in Kapitel 5.6.2 erläutert, stellt die dokumentenbasierte Anwendung dieser Methode eine Einschränkung der Benutzerfreundlichkeit sowie der Funktionalität dar. Deshalb wurde ein webbasiertes Softwarewerkzeug entwickelt, welches dieses Auswahlverfahren unterstützt und vereinfacht. Hierbei sind folgende Anforderungen relevant:

- Möglichkeit des Hinzufügens von Simulationstechniken in die Datenbank der Software
- Zuordnung von Merkmalen C_i und Eigenschaften P_j zu den Simulationstechniken
- Möglichkeit der freien Kategorisierung von Simulationstechniken, etwa über Prozessphasen oder über den in Kapitel 5.5.5 beschriebenen Konfidenzfaktor
- Freie Suche nach Simulationstechniken basierend auf Merkmalen, Eigenschaften oder Kategorien

In Abbildung 6.14 ist die Auswahl einer Simulationstechnik mithilfe dieses Softwarewerkzeuges für die strukturmechanische Auslegung der Aktorbefestigung dargestellt. Weitere Darstellungen der Benutzeroberfläche des Programms, unter anderem auch von der Definition von Simulationstechniken, sind in Anhang A.6 zu finden.

Bei der Auswahl werden zuerst die zu bestimmenden Eigenschaften in der Software ausgewählt, in diesem Fall insbesondere die mechanische Spannung. Basierend auf dieser Auswahl erfolgt bereits eine Filterung, wodurch nur noch relevante Größen dargestellt werden – alle weiteren werden in der Benutzeroberfläche der Software ausgegraut. Im nächsten Schritt können entweder Attribute zur Klassifizierung ausgewählt werden oder, wie in diesem Fall, bereits definierte Merkmale. Hier sind unter anderem Geometrie, Material sowie Belastungen bekannt. Im dritten Schritt erfolgt eine Kategorisierung, da in dieser Phase genaue Simulationsergebnisse für die Detailauslegung erforderlich sind. Auf Grundlage dieser Auswahl werden alle in der Datenbank hinterlegten Simulationstechniken dargestellt, die dieser Auswahl genügen[12]. Im konkreten Fall bedeutet dies die Anwendung der Finiten-Elemente-Methode.

[12] Die Reihenfolge, in der die Auswahl erfolgt, ist nicht auf die hier beschriebene beschränkt. Das Softwarewerkzeug schreibt hier keine Abfolge vor.

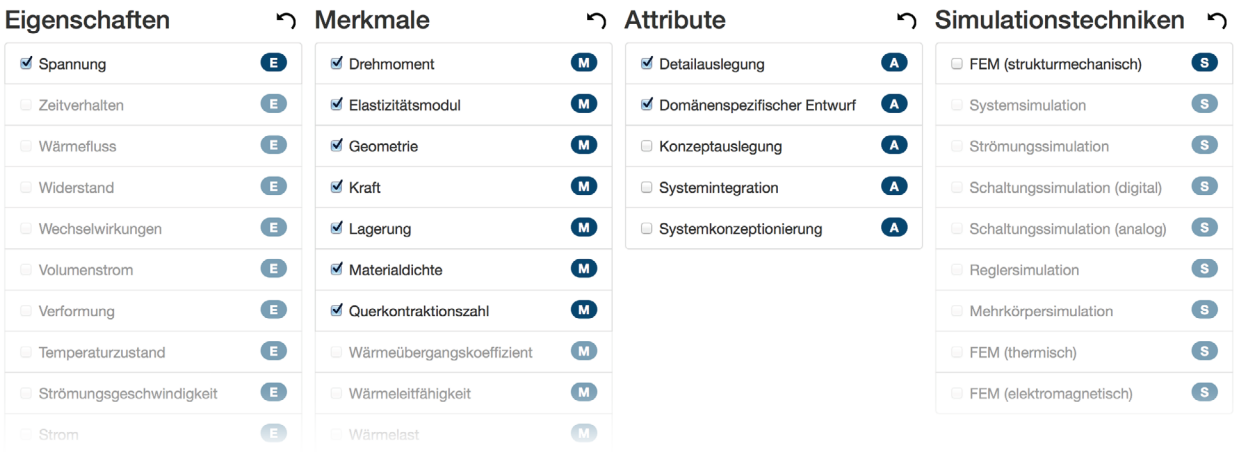


Abbildung 6.14: Auswahl einer Simulationstechnik für strukturmechanische Untersuchungen im entwickelten Softwarewerkzeug

6.3.3.3 Beschreibung von Iterationszyklen

In den zuvor beschriebenen Entwicklungsphasen erfolgten zahlreiche Analyse-Synthese-Zyklen. Dabei werden auf Basis des Abgleichs von Simulatonsergebnissen mit Analysemeilensteinen bei Bedarf Iterationszyklen ausgelöst, die von Analyse- zurück zu Entwurfstätigkeiten führen. Am Beispiel der strukturmechanischen Auslegung der Aktorbefestigung soll in diesem Kapitel der detaillierte Ablauf eines Iterationszyklus, wie er in Kapitel 5.5.5 beschrieben wurde, verdeutlicht werden. Für den statischen Belastungsfall ist im zugehörigen Analysemeilenstein hinterlegt, dass die maximale Spannung im Bauteil 150 MPa nicht überschreiten darf. Im Hinblick auf eine hohe Materialauslastung soll die Spannung jedoch maximal um 15 MPa niedriger sein. Somit ergibt sich nach Gleichung (5.2) folgendes Iterationskriterium[13]:

$$(150\ MPa - 15\ MPa) \leq P_j \leq (150\ MPa + 0\ MPa) \tag{6.1}$$

Angenommen die Interpretation der Simulationsergebnisse zeigt, dass die maximal auftretende von-Mises-Spannung bei 175 MPa liegt. Somit ist das Kriterium aus Gleichung (6.1) nicht erfüllt und eine Iteration muss eingeleitet werden. Die Merkmale, welche die simulierte Spannung bestimmen sind hauptsächlich die Geometrie sowie das Material – unter der Voraussetzung, dass das Modell und die Simulation validiert und verifiziert wurden. Da diese Merkmale in der zu der aktuellen Simulationsphase gehörenden Entwurfsphase definiert wurden, kann hier über Mikroiterationen eine Anpassung erfolgen. Lediglich wenn auch durch die Anpassung dieser

[13] Auf die Einbindung des Konfidenzfaktors gemäß Gleichung (5.5) wird hier aus Gründen der Anschaulichkeit bewusst verzichtet.

Merkmale das Kriterium nicht zu erfüllen ist, muss eine Makroiteration erfolgen, bei der insbesondere die Belastungen reduziert werden müssen. Hierfür ist beispielsweise eine Anpassung der relevanten Merkmale mittels Eingriffen in die Systemfunktionalität erforderlich. Der hier beschriebene Ablauf von Iterationszyklen ist in allen Phasen des Prozesses anwendbar.

6.3.4 Systemintegration und detaillierte Systemsimulation

Die abschließende Phase des Entwicklungsprozesses stellt die Systemintegration und Systemsimulation dar, welche in Abbildung 6.15 als Ausschnitt des Gesamtprozesses dargestellt sind.

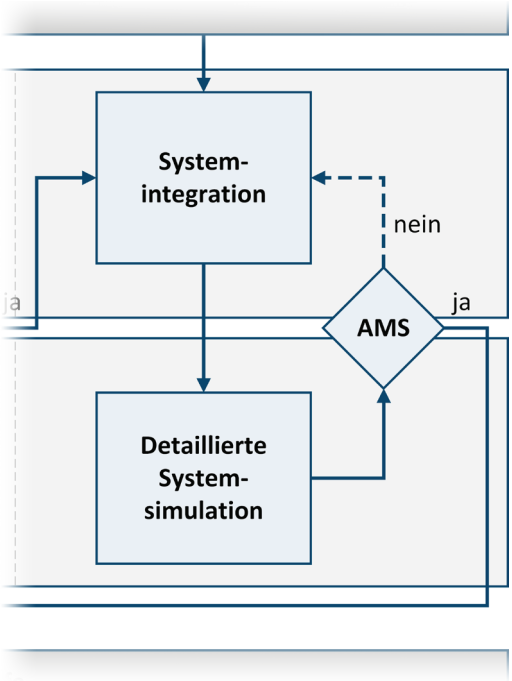


Abbildung 6.15: Prozessablauf in der Systemintegration und Systemsimulation

In einem ideal simulationsbasierten Prozess kommt der Phase der Systemintegration und der damit verbundenen detaillierten Systemsimulation eine verhältnismäßig geringe Rolle zu, da bereits sehr früh im Prozess das Gesamtsystem sowie die Einzellösungen durch Simulationen abgesichert wurden. Im Beispiel des aktiven Fahrwerks wurde auch während der domänenspezifischen Entwicklung die Kooperation zwischen den Domänen durch den Einsatz von Co-Simulationen aufrechterhalten. Somit wird die Kompatibilität der individuellen Lösungen gesichert. In der Phase der Systemintegration kann die Co-Simulation noch dahingehend ausgeweitet werden, dass das Gesamtmodell weiter detailliert wird. Beispielsweise können die Reaktionszeit des Sensors sowie das Entladeverhalten des Akkus implementiert werden. Somit ist die Abstimmung des Systems, insbesondere der Regelung, wesentlich detaillierter möglich. Darüber hinaus kann ein deutlicher Zuwachs der Systemreife nur noch über physische Tests erfol-

gen. Hierfür kann eine Hardware-in-the-Loop-Umgebung aufgebaut werden, in der die Steuerung des Systems an einem virtuellen Modell getestet wird. Alternativ kann ein vollständiger Prototyp aufgebaut werden, anhand dessen das Gesamtsystem in realen Versuchen getestet und abgestimmt wird. Auf eine Beschreibung dieses Vorgehens wird hier verzichtet, da diese für die Erläuterung der simulationsbasierten Methodik keinen direkten Mehrwert bieten würde.

6.4 Erkenntnisse der Validierung

Die Entwicklung des aktiven Fahrwerks für Fahrräder aus diesem Kapitel dient zur Validierung und Veranschaulichung der in dieser Arbeit entwickelten Methodik. Die Strukturierung des Prozessmodells zeigt sich hierbei als adäquate Abbildung des Entwicklungsprozesses: Die Aufteilung in zwei Aktivitätsstränge ermöglicht neben der Abbildung von Mikroiterationen auch die parallele und kontinuierliche Einbindung von Simulationen bereits von der Konzeptphase an. Somit werden bereits sehr früh im Entwicklungsprozess ein hohes Systemverständnis und eine hohe Systemreife erreicht, welche fundierte Konzeptentscheidungen fördern und unterstützen. Hiermit konnten die Hauptanforderungen an die Methodik erfüllt werden.

Die Aufteilung in drei Hauptphasen unterstützt das dem Systems Engineering entnommene Top-Down-Bottom-Up-Vorgehen und fördert damit den interdisziplinären Entwicklungsansatz. Darüber hinaus lässt sich das Konzept der Analysemeilensteine erfolgreich zur Strukturierung des Gesamtprozesses und insbesondere von Analyseaktivitäten anwenden. Das hierfür entwickelte Softwarewerkzeug unterstützt dabei sowohl bei der Planung und Definition als auch bei der Überprüfung der Meilensteine. Die Beschreibung der eng mit Analysemeilensteinen verbundenen Iterationszyklen sowie die hierfür erforderlichen Kriterien konnten ebenso in dem Entwicklungsbeispiel angewendet werden. Zudem konnte deren Nutzen bei der Organisation von Iterationen verdeutlicht werden.

Neben den prozessorientierten Aspekten konnte gezeigt werden, wie durch den Einsatz von Methoden zum Umgang mit Unsicherheiten bereits in frühen Phasen Simulationen zielführend und effizient angewendet werden können.

Bei der Auswahl von Simulationstechniken stellt das hierfür entwickelte Softwarewerkzeug Entwicklern ein hilfreiches und intuitives Mittel zur Verfügung. Im Rahmen des Entwicklungsbeispiels kam dabei ein Datensatz zur Anwendung, der lediglich eine Auswahl gängiger Simulationstechniken beinhaltet.

7 Fazit und Ausblick

7.1 Zusammenfassung der Arbeit

Inhalt dieser Arbeit ist die Entwicklung einer Methodik für die simulationsbasierte Entwicklung mechatronischer Systeme. Im Anschluss an die Klärung relevanter Grundlagen und Begriffe wurde in Kapitel 3 eine ausführliche Analyse des Stands der Technik durchgeführt, sowohl bezüglich Modellierungs- und Simulationstechniken als auch hinsichtlich Entwicklungsmethodiken. Die Diskussion der Erkenntnisse aus den Analysen in Kapitel 4 zeigt, dass auf technologischer Seite bereits eine Vielzahl sehr ausgereifter Möglichkeiten für die Simulation besteht. Dies bezieht sich sowohl auf die Einzeldomänen als auch auf die Mechatronik selbst. Auf methodischer Seite wird jedoch deutlich, dass der durchgängige Einsatz von Simulation in Entwicklungsmethodiken noch nicht verankert ist, weshalb hier keine vollständige Umsetzung simulationsbasierter Entwicklung zu erkennen ist. Aus dieser Lücke zwischen Technologie und Methodik leitet sich der Handlungsbedarf dieser Arbeit ab, die Entwicklung einer simulationsbasierten Methodik.

In Kapitel 5 wurde die Entwicklung einer solchen Methodik beschrieben. Hierfür erfolgten zunächst die Definition eines Gesamtrahmens sowie die Festlegung auf die Kernpunkte dieser Arbeit. Diese umfassen die Beschreibung des Entwicklungsprozesses sowie die Entwicklung von Methoden zur Unterstützung simulationsbasierter Entwicklung. Kern des entwickelten Prozessmodells stellen zwei Aktivitätsstränge dar, welche die kontinuierliche Simulation parallel zu Entwurfstätigkeiten ermöglichen sollen. Eine Gliederung des Entwicklungsprozesses in drei Phasen gemäß der VDI-Richtlinie 2206 [VDI2206] soll ein systemhierarchisch strukturiertes Vorgehen ermöglichen. Über Analysemeilensteine werden der Entwicklungsprozess sowie alle Analysetätigkeiten strukturiert, wodurch diese Meilensteine eine Schnittstelle zwischen dem Prozess und den Methoden bilden. Darüber hinaus wurden Iterationszyklen im Detail beschrieben und Kriterien für Iterationen definiert. Ebenso wurden Simulationsphasen detailliert betrachtet. Neben den prozessnahen Methoden zur Strukturierung und Organisation von Analysen durch Analysemeilensteine sowie zur Steuerung von Iterationszyklen wurden weitere Methoden betrachtet: Zum einen wurde eine Methode zur Auswahl von Analyseverfahren entwickelt, welche Entwickler anwendungsfallspezifisch entlang des gesamten Prozesses unterstützt. Zum anderen wurden, insbesondere für den Einsatz in frühen Phasen, Methoden zum Umgang mit Unsicherheiten bei Simulationen in die Entwicklungsmethodik eingebunden.

In Kapitel 6 erfolgte die Validierung der entwickelten Methodik am Beispiel eines aktiven Fahrwerks für Fahrräder. Hierbei wurde sowohl die Anwendbarkeit des Prozessmodells als auch der entwickelten Methoden gezeigt. Das beschriebene Entwicklungsprojekt veranschaulicht gleichzeitig die Anwendung der gesamten Methodik. Im Rahmen der Validierung wurden zudem Softwarewerkzeuge entwickelt, welche die Prozessstrukturierung mit Analysemeilensteinen sowie die Auswahl geeigneter Analyseverfahren unterstützen.

Als Kernelement wurde in dieser Arbeit der Fokus auf die simulationsbasierte Entwicklung gelegt. Der Einsatz physischer Prototypen wurde daher nur am Rande erwähnt und nicht im Detail erläutert. Die Definition mechatronischer Systeme als Hauptanwendungsfeld der entwickelten Methodik stellt besondere Anforderungen an diese: Aufgrund des interdisziplinären Charakters mechatronischer Systeme müssen die Einzeldomänen im Entwicklungsvorgehen integriert und deren Kommunikation und Kooperation ermöglicht werden. Aus dem Produktlebenszyklus adressiert die Methodik insbesondere den Bereich der Produktentwicklung und Konstruktion, wobei die Produktplanung als vorgeschaltete und der Produktionsanlauf als nachgeschaltete Phase nicht mehr betrachtet wurden. Für die Entwicklung einer Gesamtmethodik erfolgte die Definition von sechs Sichten, von denen im Rahmen dieser Arbeit insbesondere die Prozess- und Methodensichten adressiert wurden.

7.2 Erreichung der Ziele

In Kapitel 1 wurden initiale Forschungsfragen als Grundlage dieser Arbeit aufgestellt. Basierend auf den Erkenntnissen der Analyse des Stands der Technik aus Kapitel 3 konnte der Handlungsbedarf abgeleitet werden, der die Grundlage dieser Arbeit bildet. Mit diesen Erkenntnissen wurden in Kapitel 4.2 die initialen Forschungsfragen detailliert, welche durch die in Kapitel 5 beschriebene Methodik beantwortet werden sollen. Darüber hinaus wurden Anforderungen an die zu entwickelnde Methodik definiert, welche zu deren strukturierter Herleitung dienen.

Nachfolgend soll abschließend verdeutlicht werden, durch welche Elemente der entwickelten Methodik die einzelnen Forschungsfragen und Anforderungen adressiert wurden. Hierzu erfolgt eine Betrachtung der Forschungsfragen aus Kapitel 4.2. Ebenso wird der Bezug zu den jeweils adressierten Anforderungen aus Kapitel 5.2 hergestellt:

A1. Einbindung kontinuierlicher Eigenschaftsanalyse durch Simulation
A2. Frühe Einbindung von Simulationen
A3. Übergang von Makro- zu Mikroiterationen
A4. Bereitstellung von Methoden zur Bestimmung zu analysierender Eigenschaften
A5. Bereitstellung von Methoden zur Auswahl von Werkzeugen zur Simulation
A6. Bereitstellung von Methoden zur Iterationssteuerung
A7. Berücksichtigung des interdisziplinären Charakters der Mechatronik
A8. Möglichkeit zur Bereitstellung von Modellen und der inhärenten Informationen entlang des Entwicklungsprozesses und über Domänen hinweg
A9. Einbindung der Methodik in organisatorische Strukturen
A10. Unterstützung der Methodik durch Daten und Modelle

F1. Wie können Analysen, und insbesondere Simulationen, sinnvoll in den Prozessverlauf eingebunden werden?

Simulationen sind bereits sehr früh und kontinuierlich im Entwicklungsprozess einsetzbar. Dieser Aspekt wurde in dem entwickelten Prozessmodell durch zwei parallele Aktivitätsstränge umgesetzt. Entsprechend der Einteilung des Gesamtprozesses in drei Hauptphasen, welche ein Top-Down-Bottom-Up-Vorgehen ermöglichen, ist auch der Einsatz von Simulationen systemhierarchisch gegliedert. Zudem wird durch diese Einteilung dem interdisziplinären Charakter der Mechatronik auch im Entwicklungsprozess Rechnung getragen. Die Organisation und Strukturierung von Simulationen erfolgt durch das Konzept der Analysemeilensteine.

Durch diesen Ansatz werden die Anforderungen A1, A2 und A7 adressiert.

F2. Wie können Iterationen organisiert und dargestellt werden?

Der Einsatz von Simulationen ermöglicht es, Entwürfe anstatt in wenigen Makroiterationen in vielen Mikroiterationen zu optimieren. In der Methodik dient das Konzept der Analysemeilensteine durch die darin definierten Ziele und Kriterien von Simulationen zur Organisation von Analysen und Iterationen. Darüber hinaus erfolgte eine detaillierte Beschreibung von Iterationszyklen zwischen den beiden Aktivitätssträngen, die sowohl für Makro- als auch Mikroiterationen gültig ist. Hierbei wurden zudem Kriterien definiert, welche bei der Initiierung von Iterationen Unterstützung bieten.

Somit werden die Anforderungen A3 und A6 an die Methodik adressiert.

F3. Wie wirken sich Ergebnisse aus der Simulation auf vorangegangene und nachfolgende Entwurfsaktivitäten aus?

Die Beschreibung von Iterationszyklen beinhaltet neben den Kriterien, die Iterationen auslösen, ebenso die Festlegung, wie Ergebnisse aus Simulationen den Entwurf beeinflussen. Hierzu zählt auch die Bestimmung der Phase, in die eine Iteration zurückführt.

Entsprechend wird hierdurch die Anforderung A6 adressiert.

F4. Zu welchem Zeitpunkt können und müssen welche Eigenschaften analysiert werden?

Das Konzept der Analysemeilensteine dient dazu, bereits im Vorfeld Analysetätigkeiten auf Basis der Anforderungen zu strukturieren. Somit wird festgelegt, wann welche Eigenschaften analysiert werden sollen. Darüber hinaus unterstützt die Methode zur Auswahl von Analysetechniken bei der Wahl geeigneter Techniken.

Hiermit werden die Anforderungen A4 und A5 an die Methodik adressiert.

F5. Wie können Entwickler dabei unterstützt werden, hierfür die passende Simulation zu wählen?

Die zuvor erwähnte Methode zur Auswahl von Analyseverfahren unterstützt Entwickler bei der Wahl geeigneter Analyse- und Simulationstechniken. Zudem kann überprüft werden, welche Voraussetzungen zu erfüllen sind, um eine spezifische Technik anzuwenden. Durch die Implementierung dieser Methode in einem Softwarewerkzeug werden sowohl Anwendbarkeit als auch Funktionalität der Methode verbessert.

Somit wird durch diese Methode und das entsprechende Softwarewerkzeug die Anforderung A5 adressiert.

F6. Wie kann mit Unsicherheiten hinsichtlich Entwurfs- und Modellparametern, insbesondere in frühen Phasen, umgegangen werden?

In der Methodik sind existierende Methoden zum Umgang mit Unsicherheiten eingebettet. Hierdurch wird Entwicklern aufgezeigt, welche dieser Methoden wann und zu welchem Zweck im Entwicklungsprozess anwendbar sind.

Durch den Einsatz dieser Methoden wird die Anwendbarkeit von Simulation in frühen Phasen gefördert, weshalb hiermit die Anforderung A2 adressiert wird.

F7. Wie können Modelle, Simulationen und Simulationsergebnisse entlang des Prozesses und über die Domänen organisiert und verknüpft werden?

In der Methodik wird der durchgängige Einsatz eines Systemmodells empfohlen. Dieses Metamodell beinhaltet alle Informationen über das zu entwickelnde System und soll somit auch Modelle, Simulationen und Simulationsergebnisse bereitstellen und ver-

knüpfen. Über die Konzeptidee hinaus wurde der Einsatz eines solchen Modells jedoch nicht detailliert.

Das Konzept dieses Metamodells adressiert die Anforderungen A8 und A10 an die Methodik.

F8. Welche Softwarewerkzeuge können von Entwicklern für die einzelnen Simulationen genutzt werden?

Die entwickelte Methode zur Auswahl von Analyseverfahren ist dazu geeignet, passende Analyseverfahren für einen bestimmten Anwendungsfall zu finden. Diese Verfahren können jedoch auch so weit detailliert werden, dass konkrete Softwarewerkzeuge vorgeschlagen und ausgewählt werden können.

Wie bereits beschrieben, adressiert diese Methode somit Anforderung A5 an die Methodik.

F9. Wie können die verschiedenen Domänen und deren Simulationen verknüpft werden?

Der interdisziplinäre Charakter der Mechatronikentwicklung ist als Kernelement im Prozessmodell abgebildet: Die Aufteilung in drei Prozessphasen ermöglicht und erfordert interdisziplinäre Zusammenarbeit bei der Systemkonzeptionierung sowie bei der Systemintegration. Die entsprechenden Simulationen erfolgen dabei ebenso auf Systemebene. Obwohl die zweite Phase einen domänenspezifischen Charakter besitzt, wird auch hier gefordert, dass durch Kommunikation und Kooperation die Zusammenarbeit der Domänen gefördert wird. Hierbei soll das Systemmodell als Metamodell unterstützten. Entsprechend werden in dieser Phase idealerweise neben domänenspezifischen auch verknüpfte Simulationen durchgeführt. Das Validierungsbeispiel verdeutlicht dies.

Somit adressiert diese Herangehensweise die an die Methodik gestellte Anforderung A7.

7.3 Ausblick

Aus dem in Kapitel 5.1 beschriebenen Gesamtkonzept für die simulationsbasierte Entwicklung mechatronischer Systeme wurden in dieser Arbeit insbesondere die Prozess- und Methodenaspekte betrachtet. Die ausführliche Untersuchung der restlichen Aspekte bietet das Potential, die hier entwickelte Methodik zu erweitern und bezüglich ihres Anwendungsgebietes in den Bereichen der Modell- und Datenverwaltung (siehe Anforderung A10) zu ergänzen. Diese beiden Bereiche versprechen großes Potential bezüglich der Effizienzsteigerung von Simulationsprozessen. Die aktuellen Bestrebungen der Hersteller von PDM- und Simulationssystemen zur Integration von Simulationsdatenverwaltung zeigen diesen Bedarf. Zudem existieren bereits

Arbeiten, etwa von Eigner et al. [EGDF14], welche die Integration von modell- und simulationsbasierter Entwicklung mit PLM-Systemen adressieren.

Eine detailliertere Betrachtung der Organisationssicht (siehe Anforderung A9) bietet die Möglichkeit, die simulationsbasierte Entwicklung auch in bestehenden Unternehmensstrukturen zu integrieren. Hierfür wäre eine Untersuchung der Abteilungsstrukturen und -verantwortlichkeiten im industriellen Umfeld sinnvoll. Basierend darauf könnten die Verantwortlichkeiten im Rahmen der Methodik verteilt werden oder Empfehlungen zu Strukturanpassungen erfolgen.

Darüber hinaus wurden in dieser Arbeit Konzepte vorgeschlagen, welche für ihre Anwendung weitere Detaillierungen erfordern. Hierzu zählt das Konzept des Metamodells, welches sowohl über Domänen als auch über den Entwicklungsprozess hinweg eine gemeinsame Basis für die Systemstruktur, Simulationsmodelle, Simulationen und Entscheidungen bereitstellen soll. Dieses Konzept adressiert somit bereits ansatzweise die Daten-, Modell- und Werkzeugsichten aus dem Gesamtrahmen der Methodik (siehe Anforderung A10). Insbesondere ist zur Umsetzung des Konzeptes eine Implementierung des Metamodells erforderlich, die sich an den in Kapitel 5.5.2 erwähnten Ansätzen aus der Literatur orientieren kann.

Das beschriebene Verfahren zur Auswahl geeigneter Analyseverfahren sowie das hierfür entwickelte Softwarewerkzeug wurden im Rahmen dieser Arbeit lediglich mit einem Standarddatensatz ausgestattet, der eine Auswahl der gängigsten Simulationstechniken beinhaltet. Sowohl die Methode als auch das Werkzeug könnten vom Ausbau dieses Datensatzes profitieren. Hierfür wäre eine Analyse verschiedener Entwicklungsprojekte hilfreich, wodurch auch anwendungsfallspezifische Techniken integriert werden könnten.

Weiterhin wurden Methoden zum Umgang mit Unsicherheiten bei der Simulation in die Methodik eingebunden, wodurch bereits ein Bezug zur Modell- und Datensicht der Gesamtmethodik hergestellt wurde (siehe Anforderung A10). Dieser Bereich stellt jedoch ein äußerst breites Forschungsfeld dar, zu dem neben den Ingenieurwissenschaften auch die Naturwissenschaften beitragen. Entsprechend bieten sich hier umfangreiche Möglichkeiten, weitere Methoden in eine Gesamtmethodik zu integrieren. Insbesondere für frühe Phasen besteht dabei großes Potential, da diese aufgrund des zu jenem Zeitpunkt geringen Systemwissens von Unsicherheiten geprägt sind.

Aufbauend auf den Erkenntnissen und Erfahrungen, die im Rahmen dieser Arbeit gewonnen wurden, hat sich ein weiterer Forschungsbereich ausgegliedert. Dieser adressiert die Konzept-

generierung von Anforderungsmodellierung über Funktionsmodellierung bis hin zur simulationsbasierten Konzeptabsicherung. Somit wird die Konzeptgenerierung und -simulation, wie sie in dieser Arbeit beschrieben wurde, weiter vertieft und detailliert. Dabei steht die Konsistenz der Modellierung und des Informationsaustausches zwischen Anforderungen, Funktionen und Simulationen im Vordergrund. Entsprechend werden hier alle sechs Sichten aus dem Gesamtrahmen der Methodik betrachtet. Der Handlungsbedarf sowie erste Handlungsschritte wurden bereits in mehreren Veröffentlichungen identifiziert [DEHB14], [EDGV13], zu denen der Autor maßgeblich beigetragen hat.

Durch die Fokussierung dieser Arbeit auf mechatronische Systeme werden unweigerlich auch domänenspezifische Aspekte in die Methodik eingebunden. Somit liegt die Vermutung nahe, dass die Methodik auch auf domänenspezifische Entwicklungsprojekte anwendbar ist. Der Nachweis dieser These erfordert eine entsprechende Validierung sowie eventuell notwendige Detailanpassungen der Methodik. Hierbei kann auf die Erkenntnisse der domänenspezifischen Analysen aus Kapitel 3 zurückgegriffen werden. Aufgrund der Erkenntnisse aus dem Stand der Technik bezüglich des unterschiedlichen Entwicklungsvorgehens der einzelnen Domänen stellt die Untersuchung von Unterschieden bei der domänenspezifischen Anwendung der simulationsbasierten Methodik einen nützlichen und ebenso interessanten Aspekt dar.

Darüber hinaus wurde im Verlaufe der Arbeit bereits darauf hingewiesen, dass die Ausweitung der Methodik auf andere Phasen des Produktlebenszyklus im Sinne des Simultaneous oder Concurrent Engineering weiteres Verbesserungspotential verspricht. Insbesondere die frühe Einbindung und Absicherung von Produktionsaspekten stellt hier einen interessanten Ansatzpunkt dar, der zur weiteren Reduzierung physischer Prototypen beitragen kann.

Abschließend kann festgestellt werden, dass die simulationsbasierte Entwicklung in Zukunft großes Potential besitzt, den Entwicklungsprozess effizienter und effektiver zu gestalten. Zu deren Umsetzung stellt die im Rahmen dieser Arbeit entwickelte Methodik einen Ansatzpunkt dar und bietet zahlreiche Schnittstellen für zukünftige Arbeiten auf diesem Gebiet.

8 Literaturverzeichnis

[AbBo06] Abel, D., Bollig, A.: Rapid Control Prototyping, Springer, Berlin, 2006

[Aber06] Aberdeen Group: Simulation Driven Design Benchmark Report, Aberdeen Group, Inc., Boston, 2006

[Aber08] Aberdeen Group: Engineering Evolved – Getting Mechatronics Performance Right the First Time, Aberdeen Group, Inc., Boston, 2008

[AlNo03] Albers, A., Nowicki, L.: Integration der Simulation in die Produktentwicklung, Simulation in der Produkt- und Prozessentwicklung , Fraunhofer-IRB-Verlag, Stuttgart, 2003

[AnTa06] Ang, A. H.-S., Tang, W. H.: Probability Concepts in Engineering, John Wiley and Sons, Hoboken, 2006

[Anto00] Antony, J.: Design of Experiments for Engineers and Scientists, Butterworth-Heinemann, Burlington, 2000

[Ashe02] Ashenden, P. J.: The designer's guide to VHDL, Morgan Kaufmann, San Francisco, 2002

[AsPT03] Ashenden, P. J., Peterson, G. D., Teegarden, D. A.: The systems designer's guide to VHDL-AMS, Morgan Kaufmann, San Francisco, 2003

[BAGG11] Bauer, F., Anacker, H., Gaukstern, T., Gausemeier, J., Just, V.: Analyzing the Dynamic Behavior of Mechatronic Systems within the Conceptual Design, Proceedings of the 18th International Conference on Engineering Design (ICED11), The Design Society, Glasgow, 2011

[Bath02] Bathe, K.-J.: Finite-Elemente-Methoden, Springer, Berlin, 2002

[Beck00] Beck, K.: Extreme programming eXplained – Embrace change, Addison-Wesley, Reading, 2000

[Bend05] Bender, K.: Embedded Systems, Springer, Berlin, 2005

[BeSZ05] Becker, M. C., Salvatore, P., Zirpoli, F.: The impact of virtual simulation tools on problem-solving and new product development organization, Research Policy, 34(9), 2005, S. 1305-1321

[BGHX07] Bernard, M. L., Glickman, M., Hart, D., Xavier, P., Verzi, S., Wolfenberger, P.: Simulating Human Behavior for National Security Human Interactions, Sandia National Laboratories, U.S. Department of Commerce, Springfield, 2007

[Birk11] Birkhofer, H.: The Future of Design Methodology, Springer, London, 2011

[Bish08] Bishop, R. H.: Mechatronic Systems, Sensors, and Actuators – Fundamentals and Modeling, Taylor & Francis Group, Boca Raton, 2008

[BlCh09] Blessing, L. T. M., Chakrabarti, A.: DRM, a Design Research Methodology, Springer Verlag, Dordrecht, 2009

[Boeh79] Boehm, B. W.: Guidelines for Verifying and Validating Software Requirements and Design Specifications, Proceedings of the European Conference on Applied Information Technology of the International Federation for Information Processing, North-Holland Publishing Company, Amsterdam, 1979

[Boeh81] Boehm, B. W.: Software engineering economics, Prentice-Hall, Englewood Cliffs, 1981

[Boeh86] Boehm, B. W.: A spiral model of software development and enhancement, ACM SIGSOFT Software Engineering Notes, 11(4), 1986, S. 22-42

[Boru00] Borutzky, W.: Bondgraphen – Eine Methodologie zur Modellierung multidisziplinärer dynamischer Systeme, SCS-Europe BVBA, Ghent, 2000

[Brad10] Bradley, D.: Mechatronics – More questions than answers, Mechatronics, 20(8), 2010, S. 827-841

[Brus96] Van Brussel, H. M. J.: Mechatronics – A Powerful Concurrent Engineering Framework, IEEE/ASME Transactions on Mechatronics, 1(2), 1996, S. 127-136

[BuMV07] Burr, H., Mueller, M., Vielhaber, M.: EIMS – A framework for engineering process analysis, Proceedings of ICED 2007 - The 16th International Conference on Engineering Design, Design Society, Glasgow, 2007

[Cell91] Cellier, F. E.: Continuous system modeling, Springer, Ney York, 1991

[ChLe96] Chan, L. W., Leung, T. P.: Spiral Design Model for Consumer Mechatronic Products, Mechatronics, 6(1), 1996, S. 35-51

[CoSt09] Coleman, H. W., Steele, W. G.: Experimentation, Validation, and Uncertainty Analysis for Engineers, John Wiley and Sons, Hoboken, 2009

[CWPH08] Commerell, W., Mammen, H.-T., Panreck, K., Haase, J.: Simulation technischer Systeme – Anforderungen und Perspektiven, Advances in simulation for production and logistics applications, Fraunhofer-IRB-Verlag, Stuttgart, 2008

[DEHB14] Dohr, F., Eisenbart, B., Huwig, C., Blessing, L., Vielhaber, M.: Software support for the consistent transition from requirements to functional modeling to system simulation, Proceedings of the 10th Norddesign, Aalto University, Helsinki, 2014

[DeSi05] De Silva, C. W.: Mechatronics – An integrated approach, CRC Press, Boca Raton, 2005

[DIN66001] Deutsches Institut für Normung (DIN): DIN 66001 – Informationsverarbeitung – Sinnbilder und ihre Anwendung, Beuth, Berlin, 1983

[Döbe08] Döbler, T.: Simulation und Visualisierung in der Produktentwicklung, FAZIT-Schriftenreihe, Band 12, MFG Stiftung Baden-Württemberg, Stuttgart, 2008

[DoD01] United States Departement of Defense: Systems Engineering Fundamentals, Defense Acquisition University Press, Fort Belvoir, 2001

[DoVi12a] Dohr, F., Vielhaber, M.: Toward Simulation-based Mechatronic Design, Proceedings of the 12th International Design Conference – DESIGN 2012, D. Marjanovic, M. Štorga, N. Pavkovic, N. Bojcetic (Eds.), Dubrovnik, University of Zagreb/The Design Society, 2012, S. 411-420

[DoVi12b] Dohr, F., Vielhaber, M.: Enabling simulation-based mechatronic design by shifting of activities, Proceedings of the 9th Norddesign, Center for Industrial Production, Aalborg, 2012

[DoVi13] Dohr, F., Vielhaber, M.: Process Model for Simulation-based Mechatronic Design, DETC2013-13116, Proceedings of the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (DETC2013), American Society of Mechanical Engineers, New York, 2013

[DoVi14a] Dohr, F., Vielhaber, M.: Formalized description of a framework for simulation-based mechatronic design, Proceedings of the 24rd CIRP Design Conference, Procedia CIRP, Elsevier, 2014

[DoVi14b] Dohr, F., Vielhaber, M.: Simulation-based Concept Generation for Mechatronic Systems, Proceedings of the 13th International Design Conference – DESIGN 2014, D. Marjanovic, M. Štorga, N. Pavkovic, N. Bojcetic (Eds.), Dubrovnik, University of Zagreb/The Design Society, 2014, S. 1271-1282

[DoVi14c] Dohr, F., Vielhaber, M.: Using system-level simulation in early mechatronic design stages, Proceedings of the 10th Norddesign, Aalto University, Helsinki, 2014

[DuCh00] Du, X., Chen, W.: Methodology for Managing the Effect of Uncertainty in Simulation-Based Design, AIAA Journal, 38(8), 2000, S. 1471-1478

[DuPr93] Dubois, D., Prade, H.: Fuzzy Sets: A Survey of Engineering Applications, Computers & Chemical Engineering, 17(1), 1993, S. 373-380

[EDGV13] Eisenbart, B., Dohr, F., Gericke, K., Vielhaber, M., Blessing, L.: Potentials for realising a consistent transition between functional modelling with the IFM framework and early system simulation, Proceedings of the 19th International Conference on Engineering Design (ICED13), The Design Society, Glasgow, 2013

[EGDF14] Eigner, M., Gilz, T., Denger, A., Fritz, J.: Applicability of model-based system lifecycle management for cyber-physical systems, Proceedings of the 14th Mechatronics Forum International Conference Mechatronics 2014, Karlstads universitet, Karlstad, 2014, S. 294-302

[Ehrl09] Ehrlenspiel, K.: Integrierte Produktentwicklung, Hanser, München, 2009

[EiGB13] Eisenbart, B., Gericke, K., Blessing, L.: An analysis of functional modeling approaches across disciplines, AIEDAM, 27(3), 2013, S. 281-289

[EiGZ12] Eigner, M., Gilz, T., Zafirov, R.: Proposal for functional product description as part of a PLM solution in interdisciplinary product development, Proceedings of the 12th International Design Conference – DESIGN 2012, D. Marjanovic, M. Štorga, N. Pavkovic, N. Bojcetic (Eds.), Dubrovnik, University of Zagreb/The Design Society, 2012, S. 1667-1676

[EKBA08] Erden, M. S., Komoto, H., Van Beek, T. J., D'Amelio, V., Echavarria, E., Tomiyama, T.: A review of function modeling: Approaches and applications, AIEDAM, 22(2), 2008, S. 147-169

[ElMa97] Elmqvist, H., Mattsson, S. E.: An introduction to the physical modeling language Modelica, Proceedings of the 9th European Simulation Symposium ESS'97, The Society for Computer Simulation International, Passau, 1997

[Erke28] Erkens, A.: Beiträge zur Konstruktionserziehung, VDI-Zeitschrift, Band 72, 1928, S. 17-21

[Este08] Estefan, J. A.: Survey of Model-Based Systems Engineering (MBSE) Methodologies, ModelBased Systems Engineering (MBSE) Initiative, International Council on Systems Engineering (INCOSE), INCOSE-TD-2007-003-01, Seattle, 2008

[FAA06] United States Federal Aviation Administration (FAA), National Airspace System (NAS): System Engineering Manual, 2006

[FeGr13] Feldhusen, J., Grote, K.-H.: Pahl/Beitz Konstruktionslehre, Springer, Berlin, 2013

[FePe08] Ferziger, J. H., Perić, M.: Numerische Strömungsmechanik, Springer, Berlin, 2008

[FHPR11] Follmer, M., Hehenberger, P., Punz, S., Rosen, R., Zeman, K.: Approach for the creation of mechatronic system models, Proceedings of the 18th International Conference on Engineering Design (ICED11), The Design Society, Glasgow, 2011

[FHPZ10] Follmer, M., Hehenberger, P., Punz, S., Zeman, K.: Using SysML in the product development process of mechatronic systems, Proceedings of the 11th International Design Conference DESIGN 2010, D. Marjanovic, M. Štorga, N. Pavkovic, N. Bojcetic (Eds.), Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb und Design Society, Zagreb und Glasgow, 2010

[FoHZ12] Follmer, M., Hehenberger, P., Zeman, K.: Model-Based Approach for the Reliability Prediction of Mechatronic Systems on the System-Level, Computer aided systems theory – EUROCAST 2011, Springer, Berlin New York, 2012

[Frit04] Fritzson, P. A.: Principles of Object-Oriented Modeling and Simulation with Modelica 2.1, Wiley-IEEE Press, Piscataway, 2003

[GaBi06] Gausemeier, J., Bigl, T.: Integrative Entwicklung räumlicher elektronischer Baugruppen, Hanser, München, 2006

[GAGS09] Gajski, D. D., Abdi, S., Gerstlauer, A., Schirner, G.: Embedded System Design – Modeling, Synthesis and Verification, Springer, Dordrecht, 2009

[GaKu83] Gajski, D. D., Kuhn, R. H.: New VLSI Tools, Computer, 16(12), 1983, S. 11-14

[GeKa04] Gero, J. S., Kannengiesser, U.: The situated function-behaviour-structure framework, Design Studies, 25(4), 2004, S. 373-391

[GeKL06] Geimer, M., Krüger, T., Linsel, P.: Co-Simulation, gekoppelte Simulation oder Simulatorkopplung?, O+P Ölhydraulik und Pneumatik, 50(11-12), 2006, S. 572-576

[Gero90] Gero, J. S.: Design Prototypes: A Knowledge Representation Schema for Design, AI Magazine, 11(4), 1990, S. 26-36

[GFDK09] Gausemeier, J., Frank, U., Donoth, J., Kahl, S.: Specification technique for the description of self-optimizing mechatronic systems, Research in Engineering Design, 20(4), 2009, S. 201-223

[GiVa01] Girault, C., Valk, R.: Petri Nets for Systems Engineering – A Guide to Modelling, Verification, and Applications, Springer, New York, 2003

[GJSS09] Gamweger, J., Jöbstl, O., Strohrmann, M., Suchowerskyj, W.: Design for Six Sigma – Kundenorientierte Produkte und Prozesse fehlerfrei entwickeln, Carl Hanser Verlag, München, 2009

[GMPB04] Gausemeier, J., Müller, W., Paelke, V., Bauch, J., Shen, Q., Radkoswki, R.: Virtual Prototyping of Self-optimizing Mechatronic Systems, Proceedings of the 8th International Design Conference – DESIGN 2004, D. Marjanović (Ed.), Dubrovnik, University of Zagreb/The Design Society, Glasgow, 2004

[Hare87] Harel, D.: Statecharts: a visual formalism for complex systems, Science of Computer Programming, 8(3), 1987, S. 231-274

[Hask11] Haskins, C.: Systems engineering handbook - A guide for system life cycle processes and activities, International Council on Systems Engineering (INCOSE), San Diego, 2011

[HaTF96] Harashima, F., Tomizuka, M., Fukuda, T.: Mechatronics – „What Is It, Why, and How?“ An Editorial, IEEE/ASME Transactions on Mechatronics, 1(1), 1996, S. 1-4

[HeEr07] Heißing, B., Ersoy, M.: Fahrwerkhandbuch – Grundlagen, Fahrdynamik, Komponenten, Systeme, Mechatronik, Perspektiven, Friedr. Vieweg & Sohn Verlag, GWV Fachverlage GmbH, Wiesbaden, 2007

[Henk07] Henke, H.: Elektromagnetische Felder – Theorie und Anwendung, Springer, Berlin, 2007

[HJSS06] Helton, J. C., Johnson, J. D., Sallaberry, C. J., Storlie, C. B.: Survey of Sampling-Based Methods for Uncertainty and Sensitivity Analysis, Sandia National Laboratories, Albuquerque, 2006

[Huan02] Huang, M.: Funktionsmodellierung und Lösungsfindung mechatronischer Produkte, Dissertation, Shaker Verlag, Aachen, 2002

[HuEd88] Hubka, V., Eder, W. E.: Theory of Technical Systems – A Total Concept Theory for Engineering Design, Springer, Berlin, 1988

[HuEd92] Hubka, V., Eder, W. E.: Einführung in die Konstruktionswissenschaft – Übersicht, Modell, Ableitungen, Springer, Berlin, 1992

[HuRG07] Huang, E., Ramamurthy, R., McGinnis, L. F.: System and simulation modeling using SysML, Proceedings of the 2007 Winter Simulation Conference, Association for Computing Machinery, New York, 2007

[Huwi13] Huwig, C.: Modellierung und Simulation eines aktiven Fahrwerks am Zwei-Reifen-Modell, Studienarbeit, Lehrstuhl für Konstruktionstechnik, Universität des Saarlandes, Saarbrücken, 2013

[HWFV12] Haberfellner, R., de Weck, O. L., Fricke E., Vössner, S.: Systems Engineering - Grundlagen und Anwendung, Orell Füssli, Zürich, 2012

[IEEE97] Institute of Electrical and Electronics Engineers: IEEE Trial-Use Recommended Practice for Distributed Interactive Simulation – Verification, Validation, and Accreditation (1278.4-1997), IEEE, Piscataway, 1998

[Iser06] Isermann, R.: Fahrdynamik-Regelung – Modellbildung, Fahrerassistenzsysteme, Mechatronik, Friedr. Vieweg & Sohn Verlag, GWV Fachverlage GmbH, Wiesbaden, 2006

[Iser08] Isermann, R.: Mechatronische Systeme – Grundlagen, Springer-Verlag, Berlin Heidelberg, 2008

[Iser99] Isermann, R.: Mechatronische Systeme – Grundlagen, Springer Verlag, Berlin, 1999

[ISO1219] International Organization for Standardization (ISO): ISO 1219 Fluid power systems and components – Graphical symbols and circuit diagrams, ISO, Genf, 2012

[IsSS99] Isermann, R., Schaffnit, J., Sinsel, S.: Hardware-in-the-loop simulation for the design and testing of engine-control systems, Control Engineering Practice, 7(5), 1999, S. 643-653

[Jans10] Janschek, K.: Systementwurf mechatronischer Systeme – Methoden – Modelle – Konzepte, Springer, Berlin, 2010

[JaSa05] Janzen, D., Saiedian, H.: Test-driven development concepts, taxonomy, and future direction, Computer, 38(9), 2005, S. 43-50

[JKPB08] Johnson, T., Kerzhner, A., Paredis, C. J. J., Burkhart, R.: Integrating Models and Simulations of Continuous Dynamics Into SysML, Proceedings of the 6th International Modelica Conference, The Modelica Association und University of Applied Sciences Bielefeld, Bielefeld, 2008

[JuLa13] Jung, M., Langer, U.: Methode der finiten Elemente für Ingenieure – Eine Einführung in die numerischen Grundlagen und Computersimulation, Springer Vieweg, Wiesbaden, 2013

[KaCH74] Karnopp, D., Crosby, M. J., Harwood, R. A.: Vibration Control Using Semi-Active Force Generators, Journal of Engineering for Industry, 96(2), 1974, S. 619-626

[KAPM11] Kerley, W., Armstrong, G., Pepe, C., Moss, M., Clarkson, P. J.: Using simulation to support process integration and automation of the early stages of aerospace design, Proceedings of the 18th International Conference on Engineering Design (ICED11), The Design Society, Glasgow, 2011

[KBSS97] Kallenbach, E., Birli, O., Saffert, E., Schäffel, C.: Zur Gestaltung integrierter mechatronischer Produkte, VDI Berichte 1315: Mechatronik im Maschinen- und Fahrzeugbau, VDI-Verlag, Düsseldorf, 1997, S. 1-14

[KeMa14] Kelkel, D., Martini, T.: SysML Modellierung eines aktiven Fahrwerks in IBM Rhapsody / TopCased, Seminararbeit, Lehrstuhl für Konstruktionstechnik, Universität des Saarlandes, Saarbrücken, 2014

[KiDi09] Kiureghian, A. D., Ditlevsen, O.: Aleatory or Epistemic? Does It Matter?, Structural Safety, 31(2), 2009, S. 105-112

[Klei07] Klein, B.: FEM – Grundlagen und Anwendungen der Finite-Element-Methode im Maschinen- und Fahrzeugbau, Vieweg, Wiesbaden, 2007

[Klep11] Kleppmann, W.: Taschenbuch Versuchsplanung – Produkte und Prozesse optimieren, Carl Hanser Verlag, München, 2011

[KlKr13] Kleiner, S., Kramer, C.: Model Based Design with Systems Engineering Based on RFLP Using V6, Smart Product Engineering – Proceedings of the 23rd CIRP Design Conference, Springer, Berlin, 2013

[KLSL13] Karlberg, M., Löfstrand, M., Sandberg, S., Lundin, M.: State of the art in simulation-driven design, International Journal of Product Development, 18(1), 2013, S. 68-87

[Koss11] Kossiakoff, A.: Systems engineering - Principles and practice, Wiley-Interscience, Hoboken, 2011

[KrFG07] Krause, F.-L., Franke, H.-J., Gausemeier, J.: Innovationspotenziale in der Produktentwicklung, Carl Hanser Verlag, München, 2007

[KrMW12] Krüger, D., Miehling, J., Wartzack, S.: A Simplified Approach Towards Integrating Biomechanical Simulations into Engineering Environments, Proceedings of the 9th Norddesign, Center for Industrial Production, Aalborg, 2012

[KüHW98] Kümmel, M. A., Henke, A., Wallaschek, J.: A development strategy for mechatronic systems based on functional and geometrical modelling techniques, Mechatronics '98, Elsevier, Burlington, 1998

[KuSc05] Kumar, V., Schuhmacher, M.: Fuzzy Uncertainty Analysis in System Modelling, European Symposium on Computer-Aided Process Engineering-15, L. Puigjaner, A. Espuna (Eds.), Elsevier, Amsterdam, 2005

[KuZi04] Kundert, K. S., Zinke, O.: The designer's guide to Verilog-AMS, Kluwer Academic Publishers, 2004

[KyOh96] Kyura, N., Oho, H.: Mechatronics—an industrial perspective, IEEE/ASME Transactions on Mechatronics, 1(1), 1996, S. 10-15

[Lang09] Langermann, R.: Beitrag zur durchgängigen Simulationsunterstützung im Entwicklungsprozess von Flugzeugsystemen, Dissertation, Technische Universität Braunschweig, Braunschweig, 2009

[LaOe13] Laurien, E., Oertel jr., H.: Numerische Strömungsmechanik – Grundgleichungen und Modelle - Lösungsmethoden - Qualität und Genauigkeit, Springer Vieweg, Wiesbaden, 2013

[LeWS94] Lehmann, G., Wunder, B., Selz, M.: Schaltungsdesign mit VHDL – Synthese, Simulation und Dokumentation digitaler Schaltungen, Franzis Verlag, Poing, 1994

[Lien06] Lienig, J.: Layoutsynthese elektronischer Schaltungen – Grundlegende Algorithmen für die Entwurfsautomatisierung, Springer, Berlin, 2006

[LiMB09] Lindemann, U., Maurer, M., Braun, T.: Structural complexity management, Springer, Berlin, 2009

[Lind07] Lindemann, Udo: Methodische Entwicklung technischer Produkte, Springer, Berlin, 2007

[LüKS00] Lückel, J., Koch, T., Schmitz, J.: Mechatronik als integrative Basis für innovative Produkte, VDI Berichte 1533: Mechatronik - Mechanisch-Elektrische Antriebstechnik, VDI-Verlag, Düsseldorf, 2000, S. 1-26

[MaBu11] Mammen, H.-T., Buschmann, U.: Mechatronik Entwicklung Optimieren, MECHATRONIK, Ausgabe 11-12, 2011, S. 51-54

[McBC00] Mckay, M. D., Beckman, R. J., Conover, W. J.: A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code, Technometrics, 42(1), 2000, S. 55-61

[MeKr06] Merkt, T., Krastel, M.: Virtuelle Produktabsicherung auf Basis eines CAE Datenmanagements, Berechnung und Simulation im Fahrzeugbau, VDI-Verlag, Düsseldorf, 2006

[MiGP60] Miller, G. A., Galanter, E., Pribram, K. H.: Plans and the structure of behavior, Holt, Rinehart and Winston, Inc., New York, 1960

[MiHa06] Miles, R., Hamilton, K.: Learning UML 2.0, O'Reilly, Sebastopol, 2006

[Mode12] Modelica Association: Modelica® – A Unified Object-Oriented Language for Systems Modeling – Language Specification, Modelica Association, http://www.modelica.org, Linköping, 2012

[MöGa02] Möhringer, S., Gausemeier, J.: An Interface Specification for Principle Solutions Supporting the Cross-Domain Design of Mechatronic Systems, Proceeding of the 7th International Design Conference DESIGN 2002, M. M. Andreasen, D. Marjanovic, H. Birkhofer (Eds.), Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb und Design Society, Zagreb und Glasgow, 2002

[Möhr04] Möhringer, S.: Entwicklungsmethodik für mechatronische Systeme, Heinz Nixdorf Institut, Universität Paderborn, Paderborn, 2004

[MöSt10] Möhringer, S., Stetter, R.: A Research Framework for Mechatronic Design, Proceedings of the 11th International Design Conference DESIGN 2010, D. Marjanovic, M. Štorga, N. Pavkovic, N. Bojcetic (Eds.), Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb, Dubrovnik, 2010

[NaAn13] Nattermann, R., Anderl, R.: The W-Model – Using Systems Engineering for Adaptronics, Procedia Computer Science, Jahrgang 16, 2013, S. 937-946

[Noll09] Nollau, R.: Modellierung und Simulation technischer Systeme, Springer, Berlin Heidelberg, 2009

[ODRD02] Oberkampf, W. L., DeLand, S. M., Rutherford, B. M., Diegert, K. V., Alvin, K. F.: Error and Uncertainty in Modeling and Simulation, Reliability Engineering and System Safety, 75(3), 2002, S. 333-357

[ODRD99] Oberkampf, W. L., DeLand, S. M., Rutherford, B. M., Diegert, K. V., Alvin, K. F.: A New methodology for the Estimation of Total Uncertainty in Computational Simulation, Proceedings of the 40th Structures, Structural Dynamics, and Materials Conference and Exhibit, American Institute of Aeronautics and Astronautics, Reston, 1999

[OHJW04] Oberkampf, W. L., Helton, J. C., Joslyn, C. A., Wojtkiewicz, S. F., Ferson, S.: Challenge Problems: Uncertainty in System Response Given Uncertain Parameters, Reliability Engineering and System Safety, 85(1-3), 2004, S. 11-19

[OMG11a] Object Management Group (OMG): Unified Modeling Language (OMG UML) – Infrastructure, www.omg.org/spec/UML/2.4.1, 2011

[OMG11b] Object Management Group (OMG): Unified Modeling Language (OMG UML) – Superstructure, www.omg.org/spec/UML/2.4.1, 2011

[Paet06] Paetzold, K.: Ansätze für eine funktionale Repräsentation multidisziplinärer Produkte, Design for X – Beiträge zum 17. Symposium, Universität Erlangen-Nürnberg, Lehrstuhl Konstruktionstechnik, Erlangen, 2006

[PaSu93] Parsaei, H. R., Sullivan, W. G.: Concurrent Engineering – Contemporary issues and modern design tools, Chapman & Hall, London, New York, 1993

[PBFG07] Pahl, G., Beitz, W., Feldhusen, J., Grote, K.-H.: Konstruktionslehre, Springer-Verlag, Berlin, 2007

[PDSK01] Paredis, C. J. J., Diaz-Calderon, A., Sinha, R., Khosla, P. K.: Composable Models for Simulation-Based Design, Engineering with Computers, 17(2), 2001, S. 112-128

[PeLV06] Pecheux, F., Lallement, C., Vachoux, A.: VHDL-AMS and Verilog-AMS as alternative hardware description languages for efficient modeling of multidiscipline systems, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 24(2), 2005, S. 204-225

[Pelz03] Pelz, G.: Mechtronic systems – Modelling and simulation with HDLs, John Wiley and Sons, Chichester, 2003

[Petr62] Petri, K. A.: Kommunikation mit Automaten, Dissertation, Darmstadt, 1962

[PlTT89] Plitt, K., Thomson, B., Truckenbrodt, A.: Hydraulische Stellzylinder und aktive Rad-Federsysteme am PKW, VDI Berichte 787: Kontrollierte Bewegungen – Mechatronik im Maschinen- und Fahrzeugbau, VDI-Verlag, Düsseldorf, 1989, S. 85-100

[PoPr04] Pomberger, G., Pree, W.: Software-Engineering, Hanser, München, 2004

[Reif11] Reif, K.: Bosch Autoelektrik und Autoelektronik, Vieweg + Teubner Verlag, Wiesbaden, 2011

[Reis13] Reisig, W.: Understanding Petri Nets, Springer, Berlin, 2013

[ReMo54] Reuleaux, F., Moll, C. L.: Constructionslehre für den Maschinenbau, Vieweg, Braunschweig, 1854

[RiSc10] Rill, G., Schaeffer, T.: Grundlagen und Methodik der Mehrkörpersimulation, Vieweg + Teubner, Wiesbaden, 2010

[Rodd06] Roddeck, W.: Einführung in die Mechatronik, Teubner, Wiesbaden, 2006

[Rodd13] Roddeck, W.: Grundprinzipien der Mechatronik – Modellbildung und Simulation mit Bondgraphen, Springer Vieweg, Wiesbaden, 2013

[Ross10] Ross, T. J.: Fuzzy Logic with Engineering Applications, John Wiley and Sons, Chichester, 2010

[Roth00] Roth, K.: Konstruieren mit Konstruktionskatalogen, Springer, Berlin, 2000

[Royc70] Royce, W. W.: Managing the Development of Large Software Systems, Technical Papers of Western Electronic Show and Convention (WesCon), WesCon, Los Angeles, 1970

[Rump11] Rumpe, B.: Modellierung mit UML – Sprache, Konzepte und Methodik, Springer, Berlin Heidelberg, 2011

[RVPK01] Rajarishi, S., Vei-Chung, L., Paredis, C. J. J., Khosla, P. K.: Modeling and simulation methods for design of engineering systems, Journal of Computing and Information Science in Engineering, 1(1), 2001, S. 84-91

[SaHy06] Sandler, S. M., Hymowitz, C. E.: SPICE circuit handbook, McGraw-Hill, New York, 2006

[Sarg99] Sargent, R. G.: Validation and verification of simulation models, Proceedings of the Winter Conference on Simulation 1999, IEEE Press, Piscataway, 1999

[SBOW04] Shephard, M. S., Beall, M. W., O'Bara, R. M., Webster, B. E.: Toward simulation-based design, Finite Elements in Analysis and Design, 40(12), 2004, S. 1575-1598

[Schi97] Schiehlen, W.: Multibody System Dynamics: Roots and Perspectives, Multibody System Dynamics, 1(2), 1997, S. 149-188

[Schö99] Schön, A.: Konzept und Architektur eines Assistenzsystems für die mechatronische Produktentwicklung, Universität Erlangen-Nürnberg, Erlangen, 2000

[SCHS01] Schwarz, P., Clauß, C., Haase, J., Schneider, A.: VHDL-AMS und Modelica – ein Vergleich zweier Modellierungssprachen, Simulationstechnik: 15. Symposium in Paderborn, Society for Computer Simulation International, Delft, 2001

[Schw13] Schwarze, R.: CFD-Modellierung – Grundlagen und Anwendungen bei Strömungsprozessen, Springer Vieweg, Berlin, 2013

[Schw89] Schweitzer, G.: Mechatronik – Aufgaben und Lösungen, VDI Berichte 787: Kontrollierte Bewegungen – Mechatronik im Maschinen- und Fahrzeugbau, VDI-Verlag, Düsseldorf, 1989, S. 1-15

[ScZu06] Schäuffele, J., Zurawka, T.: Automotive Software-Engineering, Vieweg, Wiesbaden, 2006

[SeBC11] Seppäla, M., Buda, A., Coatanéa, E.: Selection of Design Concepts Using Virtual Prototyping in the Early Design Phases, Proceedings of the 18th International Conference on Engineering Design (ICED11), The Design Society, Glasgow, 2011

[Sell99] Sellgren, U.: Simulation-Driven Design – Motives, Means, and Opportunities, Department of Machine Design, The Royal Institute of Technology, KTH Tryck och Kopiering, Stockholm, 1999

[SeRa08] Seiffert, U., Rainer, G.: Virtuelle Produktentstehung für Fahrzeug und Antrieb im Kfz, Vieweg + Teubner Verlag, Wiesbaden, 2008

[SHAC07] Shishko, R., Aster, R., Cassingham, R. C.: NASA Systems engineering handbook, National Aeronautics and Space Administration, Washington, 2007

[ShKo11] Shetty, D., Kolk, R. A.: Mechatronics System Design, Cengage Learning, Stamford, 2011

[Sieg04] Siegl, J.: Schaltungstechnik - Analog und gemischt analog, Springer-Verlag, Berlin, 2005

[SiVH10] Siebertz, K., van Bebber, D., Hochkirchen, T.: Statistische Versuchsplanung – Design of Experiments (DoE), Springer Verlag, Heidelberg, 2010

[SKKR10] Stark, R., Krause, F.-L., Kind, C., Rothenburg, U., Müller P., Hayka, H., Stöckert, H.: Competing in Engineering Design – the Role of Virtual Product Creation, CIRP Journal of Manufacturing Science and Technology, 3(3), 2010, S. 175-184

[Somm11] Sommerville, I.: Software engineering, Pearson, Boston, 2011

[SSCV11] Stetter, R., Seemüller, H., Chami, M., Voos, H.: Interdisciplinary system model for agent-supported mechatronic design, Proceedings of the 18th International Conference on Engineering Design (ICED11), The Design Society, Glasgow, 2011

[Stac73] Stachowiak, H.: Allgemeine Modelltheorie, Springer, Wien, New York, 1973

[Suh90] Suh, N. P.: The principles of design, Oxford University Press, New York, 1990

[SySu10] Syal, G., Suyam-Welakwe, N. A.: A new methodology to network CAE and describe its process, Proceedings of the 8th Workshop on Integrated Product Development (IPD 2010), Otto-von-Guericke-Universität, Magdeburg, 2010

[ThFu00] Thomke, S., Fujimoto, T.: The Effect of "Front-Loading" Problem-Solving on Product Development Performance, Journal of Product Innovation Management, 17(2), 2000, S. 128-142

[ThMo08] Thomas, D. E., Moorby, P. R.: The Verilog hardware description language, Springer, Berlin 2008

[Thun03] Thunnissen, D. P.: Uncertainty Classification for the Design and Development of Complex Systems, Proceedings of the 3rd Annual Predictive Methods Conference, Veros Software, Santa Ana, 2003

[Till01] Tiller, M.: Introduction to physical modeling with Modelica, Kluwer Academic Publishers, Boston, 2001

[UlEp08] Ulrich, K. T., Eppinger, S. D.: Product Design and Development, McGraw-Hill Higher Education, Boston, 2008

[Ullm10] Ullman, D. G.: The mechanical design process, McGraw-Hill Higher Education, Boston, 2010

[VDI2206] Verein Deutscher Ingenieure: VDI-Richtlinie 2206 – Entwicklungsmethodik für mechatronische Systeme, VDI-Verlag, Düsseldorf, 2004

[VDI2221] Verein Deutscher Ingenieure: VDI-Richtlinie 2221 – Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte, VDI-Verlag, Düsseldorf, 1993

[VDI2222] Verein Deutscher Ingenieure: VDI-Richtlinie 2222 – Konstruktionsmethodik, VDI-Verlag, Düsseldorf, 1997

[VDI2223] Verein Deutscher Ingenieure: VDI-Richtlinie 2223 – Methodisches Entwerfen technischer Produkte, VDI-Verlag, Düsseldorf, 2004

[VDI2422] Verein Deutscher Ingenieure: VDI-Richtlinie 2422 – Entwicklungsmethodik für Geräte mit Steuerung durch Mikroelektronik, VDI-Verlag, Düsseldorf, 1994

[VDI3633] Verein Deutscher Ingenieure: VDI-Richtlinie 3633 – Simulation von Logistik-, Materialfluss- und Produktionssystemen, VDI-Verlag, Düsseldorf, 2000

[ViBC10] Vielhaber, M., Bergsjö, D., Catic, A.: Mechatronic Systems Engineering – Theory and Automotive Practice, Proceedings of the 11th International Design Conference DESIGN 2010, D. Marjanovic, M. Štorga, N. Pavkovic, N. Bojcetic (Eds.), Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb, Dubrovnik, 2010

[VWBZ09] Vajna, S., Weber, C., Bley, H., Zeman, K.: CAx für Ingenieure – Eine praxisbezogene Einführung, Springer, Berlin, 2009

[WaCK07] Wanke, S., Conrad, J., Köhler, C.: Verhaltensbeschreibende Produktkataloge – Ein Anwendungspotential der Solution Patterns des CPM/PDD-Ansatzes, Design for X – Beiträge zum 18. Symposium, Universität Erlangen-Nürnberg, Lehrstuhl für Konstruktionstechnik, Erlangen, 2007

[Wall07] Wall, J.: Simulation-driven Design of Complex Mechanical and Mechatronic Systems, Dissertation, Blekinge Institute of Technology, Karlskrona, 2007

[Wall95] Wallaschek, J.: Modellierung und Simulation als Beitrag zur Verkürzung der Entwicklungszeiten mechatronischer Produkte, Simulation in der Praxis – Neue Produkte effizienter entwickeln, VDI Berichte 1215: Simulation in der Praxis – Neue Produkte effizienter entwickeln, VDI-Verlag, Düsseldorf, 1995, S. 35-50

[WeAn09] Wendt, J. F., Anderson, J. D.: Computational fluid dynamics, Springer, Berlin, 2009

[Webe05] Weber, C.: CPM/PDD – An Extended Theoretical Approach to Modelling Products and Product Development Processes, Proceedings of the 2nd German-Israeli Symposium on Advances in Methods and Systems for Development of Products and Processes, Fraunhofer-IRB-Verlag, Stuttgart, 2005, S. 159-179

[Webe07] Weber, C.: Improving Product Development by Design-for-X (DfX) Support, The Future of Product Development – Proceedings of the 17th CIRP Design Conference, Springer, Berlin, 2007

[Weil08] Weilkiens, T.: Systems Engineering mit SysML/UML, dpunkt.verlag, Heidelberg, 2008

[WeWD03] Weber, C., Werner, H., Deubel, T.: A different view on Product Data Management/Product Life-Cycle Management and its future potentials, Journal of Engineering Design, 14(4), 2003, S. 447-464

[WeWe00] Weber, C., Werner, H.: Klassifizierung von CAX-Werkzeugen für die Produktentwicklung auf der Basis eines neuartigen Produkt- und Prozessmodells, Design for X – Beiträge zum 11. Symposium, Friedrich-Alexander-Universität Erlangen-Nürnberg, Nürnberg, 2000, S. 126-143

[WiMo98] Wittler, G., Moritz, W.: Mechatronic Design Methods and Software in Mechanical Engineering, Beiträge zum 9. Symposium Fertigungsgerechtes Konstruieren, Lehrstuhl für Konstruktionstechnik, Universität Erlangen-Nürnberg, Erlangen, 1998

[Woer11] Woernle, C.: Mehrkörpersysteme – Eine Einführung in die Kinematik und Dynamik von Systemen starrer Körper, Springer, Berlin, 2011

[Wöge43] Wögerbauer, H.: Die Technik des Konstruierens, Oldenbourg-Verlag, München, 1943

[WuER11] Wunram, K., Eckstein, L., Rettweiler, P.: Potenzial aktiver Fahrwerke zur Erhöhung der Fahrsicherheit von Motorrädern, Berichte der Bundesanstalt für Straßenwesen, Fahrzeugtechnik Heft F 81, Wirtschaftsverlag NW Verlag für neue Wissenschaft GmbH, Bremerhaven, 2011

[WWW1] Statistisches Bundesamt: Bruttoinlandsprodukt 2013 für Deutschland, http://www.destatis.de, letzter Aufruf: 27.02.2014

[WWW2] Bundesministerium für Wirtschaft und Energie: Fakten zum deutschen Außenhandel 2012, http://www.bmwi.de:, letzter Aufruf: 27.02.2014

[WWW3] Yaskawa America, Inc, http://www.yaskawa.com/site/products.nsf/staticPagesNewWindow/mechatronics.html, letzter Aufruf: 24.02.2014

[WWW4] TU Ilmenau, http://studieren.de/course-profile.0.mechatronik-tu-ilmenau.150.413.html, letzter Aufruf: 27.02.2014

[WWW5] Parametric Technology Corporation (PTC), http://de.ptc.com/product/creo, letzter Aufruf: 05.09.2014

[WWW6] Dassault Systèmes, http://www.3ds.com/de/produkte-und-services/catia, letzter Aufruf: 05.09.2014

[WWW7] Dassault Systèmes, http://www.3ds.com/products-services/simulia/portfolio/abaqus/overview, letzter Aufruf: 05.09.2014

[WWW8] Altair, http://www.altairhyperworks.de, letzter Aufruf: 05.09.2014

[WWW9] MSC Software, http://www.mscsoftware.com/de/product/adams, letzter Aufruf: 05.09.2014

[WWW10] SIMPACK AG, http://www.simpack.com, letzter Aufruf: 05.09.2014

[WWW11] OpenCFD Ltd, http://www.openfoam.com, letzter Aufruf: 05.09.2014

[WWW12] ANSYS, Inc., http://www.ansys.com/Products/Simulation+Technology/Fluid+Dynamics/Fluid+Dynamics+Products/ANSYS+Fluent, letzter Aufruf: 05.09.2014

[WWW13] Art Systems Software GmbH, http://www.fluidsim.de, letzter Aufruf: 05.09.2014

[WWW14] Altium Limited, http://www.altium.com/en/products/altium-designer/overview, letzter Aufruf: 06.10.2014

[WWW15] CadSoft Computer GmbH, http://www.cadsoft.de/eagle-pcb-design-software/?language=de, letzter Aufruf: 06.10.2014

[WWW16] OrCAD, http://www.orcad.com/products/orcad-ee-pspice-designer/overview, letzter Aufruf: 05.09.2014

[WWW17] ANSYS, Inc., http://www.ansys.com/Products/Simulation+Technology/Systems+&+Multiphysics/Multiphysics+Enabled+Products/ANSYS+Maxwell, letzter Aufruf: 05.09.2014

[WWW18] The MathWorks, Inc., http://www.mathworks.de/products/matlab, letzter Aufruf: 05.09.2014

[WWW19] Object Management Group, Inc. (OMG), http://www.uml.org, letzter Aufruf: 20.08.2014

[WWW20] The Eclipse Foundation, https://www.eclipse.org, letzter Aufruf: 05.09.2014

[WWW21] IBM Deutschland GmbH, http://www-03.ibm.com/software/products/de/ratirhapfami, letzter Aufruf: 05.09.2014

[WWW22] The MathWorks, Inc., http://www.mathworks.de/products/simulink, letzter Aufruf: 05.09.2014

[WWW23] Maplesoft, http://www.maplesoft.com/products/maple/, letzter Aufruf: 05.09.2014

[WWW24] iXtronics GmbH, http://www.ixtronics.com, letzter Aufruf: 05.09.2014

[WWW25] SysML Open Source Specification Project, http://www.sysml.org, letzter Aufruf: 20.08.2014

[WWW26] Modelica and the Modelica Association https://www.modelica.org, letzter Aufruf: 21.07.2014

[WWW27] Open Source Modelica Consortium, https://www.openmodelica.org, letzter Aufruf: 05.09.2014

[WWW28] Dassault Systèmes, http://www.3ds.com/products-services/catia/capabilities/systems-engineering/modelica-systems-simulation/dymola, letzter Aufruf: 05.09.2014

[WWW29] ITI Gesellschaft für ingenieurtechnische Informationsverarbeitung mbH, http://www.iti.de, letzter Aufruf: 05.09.2014

[WWW30] The MathWorks, Inc., http://www.mathworks.de/products/simscape, letzter Aufruf: 05.09.2014

[WWW31] Controllab Products, http://www.20sim.com, letzter Aufruf: 05.09.2014

[WWW32] Dassault Systèmes, http://www.3ds.com/products-services/catia/capabilities/systems-engineering/, letzter Aufruf: 05.09.2014

[WWW33] HP Velotechnik OHG, http://www.hpvelotechnik.com/images/presse/scorpionfs/Scorpion_fs_links_grau_bl.jpg, letzter Aufruf: 29.09.2014

[WWW34] Bose GmbH, http://www.bose.de/DE/de/automotive/innovations/suspension-system/the-system/, letzter Aufruf: 01.10.2014

A Anhang

A.1 Vorlage Analysemeilensteine

Tabelle A.1: Vorlage zur Definition von Analysemeilensteinen

Analysemeilenstein	Nr.	Domäne	Revision	Datum
AMS__				

Prozessschritt	Verantwort-lichkeit	Zu analysierende Eigenschaft				Analyse-Verfahren	Analysebe-dingung
		P_j	Wert für P_j	Grenzwer-te ΔP_j	Beschrei-schrei-bung		

A.2 Softwarewerkzeug zur Erstellung und Verwaltung von Analysemeilensteinen

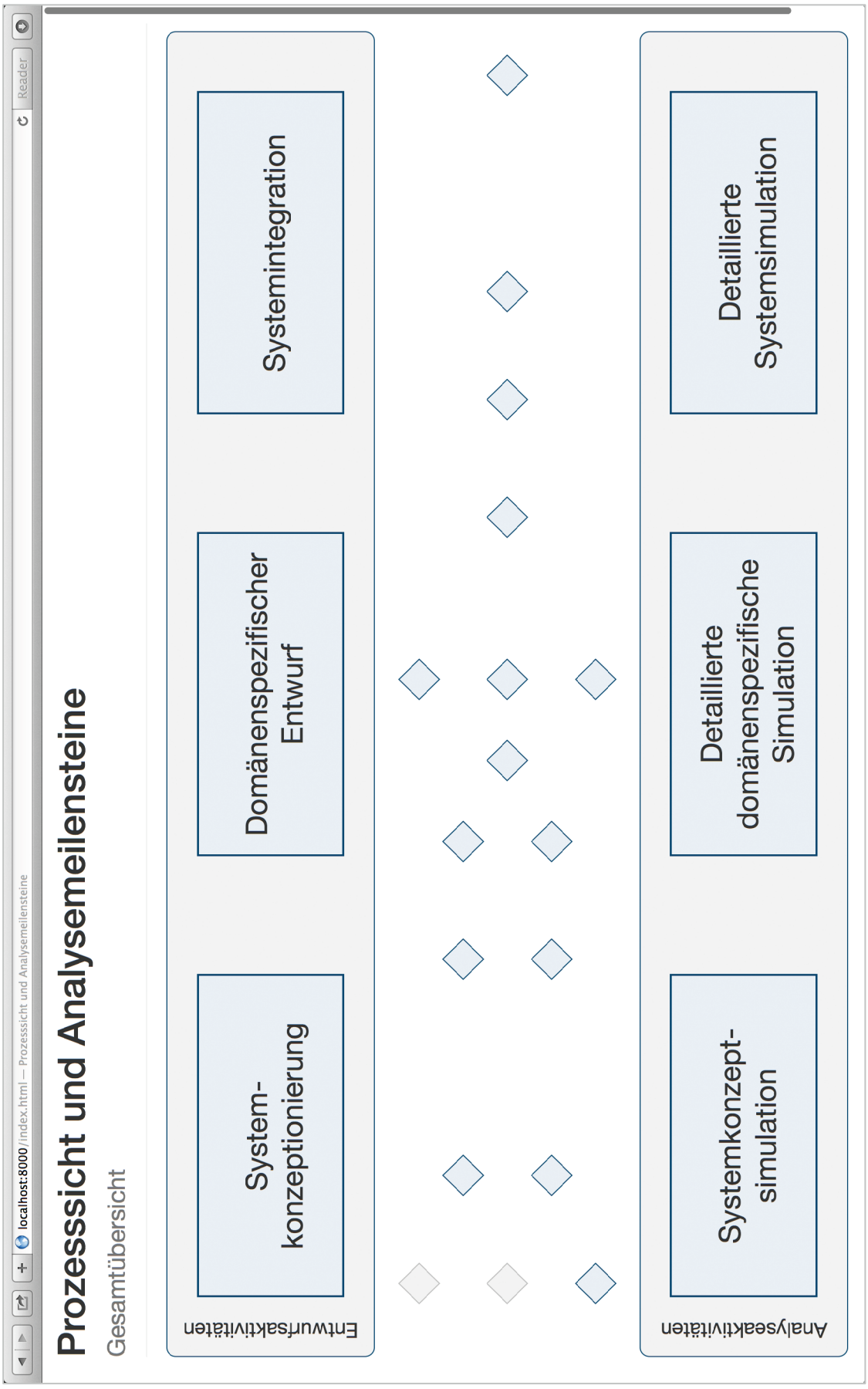


Abbildung A.1: Gesamtprozessübersicht mit Analysemeilensteinen

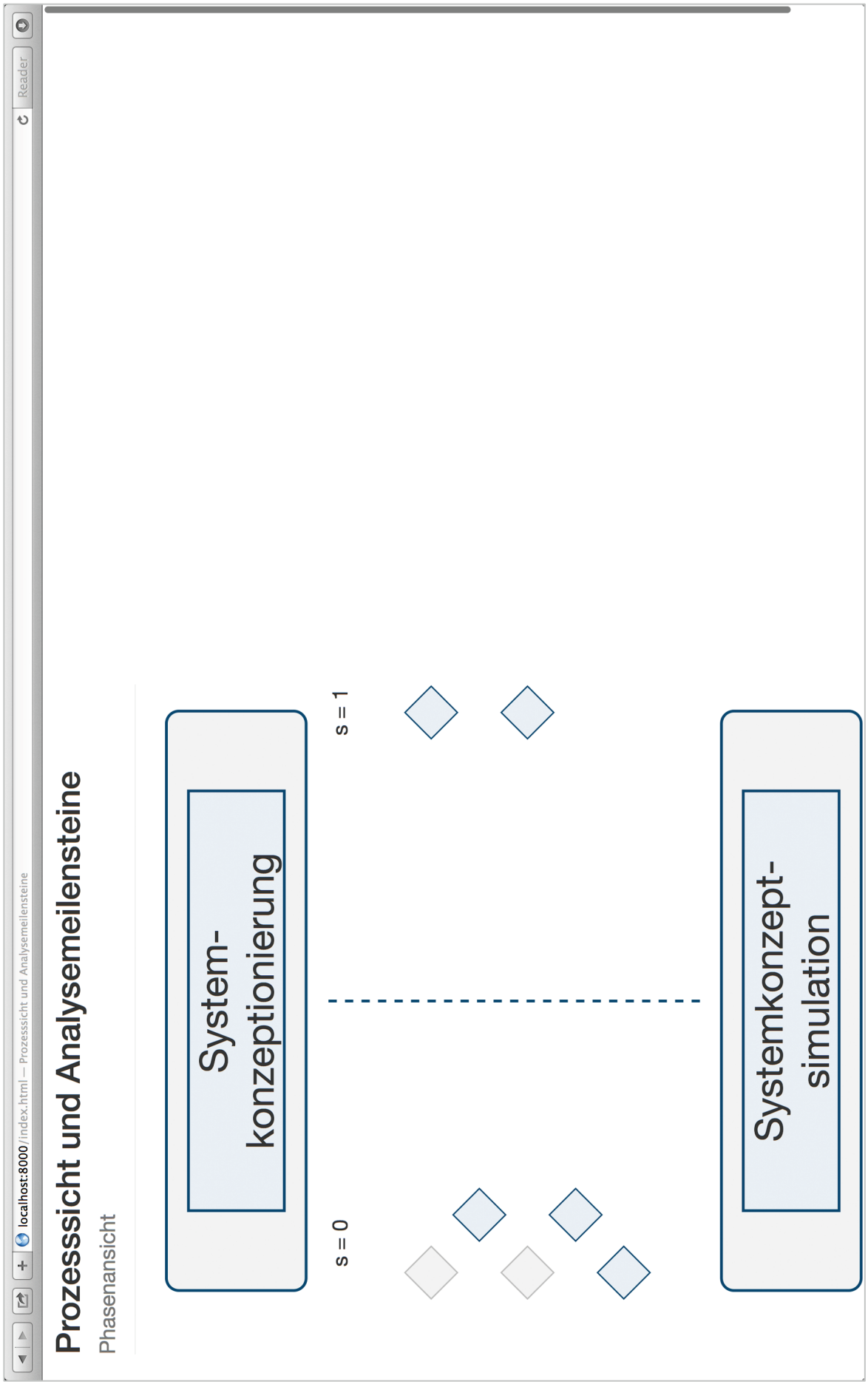


Abbildung A.2: Detailansicht von Phasen inklusive Aufteilung nach Systemleveln

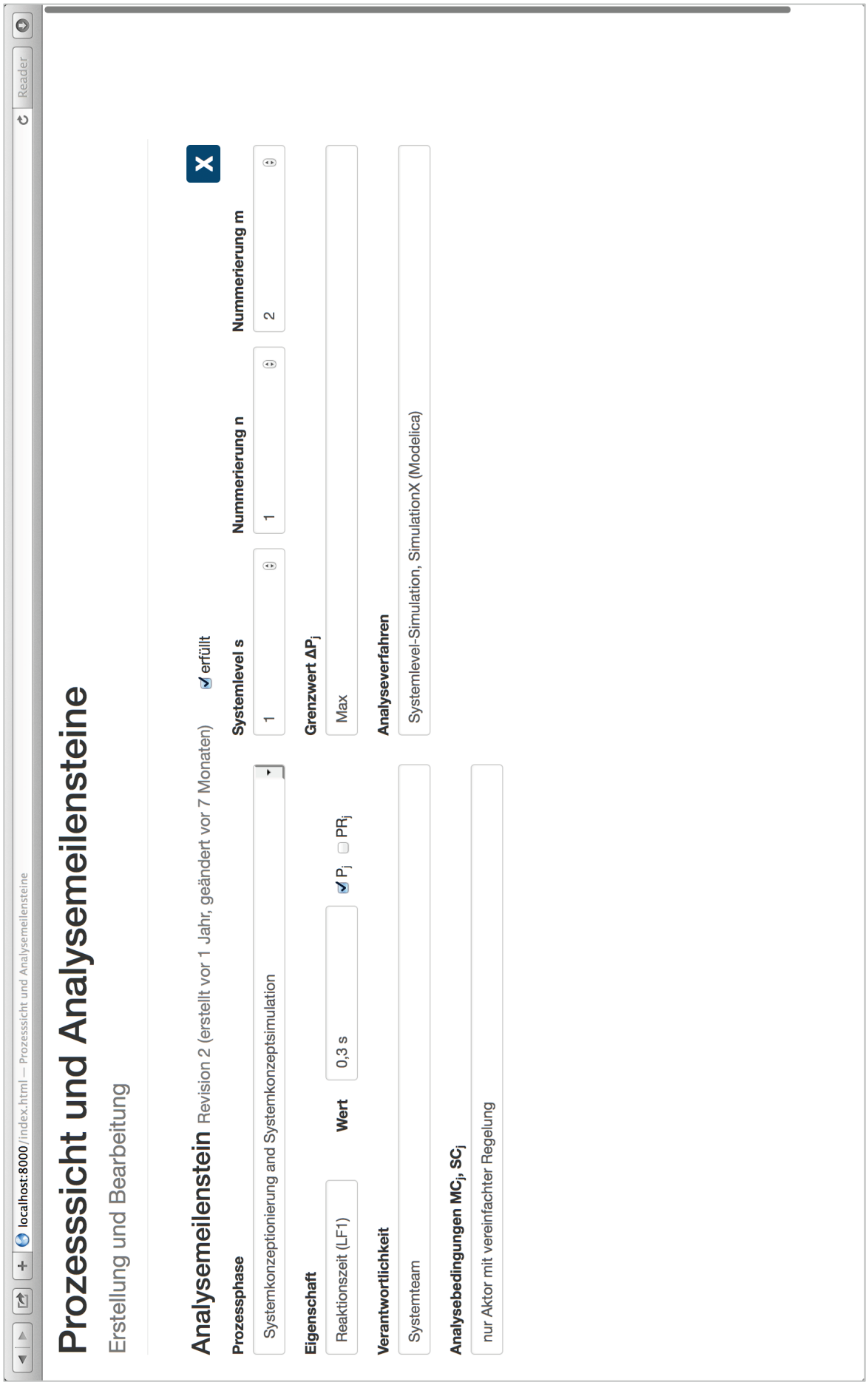


Abbildung A.3: Erstellung und Bearbeitung von Analysemeilensteinen

A.3 Beispielhafte SysML-Modellierung des aktiven Fahrwerks

Die hier dargestellte SysML-Modellierung basiert auf der Arbeit von Kelkel und Martini [KeMa14].

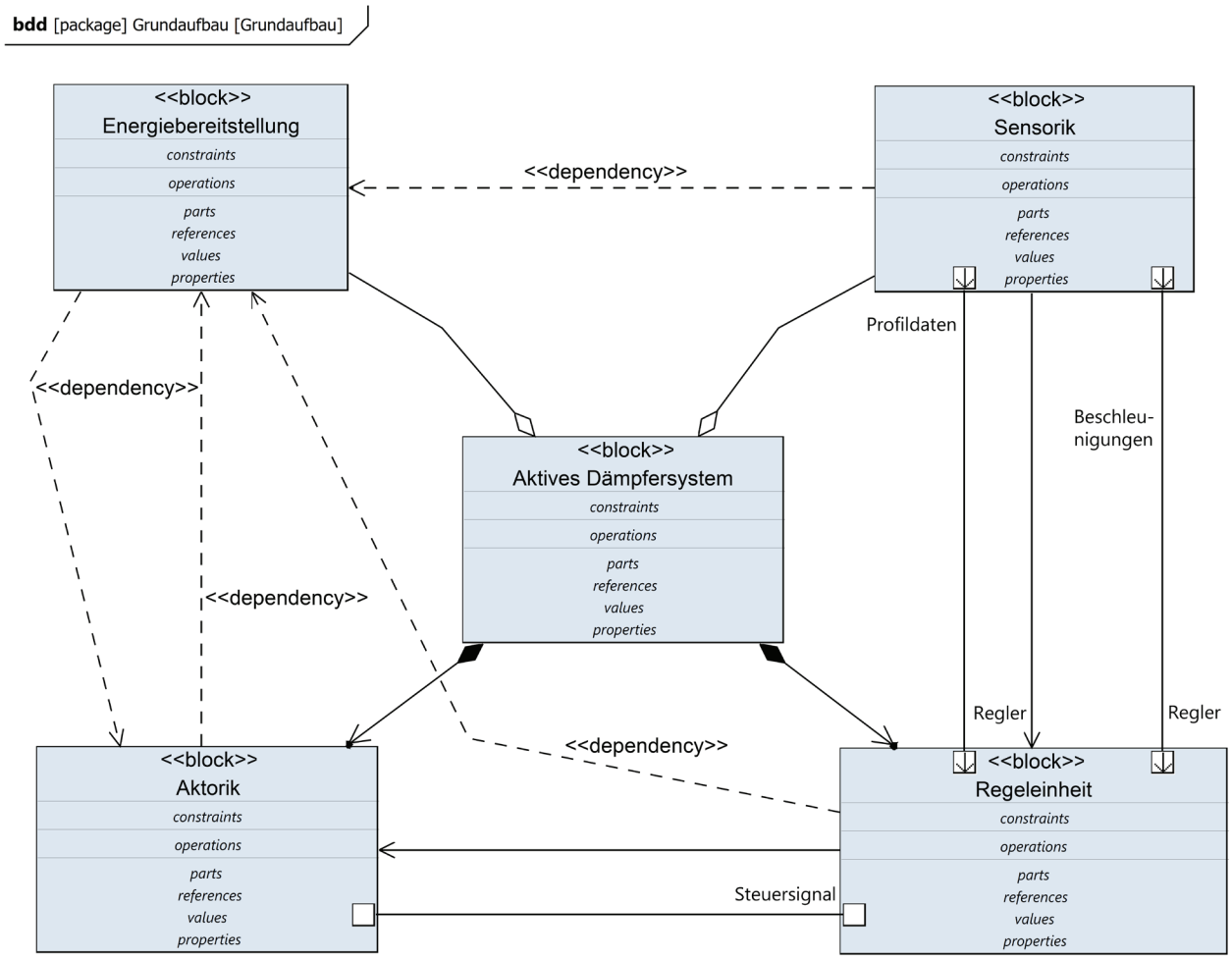


Abbildung A.4: SysML Blockdefinitionsdiagramm – Grundaufbau des aktiven Fahrwerks, nach [KeMa14]

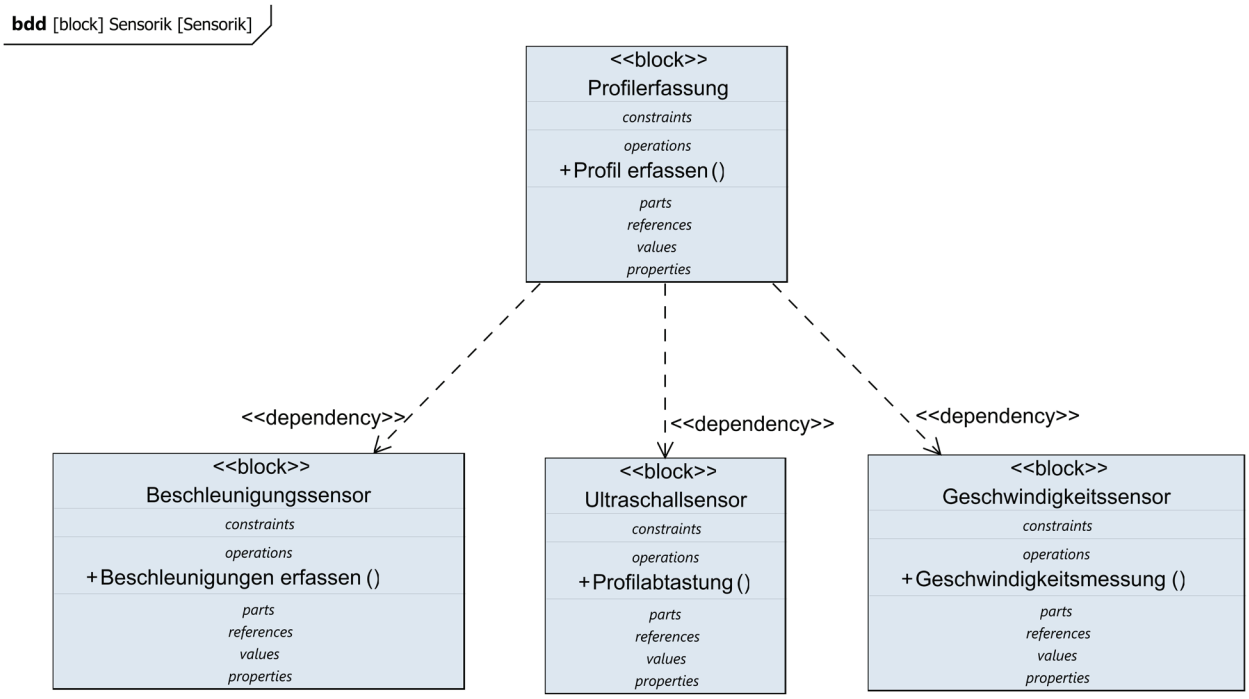


Abbildung A.5: SysML Internes Blockdiagramm – Modul Sensorik, nach [KeMa14]

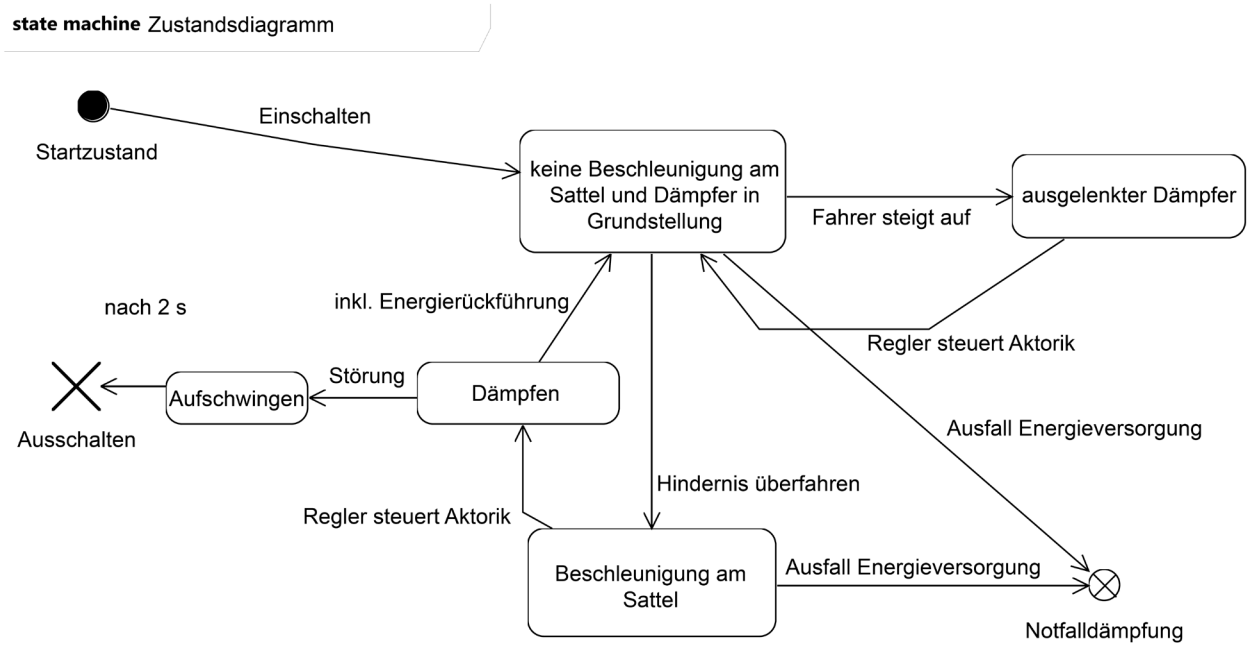


Abbildung A.6: SysML State-Machine-Diagramm – Zustände des aktiven Fahrwerks, nach [KeMa14]

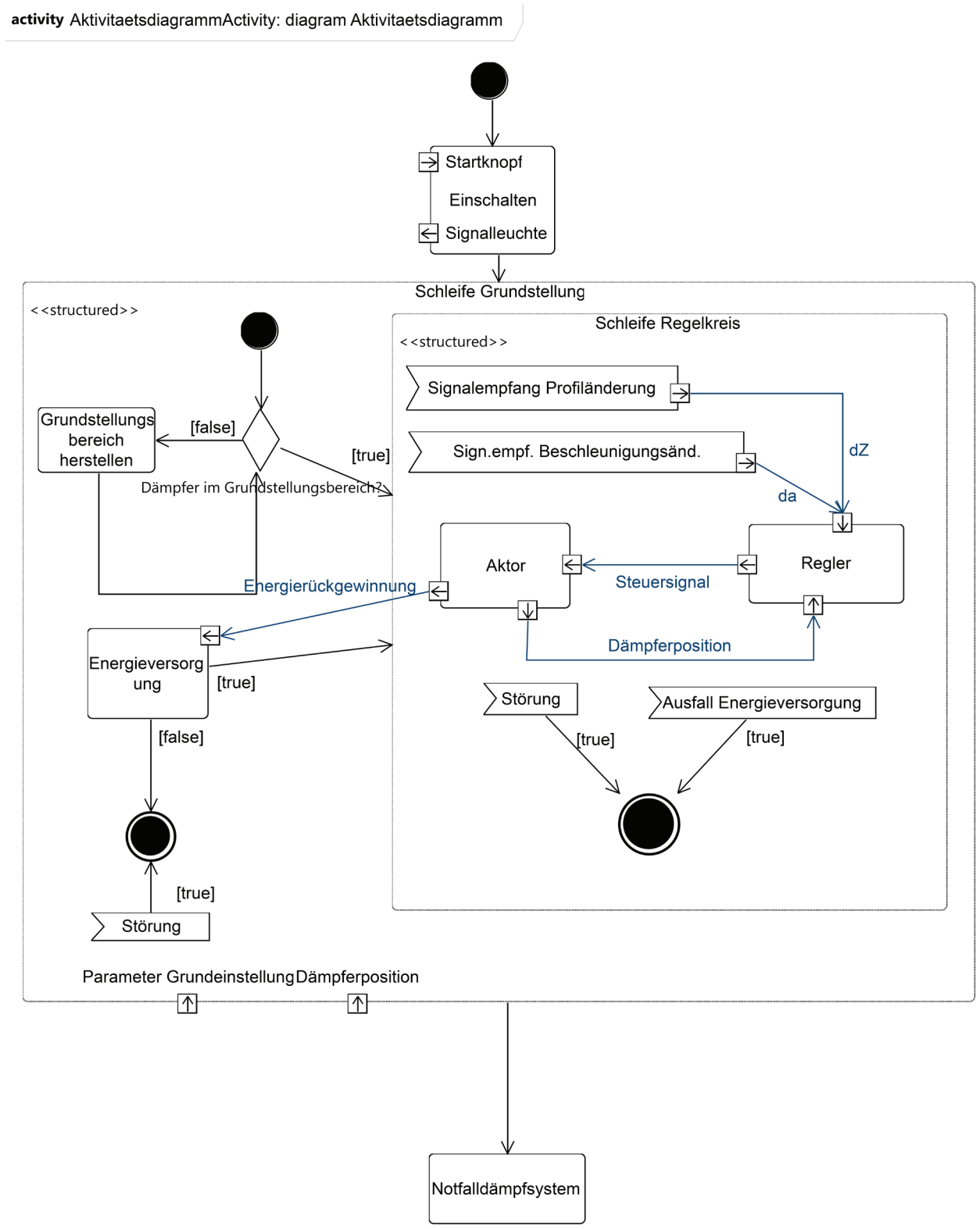


Abbildung A.7: SysML Aktivitätsdiagramm – Ablauf der Steuerung des aktiven Fahrwerks, nach [KeMa14]

A.4 Placket-Burmann-Versuchsplan

Tabelle A.2: Einflussgrößen für Plackett-Burmann-Versuchsplan

Größe	Bedeutung	Unterer Grenzwert (0)	Oberer Grenzwert (1)
A	Dichte des Materials des Fahrradrahmens	2280 kg/m³	3280 kg/m³
B	Masse des Fahrers	70 kg	100 kg
C	Translatorische Steifigkeit des Federdämpfers	100 kN/m	168 kN/m
D	Rotatorische Steifigkeit des Federdämpfers	50 kN/deg	150 kN/deg
E	Reifendruck (als Steifigkeit der Reifenfeder implementiert)	70 kN/m	100 kN/m
F	Reibung im Drehgelenk des Rahmens	0	0,5
G	Bodenamplitude	5 mm	25 mm

Tabelle A.3: Plackett-Burmann-Versuchsplan

Versuch	A	B	C	D	E	F	G
V1	1	0	0	1	0	1	1
V2	1	1	0	0	1	0	1
V3	1	1	1	0	0	1	0
V4	0	1	1	1	0	0	1
V5	1	0	1	1	1	0	0
V6	0	1	0	1	1	1	0
V7	0	0	1	0	1	1	1
V8	0	0	0	0	0	0	0

A.5 Beispiel für Auswahl von Analyseverfahren

Tabelle A.4: Zuordnung von Eigenschaften und Merkmalen zu Simulationstechniken (Zuordnung vereinfacht für eine Auswahl wichtiger Simulationstechniken)

Simulationstechnik ST_k	Eigenschaften P_j	Merkmale C_i
Finite-Elemente-Methode (strukturmechanisch, linear)	Spannungszustand **σ** (x,y,z) Verzerrungszustand **ε** (x,y,z) Eigenfrequenz f	Geometrie Materialdichte ρ Elastizitätsmodul E Querkontraktionszahl ν Kraft **F** Drehmoment **M** Randbedingungen (u. a. Lagerung)
Finite-Elemente-Methode (thermisch)	Wärmefluss Φ Temperaturzustand **T** (x,y,z)	Wärmeleitfähigkeit λ Wärmekapazität c Wärmeübergangskoeffizient h Umgebungstemperatur T_U Wärmeemissionsgrad ε Wärmelaste Q Temperaturrandbedingungen
Finite-Elemente-Methode (elektromagnetisch)	Magnetische Flussdichte **B** (x,y,z) Magnetische Feldstärke **H** (x,y,z) Elektrische Feldstärke **E** (x,y,z)	Leitfähigkeit σ Dielektrische Permittivität ε Magnetische Permeabilität μ Elektrisches Potential ϕ (x,y,z)
Strömungssimulation	Druckzustand p Volumenstrom Q Strömungsgeschwindigkeit **v** Temperaturzustand T Kraftzustand **F**	Viskosität η Anfangstemperatur T_0 Geometrie der durchströmten Umgebung Oberflächengüte der durchströmten Umgebung Druckzustand p Volumenstrom Q

Mehrkörpersimulation	Position **x** Geschwindigkeit **v** Beschleunigung **a** Kraft **F** Drehmoment **M**	Geometrie Masse und Massenverteilung Trägheitsmomente Bauteilverbindungen Lagerungen Feder-/Dämpferwirkungen Kräfte **F** Drehmomente **M** Positionsvorgaben
Schaltungssimulation (analog)	Spannung U Strom I Rauschverhalten Einschwingverhalten Zeitverhalten Werte für Widerstand, Kapazität und Induktivität	Schaltplan/Layout Widerstand R Kapazität R Induktivität L Spannung U Strom I
Schaltungssimulation (digital)	Ausgangssignale Signalflüsse Zeitverhalten	Schaltplan/Layout Eingangssignale Bauteilverhalten
Reglersimulation	Regelgröße Regelfehler Stellgröße Dynamisches Verhalten des Reglers und des Systems Reaktionszeit des Reglers und des Systems	Modell des Reglers Modell der Regelstrecke Führungsgröße
Systemsimulation	Zeitverhalten des Systems Ein-/Ausgangsverhalten des Systems Wechselwirkungen mit Umgebung	Systemelemente inkl. Verhaltensbeschreibung Systemparameter Störgrößen Rand- und Umweltbedingungen

A.6 Softwarewerkzeug zur Auswahl von Analyseverfahren

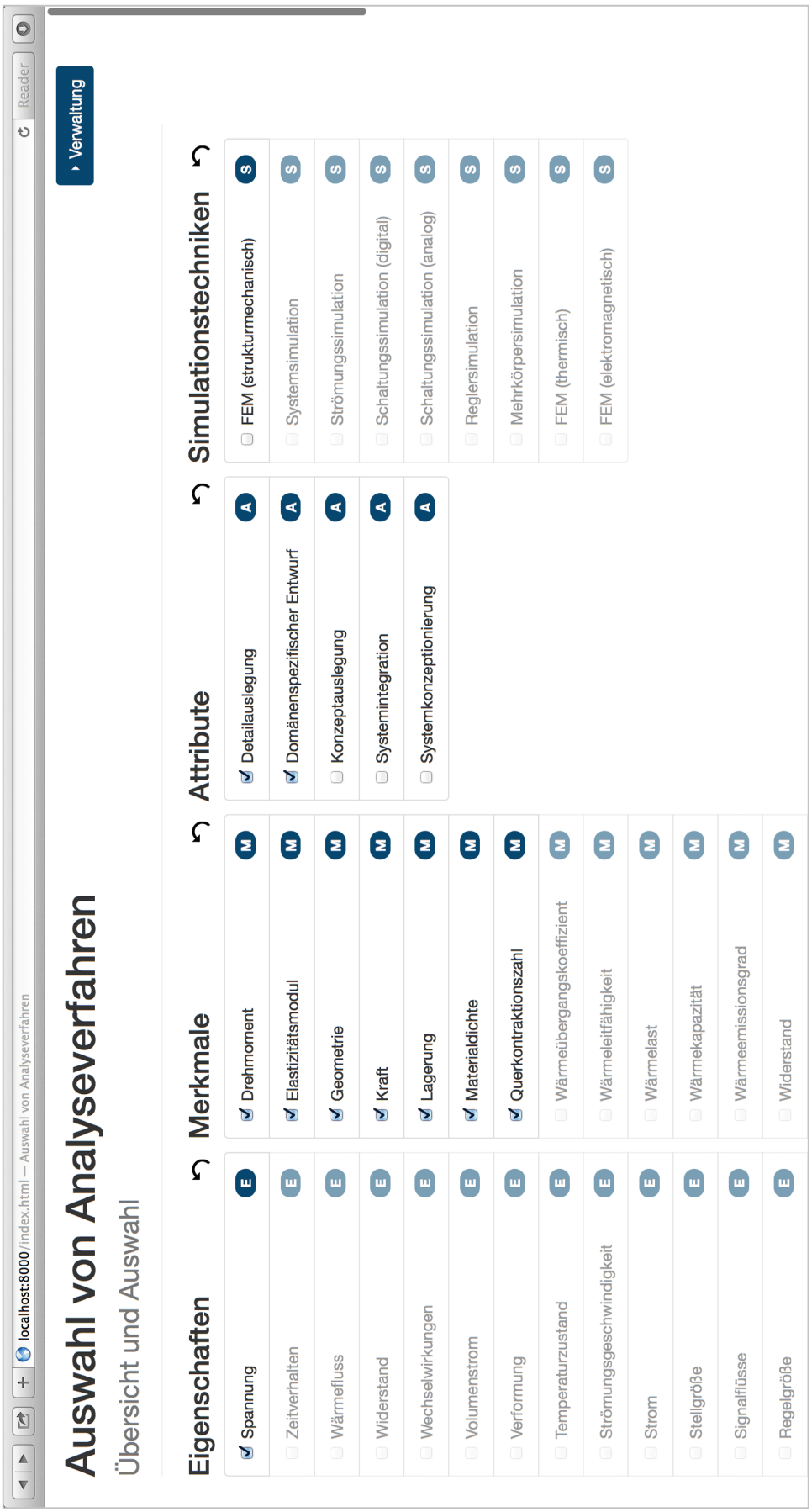
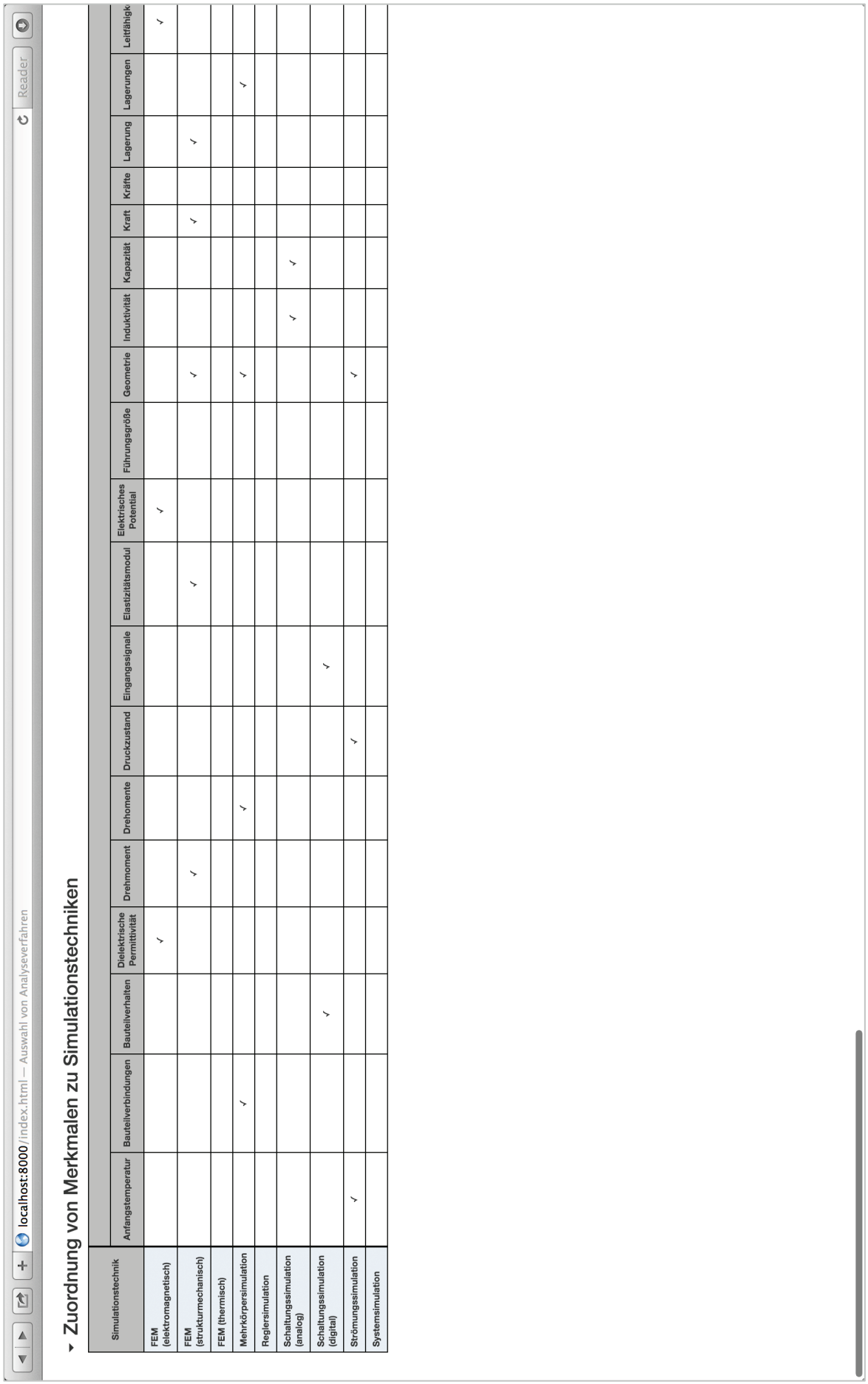


Abbildung A.8: Auswahl von Analyseverfahren

▾ Zuordnung von Simulationstechniken zu Eigenschaften

Eigenschaft	Simulationstechnik								
	FEM (elektromagnetisch)	FEM (strukturmechanisch)	FEM (thermisch)	Mehrkörpersimulation	Reglersimulation	Schaltungssimulation (analog)	Schaltungssimulation (digital)	Strömungssimulation	Systemsimulation
Ausgangssignale							✓		
Ausgangsverhalten									✓
Beschleunigung				✓					
Drehmoment				✓					
Druckzustand								✓	
Dynamisches Verhalten					✓				
Eigenfrequenz		✓							
Einschwingverhalten						✓			
Elektrische Feldstärke	✓								
Geschwindigkeit				✓					
Induktivität						✓			
Kapazität						✓			
Kraft				✓					
Kraftzustand								✓	
Magnetische Feldstärke	✓								
Magnetische Flussdichte	✓								
Position				✓					
Rauschverhalten						✓			
Reaktionszeit					✓				
Regelfehler					✓				
Regelgröße									
Signalflüsse							✓		
Spannung		✓				✓			
Stellgröße									
Strom						✓			
Strömungsgeschwindigkeit								✓	
Temperaturzustand			✓					✓	
Verformung		✓							
Volumenstrom								✓	
Wechselwirkungen									✓
Widerstand						✓			
Wärmefluss			✓						
Zeitverhalten						✓	✓		✓

Abbildung A.9: Matrix 1 – Zuordnung von Simulationstechniken zu Eigenschaften



Simulationstechnik	Anfangstemperatur	Bauteilverbindungen	Bauteilverhalten	Dielektrische Permittivität	Drehmoment	Drehmomente	Druckzustand	Eingangssignale	Elastizitätsmodul	Elektrisches Potential	Führungsgröße	Geometrie	Induktivität	Kapazität	Kraft	Kräfte	Lagerung	Lagerungen	Leitfähigk
FEM (elektromagnetisch)				✓						✓									✓
FEM (strukturmechanisch)					✓				✓			✓			✓		✓		
FEM (thermisch)																			
Mehrkörpersimulation		✓				✓						✓						✓	
Reglersimulation																			
Schaltungssimulation (analog)													✓	✓					
Schaltungssimulation (digital)			✓					✓											
Strömungssimulation	✓						✓					✓							
Systemsimulation																			

Abbildung A.10: Matrix 2 – Zuordnung von Merkmalen zu Simulationstechniken

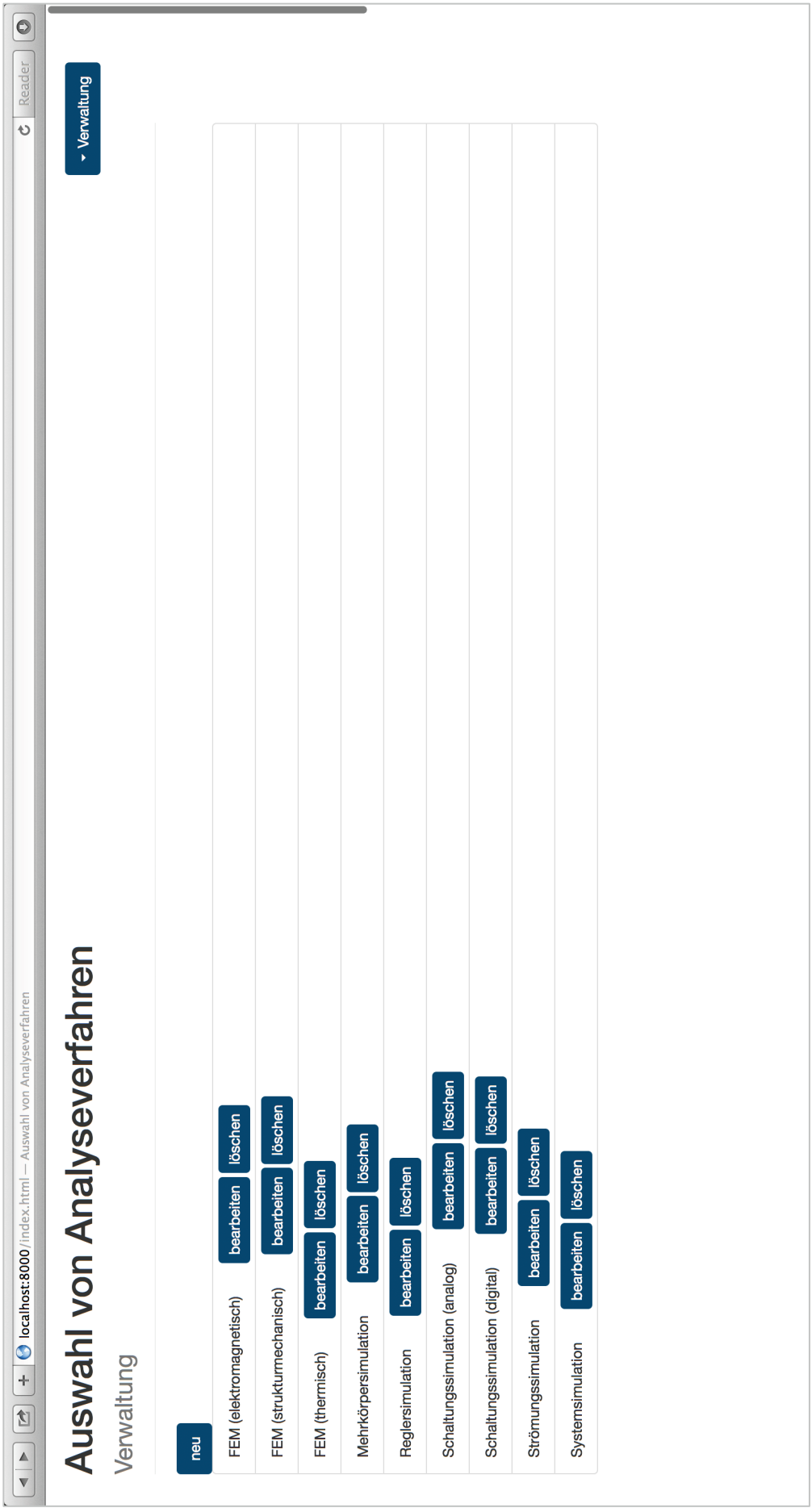


Abbildung A.11: Verwaltung von Simulationstechniken

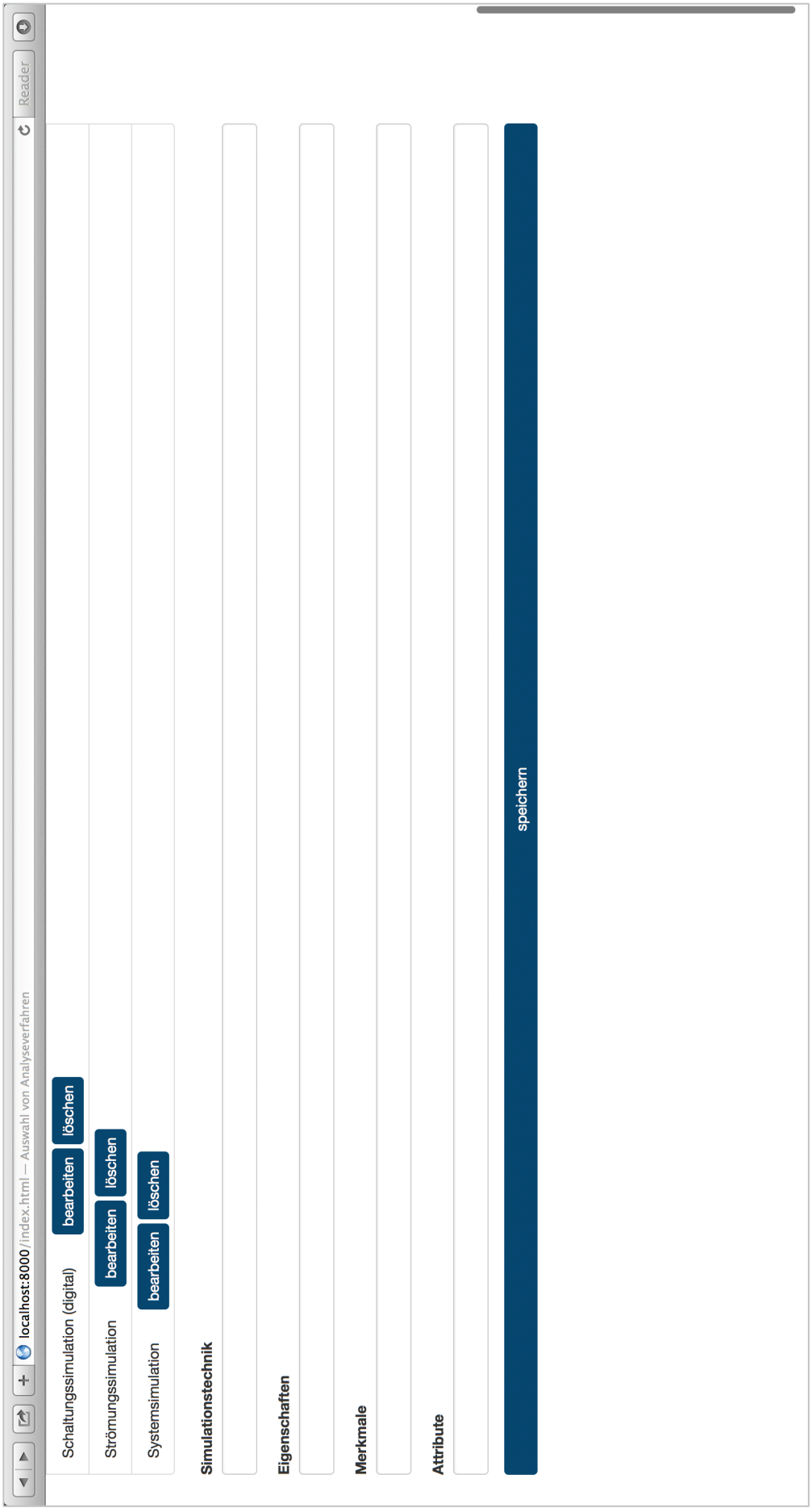


Abbildung A.12: Erstellung und Bearbeitung von Simulationstechniken