Daniel Kruse

Teilautomatisierte Parameteridentifikation für die Validierung von Dynamikmodellen im modellbasierten Entwurf mechatronischer Systeme

Partly automated parameter identification for the validation of dynamic models in the model-based design of mechatronic systems

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

Band 388 der Verlagsschriftenreihe des Heinz Nixdorf Instituts

ISSN (Print): 2195-5239
ISSN (Online): 2365-4422
ISBN: 978-3-947647-07-1

Satz und Gestaltung: Daniel Kruse

Hersteller: readbox unipress in der readbox publishing GmbH
Münster

Printed in Germany

Teilautomatisierte Parameteridentifikation für die Validierung von Dynamikmodellen im modellbasierten Entwurf mechatronischer Systeme

zur Erlangung des akademischen Grades eines
DOKTORS DER INGENIEURWISSENSCHAFTEN (Dr.-Ing.)
der Fakultät Maschinenbau
der Universität Paderborn

genehmigte
DISSERTATION

von
Dipl.-Ing. Daniel Kruse
aus Paderborn

Tag des Kolloquiums: 18. Dezember 2018
Referent: Prof. Dr.-Ing. habil. Ansgar Trächtler
Korreferent: Prof. Dr.-Ing. Jürgen Gausemeier

Vorwort

Die vorliegende Dissertation entstand während meiner Tätigkeit am Lehrstuhl für Regelungstechnik und Mechatronik (RtM) und in der Fraunhofer Projektgruppe Entwurfstechnik Mechatronik. Sie ist geprägt durch die Mitarbeit in dem EU Forschungsprojekt „Entwurfstechnik intelligente Mechatronik“ (ENTIME).

Bedanken möchte ich mich bei meinem Doktorvater, Herrn Prof. Dr.-Ing. habil. Ansgar Trächtler, der mir die Möglichkeit gegeben hat, diese Arbeit an seinem Lehrstuhl durchführen zu können. Prof. Dr.-Ing. Jürgen Gausemeier danke ich für die Übernahme des Korreferats. Ich bedanke mich auch bei Prof. Dr.-Ing. habil. Walter Sextro und Prof. Dr. rer. nat. Thomas Tröster für Ihr Mitwirken in der Promotionskommission.

Allen Kollegen am Lehrstuhl danke ich für die sehr gute Zusammenarbeit und das angenehme Arbeitsklima. Hervorzuheben sind Matthias Lochbichler, Felix Oestersötebier und Christoph Schweers, die mich durch die intensive Auseinandersetzung mit meiner Dissertation unterstützt haben. Ein großer Dank gebührt Frau Annette Bökamp-Gros für die sorgfältige Korrektur dieser Arbeit. Weiterhin danke ich den studentischen Hilfskräften, Studien-, Diplom-, Bachelor- und Masterarbeitern für ihre Unterstützung und Beiträge.

Schließlich gilt ein herzlicher Dank meinem familiären Umfeld. Einen besonderen Dank möchte ich meiner Ehefrau Carolin und meinen Eltern aussprechen, die mich stets auf meinem Weg unterstützten.

Renningen, im Januar 2019 *Daniel Kruse*

Vorveröffentlichungen

[1] Löffler, A.; Schweers, C.; Fast, V.; Kruse, D.; Trächtler, A.: Multidomänen-Modell eines Waschvollautomaten für einen Hardware-in-the-Loop-Prüfstand. In: *8. Paderborner Workshop: „Entwurf mechatronischer Systeme“*, Paderborn, 2011.

[2] Kruse, D.; Schweers, C.; Fast, V.; Trächtler, A.: Einheitliche Testumgebung für MiL und RCP mittels NI VeriStand am Beispiel eines Waschautomaten. In: *Virtuelle Instrumente in der Praxis 2012*, München, 2012.

[3] Schweers, C.; Kruse, D.; Trächtler, A.: Entwurf eines Unscented-Kalman Filters zur Zustands- und Parameterschätzung an Dymola-Modellen. In: *VDI Mechatronik-Tagung*, Aachen, 2013.

[4] Kruse, D.; Trächtler, A.; Herden, R.: Modellbasierte Entwicklung eines neuartigen Heizverfahrens für Waschautomaten. In: *Paderborner Workshop: „Entwurf mechatronischer Systeme“*, Paderborn, 2013.

[5] Schweers, C.; Kruse, D.; Fast, V.; Trächtler, A.: Online-Zustands- und Parameterschätzung an Dymola-Modellen auf NI-Echtzeithardware. In: *Virtuelle Instrumente in der Praxis 2013*, München, 2013.

[6] Fast, V.; Schweers, C.; Kruse, D.; Trächtler, A.: Methoden zur Einbindung von Fehlersimulationen in die XiL-Techniken. In: *Virtuelle Instrumente in der Praxis 2013*, München, 2013.

[7] Schweers, C.; Kruse, D.; Trächtler, A.: Automated Design of an Unscented Kalman Filter for State- and Parameter Estimation on unknown Models. In: *IEEE Conference on Control, Automation, Robotics & Embedded Systems (CARE)*, Jabalpur, 2013.

[8] Gausemeier, J. (Hrsg.); Trächtler, A. (Hrsg.); Schäfer, W (Hrsg.): *Semantische Technologien im Entwurf mechatronischer Systeme: Effektiver Austausch von Lösungswissen in Branchenwertschöpfungsketten.* Carl Hanser Verlag, München, 2014.

[9] Kruse, D.; Schweers, C.; Trächtler, A.: Methodology for a partly automated parameter identification for the validation of multi-domain models. In: *ASME International Mechanical Engineering Congress and Exposition*, Montreal, 2014.

[10] Kruse, D.; Warkentin, A.; Krüger, M.; Trächtler, A.; Rackow, S.: Multidomänenmodell zur Optimierung der Hydraulik eines Raupenlaufwerks für Landmaschinen. In: *Proc. 4. Internationales Commercial Vehicle Technology Symposium*, Kaiserslautern, 2016.

Zusammenfassung

In dieser Arbeit wird der modellbasierte Entwurf mechatronischer Systeme, in Anlehnung an die VDI-Richtlinie 2206 [VDI04], angewendet und signifikant erweitert, um der Lücke zwischen steigender Produktkomplexität und Leistungsfähigkeit von Entwicklungsmethoden entgegenzuwirken. Ein besonderes Merkmal der erarbeiteten Entwicklungsmethodik ist die ganzheitliche Betrachtung des Systems. Man hat sie fachgebietsübergreifend und funktionsorientiert gestaltet, um innovative Prinziplösungen zu erarbeiten und um auftretende Wechselwirkungen zwischen den Systemelementen der unterschiedlichen Fachdisziplinen frühestmöglich erkennen und berücksichtigen zu können. Zudem beinhaltet diese Methodik ein Vorgehen zur modellbasierten Konkretisierung, so dass detaillierte, validierte Multidomänen-Modelle der erarbeiteten Prinziplösungen für den Bereich der Systemintegration zur Verfügung stehen. Einen wesentlichen Bestandteil stellt eine entwickelte Parameteridentifikations- und Modellvalidierungsmethodik dar. Das hierzu entstandene Parameteridentifikations-Tool, bestehend aus einem FMU-Interface sowie einer MATLAB-Identifikationsumgebung, zeichnet sich besonders durch die leichte Einbindung von Dynamikmodellen aus, unabhängig von deren Modellentwicklungslandschaften, was ein Höchstmaß an Flexibilität bietet. Die Identifikationsumgebung ermöglicht es zudem, komplexe, nichtlineare Multidomänen-Modelle mittels etablierter Verfahren teilautomatisiert zu identifizieren. Die Leistungsfähigkeit der entworfenen Methodik sowie die Anwendung werden an einem Praxisbeispiel aus der Industrie aufgezeigt. Anschließend wird die Übertragbarkeit an weiteren Beispielen demonstriert.

Abstract

The thesis presents the application and a significant extension of the model-based design of mechatronic systems according to VDI Guideline 2206 [VDI04], the aim being to close the gap between an increasing product complexity and the efficiency of development methodologies. A particular feature of the extended development methodology is a holistic view of the system. It is arranged in an interdisciplinary and function-oriented manner so as to enable working out innovative principle solutions as well as detecting and taking into consideration possible interactions between the system elements from various disciplines at the earliest possible date. Furthermore, the methodology employed comprises a procedure for model-based concretization that makes detailed, validated multi-domain models of the principle solutions available to the sector of system integration. An essential element is an elaborate methodology for parameter identification and model validation. For this purpose a parameter-identification tool was designed that consists of an FMU interface and a MATLAB identification environment whose special strength lies in an easy integration of dynamics models, independently of the model-development environments these models were generated in, with the result being maximum flexibility. What is more, the identification environment allows partial identification of complex, nonlinear multi-domain models by means of well-established procedures. The efficiency of the methodology developed and its application are demonstrated by a case example from the industry. Other examples presented in some detail prove the methodology to be widely applicable.

Inhaltsverzeichnis

Abkürzungsverzeichnis

AS	Active-Set
CAD	Computer-aided Design
CONSENS	**CON**ceptual design **S**pecification technique for the **EN**gineering of complex **S**ystems
DLL	Dynamic Link Library
ENTIME	**ENT**wurfstechnik **I**ntelligente **ME**chatronik
FF	freie Flotte
FMI	Functional Mock-up Interface
FMU	Functional Mock-up Unit
GF	gebundene Flotte
GPS2N	Generalized Pattern Search Positive Basis 2N
GPSNp1	Generalized Pattern Search Positive Basis Np1
GUI	Graphical User Interface
HiL	Hardware-in-the-Loop
IP	Interior-Point
IT	Informationstechnik
it's OWL	Intelligente Technische Systeme OstWestfalenLippe
LM	Levenberg-Marquardt
LS	Line-Search
MADS2N	Mesh Adaptive Pattern Search Positive Basis 2N
MADSNp1	Mesh Adaptive Pattern Search Positive Basis Np1
MiL	Model-in-the-Loop
MKS	Mehrkörpersystem
PS	Pattern Search

QN	Quasi-Newton
RCP	Rapid Control Prototyping
RS	Relative Sensitivität
SDK	Software Development Kit
SiL	Software-in-the-Loop
SQP	Sequential Quadratic Programming
SysML	System Modeling Language
TR	Trust-Region
UML	Unified Modeling Language
XiL	X-in-the-Loop
XML	Extensible Markup Language

Symbolverzeichnis

A_C	Kontaktfläche
F_{Count}	Anzahl der Simulationsaufrufe
F_{Rad}	Wirkende Radkraft des Segway-Modells
F_{scale}	Skalierungsfaktor
H_k	Hessematrix
K	Permeabilität
KA_{Table}	Verstärkungsfaktorkennlinie
KV_{Table}	Kennlinie zur Bestimmung des drehzahlabhängigen Verstärkungsfaktors
$K_A\, K_V$	Verstärkungsfaktoren
$M_{progAblauf}$	Programmablaufmatrix
P	Identifikationsparameter
$P_{HW,n}$	Prozess für die Phase Hauptwäsche
$P_{Sp,n}$	Prozess für die Phase Spülen
$P_{VW,n}$	Prozess für die Phase Vorwäsche
RD	Aktuelle Restfeuchte
RD_{max}	Maximale Restfeuchte
S	Textilsättigung
SV_{Table}	Kennlinie zur Bestimmung des verdrängten Trommelvolumens in Abhängigkeit der Sättigung
S_V	Parameter zur Bestimmung des verdrängten Trommelvolumens
$Tank_{Table}$	Flüssigkeitsstand-Volumen-Kennlinie des Laugenbehälters
$TolX\ TolFun\ MaxFunEvals$	Optimierereinstellungen
V_d	Durch die Beladung verdrängtes Trommelvolumen
V_{Tr}	Trommelvolumen
Δh	Änderung des Flüssigkeitsstandes
Θ_{Segway}	Trägheitstensor des Segway-Modells
Θ_{yy}	Trägheitstensor des Wiesel 1 um die y-Achse
χ	Durchlässigkeitsersatzbeiwert
$\dot{\varphi}$	Gemessene Wankwinkelgeschwindigkeit des Segway-Modells

$\dot{m}_{absorption}$	Absorptionsmassenstrom
$\dot{z}$	Gemessene Ist-Geschwindigkeit des Segway-Modells in vertikaler Richtung
η_F	Viskosität einer Flüssigkeit
$\nabla f(P)$	Gradient der Fehlerfläche am Punkt P
ϕ	Goldener-Schnitt-Verhältnis
ρ_F	Dichte einer Flüssigkeit
φ	Gemessener Wankwinkel des Segway-Modells
$\vec{P}$	Identifikationsparametervektor
$\vec{U}$	Eingangsvektor
$\vec{X}$	Zustandsvektor
$\vec{Y}$	Ausgangsvektor
c_{Rad}	Rad-Federsteifigkeit des Segway-Modells
$c_{Torsion}$	Federsteifigkeit einer Torsionsfeder
d	Suchrichtung
d_{Rad}	Rad-Dämpfung des Segway-Modells
$d_{Torsion}$	Dämpfung einer Torsionsfeder
$f(\vec{P})$	Fehlerfläche
$f_{TP}\, f_{rdyn}\, f_{rKV}$	Eckfrequenz eines Tiefpassfilters 1. Ordnung
g	Erdbeschleunigung
h	Flüssigkeitsstand
lb	Untere Grenze eines Identifikationsparameters (Lower Bound)
m_B	Masse der Beladung
m_F	Masse der in der Beladung gebundenen Flüssigkeit
m_{Segway}	Masse des Segway-Modells
n	Drehzahl
pS_{Table}	Kapillardruckkennlinie
p_C	Kapillardruck
r_R	Rad-Radius des Segway-Modells
rp_{Table}	Kennlinie der drehzahlabhängigen Flüssigkeitsstandsänderung
t_{End}	Für die Simulation vorgegebene Endzeit
$t_{Int,End}$	Für das FMU-Interface vorgegebene Endzeit des auszuführenden Intervalls
$t_{Int,Start}$	Für das FMU-Interface vorgegebene Startzeit des auszuführenden Intervalls

t_{Int}	Intervalllänge in Sekunden
t_{Start}	Für die Simulation vorgegebene Startzeit
$t_{StepSize}$	Integrationsschrittweite in Sekunden
ub	Obere Grenze eines Identifikationsparameters (Upper Bound)
z	Gemessene Ist-Position des Segway-Modells in vertikaler Richtung
z_phase	Zählervariable für die Phasen
$z\_prozess$	Aktuelle Prozessvariable
z_R	Ist-Position des Radmittelpunktes des Segway-Modells in vertikaler Richtung
z_{St}	Vertikale Umgebungsänderung des Segway-Modells durch ein Hindernis

1 Motivation und Zielsetzung

Produktinnovationen des modernen Maschinenbaus sind maßgeblich von einem fachdisziplinübergreifenden Zusammenwirken von Mechanik, Elektronik und Informationstechnik abhängig, was das Wort Mechatronik zum Ausdruck bringt. Die steigende Komplexität von zu entwickelnden mechatronischen Produkten bei kürzer werdenden Entwicklungszeiten zwingt Unternehmen zu einem systematischen Vorgehen bei der Entwicklung solcher Systeme. Gleichzeitig ist ein mechatronisches Produkt in der Regel das Ergebnis einer Branchenwertschöpfungskette, bei der Unternehmen auf die Lösungen von spezialisierten Lieferanten zurückgreifen. Hierdurch steigt weiterhin die Komplexität des gesamten Entwicklungsprozesses, was ebenfalls in einem systematischen Vorgehen berücksichtigt werden muss.

Zahlreiche Entwicklungsmethodiken beschäftigen sich mit genau diesem Thema [Ise08, Jan09, Ehr09]. Oftmals werden hierbei jedoch nur einzelne Bereiche betrachtet, wodurch eine Lücke zwischen der Leitungsfähigkeit der Entwicklungsmethoden und der Produktkomplexität entsteht (vgl. Bild 1-1).

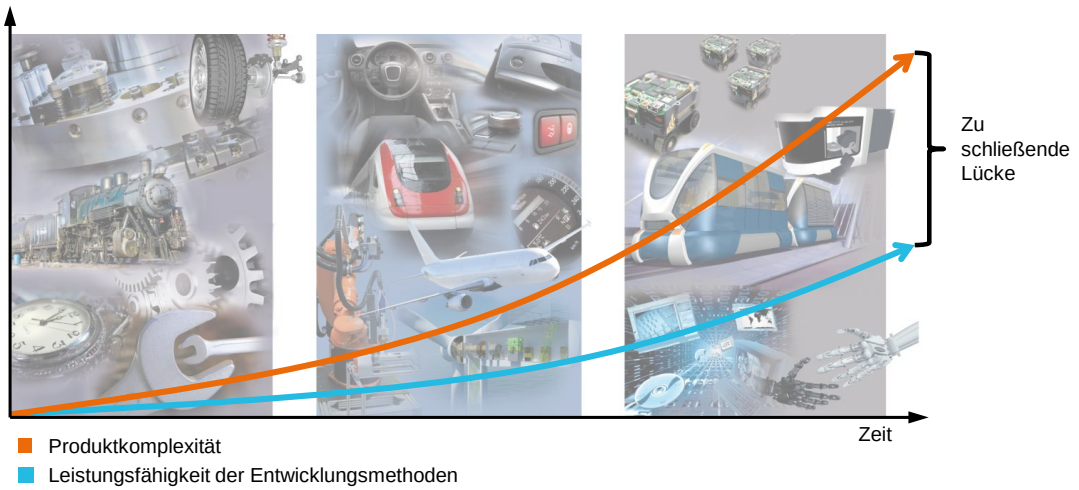

Bild 1-1: Zu schließende Lücke zwischen steigender Produktkomplexität und der Leistungsfähigkeit der Entwicklungsmethoden (in Anlehnung an [GDS+13])

1.1 Motivation

Einer der verbreitetsten Ansätze, um die Lücke zu verkleinern, ist die fachgebietsübergreifende modellbasierte Entwicklungsmethodik auf Basis des V-Modells (vgl. Bild 1-2).

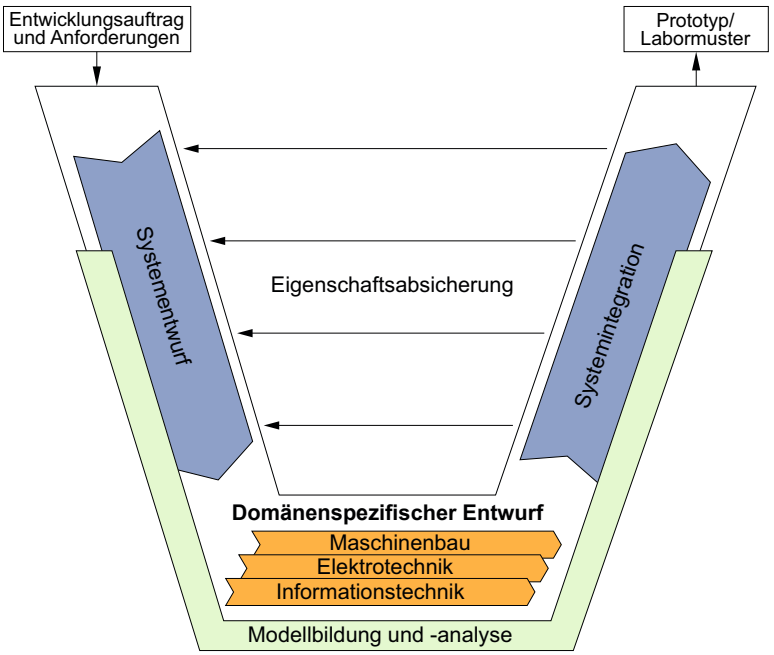

Bild 1-2: V-Modell [VDI04]

Das V-Modell lässt sich im Wesentlichen in die drei Phasen Systementwurf, domänenspezifischer oder auch disziplinspezifischer[1] Entwurf und die Systemintegration aufteilen [VDI04]. Unterstützt werden die drei Phasen durch Modellbildung und -analyse, bei der unter anderem Simulationsmodelle und entsprechende Simulationstools zum Einsatz kommen. Das Ziel eines solchen Simulationsmodells ist es, das dynamische Systemverhalten eines mechatronischen Systems hinreichend genau abzubilden. Entscheidenden Einfluss auf das jeweilige dynamische Verhalten besitzen die im System mitwirkenden Komponenten, teils aus unterschiedlichsten Domänen. Aufgrund der starken Verbreitung mechatronischer Systeme (Fahrzeugtechnik, Haushaltsgerätetechnik, Medizintechnik, Energietechnik ...) steigt die Anzahl der zu berücksichtigenden Domänen (Mechanik, Elektronik, Informationstechnik, Regelungstechnik, Thermodynamik, Hydraulik, Fluidtechnik, Bionik etc.). Für die Analyse des dynamischen Systemverhaltens werden somit vermehrt Multidomänen-Modelle verwendet, deren Detaillierungsgrad

[1] Domäne und Disziplin werden in dieser Arbeit synonym verwendet.

in den jeweiligen Phasen variiert. Findet im Systementwurf zunächst eine domänenübergreifende Betrachtung des Gesamtsystems in enger Abstimmung aller beteiligten Experten statt, wird die Entwicklung der einzelnen Domänenmodelle während des disziplinspezifischen Entwurfs von den jeweiligen Experten separat durchgeführt. Durch moderne Kommunikations- und Austauschmöglichkeiten, wie sie zum Beispiel das Internet zur Verfügung stellt, können verschiedenste Experten Teilsysteme parallel bearbeiten, bevor diese in der abschließenden Systemintegration als Gesamtmodell eingesetzt werden. Die zur Verfügung stehende Zeit für die jeweiligen Entwicklungsabschnitte verkürzt sich jedoch aufgrund des steigenden Wettbewerbs, da Unternehmen dazu gezwungen sind, ihre Produkte früher auf den Markt zu bringen. Um dennoch den steigenden Anforderungen an das jeweilige Produkt trotz verkürzter Entwicklungszeit gerecht zu werden, greifen die Entwickler vermehrt auf bestehendes Wissen in Form von sogenannten Lösungselementen zurück. Als Lösungselemente werden bewährte und realisierte Lösungen zur Erfüllung einer Funktion des Gesamtsystems bezeichnet [GEK01, GSA+11]. Diese Lösungselemente können für die Modellierung aus bestehenden Modellbibliotheken entnommen werden. Hierdurch ergeben sich viele Vorteile: Komplexe Modelle können schneller erstellt werden, sie sind weniger fehleranfällig, man kann schneller Lösungen vergleichen etc. Es ist ein Ansatz, bewährte Lösungen wieder zu verwenden und somit die Komplexität in der zur Verfügung stehenden Zeit beherrschbar zu gestalten. Zugleich wird es jedoch unwahrscheinlicher, einen Experten für das Gesamtsystem zu finden, da in Anbetracht der verkürzten Zeit nicht jedes Teilsystem aus einer Modellbibliothek bis ins Detail nachvollzogen werden kann. Die Verwendung von bestehenden Lösungselementen auch in Form von Modellbibliotheken hat des Weiteren den Nachteil, dass viele Experten oftmals nur ihnen bekannte Lösungen verwenden. Das Innovationspotenzial wird somit nicht vollkommen ausgeschöpft.

Um Simulationsmodelle anzuwenden bzw. um diese wiederzuverwenden, muss man sie validieren.

„Jedes (!) Modell muss validiert werden – Modelle, die nicht überprüft werden können, brauchen nicht erstellt zu werden.“([Ada14], S. 32).

Ziel der Modellbildung ist nicht, das reale System so genau und detailliert wie möglich abzubilden, sondern so genau wie erforderlich. Die Modellvalidierung stellt genau diesen Aspekt sicher. Die Basis der Modellvalidierung bildet die Parameteridentifikation, für die es verschiedenste Ansätze gibt [Lju11, Ise08, Nye06]. Sehr verbreitet ist die Parameteridentifikation durch einen Experten. Vor dem genannten Hintergrund ist dieses Verfahren jedoch zukünftig immer weniger anwendbar, da es keinen Experten für das Gesamtmodell gibt und die steigende Komplexität eine ganzheitliche Parameteridentifikation für das Gesamtmodell durch *trial and error* in einer angemessenen Zeit verhindert. Eine effizientere

Möglichkeit zur Parameteridentifikation ist die Verwendung von Optimierungsverfahren. Handelt es sich um eine große Anzahl zu identifizierender Parameter und ist zudem das Simulieren des Gesamtmodells rechenintensiv, ist diese Methode jedoch sehr zeitintensiv, und das Auffinden von optimalen Parametern ist nicht sichergestellt. Um nur die wesentlichsten Parameter festzulegen, die für die Parameteridentifikation von Bedeutung sind, braucht es wiederum Expertenwissen aus den jeweiligen Domänen.

1.2 Zielsetzung und Aufbau der Dissertation

In dieser Arbeit wird eine Methodik aufzeigt, bei der zwei wesentliche Aspekte im Vordergrund stehen. Zum einen wird aufgezeigt, wie lösungsneutrale Anforderungen an ein zu entwickelndes System strukturiert erarbeitet werden können, um bei der Erarbeitung von neuen, innovativen Prinziplösungen einen möglichst großen Lösungsraum aufzuspannen (Innovationspotenzial ausschöpfen). Zum anderen ist eine Parameteridentifikations- und Modellvalidierungsmethodik integriert, die der zuvor genannten Problematik (fehlender Experte für das Gesamtmodell, viele zu identifizierende Parameter aufgrund bestehender Modellbibliotheken, verkürzte Entwicklungszeit ...) entgegenwirkt.

Ein besonderer Fokus der entwickelten Methodik liegt auf einer ganzheitlichen Betrachtung des Systems. Die Vorgehensmethodik ist hierzu fachgebietsübergreifend und funktionsorientiert gestaltet, damit auftretende Wechselwirkungen zwischen den Systemelementen der unterschiedlichen Fachdisziplinen frühestmöglich erkannt und berücksichtigt werden können. Des Weiteren beinhaltet diese Methodik ein Vorgehen zur modellbasierten Konkretisierung, so dass detaillierte, validierte Multidomänen-Modelle der erarbeiteten Prinziplösungen für den Bereich der Systemintegration zur Verfügung stehen.

Kern der entstandenen Methodik ist ein Vorgehensmodell, das im Wesentlichen die im **ENT**wurfstechnik **I**ntelligente **ME**chatronik (ENTIME)-Projekt (vgl. Kapitel 3) mitentwickelten Ergebnisse beinhaltet. Ein besonderes Augenmerk in dieser Arbeit liegt auf dem Bereich der Validierung, was eine Erweiterung der ENTIME-Entwurfsmethodik darstellt. Durch das gezielte Erfassen von Referenzdaten an entsprechenden Prüfmustern kann das Modell zu einem frühen Zeitpunkt validiert werden. Hierdurch sind schon frühzeitig Rückschlüsse auf die richtige Wahl der Modellierungstiefe und der Modellierungsgüte möglich. Des Weiteren kann das somit vorliegende validierte Modell für die anschließende Systemintegration verwendet werden.

Bild 1-3 zeigt das angewendete Vorgehensmodell, integriert in das V-Modell. Auf die in Bild 1-3 genannten Meilensteine und Ergebnisse wird im weiteren Verlauf

dieser Arbeit ein besonderes Augenmerk gelegt. Wegen der Vielzahl an mitwirkenden Disziplinen werden spezialisierte Informationstechnik (IT)-Werkzeuge benötigt. Zur Vermeidung von Systembrüchen ist die Nutzung einer durchgängigen Toolkette hierbei ein wesentlicher Bestandteil.

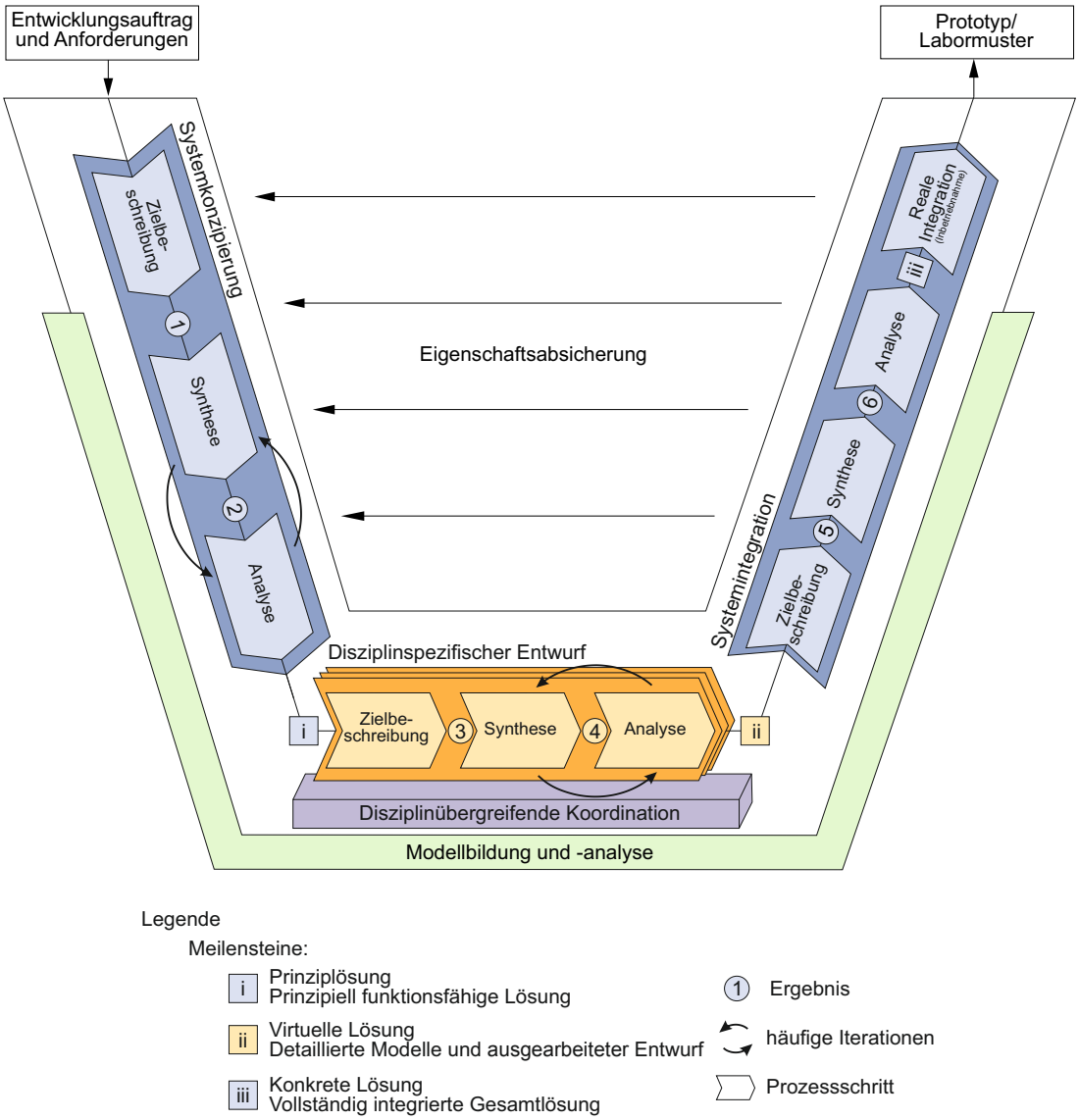

Bild 1-3: Vorgehensmodell, integriert in das V-Modell

Das Vorgehen beginnt mit einer fachdisziplinübergreifenden Systemkonzipierung. Ein Schwerpunkt liegt hierbei auf einer strukturierten Vorgehensweise, bei der die Anforderungen an ein zu entwickelndes System so aufbereitet werden, dass auf ein breit gefächertes Lösungsangebot zurückgegriffen werden kann. Hierdurch

werden innovative Produktkonzepte ganzheitlich erarbeitet und in Form einer Prinziplösung beschrieben.

Die Prinziplösung legt den grundsätzlichen Aufbau und die Wirkungsweise des zu betrachtenden mechatronischen Systems fest und bildet die Grundlage der nachfolgenden Konkretisierung. In der Konkretisierung erfolgt durch eine domänenspezifische Betrachtung der jeweiligen Teilsysteme eine Detaillierung der Prinziplösung bis hin zur konkreten Lösung in der Systemintegration. Um sicherzustellen, dass während des disziplinspezifischen Entwurfs die jeweiligen Disziplinen nicht auseinanderlaufen, findet parallel eine disziplinübergreifende Koordination statt. In dieser Phase sammelt und überwacht ein Koordinator gesamtsystemrelevante Änderungen, analysiert sie und legt Strategien für die modellbasierte Integration sowie die anschließende Inbetriebnahme fest. In [Jus13] wird diese Person auch als Mechatroniker bezeichnet.

Einen wichtigen Bestandteil aller Phasen stellt die Modellbildung dar. Es entsteht ein detaillierter werdendes Dynamikmodell des zu betrachtenden Systems. Für die in der Konkretisierung stattfindende Modellvalidierung ist eine Parameteridentifikations- und Modellvalidierungsmethodik erarbeitet worden. Einen wesentlichen Bestandteil dieser Methodik stellt ein entwickeltes Parameteridentifikations-Tool dar (siehe Kapitel 4). Die Methodik zur Identifikation und Validierung gliedert sich in die in ENTIME mitentwickelte Methodik zur modellbasierten Entwicklung mechatronischer Systeme ein und erweitert diese entsprechend.

Die Notwendigkeit der entworfenen Methodik sowie die Anwendung werden an einem Praxisbeispiel aus der Industrie, das im Rahmen des ENTIME-Projektes entstanden ist, in Kapitel 3 aufgezeigt. Da es sich bei den erarbeiteten Ergebnissen teilweise um sensible Daten aus der Vorentwicklung eines Industrieunternehmens handelt, werden nicht alle Schritte bzw. Ergebnisse im Detail vorgestellt. Es wird dennoch ein allgemeingültiger Eindruck vermittelt, wie die Entwicklungsmethodik konkret angewendet werden kann, um bestmögliche, innovative Ergebnisse mit Hilfe von modernsten Entwicklungstools zu erzielen.

Die Übertragbarkeit der Ergebnisse auf weitere Bereiche zeigt Kapitel 5. Die erzielten Ergebnisse werden in Kapitel 6 bewertet, bevor ein Ausblick die Dissertation abschließt.

2 Stand der Technik und Handlungsbedarf

In diesem Kapitel wird eine kurze Übersicht über den diese Arbeit zugrundeliegenden Stand der Technik und Wissenschaft gegeben. Hierzu wird zunächst auf den allgemeinen Aufbau mechatronischer Systeme eingegangen. Anschließend werden Vorgehensweisen zur Entwicklung solcher Systeme vorgestellt. Einen wichtigen Bestandteil bildet dabei die Modellbildung. Es werden unterschiedliche Modellierungsarten aufgezeigt, bevor auf die Modellvalidierung mithilfe von Parameteridentifikationsverfahren eingegangen wird. Hierzu werden unterschiedliche Optimierungsverfahren vorgestellt, die für diese Zwecke verwendet werden können. Ein sich aus dem aktuellen Stand der Technik ergebender Handlungsbedarf schließt dieses Kapitel ab.

2.1 Aufbau mechatronischer Systeme

Das Zusammenwirken von Mechanik, Elektronik und Informationstechnik wird im Wort Mechatronik zum Ausdruck gebracht. Definiert ist der Begriff im deutschsprachigen Raum wie folgt:

„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“ ([Ise08], S. 18).

Die Grundstruktur eines mechatronischen Systems ist in Bild 2-1 dargestellt. Sie besteht aus einer logischen und einer physikalischen Ebene. Sie setzt sich immer aus dem Grundsystem, der Sensorik, einer Informationsverarbeitung und der Aktorik zusammen, weshalb diese in Bild 2-1 als notwendige Einheiten bezeichnet sind. Die jeweiligen Elemente können dabei in Wechselwirkung mit der Umgebung treten. Zudem gibt es die Möglichkeit einer Mensch-Maschine-Schnittstelle zwischen dem Nutzer und der Informationsverarbeitung (vgl. optionale Einheit

Bild 2-1). Im gesamten System spielen neben den Informationsflüssen die Energie- sowie die Stoffflüsse eine entscheidende Rolle.

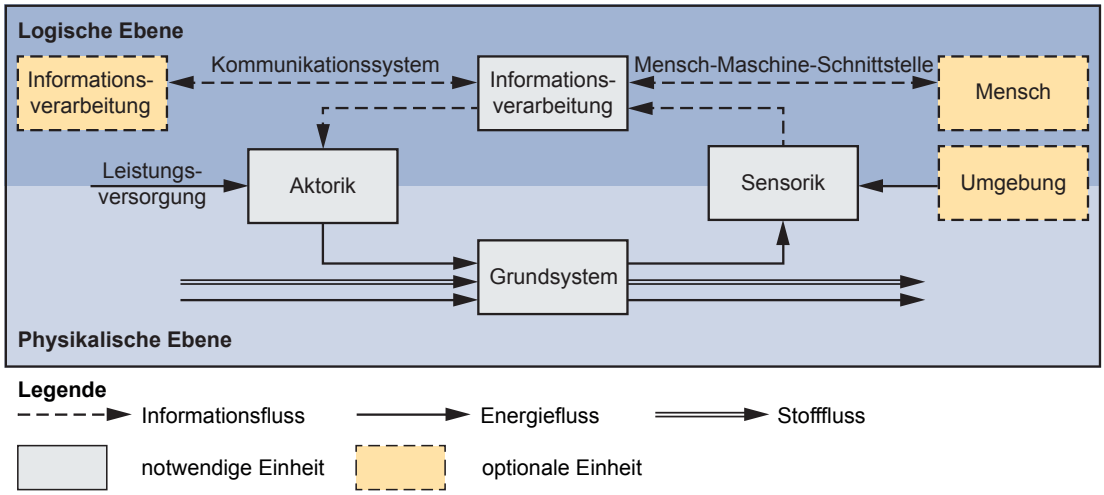

Bild 2-1: Grundstruktur eines mechatronischen Systems ([GTS+14] nach [VDI04])

Das Grundsystem beschreibt ein physikalisches System, das sich durch die Verknüpfung von interdisziplinären Komponenten darstellen lässt. Die Sensorik dient der Erfassung physikalischer Größen des Gesamtsystems. Aufgrund der großen Differenzen hinsichtlich der Komplexität und der Einsetzbarkeit von bereitstehenden Sensoren wird zwischen Sensorelementen, Sensorsystemen und intelligenten Sensoren unterschieden [Rod06]. In der Informationsverarbeitung können die gemessenen Sensorsignale gezielt aufbereitet werden, wodurch der Ist-Zustand des Gesamtsystems erfasst wird. Des Weiteren dient die Informationsverarbeitung zur Ermittlung von Strategien, die das Gesamtsystem in einen gewünschten Zustand überführen sollen. Hierzu werden die Ergebnisse der Informationsverarbeitung an die Aktorik übergeben. Durch die Wirkung der Aktorik auf das Gesamtsystem verändern sich die physikalischen Größen des Grundsystems, und der Wirkungskreis der mechatronischen Grundstruktur wird geschlossen.

2.2 Entwicklungsmethodiken

Für die Entwicklung von mechatronischen Systemen gibt es eine Vielzahl von Entwicklungsmethodiken [Ise08, Jan09, Ehr09]. Zwei der Verbreitetsten sind die mobellbasierte Entwicklung nach dem V-Modell, beschrieben in der VDI-Richtlinie 2206 [VDI04], sowie der erweiterte konstruktive Entwurf nach [PB97], die nachfolgend kurz vorgestellt werden.

2.2.1 Erweiterter konstruktiver Entwurf

Die Vorgehensweise beim erweiterten konstruktiven Entwurf für mechatronische Systeme ist in Bild 2-2 gezeigt. Dabei wird der zugrundeliegende klassische konstruktive Entwurf technischer Systeme nach [PB97], der den Anforderungen mechatronischer Systeme nicht gerecht wird, um einen mechatronischen Entwurf erweitert [Toe02]. Dies entspricht dem Systementwurf in Bild 2-2.

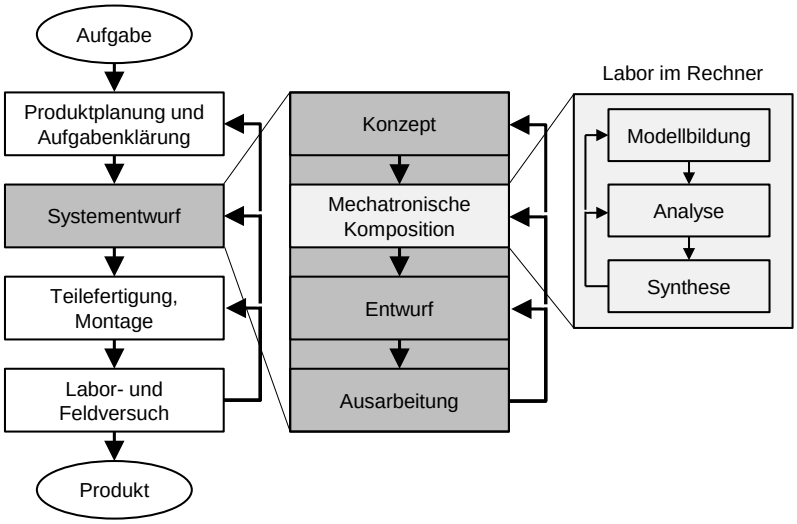

Bild 2-2: Mechatronische Komposition im Produktentwicklungszyklus [Toe02]

Ein wichtiger Teil des Systementwurfs ist die mechatronische Komposition. Sie setzt unmittelbar nach der Konzeptphase an. In ihr wird das zu entwickelnde mechatronische System ganzheitlich im Rechner abgebildet, analysiert und zusammengesetzt. Einen wesentlichen Bereich der mechatronischen Komposition bildet die Modellbildung. In ihr wird im Rechner ein Abbild der Realität geschaffen, das im weiteren Verlauf zur Analyse und Synthese verwendet wird. Die Analyse bezeichnet hierbei die Untersuchung des Systems im Rechner, wohingegen die Synthese die Berücksichtigung der Analyseergebnisse in Bezug auf das zu untersuchende System darstellt. Das Ziel der mechatronischen Komposition ist eine funktionsorientierte Gesamtauslegung im Rechner. Das Vorgehen ist iterativ, wobei das zu behandelnde System stets ganzheitlich betrachtet wird. Mit zunehmender Produktreife erfolgt schrittweise ein Übergang von einer Komposition des Grundsystems über eine idealisierte Komposition hin zu einer ganzheitlichen Komposition [Ill14].

2.2.2 V-Modell

Einen weiteren Ansatz, ein System ganzheitlich zu betrachten, liefert das V-Modell nach [VDI04]. Dieses beschreibt den Makrozyklus beim Entwurf mechatronischer Systeme [VDI04]. Seine in der Softwaretechnik liegenden Ursprünge [BD93] wurden an die Entwicklungsanforderungen für mechatronische Systeme angepasst. Das V-Modell startet mit den zuvor definierten Anforderungen (vgl. Bild 1-2, S. 2). Anhand der Anforderungen findet im Systementwurf die Produktkonzeption statt. Als Ergebnis des Systementwurfs werden erste Lösungsansätze erarbeitet. In dem darauffolgenden domänenspezifischen Entwurf werden die Lösungsansätze konkretisiert. Die Konkretisierung erfolgt parallel in der jeweils beteiligten Domäne. Ist die Funktionserfüllung einer jeden Domäne für die erarbeiteten Lösungsansätze sichergestellt, werden die jeweils detaillierten Lösungen aus jeder Domäne in der Systemintegration zusammengeführt. Das Ergebnis ist ein Gesamtentwurf des zu betrachtenden mechatronischen Systems. Die Eigen-

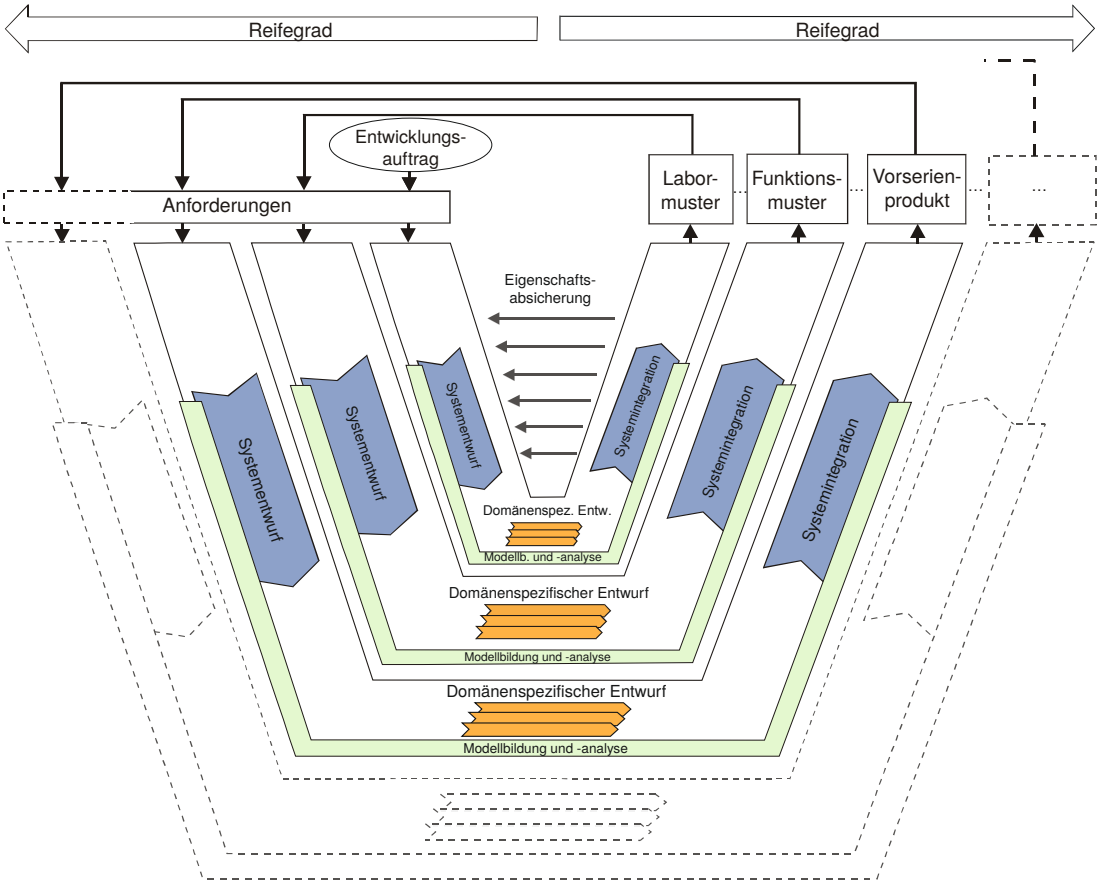

Bild 2-3: Durchlauf mehrerer Makrozyklen mit zunehmender Produktreife [VDI04]

schaftsabsicherung während der Systemintegration dient der Ergebnisprüfung. Hierzu werden die Ergebnisse mit den zuvor definierten Anforderungen abgeglichen. Ein weiterer wesentlicher Bereich des V-Modells befasst sich mit Modellbildung und -analyse. In diesem Bereich werden dynamische Modelle des zu untersuchenden mechatronischen Systems entwickelt, die sich kontinuierlich erweitern und detaillieren. Hierdurch können wesentliche Erkenntnisse über das zu untersuchende System getroffen werden. Das V-Modell endet mit dem Produkt, jedoch ist es üblich, das Vorgehensmodell mehrfach zu durchlaufen, um den Reifegrad des Produktes zu erhöhen. Übliche Zwischenstufen vor dem eigentlichen Produkt sind Labormuster, Funktionsmuster, Vorserienprodukt ... (vgl. Bild 2-3).

2.3 Domänenübergreifender Systementwurf mit CONSENS

Bei den zuvor vorgestellten Entwurfsmethodiken findet jeweils ein domänenübergreifender Systementwurf statt. Beim V-Modell geschieht dies während des Systementwurfes, beim konstruktiven Entwurf in der Produktplanung und Aufgabenklärung sowie in der Konzeptphase. Nachfolgend wird die für diese Arbeit etablierte Spezifikationstechnik für die Entwicklung mechatronischer Systeme mit dem Namen **CON**ceptual design **S**pecification technique for the **EN**gineering of complex **S**ystems (CONSENS) vorgestellt.

Entwickelt wurde die Spezifikationstechnik am Heinz Nixdorf Institut [GLL12, GFD+08]. Sie kommt während der Konzeptphase zum Einsatz und bietet die nötige ganzheitliche Beschreibung der Prinziplösung. Diese beinhaltet eine abstrakte Sicht der Lösung in Form von idealisierten Modellen. Den Kern der Spezifikationstechnik stellen die acht Partialmodelle Umfeld, Anwendungsszenarien, Anforderungen, Funktionen, Wirkstruktur, Verhalten[1], Gestalt und Zielsystem dar (vgl. Bild 2-4). Für die rechnerinterne Abbildung der jeweiligen Partialmodelle kann neben *Microsoft Visio* die Modellierungssprache System Modeling Language (SysML) verwendet werden. SysML ist eine graphische, semiformale Sprache, basierend auf der Unified Modeling Language (UML). Im Gegensatz zur UML, die für die Informationstechnik definiert wurde, ist SysML auf die ganzheitliche domänenübergreifende Modellierung technischer Systeme ausgerichtet [Wei08, FMS12, Alt12]. Für die Einbindung der CONSENS-Partialmodelle unter SysML ist im Rahmen des Sonderforschungsbereichs 614 (SFB 614) ein SysML4CONSENS-Profil entstanden [IKD+13, KDH+13]. Eine weitere Möglichkeit für die rechnerinterne Abbildung stellt das IT-Werkzeug *Mechatronic Modeller* dar, in dem die entwickelte Spezifikationstechnik ebenfalls prototypisch umgesetzt wurde [GTS+14]. Hierbei wird vor allem Wert auf die durchgängige Modellierung der Partialmodelle gelegt. Weitere vielversprechende Softwarewerkzeuge

[1] Hier ist das ereignisdiskrete Verhalten der Steuerung gemeint.

für die Umsetzung der Spezifikationstechnik sind *Enterprise Architect*, *Papyrus* sowie die *3DEXPERIENCE Plattform* von Dassault [IKD+13, PHM14, KDH+13, KKA+13]. Bei der Erarbeitung der einzelnen Partialmodelle können mehrere Iterationen erfolgen, wodurch speziell die partialmodellübergreifenden Beziehungen beschrieben werden können. Nachfolgend werden die acht Partialmodelle kurz vorgestellt.

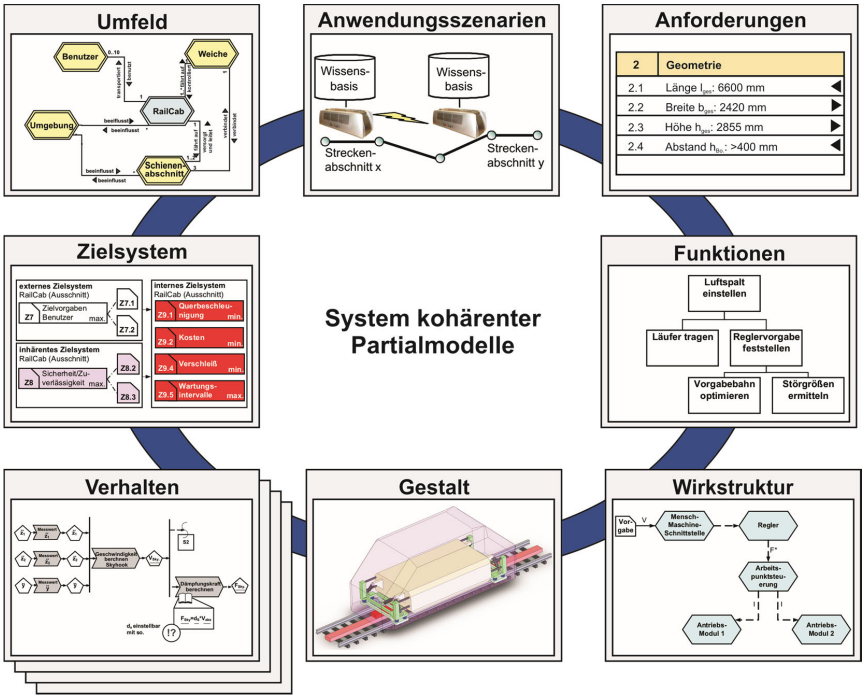

Bild 2-4: Partialmodelle der Spezifikationstechnik CONSENS [GFD+08]

Umfeldmodell: Das zu entwickelnde System wird als Systemelement zunächst in Form einer Black Box dargestellt. Die Ein- und die Ausgänge dieses Systemelementes werden durch die Wechselwirkungen zwischen dem zu entwickelnden System und seiner Umwelt definiert und durch Stoff-, Energie- und Informationsflüsse beschrieben. Störgrößen, die auf das System einwirken, werden gesondert gekennzeichnet.

Anwendungsszenarien: Hierbei werden Situationen aufgenommen, in denen sich das System befinden kann. Diese werden als eine kurze Situationsbeschreibung dokumentiert. Des Weiteren wird festgehalten, wie sich das System in der beschriebenen Situation verhalten soll. Hierdurch können frühzeitig weitere Anforderungen, die das zu untersuchende System erfüllen soll, gewonnen werden.

Anforderungen: Ein wesentliches Ergebnis der Bearbeitung der Partialmodelle stellt die Anforderungsliste dar. In ihr werden alle Anforderungen an das zu

entwickelnde System festgehalten. Sie bildet die Grundlage für die nachfolgende Entwicklung.

Funktionen: Funktionen werden in Form einer Funktionshierarchie aufgelistet. Eine Gesamtfunktion, die das zu entwickelnde System erfüllen soll, wird so lange in Unterfunktionen beschrieben, bis ein sinnvolles Lösungsmuster gefunden werden kann. Um bei der Erarbeitung von neuen, innovativen Prinziplösungen einen möglichst großen Lösungsraum aufspannen zu können, ist es wichtig, dass die erstellte Funktionshierarchie so lösungsneutral wie möglich aufgebaut wird.

Wirkstruktur: Das bei dem Umfeldmodell zunächst als Black Box dargestellte Systemelement wird durch die Wirkstruktur beschrieben. Sie bildet eine konkrete Prinziplösung für das zu entwickelnde System graphisch durch unterschiedliche Systemelemente ab. Bei den Systemelementen in einer Wirkstruktur kann es sich um Systeme, Module, Bauteile oder Softwarekomponenten handeln [GTS+14]. Die Wechselwirkungen der einzelnen Systemelemente werden wie beim Umfeldmodell durch Stoff-, Energie- und Informationsflüsse beschrieben. Störgrößen werden auch hier gesondert gekennzeichnet.

Gestalt: Schon während des Systementwurfes können erste Angaben zur späteren Gestalt des zu entwickelnden Systems getroffen werden. Hierzu können 3D-Computer-aided Design (CAD)-Systeme genutzt werden, um die entsprechenden Modelle zu erstellen.

Verhalten: In diesem Partialmodell wird das gewünschte Verhalten des zu entwickelnden Systems mithilfe von Aktivitäten, Zuständen und Zustandsübergängen beschrieben.

Zielsystem: Das Zielsystem beschreibt die internen und die externen Ziele eines Systems sowie deren Verknüpfungen untereinander. In einer Einflussmatrix werden die Beeinflussungen der Ziele untereinander dargestellt.

Durch die Bearbeitung der Partialmodelle kann während des Systementwurfes implizites Wissen in explizites überführt werden. Wegen der allgemeinverständlichen Beschreibung eignet sich CONSENS für die Bearbeitung in unterschiedlichsten Fachdisziplinen und ist somit fachdisziplinunabhängig einzusetzen.

Das Ergebnis des domänenübergreifenden Systementwurfes stellt die Prinziplösung des zu betrachtenden Systems dar; diese bildet die Ausgangsbasis für den anschließenden fachdisziplinspezifischen Entwurf.

2.4 Modellierung von kontinuierlichen und ereignisdiskreten Systemen

Im fachdisziplinspezifischen Entwurf erfolgt eine detaillierte Ausarbeitung der Prinziplösung in den jeweils beteiligten Fachdisziplinen. Für die Konkretisierung der Prinziplösung sind detaillierte Auslegungen und Berechnungen nötig, um insbesondere bei kritischen Systemfunktionen die Funktionserfüllung sicherstellen zu können. Ein wesentliches Element bei der Detaillierung der Prinziplösung stellt die Modellbildung dar. Bereits während des Systementwurfes werden Modelle des zu untersuchenden Systems erstellt und für Analysen verwendet. Handelt es sich bei den Modellen während des Systementwurfes um idealisierte, werden für den fachdisziplinspezifischen Entwurf detaillierte Modelle erarbeitet. Die Grundlage für diese Modelle stellen die idealisierten Modelle des Systementwurfes dar. Für die jeweilige Erweiterung und Detaillierung der Modelle kommen in den jeweiligen Fachdisziplinen unterschiedliche Modellierungsansätze und Modellierungs-Tools zum Einsatz. Die für diese Arbeit relevanten Modellierungsansätze werden im Folgenden kurz vorgestellt.

2.4.1 Modellierung der kontinuierlichen Dynamik

Unterschiedliche Fachdisziplinen verwenden unterschiedliche Modelle, die speziell auf die für ihre Domäne vorliegenden Probleme bzw. Fragestellungen ausgerichtet sind. Die jeweils vorhandenen Experten können somit Begriffe, Konzepte und Beziehungen direkt aus ihrer Fachdisziplin für die Modellierung verwenden. Bevor auf die domänenspezifischen Modellierungsarten für die Erstellung von kontinuierlichen Dynamikmodellen eingegangen wird, wird zunächst auf die allgemeine Vorgehensweise bei der Erstellung eines Modells eingegangen (vgl. Bild 2-5). Das Vorgehen gilt für das Erstellen sowohl von detaillierten als auch von idealisierten Modellen, wobei bei letzteren die Parameteridentifikation sowie die Modellvalidierung nicht zwangsläufig erfolgen müssen.

Zunächst wird die zu betrachtende technische Anlage zu ihrer Umwelt abgegrenzt. Hierbei erfolgt eine erste Definition von Eingangs-, Ausgangs- und internen Größen, die in einem System abgebildet werden. In einem nächsten Schritt kann das System mithilfe von Apriori-Kenntnissen als physikalisches Ersatzschaltbild dargestellt werden. In diesem Schritt wird der Detaillierungsgrad des Modells festgelegt. Es braucht besonderes Expertenwissen, damit das erzeugte Modell nur die Effekte abbildet, die für die gegebene Aufgabenstellung relevant sind. Grundlegend gilt, dass ein Modell „so einfach wie möglich, so genau wie nötig“ sein soll [Wal95, Rob94, Sal93]. Durch die Verwendung von physikalischen Gesetzen wird anschließend das physikalische Ersatzbild als qualitatives mathematisches Modell ausgedrückt. Unter der Einbeziehung von physikalischen Parametern wird

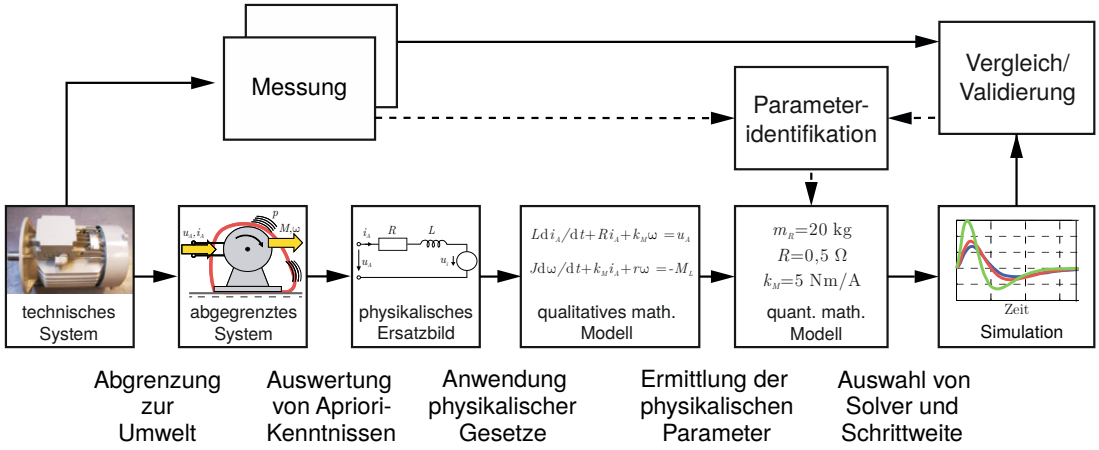

Bild 2-5: Vorgehensweise zur Erstellung von Dynamikmodellen realer Systeme (in Anlehnung an [GTS+14])

das qualitative mathematische Modell in ein quantitatives mathematisches Modell überführt. Die hierzu benötigten Parametergrößen können aus Datenblättern, Messungen oder ähnlichem entnommen werden. Das quantitative mathematische Modell kann mittels geeigneter Solver[2] numerisch gelöst und somit simuliert werden. Die erzeugten Simulationsergebnisse sollten zwingend mit den Ergebnissen von zuvor durchgeführten Messungen an der zu untersuchenden technischen Anlage verglichen werden. Erst durch diesen Schritt, der sogenannten Modellvalidierung, kann sichergestellt werden, dass das erzeugte Modell die reale technische Anlage hinreichend genau abbildet. Sollte dies nicht der Fall sein, können die zuvor eingegebenen Werte der physikalischen Parameter mittels einer Parameteridentifikation neu ermittelt werden, bevor eine erneute Modellvalidierung erfolgt. Sollte das Modell trotz der durchgeführten Parameteridentifikation das reale System nicht hinreichend genau abbilden, muss ein anderer Modellierungsansatz bzw. Detaillierungsgrad des Modells gewählt werden. Ansätze für eine methodische Wahl des Detaillierungsgrads von Dynamikmodellen finden sich zum Beispiel in [LSB+12].

Um ein solches Dynamikmodell zu erstellen, gibt es eine Vielzahl von rechnergestützten Simulationswerkzeugen, die unterschiedliche Modellierungsarten unterstützen. Für diese Arbeit sind besonders die Bereiche Regelungstechnik und Softwaretechnik von Bedeutung, weshalb auf sie genauer eingegangen wird. In der Domäne Regelungstechnik werden im Allgemeinen topologie-, signalfluss- und CAD-orientierte Modellierungen angewendet, um ein Dynamikmodell zu erstellen. Bild 2-6 zeigt exemplarisch einige dieser Modellierungsarten.

[2] Solver sind spezielle mathematische Algorithmen, die mathematische Probleme numerisch lösen können.

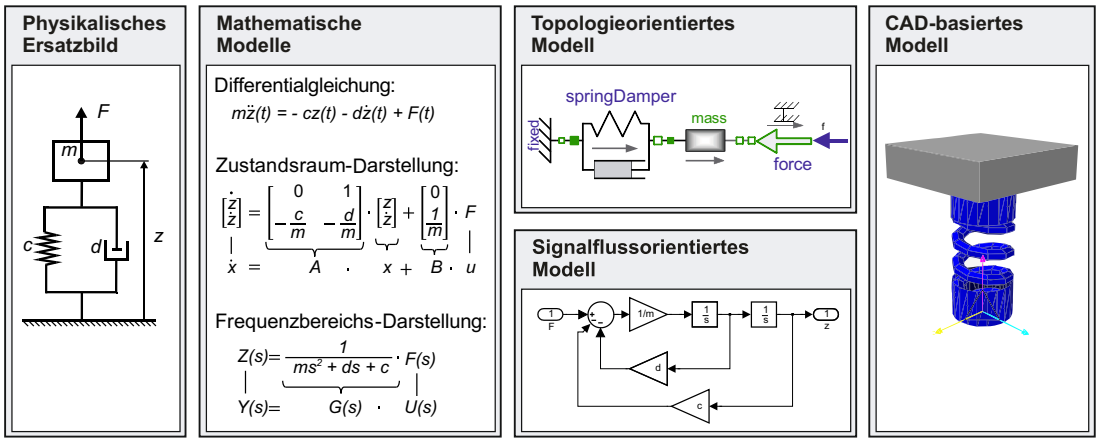

Bild 2-6: Beispielhafte Übersicht von Modellarten des Einmassenschwingers (in Anlehnung an [GTS+14])

Die **signalflussorientierte Modellierung** basiert auf mathematischen Modellen. Es werden Blöcke verwendet, die mithilfe von gerichteten Verbindungen die mathematischen Modelle abbilden. Da es sich um gerichtete Verbindungen handelt, müssen Rückführungen in Form von Schleifen eingebunden werden. Da die signalflussorientierte Modellierung auf mathematischen Modellen beruht, entsteht der Vorteil, dass sie hierdurch fachdisziplinunabhängig angewendet werden kann. Ein möglicher Nachteil besteht darin, dass die mathematische Beschreibung des physikalischen Systems zunächst erfolgen muss, bevor ein signalflussorientiertes Modell erstellt werden kann.

Bei einer **topologieorientierten Modellierung** handelt es sich um eine Modellierung auf physikalischer Ebene. Das System wird auf einer höheren Abstraktionsebene auf Basis des physikalischen Ersatzbildes beschrieben. Die eigentliche Gleichungserstellung und -umformung übernimmt hierbei ein Compiler, weshalb diese aufwändige und fehlerträchtige Arbeit nicht händisch durchgeführt werden muss. Möglich wird diese Form der Modellierung durch die hinterlegte akausale Beschreibung vordefinierter Komponenten, die auf echten Gleichungen statt auf Zuweisungen basiert. Im Gegensatz zur signalflussorientierten Beschreibung werden erst durch die Definition von Systemein- und -ausgängen am Ende des Modellierungsprozesses Ursache und Wirkung festgelegt. Vor allem wegen der steigenden Anzahl an vordefinierten Komponenten, die in Form von Modellbibliotheken (domänenübergreifend) abgelegt und zur Verfügung gestellt werden, ist die topologieorientierte Modellierung für den fachdisziplinübergreifenden Einsatz gut geeignet.

Die **CAD-basierte Modellierung** erfolgt ebenfalls auf der physikalischen Ebene. Sie ist stark domänenspezifisch (Mechanik) und durch spezialisierte Tools

für Fachexperten intuitiv zu bedienen. Die Kopplung zu anderen Disziplinen ist hierbei meist nur eingeschränkt möglich.

2.4.2 Modellierung des ereignisdiskreten Verhaltens

Im Gegensatz zu den zuvor genannten Modellierungsarten, die verstärkt im Bereich der Regelungstechnik verwendet werden, ist eine der verbreitetsten Modellierungssprachen im Bereich der Softwaretechnik die UML. In ihr werden Modelle in Form von unterschiedlichen Diagrammarten dargestellt [ISO05]. Ein wesentlicher Aspekt bei der Verwendung solcher Diagrammarten liegt in der Abbildung des ereignisdiskreten Verhaltens. Hierzu wird nachfolgend das Zustandsdiagramm, basierend auf endlichen Automaten, genauer vorgestellt.

Endliche Automaten und Zustandsdiagramme

Endliche Automaten werden eingesetzt, um ein gewünschtes Verhalten für ein System zu modellieren. Hauptelemente eines endlichen Automaten sind Zustände, Zustandsübergänge und Aktionen. Der Begriff „endlich“ bezieht sich auf die Anzahl der verwendeten Zustände eines solchen Automaten. Zustandsdiagramme (engl. *state charts*) sind eine gebräuchliche Darstellungsform für endliche Automaten. Sie wurden erstmals 1987 von David Harel vorgestellt [Har87]. Bei einem Zustandsdiagramm wird zwischen Entry-, During- und Exit-Aktionen (kurz: en, du, ex) unterschieden. Zustandsübergänge werden durch Transitionen ermöglicht. Diese können mit Bedingungen versehen werden. Entscheidungsknoten ermöglichen es, eine Transition in mehrere aufzuspalten. Die jeweiligen Transitionen unterscheiden sich durch unterschiedliche Bedingungen. Die Reihenfolge, in der die Bedingungen der einzelnen Transitionen überprüft werden sollen, kann flexibel vorgegeben werden. Zustandsdiagramme ermöglichen es, endliche Automaten zu hierarchisieren. Einzelne Zustände können wiederum eigene Zustände oder ganze endliche Automaten beinhalten. Solche übergeordneten Zustände werden als *Superzustände* bezeichnet, während die untergeordneten Zustände *Unterzustände* genannt werden. Die Anordnung von Unterzuständen kann seriell oder parallel erfolgen. Durch diese Kombinationen sind Zustandsdiagramme für die Abbildung komplexer Steuerungssysteme sehr gut geeignet.

Stateflow

Die Umsetzung von Zustandsdiagrammen ist unter anderem in Stateflow® zu finden. Es ist ein kommerzielles Programm von The MathWorks, das eine grafische Erweiterung zu MATLAB-Simulink®[3] darstellt [ABR+11]. Stateflow-Model-

[3] MATLAB®, Simulink® und Stateflow® sind eingetragene Warenzeichen der Firma "The MathWorks, Inc." (www.mathworks.com)

le sind ereignisgesteuert. Ihre enthaltenen Zustände können entweder aktiv oder inaktiv sein. Ein Stateflow-Modell ist immer in ein Simulink-Modell eingebettet und bildet ein eigenes Untersystem. Für die Kommunikation zwischen einem Stateflow- und einem Simulink-Modell werden vor Simulationsbeginn äquivalente Beschreibungen in einer universellen Programmiersprache generiert. Durch dieses Kompilieren erfolgt eine Übersetzung in die Programmiersprache C. Für die Parametrierung von Stateflow-Modellen können Eingangssignale des Modells oder der MATLAB-Workspace verwendet werden, wodurch sich große Datenmengen einfach verarbeiten lassen. Stateflow verwendet eine eigene Syntax, die als Action Language bezeichnet wird. Zudem werden zusätzliche Elemente zum Durchführen von Vergleichen, zum Anstoßen von Aktionen und für das Aufrufen von Funktionen bereitgestellt. Um bei der Modellierung komplexer Systeme Zustände eindeutig identifizierbar zu machen, wird in der Bezeichnung von *Unterzuständen* der übergeordnete Zustand (*Superzustand*) vorangestellt. Die erste ausgeführte Transition der obersten Hierarchieebene sowie aller Unterzustände heißt Standardtransition und besitzt keinen Quellzustand. In Bild 2-7 wird die Hierarchisierung von Zuständen anhand einer beispielhaften Stateflow-Modellierung dargestellt.

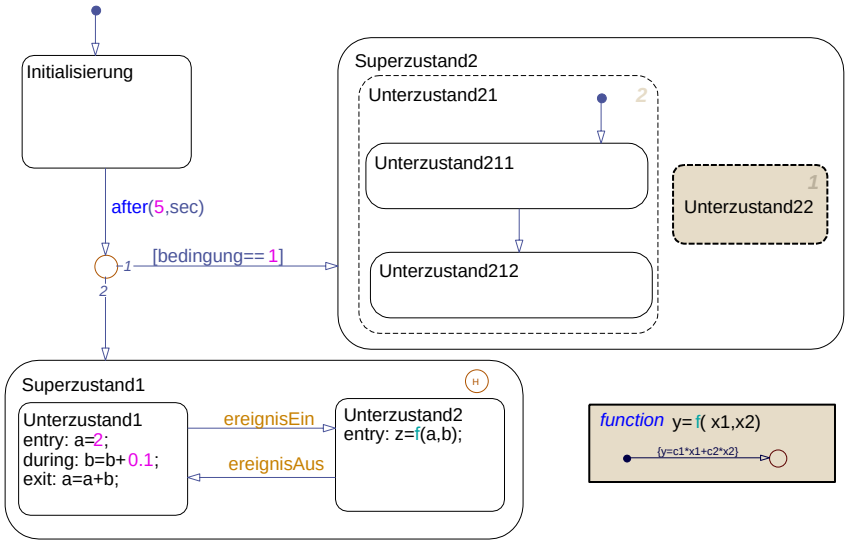

Bild 2-7: Beispielhaftes Zustandsdiagramm in Stateflow

2.4.3 Modellaustausch durch das Functional Mock-up Interface

Aufgrund der Vielzahl an unterschiedlichen Modellierungswerkzeugen und Modellierungsarten für den fachdisziplinspezifischen Entwurf gibt es fortlaufend Arbeiten mit dem Ziel, ein möglichst allgemeingültiges Austauschformat zu definieren bzw. zu erzeugen, mit dessen Hilfe die einzelnen Teilmodelle zu einem

Gesamtmodell zusammengeführt und ganzheitlich simuliert werden können. Eine vielversprechende Lösung für ein solches Austauschformat stellt das Functional Mock-up Interface (FMI) dar.

Bei dem FMI handelt es sich um ein simulationstoolunabhängiges Standardaustauschformat für Dynamikmodelle. Es wurde im Rahmen des ITEA2-MODELISAR-Projektes [FMI12] entwickelt. Initiiert und organisiert wurde das Projekt durch die Daimler AG mit dem Ziel, ein Austauschformat für Dynamikmodelle zwischen Zulieferern und Herstellern zu entwickeln [BOA+11, Cho13]. Bild 2-8 zeigt diesen Austausch exemplarisch. Partnerunternehmen bei diesem Projekt waren unter anderem Atego, Bosch GmbH, Dassault Systèmes SE und Siemens AG. 2010 wurde die erste Version (FMI 1.0) veröffentlicht.

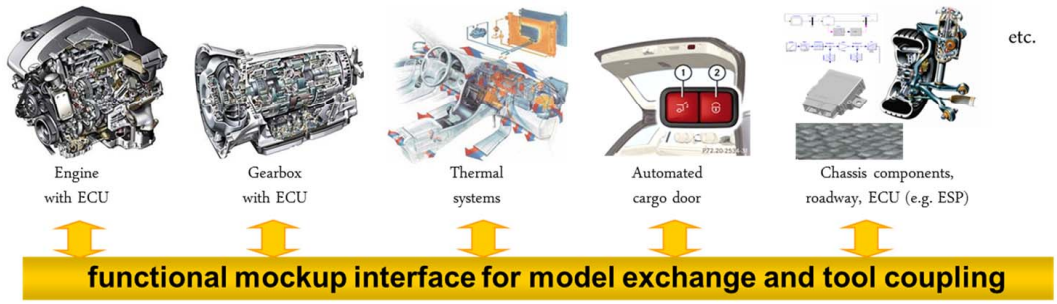

Bild 2-8: Möglichkeit zur Kopplung unterschiedlicher Modelle mittels des FMI-Standards [BOA+11]

Die Grundidee ist, ein erstelltes Modell als C-Code zu exportieren und somit für andere Simulationstools zur Verfügung zu stellen. Bei dem Export eines Modells entsteht ein zip-File mit der Endung *.fmu*, was für Functional Mock-up Unit (FMU) steht. Das zip-File beinhaltet den erzeugten C-Code in Form einer Dynamic Link Library (DLL) sowie einer systembeschreibenden Extensible Markup Language (XML) mit allen notwendigen Modellinformationen. Eine FMU kann als Co-Simulation (mit Solver) oder als Model Exchange (ohne Solver) durch das jeweilige Simulationstool exportiert werden (vgl. Bild 2-9). Weiterentwickelt wird das FMI von der Modelica Association [Mod09]. Es wird derzeit von über 35 Simulationstools unterstützt, deren Zahl weiter steigt [FMI12].

In dem bisher dargestellten Stand der Technik wurde zunächst der grundsätzliche Aufbau mechatronischer Systeme erläutert. Darauf aufbauend sind etablierte Entwicklungsmethodiken vorgestellt worden. Diese bilden den Grundbaustein für die in dieser Arbeit entwickelte Methodik, damit lösungsneutrale Anforderungen an ein zu entwickelndes System strukturiert erarbeitet werden können und somit bei der Erarbeitung von neuen, innovativen Prinziplösungen ein möglichst großer

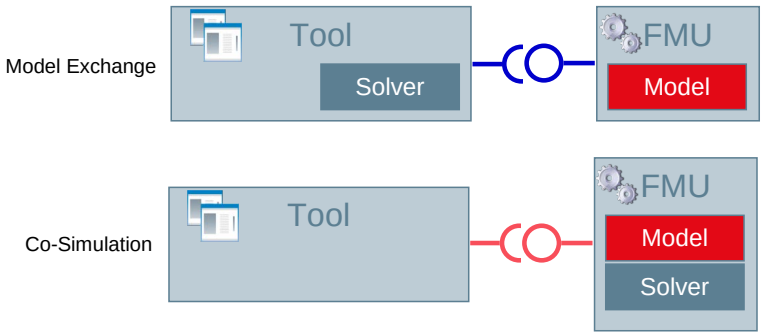

Bild 2-9: FMI Model Exchange und Co-Simulation (in Anlehnung an [FMI12])

Lösungsraum aufgespannt werden kann (Innovationspotenzial ausschöpfen). Demonstriert wird diese Methodik anhand eines Praxisbeispiels (vgl. Kapitel 3), bei dem die zuvor gezeigten Modellierungsansätze eine wesentliche Rolle spielen. Im nachfolgenden Teil wird der zugrundeliegende Stand der Technik für den zweiten wichtigen Aspekt dieser Arbeit vorgestellt, eine in das Vorgehen integrierte Parameteridentifikations- und Modellvalidierungsmethodik. Auch diese wird anhand von ausgewählten Beispielen demonstriert (vgl. Kapitel 4 und Kapitel 5).

2.5 Modellvalidierung und Parameteridentifikation

Werden während des Systementwurfes idealisierte Modelle verwendet, um einen grundsätzlichen Nachweis zu erbringen, dient die Analyse der detaillierten Modelle während des domänenspezifischen Entwurfes dazu, belastbare Aussagen und Erkenntnisse zu liefern. Diese können dem Modell erst nach der Modellvalidierung entnommen werden. Im Unterschied zur Modellverifikation, die besagt, ob ein Modell grundsätzlich plausibel bzw. richtig ist, wird bei der Modellvalidierung sichergestellt, dass ein erstelltes Modell das zu betrachtende System hinsichtlich der zuvor gestellten Anforderungen hinreichend genau beschreibt [Ise08]. Um dies sicherzustellen, kommen Identifikationsverfahren zum Einsatz, die im Folgenden genauer beschrieben werden.

Identifikationsverfahren

Je nach Anforderung gibt es eine große Vielzahl von Identifikationsmethoden, die sich in den letzten Jahren entwickelt haben. Literatur ist auf diesem Gebiet zahlreich zu finden. Dabei sind wichtige renommierte Autoren Eykhoff [Eyk74], Strobel [Str75], Leonhard [Leo73], Young [You84], Ljung [Lju87], Söderström [LS86] sowie Isermann [Ise99, Ise08]. Während Eykhoff einen eher generellen Überblick

über die Identifizierung bereitstellt, präsentiert Söderström das Thema sehr theoretisch. Rekursive Identifikationsmethoden werden vor allem bei Söderström und Ljung vorgestellt [LS86]. In der vorliegenden Arbeit wird im Wesentlichen auf die Werke von Isermann [Ise99, Ise08] und Ljung [Lju87] eingegangen, da diese sich vermehrt mit der Parameteridentifikation mittels Optimierungsverfahren beschäftigen, was ein wichtiger Aspekt in dieser Arbeit ist.

Nach Isermann [Ise08] kann grob zwischen Methoden für parametrische (Gleichungen, welche die Parameter explizit erhalten) und nichtparametrische Modelle (Funktionen in Form von Wertetafeln oder Kurvenverläufen) unterschieden werden. Bei nichtparametrischen Modellen ist die Modellstruktur nicht a priori festgelegt. Hierbei bieten sich beispielsweise die Identifikationsmethoden über eine Frequenzgangmessung [Nye06], eine Fourieranalyse oder über eine Korrelationsanalyse an, was jedoch einen linearisierbaren Prozess voraussetzt.

Bei parametrischen Modellen muss eine bestimmte Modellstruktur angenommen werden. Ermöglicht diese Struktur das Abbilden eines realen Verhaltens durch entsprechendes Variieren der enthaltenen Parameter, so ist aufgrund der berücksichtigten Apriori-Informationen ein genaueres Ergebnis zu erwarten als mit den nichtparametrischen Methoden [Ise08]. Eine der einfachsten Methoden bei der Identifikation parametrischer Modelle ist die Kennwertermittlung. Dabei werden aus gemessenen Antwortfunktionen auf nichtperiodische Testsignale Kennwerte ermittelt. Diese lassen sich anhand von Tabellen und Diagrammen für einfache Modelle bestimmen. Ein solcher Kennwert kann z. B. die Knickfrequenz eines Verzögerungsgliedes darstellen.

Sollten nichtlineare Prozesse identifiziert werden müssen, zu denen kein wesentliches Vorwissen in Bezug auf die Modellstruktur vorliegt, so eignet sich die Verwendung von neuronalen Netzen [Sch10]. In dieser Arbeit werden hauptsächlich nichtlineare Dynamikmodelle betrachtet, deren Modellstrukturen bekannt sind, weshalb auf das Verwenden von neuronalen Netzen nicht genauer eingegangen wird.

Für parametrische Modelle mit bekannter Modellstruktur eignen sich vor allem die Parameterschätzmethoden. Hierbei werden zu identifizierende Parameter P_i[4] eines Modells variiert, bis das Modell ein vorgegebenes Verhalten hinreichend genau abbildet. Hierzu benötigt es Referenzgrößen, die mit den entsprechenden Simulationsgrößen des zu untersuchenden Modells abgeglichen werden. Ein solcher Abgleich erfolgt durch die Minimierung der Differenz Δ_{RefSim} zwischen den Referenzgrößen $\vec{Y}_{Ref}$ und den simulierten Größen $\vec{Y}_{Sim}$ (vgl. Bild 2-10). Die hierzu benötigten Referenzgrößen können zum Beispiel in Form von realen Messungen oder von Simulationsgrößen eines Referenzmodells vorliegen. Wichtig ist bei den

[4] Der Index i steht für den jeweiligen Identifikationsparameter.

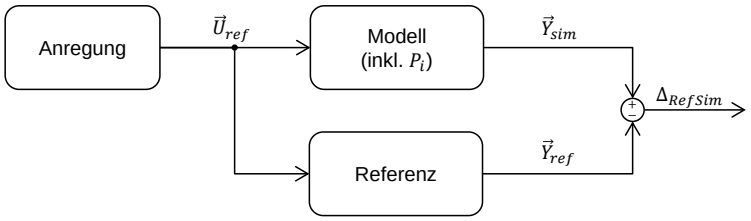

Bild 2-10: Schematische Darstellung eines Vergleichs von Simulations- und Referenzgrößen

Referenzgrößen, dass die zu identifizierenden Parameter P_i Einfluss auf sie besitzen. Sollte es sich um eine Vielzahl von zu identifizierenden Parametern handeln, bietet es sich an, unterschiedliche Referenzgrößen einzubinden, bei denen jeweils nicht alle Parameter Einfluss haben. Somit kann eine sequentielle Identifikation stattfinden. Dieser Ansatz wird in der vorliegenden Arbeit verwendet und detailliert an verschiedenen Beispielen demonstriert (siehe Abschnitt 4.3).

Für die Parameteridentifikation selbst gibt es verschiedene Methoden, von denen *trial and error* weit verbreitet ist. Hierbei werden die Identifikationsparameter manuell variiert, bis sich ein gewünschtes Verhalten einstellt. Dieses Vorgehen bedarf genauester Apriori-Kenntnisse über das zu untersuchende Modell und bietet sich nur bei einer geringen Anzahl an zu identifizierenden Parametern an, da es anderenfalls sehr zeitaufwändig ist. Aufgrund der steigenden Komplexität mechatronischer Systeme und der damit zusammenhängenden Verwendung von Multidomänen-Modellen steigt die Anzahl zu identifizierender Parameter, weshalb eine Parameteridentifikation mittels *trial and error* nicht in angemessener Zeit durchzuführen ist. Alternativ ist eine Parameteridentifikation mittels Optimierungsverfahren möglich. Da diese einen wesentlichen Bestandteil dieser Arbeit darstellen, wird die Parameteridentifikation mittels Optimierungsverfahren nachfolgend genauer beschrieben.

2.6 Parameteridentifikation mittels Optimierung

Das Ziel einer Optimierung besteht darin, optimale Parameter $\vec{P}$ für das Minimieren oder Maximieren einer oder mehrerer Zielfunktionen $\vec{f}(\vec{P})$ zu finden. Bei Identifikationsverfahren bestehen die Zielfunktionen aus einer Auswertung von Mess- und Simulationsgrößen. Hierbei müssen keine Apriori-Informationen über das Modell vorliegen, das die Simulationsgrößen generiert. Als Auswertung der

Zielfunktion $f(\vec{P})$ wird häufig die quadratische Fehlerfläche der Differenz Δ_{RefSim} zwischen Messungen $\vec{Y}_{Ref}$ und Simulationswerten $\vec{Y}_{Sim}$ verwendet, die sich durch

$$f(\vec{P}) = \int_{t_{Start}}^{t_{End}} \Delta^2_{RefSim}(\vec{P}, t)dt = \int_{t_{Start}}^{t_{End}} (\vec{Y}_{ref}(t) - \vec{Y}_{sim}(\vec{P}, t))^2 dt \qquad (2\text{-}1)$$

beschreiben lässt (vgl. Bild 2-11). Ein möglicher Vorteil bei der Verwendung von Optimierungsverfahren liegt in einer verkürzten Entwicklungszeit und darin, dass kein Experte für das Gesamtmodell vorhanden sein muss, der alle Details zum Modell kennt. Jedoch muss berücksichtigt werden, dass es meist keine Garantie für ein Optimum gibt und dass dieses, falls vorhanden, evtl. nur mit erhöhtem Rechenaufwand gefunden werden kann, was dem möglichen Vorteil einer verkürzten Entwicklungszeit entgegenwirkt. Jedes Optimierungsverfahren benötigt zudem spezielle Einstellungen und besitzt Vor- und Nachteile für den jeweiligen Anwendungsfall.

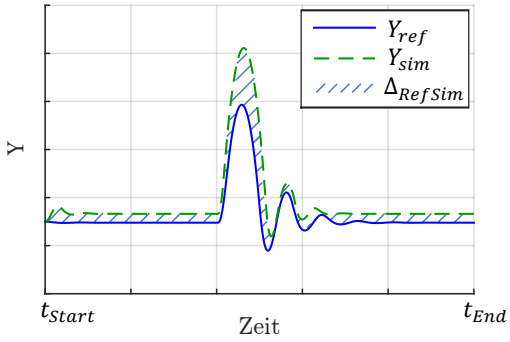

Bild 2-11: Schematische Darstellung einer Fehlerfläche zwischen zwei Funktionen

Aufgrund der unterschiedlichen Einsatzbereiche in denen Optimierungsverfahren eingesetzt werden können, sind diese zahlreich (vgl. Bild 2-12). Sie unterscheiden sich grundlegend. Nachfolgend werden exemplarisch einige Optimierungsverfahren genannt und deren Unterschiede kurz erläutert.

Um lineare Optimierungsverfahren handelt es sich, wenn sowohl die Zielfunktion als auch die Nebenbedingungen durch ein System linearer Gleichungen und Ungleichungen beschrieben werden. Entsprechend wird von nichtlinearer Optimierung gesprochen, wenn entweder die Zielfunktion oder die Nebenbedingungen oder beide nichtlineare Gleichungen bzw. Ungleichungen aufweisen [Kun11].

Bei Optimierungsverfahren mit Nebenbedingungen können für die zu optimierenden Parameter Wertebereiche vorgegeben werden. Diese werden durch Beschrän-

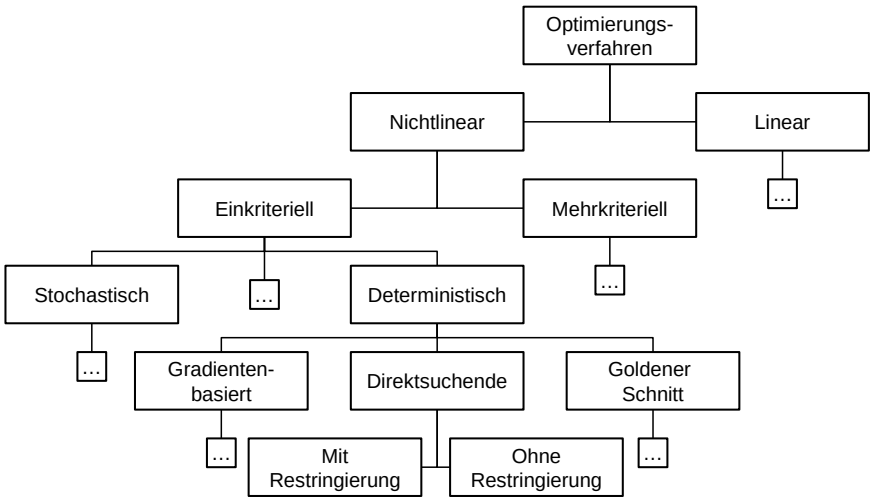

Bild 2-12: Ein Beispiel unterschiedlicher Optimierungsverfahren (in Anlehnung an [Mor95])

kungen berücksichtigt, was man in der Literatur häufig als Optimierungsverfahren mit Restringierung (engl. *constrained optimization*) bezeichnet [NW99]. Bei der Verwendung von Optimierungsverfahren ohne Restringierung ist darauf zu achten, dass die zu identifizierenden Parameter keinen vorgegebenen Wertebereich einhalten müssen. Ist ein fester Wertebereich einzuhalten, so sollten keine Optimierungsverfahren ohne Restringierung verwendet werden, da die Ergebnisse zum einen physikalisch unplausibel werden können (z. B. negative Massen) und es zum anderen zu Instabilitäten während der Simulation kommen kann (z. B. durch Null teilen). Hierzu muss jedoch das Wissen über die einzuhaltenden Grenzen der jeweiligen Identifikationsparameter vorhanden sein.

Einkriterielle Optimierungsverfahren basieren auf der Optimierung einer Zielfunktion, wohingegen mehrkriterielle Optimierungsverfahren mehrere Zielfunktionen besitzen. In dieser Arbeit werden nur die einkriteriellen verwendet, daher werden die mehrkriteriellen Optimierungsverfahren nicht vorgestellt. Ein wichtiger Aspekt bei Optimierungsverfahren ist die Skalierung der Identifikationsparameter, die nachfolgend vorgestellt wird.

2.6.1 Skalierung der Parameter

Der Wertebereich der zu identifizierenden Parameter kann stark unterschiedlich sein. Dies wird exemplarisch an den Wertebereichen für eine Federsteifigkeit P_c und eine ungestreckte Federlänge P_s in Bild 2-13 durch die jeweilige obere und untere Grenze gezeigt. Für eine adäquate Optimierung von Parametern hinsichtlich

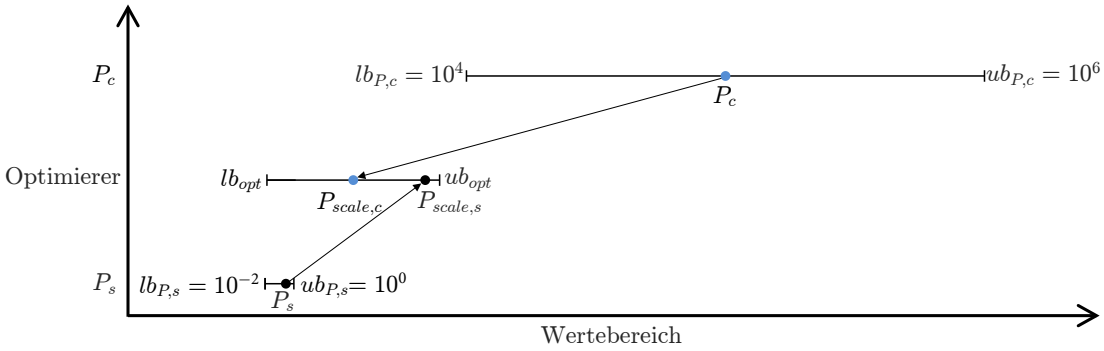

Bild 2-13: Schematische Darstellung der Skalierung von Identifikationsparametern

der Minimierung einer Zielfunktion ist es notwendig, den Raum der Parameter angemessen zu skalieren. Besonders gradientenbasierte Optimierungsverfahren sind abhängig von einer Skalierung [Mar63, Mat14]. Hierzu wird zunächst ein Wertebereich für das anzuwendende Optimierungsverfahren von lb_{opt} bis ub_{opt} festgelegt. Die jeweiligen Parameter-Wertebereiche werden auf diesen skaliert, bevor sie an das Optimierungsverfahren übergeben werden. Bild 2-13 zeigt schematisch eine entsprechende Skalierung der Parameter P_c und P_s zu den Parametern $P_{scale,c}$ und $P_{scale,s}$, die sich in dem Wertebereich des Optimierungsverfahrens befinden.

Der für die jeweilige Skalierung benötigte Skalierungsfaktor F_{scale} berechnet sich nach folgender Gleichung:

$$F_{scale,i} = \frac{ub_{P,i} - lb_{P,i}}{ub_{opt} - lb_{opt}}. \tag{2-2}$$

Bild 2-14 zeigt schematisch das Vorgehen für die Parameteridentifikation mit der entsprechenden Skalierung und Rückskalierung. Zunächst erfolgt die Simulation mit den Eingangswerten $\vec{U}_{Ref}$ über den vorgegebenen Simulationsbereich t_{Start} bis t_{End} und mit den initialen Werten der Identifikationsparameter $\vec{P}_{init}$, die sich in dem Simulationsmodell befinden. Mithilfe der Referenzgrößen $\vec{Y}_{Ref}$ und der simulierten Größen $\vec{Y}_{Sim}$ findet die Auswertung der Zielfunktion statt, und $f(\vec{P})$ als Ergebnis der Zielfunktion wird an das Optimierungsverfahren übergeben. Zusätzlich findet initial eine Skalierung der zu identifizierenden Parameter

$$P_{scale,i} = lb_{opt} + \frac{(P_i - lb_{P,i})}{F_{scale,i}} \tag{2-3}$$

statt, die ebenfalls an das Optimierungsverfahren übergeben werden. Mit den initial skalierten Parametern $\vec{P}_{scale,init}$, dem berechneten Wert der Zielfunktion $f(\vec{P})$ sowie den gewählten Einstellungen für das Optimierungsverfahren $TolX$, $TolFun$,..., auf die später genauer eingegangen wird (siehe Bild 2-20), startet die Optimierung. Ist diese erfolgt, so wird entsprechend rückskaliert:

$$P_i = (P_{scale,i} - lb_{opt}) \cdot F_{scale,i} + lb_{P,i},$$

und die Simulation mit den jeweils variierten Parametern $\vec{P}_{var}$ erneut durchgeführt. Diese Iteration erfolgt so lange, bis entsprechende Abbruchkriterien, die durch die Variablen $TolX$, $TolFun$,... vorgegeben werden, erfüllt sind. Mittels einer anschließenden Rückskalierung liegen optimierte Parameter $\vec{P}_{opt}$ vor, und die Parameteridentifikation endet (vgl. Bild 2-14). In Kapitel 4 wird dieses Vorgehen an einem Beispiel genauer vorgestellt.

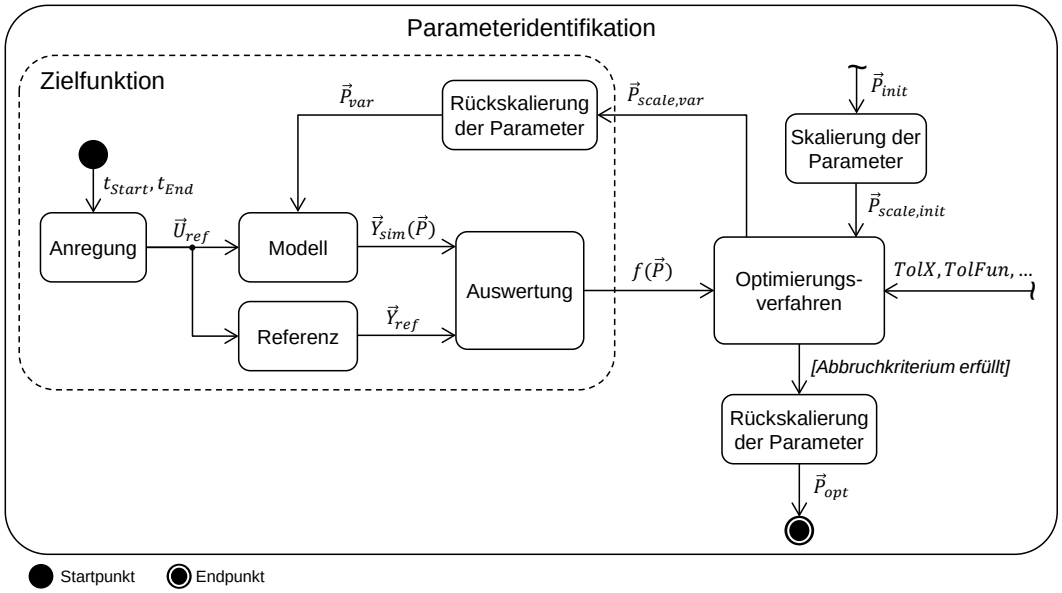

Bild 2-14: Schematische Darstellung des Vorgehens bei der Parameteridentifikation mit entsprechender Parameter-Skalierung und -Rückskalierung

Bei der zuvor angesprochenen Unterscheidung zwischen linearen und nichtlinearen Optimierungsverfahren werden in dieser Arbeit, aufgrund der Einbindung von nichtlinearen Multidomänen-Modellen, die linearen nicht genauer beschrieben. Bei den nichtlinearen Optimierungsverfahren kann des Weiteren zwischen deterministischen, stochastischen und Hybrid-Verfahren unterschieden werden (vgl.

Bild 2-12). Nachfolgend werden die deterministischen Optimierungsverfahren genauer vorgestellt, da sie im weiteren Verlauf der Arbeit angewendet werden. Weitere Informationen zu den anderen Verfahren sind unter anderem in [NW99] und [Mat14] zu finden.

2.6.2 Nichtlineare, deterministische Optimierungsverfahren

Deterministische Verfahren zeichnen sich dadurch aus, dass sie immer ein identisches Verhalten aufweisen, wenn man sie an einem gleichen Startpunkt beginnen lässt. Sie lassen sich in die Bereiche der gradientenbasierten (indirekten) und der direktsuchenden Verfahren einteilen. Des Weiteren gibt es noch ein Verfahren, das auf dem Prinzip des sog. Goldenen Schnittes beruht (siehe Abschnitt 2.6.2). Der jeweilige Unterschied der einzelnen Verfahren liegt im Wesentlichen in der Art, wie die Suchrichtung bestimmt wird.

Nachfolgend werden hierzu ausgewählte Beispiele exemplarisch beschrieben. Hierbei handelt es sich um die Verfahren, die in die entwickelte Parameteridentifikations- und Modellvalidierungsmethodik integriert sind und somit im weiteren Verlauf dieser Arbeit angewendet werden.

Gradientenbasierte Verfahren

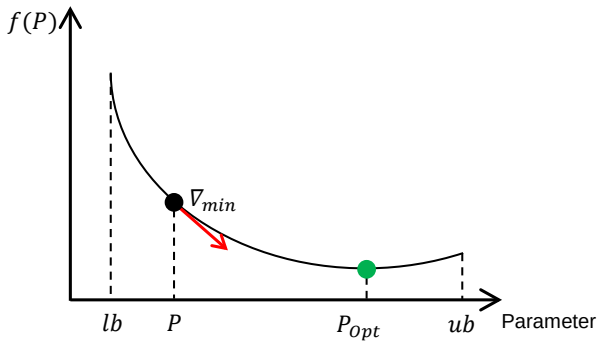

Bild 2-15: Schematische Darstellung der Suche eines Punktes zur Minimierung einer Zielfunktion mithilfe gradientenbasierter Verfahren

Die gradientenbasierten Verfahren bilden bei der Suche nach dem Optimum einer Zielfunktion $f(P)$ den Gradienten $\nabla f(P)$ an einem Punkt P (vgl. Bild 2-15). Die Nutzung dieses Gradienten wird auf unterschiedlichste Art und Weise zum Fortschreiten von einem initialen Startpunkt zu einem Optimum genutzt. Zwei fundamentale Strategien sind bei den gradientenbasierten Verfahren die *Trust-Region (TR)*- und die *Line-Search (LS)*-Methoden [NW99]. Viele Algorithmen

basieren auf diesen Methoden, weshalb sie an dieser Stelle genauer beschrieben werden.

Bei der *TR*-Methode wird die Zielfunktion $f(P)$ durch eine quadratische Funktion

$$f(P^k + d) = f(P^k) + \nabla f(P^k)^T d + \frac{1}{2} d^T H_k d$$

um den aktuellen Punkt P^k mit der Richtungsvorgabe d und unter Zuhilfenahme der Hessematrix H_k approximiert. Da die Funktion für alle P, die weit entfernt von P^k liegen, keine gute Approximation von $f(P)$ darstellt, wird die Suche eines Minimums auf eine Region um P^k beschränkt. Ein Kandidat für eine Schrittweite wird gefunden, indem folgendes Subproblem in Abhängigkeit der Richtung d gelöst wird:

$$min_d f(P^k + d).$$

Sollte die Lösung für den Schrittweitenkandidaten keine hinreichende Minimierung von $f(P)$ bewirken, ist die Vertrauensregion zu groß. Folglich wird diese verkleinert und das Problem wiederholt gelöst. Die Vertrauensregion kann dabei verschiedene Formen annehmen (elliptisch, box-förmig, kugelförmig) [NW99]. Ein wichtiger Vertreter dieser Methode ist das *Sequential Quadratic Programming (SQP)*.

Bei der *LS*-Methode wählt der Algorithmus in jeder Iteration eine Richtung d_k und folgt somit der Richtung des steilsten Abstiegs. Die Distanz, die dabei in Richtung d_k gelaufen werden soll, kann näherungsweise durch folgendes Minimierungsproblem gefunden werden, wobei die Schrittweite durch α dargestellt wird:

$$min_{\alpha>0} f(P_k + \alpha d_k).$$

Da eine genaue Lösung des Minimierungsproblems zu aufwändig wäre, wird stattdessen eine begrenzte Anzahl von Versuchsschrittweiten erzeugt, wovon eine letztlich das Minimum annähernd erreicht [NW99].

LS-Methode wählen erst eine Suchrichtung d; entlang dieser Richtung (line) wird anschließend eine Schrittweite α bestimmt (search). Bei *TR*-Methoden wird die Zielfunktion zunächst durch eine quadratische Funktion approximiert. Innerhalb eines Gebietes (region) vertraut (trust) man auf die Güte dieser Approximation

und legt dadurch eine maximale Schrittweite fest. Dann erst bestimmt man die beste Suchrichtung. Daher benötigen die *LS*-Methoden einen qualitativ besseren Anfangsschrittvektor als die *TR*-Methoden, da letztere im Laufe des Verfahrens die Suchrichtung ändern können. Für die Ermittlung der Suchrichtung gibt es zwei wesentliche Verfahren, das *Newton-* und das *Quasi-Newton (QN)-Verfahren.* Mehr Informationen diesbezüglich sowie eine detaillierte Beschreibung weiterer Methoden für gradientenbasierte Verfahren, wie unter anderem die *Active-Set (AS)-*, *Interior-Point (IP)-* sowie die *Levenberg-Marquardt (LM)*-Methoden, sind zum Beispiel in [NW99] zu finden.

Direktsuchende Verfahren

Neben den gradientenbasierten Verfahren gibt es Optimierungsverfahren, die ohne die Berechnung eines Gradienten die Minimierung einer Zielfunktion vornehmen können. Diese werden als direktsuchende Verfahren bezeichnet. Die wichtigsten Methoden stellen dabei die *Pattern Search-* sowie die *Nelder-Mead-Simplex*-Methode dar.

Der *Pattern Search* gehört zur Familie der ableitungsfreien, numerischen Optimierungsmethoden. Er legt ein festes Muster (Pattern) über einen aktuellen Punkt im Suchraum, um entlang des Musters einen neuen Punkt zu finden, der eine Zielfunktion minimiert. Die Dimension des Musters ist dabei abhängig von der Wahl des Solvers und entspricht entweder $n+1$ oder $2n$, wobei n die Dimension des Basisvektors angibt. Ist ein solcher Punkt gefunden, so stellt dieser den Mittelpunkt des neuen Musters dar, entlang dessen weiter gesucht wird. Sollte kein Punkt zur Minimierung entlang des Musters gefunden werden, so wird die Suchschrittweite bzw. die Mustergröße halbiert und erneut gesucht. Die Suche endet, sobald eine minimale Schrittweite Δk unterschritten wird. Bild 2-16 zeigt schematisch die Suche eines Punktes zur Minimierung einer Zielfunktion im dreidimensionalen Raum $n = 3$.

Ein großer Vorteil dieser Methode ist, dass für die Ausführung des nächsten Optimierungsschrittes nur eine kleine Verbesserung bzw. Verkleinerung der Zielfunktion $f(P)$ nötig ist, wodurch der nächste Punkt (rot) schnell gefunden wird. Die *Pattern Search*-Methode ist aufgrund ihres großen Anwendungsbereiches und ihrer geringen Komplexität so populär geworden. Trotz des simplen Algorithmus weist sie nach [Lew98] immer lokale Konvergenz auf, die annähernd mit der von *LS-* und *TR*-Algorithmen mithalten kann. Somit ist der *Pattern Search* sehr gut geeignet, um eine initiale Schätzung zu verbessern. Dabei darf jedoch nicht von einer schnellen Konvergenz ausgegangen werden. Die Konvergenzrate ist lediglich linear [Lew98].

Neben dem *Pattern Search* ist *Nelder-Mead-Simplex* eine weitere, häufig verwendete Methode für die direktsuchenden Verfahren. Die Grundidee des Verfahrens

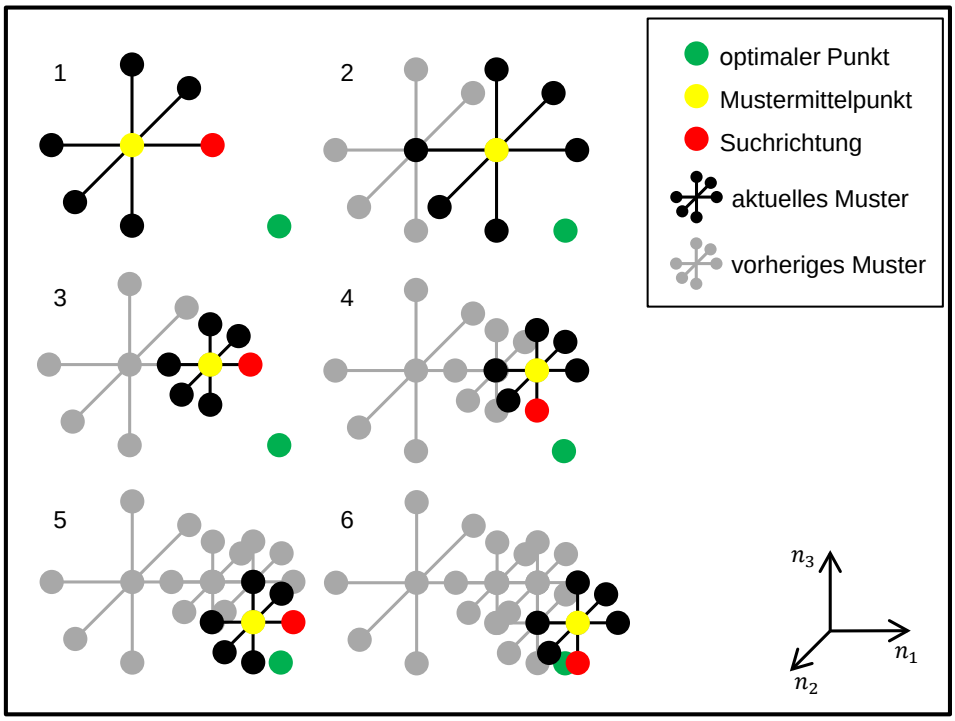

Bild 2-16: Schematische Darstellung der Suche eines Punktes zur Minimierung einer Zielfunktion im dreidimensionalen Raum mithilfe der Pattern Search-Methode (in Anlehnung an [BBB+14])

ist, ausgehend von einem initialen Simplex, den Punkt mit dem schlechtesten Zielfunktionswert zu finden und diesen zu entfernen. Die restlichen Punkte bilden einen Simplex, um dessen Schwerpunkt der zuvor entfernte Punkt reflektiert wird. Der Abstand des reflektierten Punktes kann dabei variiert werden, um einem möglichen optimalen Bereich näher zu kommen. Dieser Vorgang wird iterativ wiederholt, bis keine Verbesserung mehr erreicht werden kann. Eine detaillierte Beschreibung dieser Methode ist unter anderem in [NM65, WWH14] zu finden.

Goldener-Schnitt-Verfahren mit parabolischer Interpolation

Neben den indirekt- und direktsuchenden gibt es das *Goldener-Schnitt-Verfahren mit parabolischer Interpolation*. Dieses beruht ebenfalls auf der zuvor vorgestellten *LS*-Methode, benötigt jedoch keinen Gradienten [NW99]. Dieses Verfahren gilt nur für die Optimierung eines Parameters. Dessen Startwert spielt hierbei keine Rolle. Bild 2-17 zeigt schematisch das Vorgehen bei diesem Verfahren.

Zunächst wird das Suchintervall durch die obere (*ub*) und untere Grenze (*lb*) beschränkt und mithilfe des *Goldener-Schnitt-Verhältnisses* ϕ= 1.6180... in drei

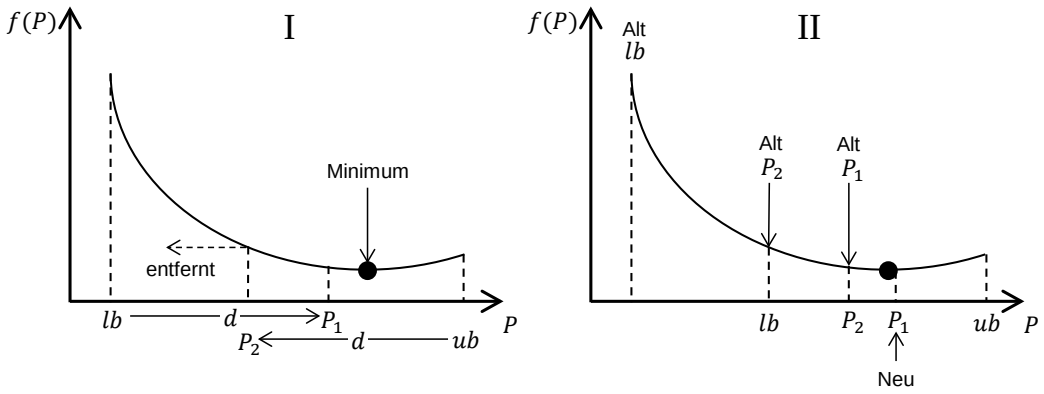

Bild 2-17: Schematische Darstellung des Goldener-Schnitt-Verfahrens (angelehnt an [Mar09])

Teilintervalle zerlegt (vgl. Gleichung 2-4) [Mar09]. Als *Goldenen Schnitt* bezeichnet man das Teilungsverhältnis einer Größe, bei dem das Verhältnis des Ganzen zu seinem größten Teil dem Verhältnis des größten zum kleineren Teil entspricht.

$$\begin{aligned} d &= (\phi - 1) \cdot (ub - lb) \\ P_1 &= lb + d \\ P_2 &= ub - d \end{aligned} \tag{2-4}$$

Falls der Funktionswert von P_1 kleiner als der Funktionswert von P_2 ist (vgl. Bild 2-17 (I)), so wird aus P_2 die neue untere Grenze, und aus P_1 wird P_2. Anderenfalls würde P_1 die neue obere Grenze, und aus P_2 würde P_1. Dieser Vorgang erfolgt iterativ, um sich dem Minimum zu nähern.

Neben dem *Goldener-Schnitt-Verfahren* kann zusätzlich die *parabolische Interpolation* zum Auffinden eines Minimums verwendet werden (vgl. Bild 2-18). Hierzu wird die tatsächliche Funktion durch eine mittels drei Stützstellen berechnete Parabel angenähert. Ein neuer Punkt P_4 wird berechnet und zusammen mit den zwei ihn umgebenden Stützstellen für die nächste Iteration verwendet.

Der große Anwendungsbereich, in dem Optimierungsverfahren eingesetzt werden können, hat zu einer großen Vielfalt an Optimierungsmethoden geführt, die wiederum in unterschiedlichsten IT-Werkzeugen zur Verfügung stehen. Jedes IT-Werkzeug hat dabei seine eigenen Vorgaben, wie der jeweilige Optimierer und die entsprechenden Referenz- und Simulationsgrößen anzuwenden bzw. einzubinden sind. In dieser Arbeit wird das IT-Werkzeug MATLAB zur Identifikation

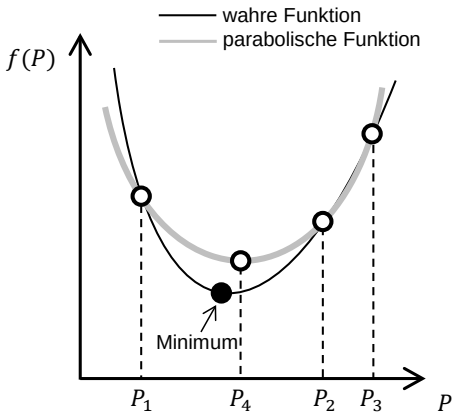

Bild 2-18: Parabolische Interpolation (angelehnt an [Mar09])

verwendet, da es ein etabliertes Tool in dem ingenieurtechnischen Bereich darstellt. Zudem beinhaltet es eine Vielzahl an etablierten Optimierungsverfahren und bietet des Weiteren sehr gute Möglichkeiten zur Datenaufbereitung (Filtern, Dateninterpolation, Datenextrapolation ...), was für die Einbindung von Referenzgrößen ebenfalls von entscheidender Bedeutung ist. Nachfolgend wird auf die Anwendung von MATLAB-Optimierungsverfahren genauer eingegangen.

2.6.3 MATLAB-Optimierungsverfahren

MATLAB stellt eine Reihe von etablierten Optimierungsverfahren zur Verfügung. Wie zuvor genannt (vgl. Bild 2-12, S. 24), kann hier unter anderem zwischen linearen und nichtlinearen oder deterministischen und stochastischen Verfahren unterschieden werden. Eine weitere wichtige Unterscheidung besteht in der Verfügbarkeit. Dabei gibt es bereits vorhandene Optimierungsverfahren sowie solche, für die zusätzlich kostenpflichtige Toolboxen benötigt werden. Hierbei handelt es sich um die „*Optimization Toolbox*“ und die „*Global Optimization Toolbox*“. Mit Hilfe dieser kostenpflichtigen Toolboxen bietet MATLAB zudem die Möglichkeit, die Optimierungsverfahren für Modelle unter MATLAB mittels graphischer Benutzeroberflächen anzuwenden. Bild 2-19 zeigt einen Ausschnitt der graphischen Benutzeroberfläche des Optimization Tools unter MATLAB. Diese bietet die Auswahl umfangreicher Einstellungen für die Optimierungsverfahren, wodurch es dem Experten ermöglicht wird, die optimale Einstellung – für ein unter MATLAB vorliegendes Modell – auszuwählen.

Wie schon zuvor werden nachfolgend nur die zur Verfügung stehenden nichtlinearen, deterministischen, einkriteriellen Optimierungsverfahren von MATLAB sowie einige der möglichen Einstellungen genauer dargestellt.

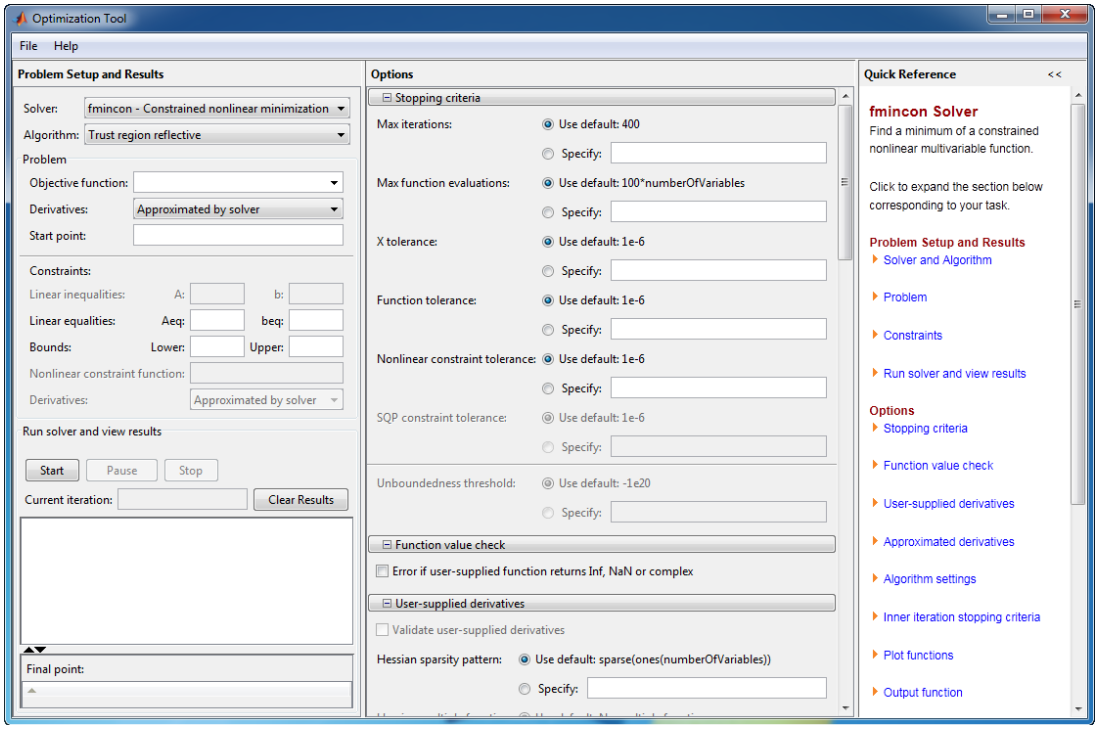

Bild 2-19: Graphische Benutzeroberfläche des Optimization Tools unter MATLAB [Mat14])

Für **gradientenbasierte Verfahren ohne Restringierung** stehen neben *fzero* noch *fsolve* und *fminunc* zur Verfügung. *fzero* und *fsolve* versuchen die Zielfunktion zu minimieren, indem sie die numerische Nullstelle dieser Funktion bestimmen. Dabei sind *fzero* der Befehl für skalare und *fsolve* der Befehl für vektorwertige nichtlineare Funktionen. *fminunc* hingegen bedient sich dabei der *TR*- oder der *Quasi-Newton*-Methode zur Berechnung des Gradienten [ABR+11].

Für **direktsuchende Verfahren ohne Restringierung** steht *fminsearch* zur Verfügung. Dieser basiert auf der *Nelder-Mead-Simplex*-Methode. Eine genauere Beschreibung ist unter anderem in [Mat14] zu finden.

Der *fminbnd* kombiniert das **Goldener-Schnitt-Verfahren** mit der **parabolischen Interpolation** [ABR+11]. Bei der Anwendung ist darauf zu achten, dass dieses Verfahren nur für die Optimierung einzelner Parameter gilt. Die Zielfunktion muss stetig und (mindestens) einmal differenzierbar sein, zudem darf sie nur reale Werte annehmen. Liegt das zu findende Optimum auf oder in der Nähe einer vorgegebenen Grenze, so ist dieses Verfahren zeitintensiv [Mat14].

Bei *lsqnonlin*, *fseminf* und *fmincon* handelt es sich um die **gradientenbasierten Verfahren mit Restringierung**. Für die Wahl der Solver kann bei *lsqnonlin*

zwischen *TR*- und *LM*-Methode entschieden werden. Des Weiteren verwendet *lsqnonlin* bei der Bestimmung der Zielfunktion immer die Methode der kleinsten Quadrate; somit ist die Art der Zielfunktion nicht frei wählbar. Anders ist dies bei der Verwendung von *fmincon*. Zudem besteht hierbei die Möglichkeit, zwischen vier Solvern (*TR*-, *AS*-, *IP*- und *SQP-Methode*) zu wählen, die alle bestimmte Vor- sowie Nachteile für das spezifische Optimierungsproblem besitzen. *fseminf* kann bei Problemen angewendet werden, bei denen die Lösung der zu optimierenden Zielfunktion außerhalb der gestellten Grenzen liegt [Mat14].

Für **direktsuchende Verfahren mit Restringierung** steht der *Pattern Search* zur Verfügung. Hierbei kann zwischen vier Solvern (*Generalized Pattern Search Positive Basis 2N (GPS2N)*, *Generalized Pattern Search Positive Basis Np1 (GPS-Np1)*, *Mesh Adaptive Pattern Search Positive Basis 2N (MADS2N)* und *Mesh Adaptive Pattern Search Positive Basis Np1 (MADSNp1)*) gewählt werden. Bei den Solvern *GPS2N* und *GPSNp1* handelt es sich um die zuvor beschriebenen Grundlagen, wobei $Np1 \mathrel{\widehat{=}} n+1$ bzw. $2N \mathrel{\widehat{=}} 2 \cdot n$ die Anzahl der Punkte für das Muster angibt; dabei stellt n die Dimension der zu optimierenden Parameter dar. Bei den Solvern *MADSNp1* und *MADS2N* handelt es sich um eine Erweiterung der zuvor beschriebenen Grundlagen. Im Gegensatz zu den *GPS*-Solvern, bei denen entlang eines Musters gesucht wird, kann bei den *MADS*-Solvern das Muster für die Suche verdreht werden [LD10]. Hierdurch steigen die Rechenzeit sowie die Genauigkeit der Ergebnisse.

Zur Verwendung der zuvor beschriebenen Optimierungsverfahren ist nachfolgend die Grundstruktur eines Optimierungsverfahrens unter MATLAB dargestellt.

Quellcode 2.1: Beispielhaftes MATLAB-Skript für ein Optimierungsverfahren

```
function(Ergebnis)Optimierung
% Identifikationsparameter inklusive Nebenbedingungen
P=1;
lb=0.1;
ub=5;

% Einstellungen für Optimierungsverfahren
TolX=1e-6;
TolFun=1e-6;
...

options=optimset('TolX',TolX,'TolFun',TolFun,...);

% Aufruf eines Optimierungsverfahrens
Ergebnis=ALGORITHMUS(@Zielfunktion,P,lb,ub,options);
```

Hierbei handelt es sich um eine MATLAB-Funktion mit dem Namen *Optimierung*. Als zu identifizierender Parameter wird P mit dem Startwert 1 sowie der unteren

Grenze $lb = 0{,}1$ und der oberen Grenze $ub = 5$ gesetzt. Der optimierte Parameter wird in der Variable *Ergebnis* gespeichert. Die Einstellungen für das jeweilige Optimierungsverfahren können mit dem Befehl *optimset* getroffen werden. Einige der wichtigsten sind hierbei *TolX*, *TolFun* und *MaxFunEvals*. Dabei beschreibt die Variable *TolX* die Abweichung der zu optimierenden Parameter P zwischen zwei Iterationen. Ist diese kleiner als die mit *TolX* eingestellte Abweichung, so endet das Verfahren. Bei *TolFun* endet das Verfahren, sobald die Abweichung der Zielfunktion $f(P)$ zwischen zwei Iterationen kleiner ist als der vorgegebene Wert. Bild 2-20 zeigt den Einfluss der beiden Abbruchkriterien, dargestellt an einem Identifikationsparameter P über einer Zielfunktion $f(P)$.

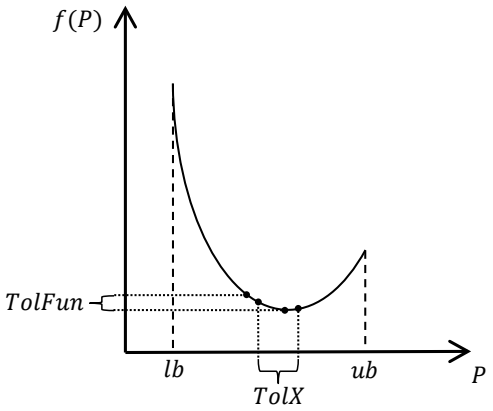

Bild 2-20: Darstellung von TolX und TolFun an einer Zielfunktion $f(P)$ (angelehnt an [Mat14])

Das Abbruchkriterium *MaxFunEvals* bietet schließlich noch die Möglichkeit, eine maximale Anzahl von Funktionsevaluierungen vorzugeben. Durch die Vorgabe der oberen und der unteren Grenze wird der Lösungsraum für den Optimierer, wie zuvor beschrieben, beschränkt. Sollte es sich um mehrere Identifikationsparameter $\vec{P}$ handeln, können neben den direkten Parametergrenzen auch Parameterabhängigkeiten untereinander vorgegeben werden. Diese Abhängigkeiten werden mit den konstanten Matrizen A und A_{eq} sowie den konstanten Vektoren $\vec{b}$ und $\vec{b}_{eq}$ festgelegt. Hierzu werden lineare Nebenbedingungen definiert:

$$A \cdot \vec{P} \leq \vec{b},$$
$$A_{eq} \cdot \vec{P} = \vec{b}_{eq},$$
$$\vec{lb} \leq \vec{P} \leq \vec{ub}.$$

Sollen keine Abhängigkeiten der Identifikationsparameter untereinander vorgegeben werden, so können die Einträge durch ein leeres Element [ ] ersetzt werden. Gleiches gilt für die anderen vorzugebenden Größen im Befehl *optimset*.

Tabelle 2-1 zeigt abschließend noch einmal die unter MATLAB zur Verfügung stehenden Optimierer für nichtlineare, deterministische Verfahren sowie deren Besonderheiten in Bezug auf Restringierungen und Verfügbarkeit.

Tabelle 2-1: Vorgestellte Optimierungsalgorithmen unter MATLAB für die nichtlineare, deterministische Optimierung einer Zielfunktion [Bru14] (∗ = OptimizationToolbox, ∗∗ = GlobalOptimizationToolbox)

Name	In MATLAB auszuwählende Methode	Restringierung	Toolbox
Gradientenbasierte Methode			
fzero	-	nein	∗
fsolve	TR, LM	nein	∗
fminunc	TR, QN	nein	∗
fseminf	-	ja	∗
fmincon	IP, AS, TR, SQP	ja	∗
lsqnonlin	TR, LM	ja(TR), nein(LM)	∗
Goldener Schnitt			
fminbnd	-	ja	-
Direktsuchmethode			
fminsearch	-	nein	-
Pattern Search (PS)	$GPS2N, GPSNp1$, $MADS2N, MADSNp1$	ja	∗∗

2.7 Handlungsbedarf

Unternehmen können bei der Entwicklung von neuen technischen Produkten etablierte Entwicklungsmethodiken verwenden. Zugleich kann auf bestehendes Wissen durch die im Internet zur Verfügung stehenden Online-Kataloge sowie durch die steigende Anzahl an zugänglichen Modellbibliotheken zurückgegriffen werden, um so Entwicklungszeiten zu verkürzen. Es hat sich jedoch gezeigt, dass viele Experten oftmals nur auf ihnen bekannte Lösungen zurückgreifen. Das Innovationspotenzial wird somit nicht vollkommen ausgeschöpft. Hier benötigt man eine Methodik, in der das zu untersuchende System ganzheitlich betrachtet wird

und die Entwickler durch eine systematische Vorgehensweise bei der Suche nach innovativen Lösungen unterstützt werden. Hinzu kommt, dass die modellbasierte Entwicklung beim Entwurf mechatronischer Systeme zunehmend an Bedeutung gewinnt und die Anzahl an Modellierungswerkzeugen in den einzelnen Domänen wächst. Für die gesamtheitliche Betrachtung eines mechatronischen Systems ist eine Zusammenführung aller erstellten Teilsysteme unumgänglich. Da es Modellungenauigkeiten im Gesamtmodell immer geben wird, ist des Weiteren eine Modellvalidierung mit einer dazugehörigen Parameteridentifikation des Gesamtmodells zwingend erforderlich. Die Zeiten, in denen es einen Experten für das Gesamtmodell gab, sind jedoch vorbei, und die steigende Komplexität verhindert eine ganzheitliche Parameteridentifikation für das Gesamtmodell durch *trial and error* in einer angemessenen Zeit. Der Einsatz von Optimierungsverfahren bietet hierzu gute Ansätze, dem Abhilfe zu schaffen. Die große Vielfalt an Modellierungswerkzeugen gestaltet die Einbindung der Modelle in die Optimierungsverfahren derzeit noch sehr aufwändig. Hier bedarf es einer Identifikationsumgebung, welche die Einbindung von Simulationsmodellen, unabhängig von dem zuvor gewählten Modellierungswerkzeug, unkompliziert ermöglicht. Zudem erfordert die große Vielzahl an Optimierungsverfahren, die jeweils für bestimmte Optimierungsprobleme Stärken und Schwächen besitzen, für die Auswahl des geeignetsten Verfahrens detailliertes Vorwissen. Das Vorwissen bezieht sich sowohl auf das Optimierungsverfahren als auch auf das zu lösende Optimierungsproblem. Das jeweilige Optimierungsproblem hängt in diesem Fall wiederum maßgeblich vom erstellten Modell und den zu identifizierenden Parametern ab. Mit der falschen Wahl des jeweiligen Optimierungsverfahrens und den dazugehörigen -eigenschaften ist das Auffinden eines Optimums unwahrscheinlich. An dieser Stelle benötigt man eine Identifikationsumgebung, welche die Wahl des jeweils zu verwendenden Optimierungsverfahrens sowie der hierzu benötigten Einstellungen unterstützt.

Die nachfolgenden Kapitel zeigen, wie dieser Handlungsbedarf gedeckt werden kann. Hierzu werden anhand eines durchgängigen Beispiels des Umflut-Waschverfahrens eine mitentwickelte Entwurfsmethodik aus dem ENTIME-Projekt angewendet und deren Effizienz nachgewiesen (siehe Kapitel 3). Eine besondere Stärke dieses Vorgehens liegt in der ganzheitlichen Betrachtung des zu entwickelnden Systems unter Verwendung modernster Entwicklungswerkzeuge, um innovative Lösungen zu liefern. Eine erarbeitete Parameteridentifikations- und Modellvalidierungsmethodik gliedert sich in die mitentwickelte Entwurfsmethodik ein und ergänzt diese entsprechend. Sie besteht aus einem entwickelten Interface und aus einer MATLAB-Identifikationsumgebung, mit deren Hilfe das aus der Entwurfsmethodik des ENTIME-Projektes erzeugte nichtlineare Multidomänen-Modell teilautomatisiert eingebunden und mittels etablierter Verfahren identifiziert werden kann. Da es sich um eine standardisierte Schnittstelle handelt, ist die Identifikationsumgebung unabhängig von der verwendeten Modellentwicklungs-

landschaft, was ein Höchstmaß an Flexibilität bietet. Weitere besondere Stärken der entwickelten Parameteridentifikations- und Modellvalidierungsmethodik werden in Kapitel 4 dargestellt.

3 Modellbasierter Entwurf mechatronischer Systeme am Beispiel des Umflut-Waschverfahrens

Dieser Abschnitt befasst sich mit dem modellbasierten Entwurf mechatronischer Systeme am Beispiel des Umflut-Waschverfahrens (vgl. Abschnitt 3.2). Dieses soll den zuvor angesprochenen Handlungsbedarf an einer neuen Methodik zur modellbasierten Entwicklung für aktuelle Herausforderungen für mechatronische Systeme erfüllen. Es wird aufgezeigt, wie mittels der neuen Methodik anhand von Anforderungen eine virtuelle Lösung für ein zu untersuchendes System erstellt wird, die hohes Innovationspotenzial beinhaltet und gezielt für die modellbasierte Entwicklung mechatronischer Systeme eingesetzt werden kann. Bevor das konkrete Vorgehen am modellbasierten Entwurf des Umflut-Waschverfahrens gezeigt wird, wird zunächst allgemein die angewendete Entwurfsmethodik vorgestellt.

3.1 Vorstellung und Einordnung der Entwurfsmethodik

Bei dem Vorgehensmodell handelt es sich um eine Erweiterung der im Rahmen des ENTIME-Projektes mitentwickelten Entwurfsmethodik [GTS+14]. Es ist ein ganzheitlicher und funktionsorientierter Entwurf, der in Anlehnung an die Konstruktionslehre nach Pahl/Beitz [PB97] sowie an das V-Modell nach der VDI-Richtlinie 2206 [VDI04] entstanden ist. Die Entwurfsmethodik besteht aus den Phasen Systemkonzipierung, Disziplinspezifischer Entwurf, Disziplinübergreifende Koordination und Modellgestützte Systemintegration, wobei in dieser Arbeit ein Schwerpunkt auf der Betrachtung der ersten beiden Phasen liegt (vgl. Bild 3-1). Die jeweils durchzuführenden Schritte einer jeden Phase sowie die dazugehörigen Tätigkeiten werden nachfolgend anhand der Einordnung in das V-Modell kurz beschrieben.

3.1.1 Einordnung in das V-Modell

Bild 3-1 zeigt das Vorgehen bei der angewendeten Entwurfsmethodik sowie die Einordnung in das V-Modell. Um u. a. Produkteigenschaften abzusichern, erfolgen bei dem Vorgehen grundsätzlich immer Iterationen. Besonders häufige Iterationen sind in Bild 3-1 mit zusätzlichen Pfeilen dargestellt. Die Entwurfsmethodik beginnt mit einer fachgebietsübergreifenden `Systemkonzipierung`. Diese lässt sich in die drei Schritte `Zielbeschreibung`, `Synthese`[1] und `Analyse` aufteilen.

[1] Als Synthese wird das Zusammenführen von Elementen zu einer neuen Einheit bezeichnet [Klu02].

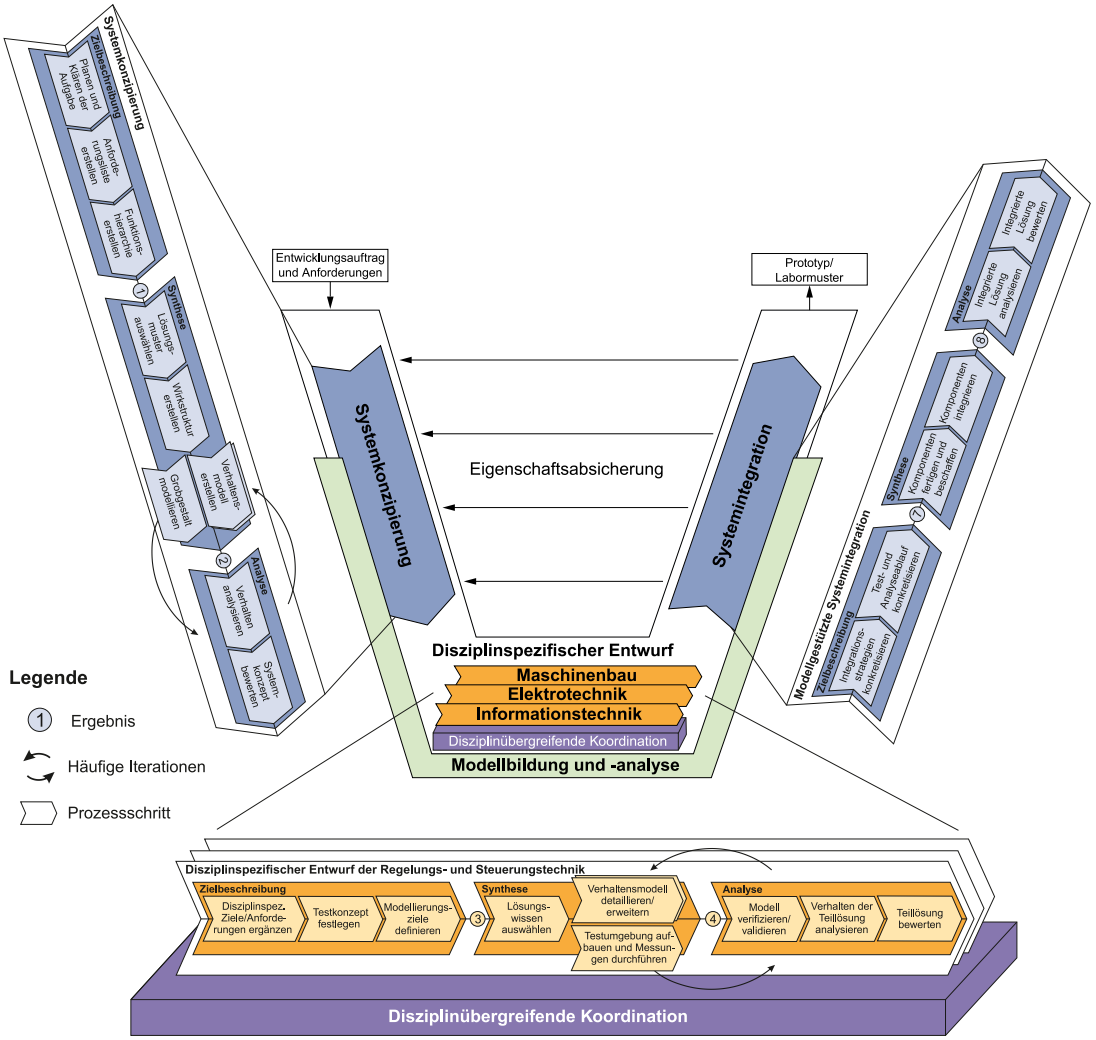

Bild 3-1: Einordnung der Entwurfsmethodik in das V-Modell (siehe auch [Loc, Oes18])

Um die grundlegende Entwicklungsaufgabe festzulegen, wird zunächst während der `Zielbeschreibung` mit dem Planen und Klären der Aufgabe begonnen. In dieser Tätigkeit werden, in Anlehnung an die Spezifikationstechnik CONSENS (vgl. Abschnitt 2.3), die Partialmodelle Umfeld und Anwendungsszenarien erarbeitet. Auf dieser Grundlage wird als nächste Tätigkeit eine Anforderungsliste erstellt, die es im Laufe der Entwicklung zu detaillieren gilt. Anschließend werden die Funktionen in Form einer Funktionshierarchie erarbeitet, welche die Grundlage für die darauffolgende `Synthese` bildet. Diese startet mit einer Auswahl von Lösungsmustern, welche die zuvor erarbeiteten Funktionen erfüllen. Als Lösungsmuster werden abstrakte Darstellungen einer Klasse von Lösungselementen

bezeichnet, deren Struktur und Verhalten dabei in allgemeiner Form beschrieben werden [GTS+14]. Die ausgewählten Lösungsmuster werden in Form einer Wirkstruktur kombiniert. Eine idealisierte Modellierung des Verhaltens der Steuerung sowie eine idealisierte physikalische Modellierung des Systems erfolgen in einem nächsten Schritt parallel. Zugleich kann an dieser Stelle ein erstes, grobes, rechnerinternes 3D-CAD-Gestaltmodell des zu entwickelnden Systems erstellt werden. An dieser Stelle endet die Synthese, und der Schritt der `Analyse` beginnt. Hierbei wird mithilfe der zuvor erstellten Modelle die Systemdynamik analysiert und bewertet. Zugleich können durch eine Bewertung der Systemkonzepte erste Kostenabschätzungen erarbeitet werden. Als Ergebnis der Systemkonzipierung entsteht die *Prinziplösung*. Diese beinhaltet eine abstrakte Sicht auf die Lösung in Form von idealisierten Verhaltensmodellen der Steuerung sowie der Dynamik. Zugleich bildet die *Prinziplösung* die Grundlage des nachfolgenden disziplinspezifischen Entwurfs. In diesem finden parallel die fachdisziplinspezifischen Ausarbeitungen der jeweils beteiligten Domänen statt. Jede Domäne kann an dieser Stelle ihr eigenes Vorgehensmodell verwenden, um den jeweils zu betrachtenden Bereich bestmöglich zu gestalten. Das Vorgehen für den `disziplinspezifischen Entwurf der Regelungs- und Steuerungstechnik` ist exemplarisch in Bild 3-1 dargestellt.

Anhand von der zuvor erarbeiteten *Prinziplösung* werden in der `Zielbeschreibung` der Disziplin Regelungs- und Steuerungstechnik die Anforderungen an die disziplinspezifischen Lösungen ergänzt bzw. definiert. Des Weiteren werden mögliche Testkonzepte und erforderliche Modellierungsziele festgelegt. Während der `Synthese` werden die zuvor definierten Modellierungsziele und Testkonzepte umgesetzt. Hierzu findet zunächst eine Auswahl der Lösungselemente bzw. disziplinspezifischer Lösungsmuster statt. Lösungselemente weisen im Vergleich zu den Modellen der Lösungsmuster während der Konzipierung eine deutlich höhere Modellierungstiefe auf und können somit das reale Verhalten genauer abbilden. Parallel zum Aufbau einer Testumgebung für die Durchführung von Messungen kann die Detaillierung der zuvor idealisiert modellierten Verhaltensmodelle erfolgen. Hierzu werden die Lösungsmuster durch Lösungselemente, die zuvor in der Zielbeschreibung ausgewählt wurden, ausgetauscht. Müssen für die in der anschließenden `Analyse` stattfindende Bewertung der jeweiligen domänenspezifischen Teillösung weitere physikalische Eigenschaften im Modell abgebildet sein, die zuvor bei der idealisierten Betrachtung nicht von Bedeutung waren, ist das Modell während der Synthese um die entsprechenden Komponenten zu erweitern. Für die Schritte Synthese und Analyse erfolgen somit ebenfalls häufige Iterationen. Einen wichtigen Teil der Analyse stellt die Modellvalidierung dar, auf die im weiteren Verlauf dieser Arbeit genauer eingegangen wird (siehe Kapitel 4). Erst nach der Validierung kann die Teillösung analysiert und bewertet werden, bevor

die benötigten disziplinspezifischen Modelle sowie entsprechend ausgearbeitete Regelungen und Steuerungen als *virtuelle Teillösungen* vorliegen.

Parallel zu den jeweiligen Vorgehensmodellen der einzelnen Domänen während des disziplinspezifischen Entwurfs findet eine `disziplinübergreifende Koordination statt`. In dieser Phase sammelt und überwacht ein Koordinator gesamtsystemrelevante Änderungen, analysiert sie und legt Strategien für die modellbasierte Integration sowie die anschließende Inbetriebnahme fest [Jus13]. Die disziplinübergreifende Koordination ist ebenfalls in die drei Schritte Zielbeschreibung, Synthese und Analyse aufgeteilt. Der disziplinübergreifende Koordinator erarbeitet während der `Zielbeschreibung` die Integrationsstrategien und legt das Test- und das Analysekonzept fest. In der anschließenden `Synthese` werden die jeweils relevanten Teilmodelle aus den unterschiedlichen Domänen ausgewählt und zu einem Gesamtsystem zur Eigenschaftsabsicherung integriert. Durch die ganzheitliche Modellvalidierung im Schritt der `Analyse` wird sichergestellt, dass die zuvor validierten Teillösungen auch im integrierten Gesamtmodell gültig sind. Durch die anschließende Systemanalyse und -bewertung des Gesamtsystems und die rückwirkende Absicherung aller zu erfüllender Anforderungen endet die disziplinübergreifende Koordination mit dem Ergebnis der *virtuellen Lösung*, und es geht zur `modellgestützten Systemintegration` über.

In der modellgestützten Systemintegration werden während der `Zielbeschreibung` die zuvor erarbeiteten Integrationsstrategien sowie die festgelegten Test- und Analysekonzepte konkretisiert. In der darauffolgenden `Synthese` werden die hierzu benötigten Komponenten gefertigt bzw. beschafft. Anschließend werden diese in das zuvor definierte Testkonzept integriert, so dass eine Testumgebung zur Eigenschaftsabsicherung für den Schritt der `Analyse` bereit steht. Mit der Analyse und der Bewertung der integrierten Lösung endet die modellgestützte Systemintegration mit dem Ergebnis der *konkreten Lösung*, die anschließend gefertigt und in Betrieb genommen werden kann.

Grundlage des beschriebenen Vorgehens, speziell für die beiden Phasen der Systemkonzipierung und des disziplinspezifischen Entwurfs, bildet die im Rahmen des ENTIME-Projektes mitentwickelte Entwurfsmethodik [GTS+14], die nachfolgend kurz vorgestellt wird.

3.1.2 Entwurfstechnik Intelligente Mechatronik

Die ENTIME-Entwurfsmethodik ermöglicht eine Beschreibung lösungsneutraler Anforderungen an das zu entwickelnde System für die Entwicklung innovativer technischer Produkte. Sie ist unterteilt in die beiden Phasen `fachgebietsübergreifende Konzipierung`, auch Systemkonzipierung genannt, und `fachspezifische Ausarbeitung`, auch disziplinspezifischer Entwurf genannt (vgl. Bild 3-2). Während der ersten Phase wird die Prinzipiösung des Produktes erarbeitet, die in der darauffolgenden zweiten Phase zur detaillierten Lösung modellbasiert weiterentwickelt wird. Ein wesentlicher Aspekt bei dieser Entwurfsmethodik liegt auf der ganzheitlichen Betrachtung eines zu untersuchenden Systems sowie der Wiederverwendung von bestehendem Lösungswissen [Oes18]. Hierzu kommen modernste Technologien, wie z. B. das Semantic Web, zum Einsatz, mit deren Hilfe Entwickler nach bestehenden Lösungselementen für die zuvor definierten Anforderungen suchen können. Beim Semantic Web handelt es sich um eine Erweiterung des World Wide Web (WWW) mit dem Ziel, inhaltliche Bedeutungen von

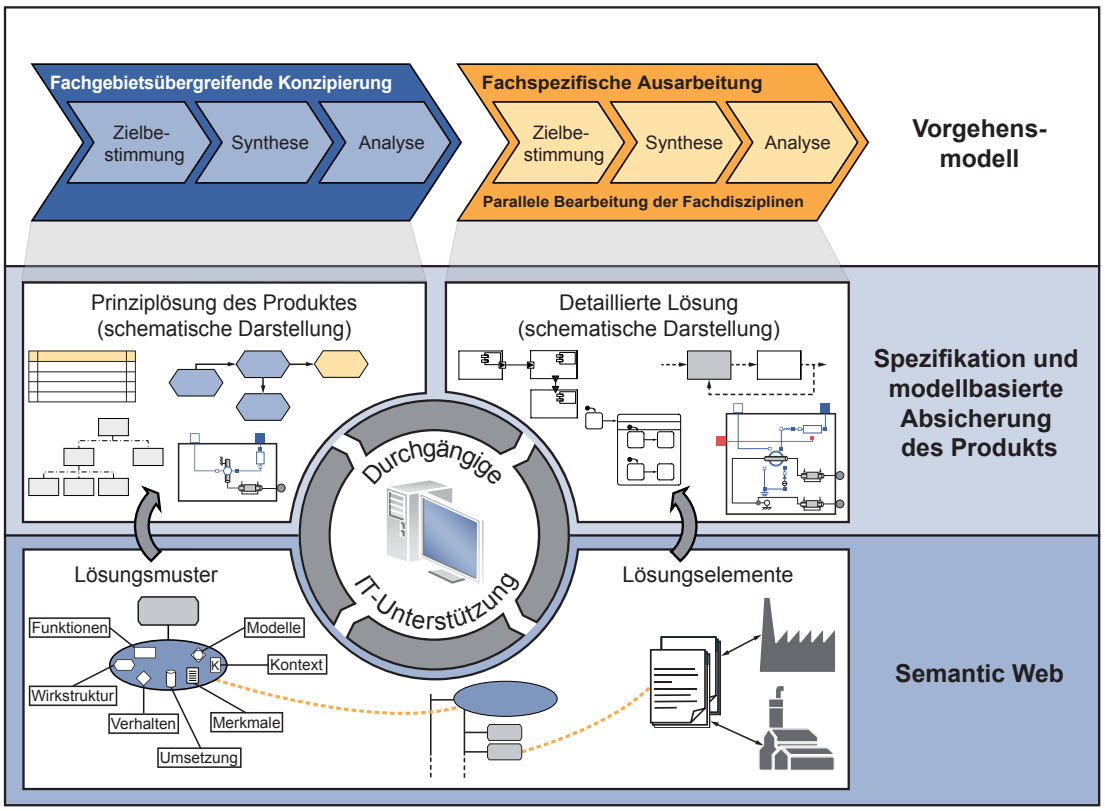

Bild 3-2: Struktur des im ENTIME-Projekt entwickelten Instrumentariums [GTS+14]

Informationen durch einen Computer interpretierbar zu gestalten, damit diese maschinell logisch verarbeitet und verknüpft werden können [BF99]. Einen guten und umfangreichen Überblick zu Semantic-Web-Werkzeugen bietet eine Übersicht des World Wide Web-Konsortiums [Wor13]. Die Lösungselemente werden von den entsprechenden Herstellern in Form von Datenbanken im Semantic Web hinterlegt. Hierdurch soll die Lücke zwischen den Entwicklern und den Lösungsherstellern geschlossen werden [GTS+14]. In dieser Arbeit liegt der Fokus auf dem Vorgehensmodell während der fachgebietsübergreifenden Konzipierung und der nachfolgenden fachdisziplinspezifischen Ausarbeitung sowie der dazugehörigen Spezifikation und modellbasierten Absicherung des zu entwickelnden technischen Produktes. Ergänzt wird dieses Vorgehen um die weiteren Schritte während der disziplinübergreifenden Koordination und der anschließenden modellgestützten Systemintegration. Für weitere Informationen bzgl. des Semantic Webs siehe [BF99, BHL01, GTS+14].

Das Vorgehen bei der angewendeten Entwurfsmethodik ist nicht grundlegend neu. Es kombiniert die Stärken etablierter Entwurfsmethodiken und ergänzt diese entsprechend. Die Einordnung des Vorgehens in das etablierte V-Modell wurde bereits gezeigt; nachfolgend wird der Zusammenhang mit der mechatronischen Komposition aufgezeigt.

3.1.3 Einordnung in die mechatronische Komposition

Wie schon in Abschnitt 2.2.1 gezeigt, stellt die mechatronische Komposition eine Erweiterung des konstruktiven Entwurfs auf mechatronische Systeme dar, wobei eine ganzheitliche Betrachtung des Systems im Vordergrund steht. Bild 3-3 zeigt die Einordnung der mechatronischen Komposition in die ENTIME-Entwurfsmethodik. Sie ist sowohl in der fachgebietsübergreifenden Konzipierung als auch in der fachdisziplinspezifischen Ausarbeitung zu finden und beschreibt jeweils die angesprochene Modellbildungsphase.

Auf Basis der Partialmodelle finden während der `fachgebietsübergreifenden Konzipierung` eine Komposition des Gesamtsystems sowie eine idealisierte Komposition statt, bei der jeweils die Schritte `Modellbildung`, `Analyse` und `Synthese` durchlaufen werden. In der hier verwendeten Entwurfsmethodik wird dies durch die Iterationen zwischen den Schritten `Synthese` und `Analyse` dargestellt. Hierdurch wird die prinzipielle Machbarkeit nachgewiesen (zu klären: Kann es überhaupt funktionieren?). In der `fachspezifischen Ausarbeitung` findet anschließend die ganzheitliche Komposition statt, indem detaillierte Modelle in Form von Lösungselementen eingebunden werden (zu klären: Wie kann es genau funktionieren?). Sowohl die Partialmodelle als auch die mechatronische Komposition bilden Methoden, die sich in die Entwurfsmethodik einordnen lassen und diese somit unterstützen.

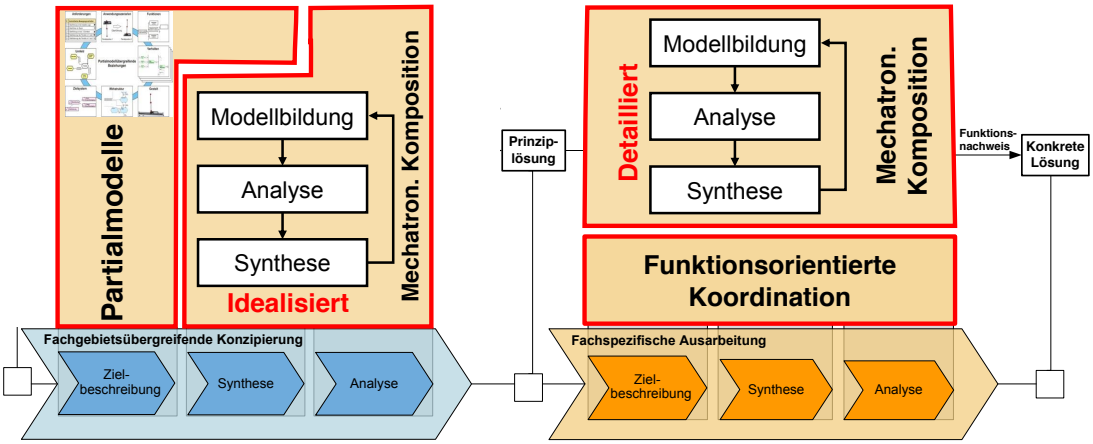

Bild 3-3: Einordnung der mechatronischen Komposition in die ENTIME-Entwurfsmethodik [Jus13]

Nachdem gezeigt wurde, wie die Entwurfsmethodik aufgebaut ist und wie sie bestehende etablierte Entwicklungsmethodiken beinhaltet, wird im Folgenden die konkrete Anwendung dieser Entwurfsmethodik an einem Anwendungsbeispiel gezeigt, das zunächst kurz vorgestellt wird.

3.2 Vorstellung des Umflut-Waschverfahrens

Das Anwendungsbeispiel befasst sich mit der modellbasierten Entwicklung des Umflut-Waschverfahrens [KTH13, GTS+14]. Motiviert ist dieses Anwendungsbeispiel durch die wachsende Bedeutung von energiesparenden Systemen. Zunehmendes Umweltbewusstsein der Bevölkerung und eine verstärkte Betonung von Energiepolitik zwingen Unternehmen zur Herstellung von energieeffizienteren Produkten. Einen signifikanten Anteil an der benötigten Energiemenge in Privathaushalten macht noch immer der Waschautomat aus. Ein moderner Waschautomat stellt ein komplexes mechatronisches System dar, das Aufgabenstellungen aus verschiedenen technischen Fachdisziplinen kombiniert. Diese enge Verknüpfung stellt besonders hohe Ansprüche an Planung und Vorgehen bei der Entwicklung. Um diesen Anforderungen gerecht zu werden, soll für die modellbasierte Entwicklung des energiesparenden Umflut-Waschverfahrens die vorgestellte Entwurfsmethodik angewendet werden. Bevor die einzelnen Phasen und Schritte der angewendeten Entwurfsmethodik beschrieben werden, werden zunächst der Waschprozess und speziell das Umflut-Waschverfahren vorgestellt.

Die Hauptaufgabe eines Waschautomaten besteht darin, Wäsche zu reinigen. Die wesentlichen Größen Mechanik, Temperatur, Zeit und Chemie und deren Abstimmung untereinander haben wesentlichen Einfluss auf die Waschwirkung, was

durch den Sinnerschen Kreis beschrieben werden kann [Geu98]. Bild 3-4 stellt jeweils einen Sinnerschen Kreise für zwei unterschiedliche Waschprogramme mit identischer Waschwirkung dar. Wird eine der Größen verringert, muss eine andere zwangsläufig vergrößert werden, um eine gleiche Waschwirkung zu liefern.

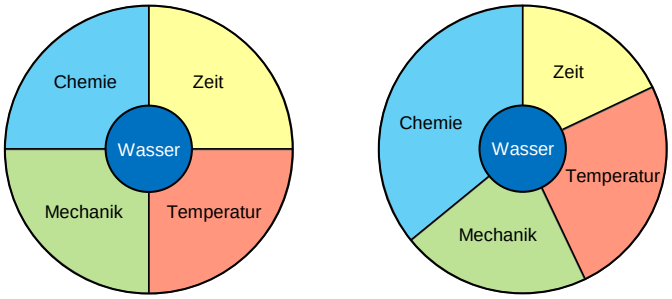

Bild 3-4: Sinnerscher Kreis (in Anlehnung an [Geu98])

Der Waschprozess selbst lässt sich vereinfacht wie folgt beschreiben: Verschmutzte Wäsche wird in eine rotationsfähige Trommel gegeben. Diese befindet sich innerhalb eines sogenannten Laugenbehälters, der wiederum mittels Feder- und Dämpfersystemen an einem Gehäuse befestigt ist. Über ein Wassereinspülsystem gelangt die Waschlauge (ein Gemisch aus Wasser und Chemie) in den Laugenbehälter. Diese Lauge wird im weiteren Verlauf als freie Flotte (FF) bezeichnet [KM02]. Nimmt die Wäsche einen Teil der FF auf, wird diese als gebundene Flotte (GF) bezeichnet. Mit Hilfe eines Heizelementes können sowohl die FF als auch die GF aufgeheizt werden. Unter Hinzunahme eines Antriebsmotors, der die Trommel in Bewegung versetzt, kann, vereinfacht ausgedrückt, die Wäsche gereinigt werden. Über eine Laugenpumpe wird am Ende die Lauge aus dem System geführt. Der wesentliche Unterschied zwischen Standard-Waschverfahren und Umflut-Waschverfahren ist der Weg, auf dem die Durchfeuchtung der Beladung erreicht wird. Das Prinzip des Umflut-Waschverfahrens im Vergleich zum Standard-Waschverfahren ist in Abbildung 3-5 dargestellt.

Beim Standard-Waschverfahren wird die Beladung durch einen hohen Stand der FF durchfeuchtet, was beim Umflut-Waschverfahren durch einen kontinuierlich umflutenden Massenstrom erzielt wird. Dazu wird ein zusätzliches Umflutsystem benötigt, das einen Teil der FF aus dem Laugenbehälter in die Trommel befördert und von dort aus die Wäsche befeuchtet. Auf diese Weise kann ein vergleichbarer Durchfeuchtungsgrad bei verringerter Menge an FF erreicht werden.

Mit der Entwicklung des Umflut-Waschverfahrens werden gleichzeitig neue Konzepte zur Erhitzung der Beladung benötigt, da der konventionell eingesetzte Heizstab nicht in dauerhaftem Kontakt zur FF steht. Für die Entwicklung eines neuen

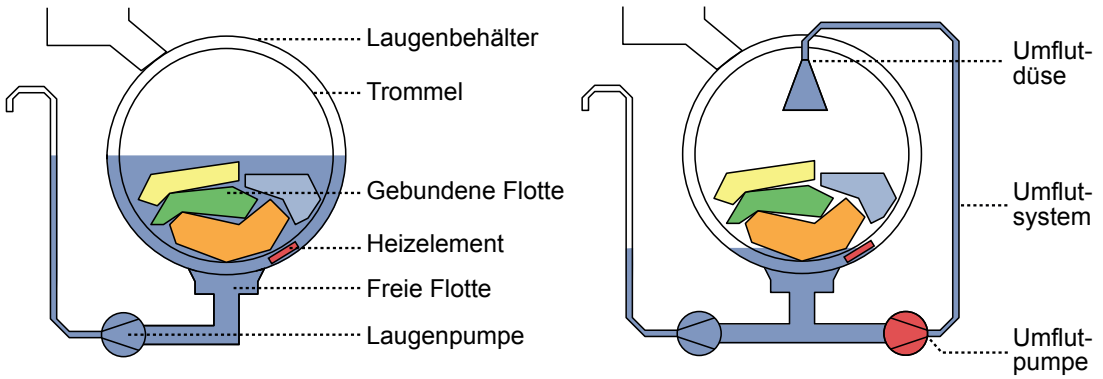

Bild 3-5: Gegenüberstellung des prinzipiellen Aufbaus des Standard-Waschverfahrens (links) und des Umflut-Waschverfahrens (rechts) [KTH13]

Heizkonzeptes für das Umflut-Waschverfahren wird die gezeigte Entwicklungsmethodik verwendet.

3.3 Systemkonzipierung

Dieser Abschnitt befasst sich mit der Phase der fachgebietsübergreifenden Systemkonzipierung für die Entwicklung des Umflut-Waschverfahrens. Diese lässt sich in die drei Schritte `Zielbeschreibung`, `Synthese` und `Analyse` aufteilen (vgl. Bild 3-6).

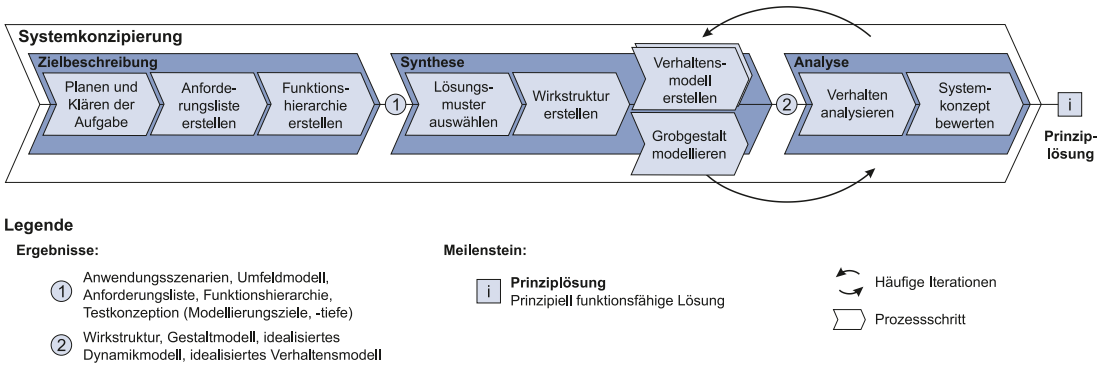

Bild 3-6: Vorgehensmodell für die Systemkonzipierung (vgl. [Loc, Oes18])

Einen wesentlichen Bestandteil der Systemkonzipierung bilden die mittels CONSENS zu erarbeitenden Partialmodelle. Bevor auf die Erarbeitung der jeweiligen Modelle genauer eingegangen wird, zeigt Bild 3-7 zunächst schematisch eine Übersicht der mittels der IT-Werkzeuge *Microsoft Visio* und *Mechatronic Modeller* erarbeiteten Partialmodelle. Da es sich bei den Ergebnissen, wie bereits

angesprochen, teilweise um sensible Daten aus der Vorentwicklung eines Industrieunternehmens handelt, werden nicht alle Schritte bzw. Ergebnisse im Detail vorgestellt. Es soll dennoch ein allgemeingültiger Eindruck vermittelt werden, wie die Entwicklungsmethodik konkret angewendet werden kann, um bestmögliche, innovative Ergebnisse mit Hilfe von modernsten Entwicklungstools zu liefern.

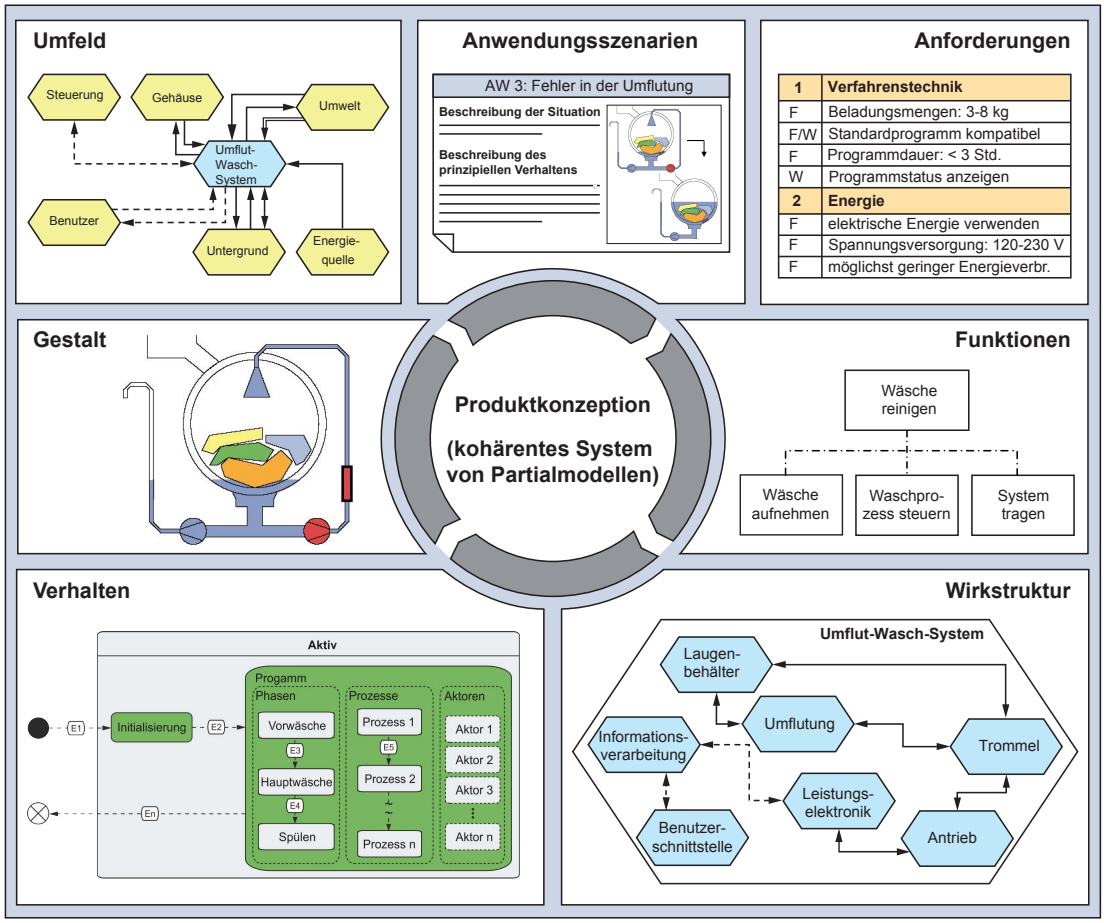

Bild 3-7: Schematische Darstellung der entstandenen Partialmodelle

3.3.1 Zielbeschreibung

Begonnen wird mit der `Zielbeschreibung` des zu entwickelnden Umflut-Waschverfahrens. Diese wird mithilfe der vier Partialmodelle *Umfeld*, *Anwendungsszenarien*, *Anforderungen* und *Funktionen* erarbeitet, die zugleich die Ergebnisse dieses Schrittes darstellen (vgl. Bild 3-6).

Für die Schritte `Planen und Klären der Aufgabe` wird das Partialmodell *Umfeld* erarbeitet (vgl. Bild 3-8). Hierzu wird das zu untersuchende System als Black

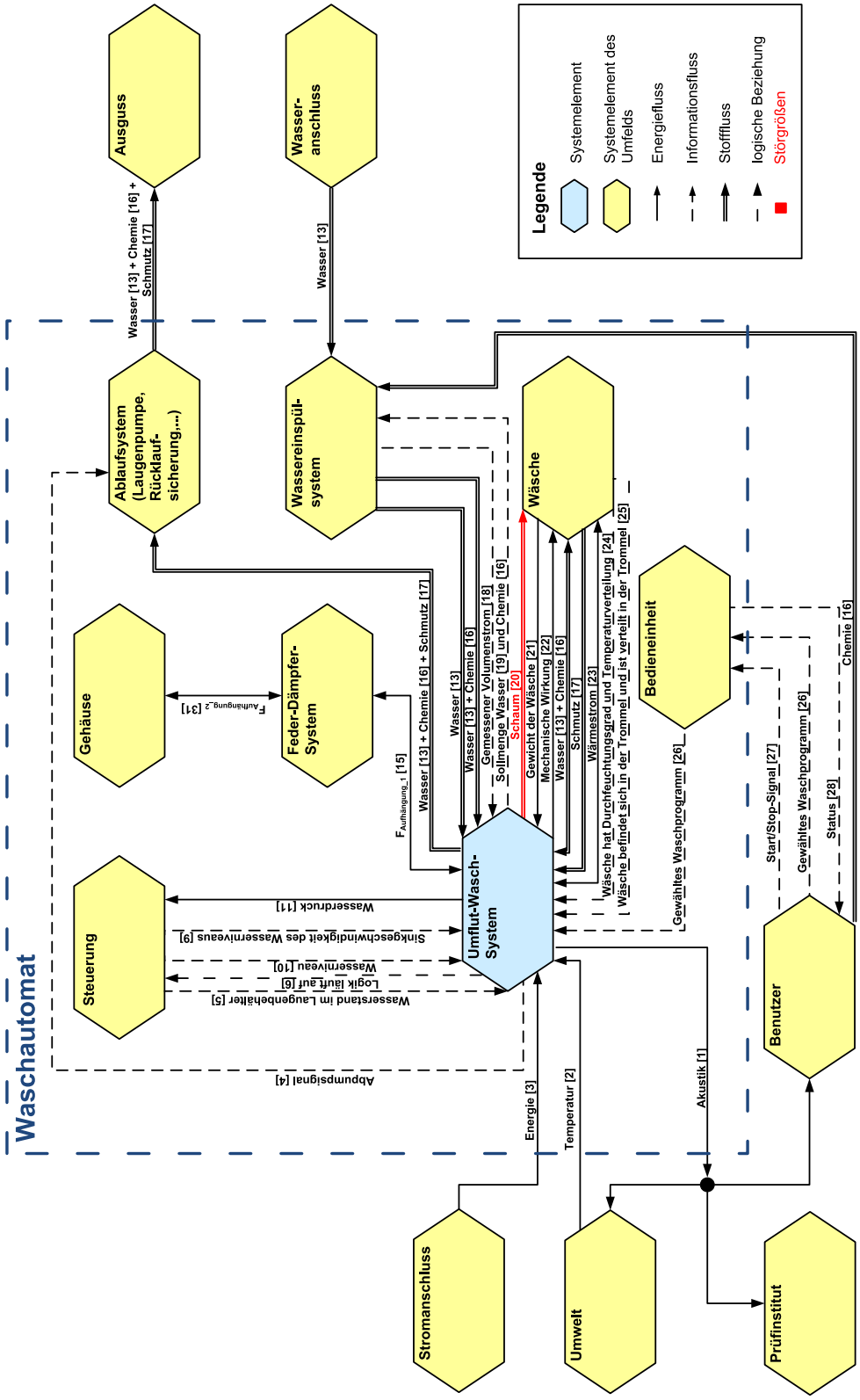

Bild 3-8: Umfeldmodell für das Umflut-Waschverfahren [KTH13]

Box modelliert. In einem Workshop, an dem alle beteiligten Experten teilnehmen, werden daraufhin die Grenzen des Systems zu seiner Umwelt abgeklärt und ebenfalls in dem Modell festgehalten. Neben den Systemgrenzen werden relevante Einflüsse, die auf das zu untersuchende System wirken, identifiziert und eingetragen. Dieses Vorgehen führt zu einer ganzheitlichen Betrachtung des Systems in einem frühen Entwicklungsstadium. Des Weiteren dient das Umfeldmodell als Unterstützung bei der Definition von Anforderungen.

Durch die Erarbeitung des Umfeldmodells soll allen beteiligten Personen zu einem möglichst frühen Zeitpunkt die Aufgabengebiete für das zu entwickelnde System verdeutlicht werden. Es soll zeigen, was genau am Gesamtsystem verändert werden darf bzw. muss und mit welchen Umfeldeinflüssen das System umgehen muss. Die jeweiligen Einflüsse werden in Informationsfluss, Energiefluss und Stofffluss kategorisiert und beschriftet. Zudem bekommt jede Beschriftung eine Nummer. Die Nummern werden in einem separaten Dokument aufgelistet, das auszugsweise in Bild 3-9 dargestellt ist. Dies dient zur genaueren Beschreibung der einzelnen Einflüsse. Sollten zu Beginn nicht alle Informationen zu den jeweiligen Feldern bekannt sein, können diese im Laufe der Entwicklung zu einem späteren Zeitpunkt eingegeben werden.

Nr.	Enflussfaktor	Wertebereich	Einheit	störend	neutral	unterstützend	Anmerkung
1	Temperatur	10 -30 °C	Grad Celsius [°C]				Temperatur, die von außen in den Waschautomat und auf das Umflut-Wasch-System einwirkt
2	Akustik		Dezibel [db]				Beitrag des Umflut-Wasch-Systems zur Geräuschentwicklung des Waschautomaten
3	Energie	120 - 230 V	Volt [V]				
		50 - 60 Hz	Herz [Hz]				
		16 A	Ampere [A]				
4	Abpumpsignal	digitales Signal					Am Ende des Umflut-Wasch-Verfahrens soll abgepumpt werden
5	Logik läuft auf	logische Zuordnung					Die Logik des Systemelements läuft auf der genannten Zielhardware
6	Schaumerkennungs-signal	digitales Signal					Hier ist noch unklar, wie Schaumerkennung und Umflut-Wasch-Verfahren interagieren

Bild 3-9: Ergänzende Angaben zum Umfeldmodell

Als nächstes werden mögliche Anwendungsszenarien durchdacht und dokumentiert. Dies erfolgt ebenfalls in dem zuvor genannten Workshop, an dem alle beteiligten Experten teilnehmen. Somit kann frühzeitig auf weitere Anforderungen, die das zu untersuchende System erfüllen soll, geschlossen werden. Zudem werden schon an dieser Stelle mögliche Systemfunktionen in unterschiedlichen Betriebsfällen erfasst. Sollten zu einem späteren Zeitpunkt weitere Anwendungsszenarien in Betracht gezogen werden, so ergänzt man diese nachträglich. Anhand des Umfeldmodells werden 11 Anwendungsszenarien erfasst, in die das Gesamtsystem gelangen kann:

1) Standardwaschvorgang,
2) Überbeladung,
3) Ungeeignetes Programm durch Benutzer gewählt,
4) Unterbeladung,
5) Nicht ausreichend Waschmittel,
6) Zu viel Waschmittel,
7) Temperatursensor defekt,
8) Heizelement defekt,
9) Drucksensor defekt,
10) Abpumpsystem defekt,
11) Fehler in der Umflutung (defekte Pumpe, verstopfte Leitungen etc.).

Anschließend wird überlegt, auf welche Weise das System auf die gegebene Situation reagieren soll. In Bild 3-10 ist exemplarisch ein Anwendungsszenario aufgelistet. Da das Systemverhalten nicht in allen Situationen genau definiert ist, können zunächst Vermutungen aufgestellt werden, die im weiteren Verlauf zu konkretisieren sind.

Nach der Abgrenzung des zu betrachtenden Systems von seiner Umwelt und der Erfassung der wirkenden Einflüsse bei unterschiedlichen Anwendungsszenarien wird als nächste Tätigkeit eine `Anforderungsliste` erstellt (vgl. Bild 3-6). Sie stellt ein wesentliches Ergebnis der Partialmodelle dar und bildet die Grundlage für die gesamte Entwicklung. In der Anforderungsliste werden lösungsneutrale Anforderungen (Lasten) an das zu entwickelnde System definiert. Im Laufe der Entwicklung werden nach und nach konkrete Lösungen für die Anforderungen in Form von Pflichten dokumentiert. Ein exemplarischer Ausschnitt der Anforderungsliste ist in Bild 3-7 dargestellt.

Anwendungsszenario	Nr. 11
Titel: Fehler in der Umflutung (defekte Pumpe, verstopfte Leitungen,…)	
Situationsbeschreibung: Der Waschautomat wurde vom Benutzer mit einer korrekten Menge an Wäsche gefüllt. Die Wäsche ist Standardwäsche mit Standardverschmutzung. Der Benutzer hat eine angemessene Menge an Chemie in das Wassereinspülsystem des Waschautomaten gegeben. Ferner hat er ein Waschprogramm gewählt, dass durch ein Umflut-Waschverfahren realisiert werden kann. Der Wasseranschluss des Waschautomaten ist in Ordnung. Das Umflutsystem ist defekt. Der Benutzer hat den Startknopf gedrückt.	
Beschreibung des Systemverhaltens: Der Waschautomat startet und Wasser und Chemie werden in den Laugenbehälter eingespült. Die Maschine durchläuft das gewählte Waschprogramm (i.d.R. Vorwäsche, Hauptwäsche, Spülen und Schleudern). Während der Umflutungsphase wird der Fehler detektiert. Wenn möglich wird auf ein Standard-Waschprogramm gewechselt. Am Ende erscheint eine Fehlermeldung. **Anmerkung:** Die Möglichkeit des Wechselns zwischen einem Umflutungsprogramm und einem Standard-waschprogramm ist stark von dem eingesetzten Heizverfahren abhängig. Befindet sich das Heizelement in dem Umflutungssystem, so kann nicht gewechselt werden. ➔ Es ist wichtig wechseln zu können (Vorschlag: Aufnahme in die Anforderungsliste als F (mind. W))	

Bild 3-10: Exemplarisches Anwendungsszenario

Im nächsten Schritt werden lösungsneutrale Funktionen erstellt, die zum Ziel haben, die erfassten Anforderungen zu erfüllen. Diese werden in Form einer `Funktionshierarchie` aufgelistet. Eine Funktion wird hierbei so lange durch Unterfunktionen beschrieben, bis ein sinnvolles Lösungsmuster gefunden werden kann. Bild 3-11 zeigt einen Ausschnitt der erstellten Funktionshierarchie.

Im Gegensatz zu einer Funktionsstruktur werden bei einer Funktionshierarchie keine Querverbindungen zwischen den aufgestellten Funktionen gezogen. Das bedeutet, dass theoretisch gleiche Funktionen mehrmals in einer Hierarchie auftreten können. Durch die fehlenden Querverbindungen wird das Dokument jedoch deutlich übersichtlicher.

3.3.2 Synthese

Nachdem die Funktionshierarchie erstellt ist, endet der Schritt der Zielbeschreibung, und es beginnt die Synthese der fachgebietsübergreifenden Systemkonzipierung. Diese startet mit der `Auswahl von Lösungsmustern`, welche die zuvor erfassten Funktionen erfüllen (vgl. Bild 3-6). An dieser Stelle kann mithilfe von intelligenten Suchmaschinen, wie das zuvor angesprochene Semantic Web, nach neuen innovativen Lösungsmustern gesucht werden. Um dabei einen möglichst

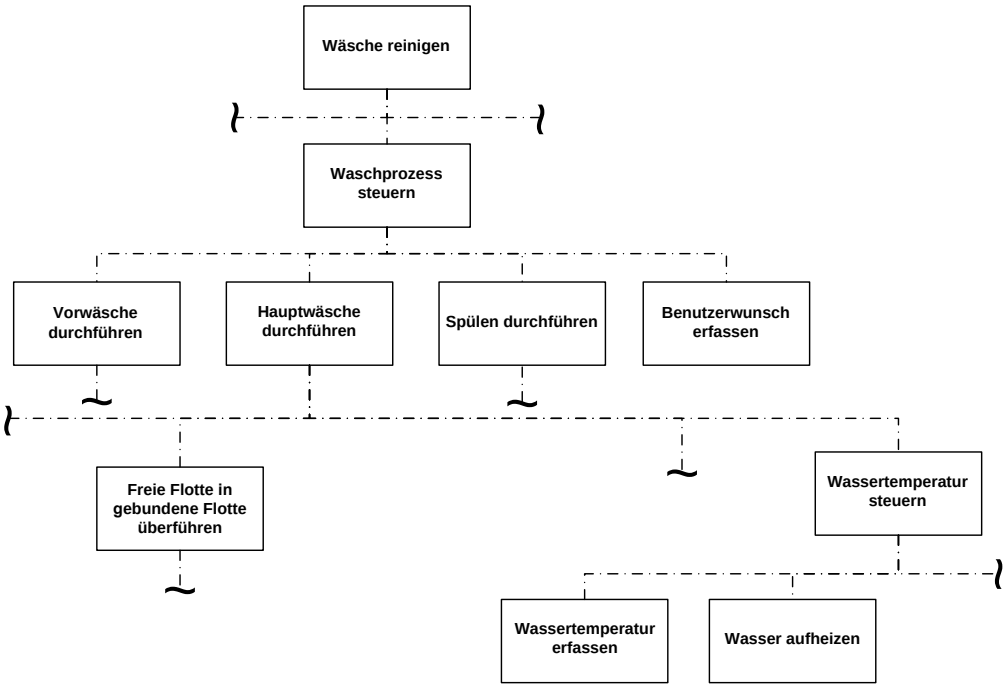

Bild 3-11: Exemplarischer Ausschnitt der erstellten Funktionshierarchie

großen Lösungsraum aufspannen zu können, ist es wichtig, dass die zuvor erstellte Funktionshierarchie so lösungsneutral wie möglich aufgebaut ist. Eine genauere Beschreibung, wie eine Suche im Semantic Web funktioniert, ist in [GTS+14] zu finden. Sollten nicht für alle Anforderungen und Funktionen Lösungsmuster im Semantic Web gefunden werden, müssen an dieser Stelle eigene Lösungsideen gefunden werden. Für die Dokumentation der erfassten Lösungsmuster zu den jeweiligen Funktionen kann ein morphologischer Kasten verwendet werden. Durch Auswahl und Kombination verschiedener Lösungsmuster können unterschiedliche potentielle Prinziplösungen erarbeitet werden. Um die Auswahl möglichst objektiv zu treffen, kann sowohl für die einzelnen Lösungsmuster als auch für die Kombination von Lösungsmustern eine Nutzwertanalyse angewendet werden.

Analog zu dem Umfeldmodell wird anschließend für jede potentielle Prinziplösung eine `Wirkstruktur` erstellt (vgl. Bild 3-6). In diesem Partialmodell werden wiederum die relevanten Einflüsse, die auf die einzelnen Systemkomponenten wirken, erfasst und eingetragen. Durch die detaillierte Erarbeitung der Wirkstruktur können schon frühzeitig die jeweils benötigten Komponenten, sowie deren Zusammenspiel untereinander für verschiedene potentielle Prinziplösungen erfasst werden.

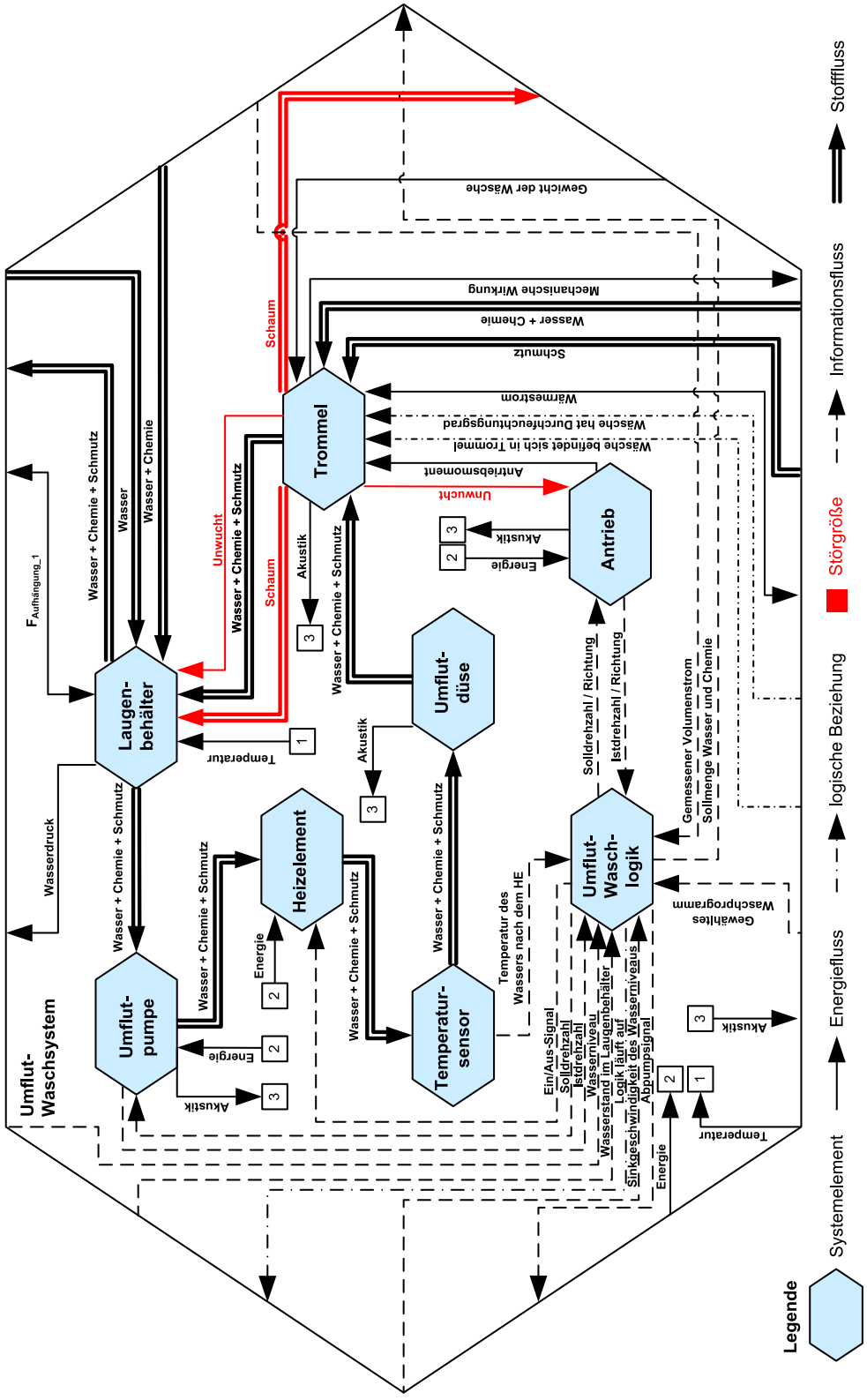

Bild 3-12: Darstellung einer Wirkstruktur-Variante

Als eine von mehreren potentiellen Prinziplösungen für ein neues Heizkonzept ist das Heizen mittels Durchlauferhitzer erarbeitet worden. In Bild 3-12 ist die entsprechende Wirkstruktur abgebildet. Der Durchlauferhitzer befindet sich im Umflutsystem zwischen der Umflutpumpe und der Umflutdüse. Durch Aktivierung der Umflutpumpe gelangt die Lauge zu dem Heizelement, wird erwärmt und über die Umflutdüse in die Trommel und der darin befindlichen Wäsche geleitet.

Nachdem die Wirkstrukturen für potentielle Prinziplösungen erarbeitet wurden, kann in einem nächsten Schritt die jeweilige `Grobgestalt modelliert` werden. Da es sich bei dem zu entwickelnden System um ein Heizkonzept für ein Waschverfahren handelt, bei dem nicht die Veränderung der Mechanik im Vordergrund steht, wird an dieser Stelle kein 3D-CAD-Modell des Systems erstellt, sondern eine Prinzipskizze, welche das Verfahren inklusive der gewählten potentiellen Prinziplösung darstellt. Die Prinzipskizze ist in Bild 3-7 unter dem Bereich Gestalt abgebildet.

Parallel zu dem Partialmodell der Gestalt kann das `Verhalten der Steuerung` modelliert werden. Bei diesem Partialmodell wird das Verhalten des zu entwickelnden Systems durch diskrete Zustände sowie Zustandsübergänge beschrieben. Da es sich hier um die Entwicklung eines Verfahrens (Umflut-Waschverfahren) handelt, stellt die Modellierung des Steuerungsverhaltens einen besonders wichtigen Aspekt dar. Es soll eine Logik abgebildet werden, mit dessen Hilfe das gesamte Umflut-Waschverfahren inklusive des Heizens modelliert werden kann.

Wesentliche Kriterien bei der Ausarbeitung dieses Verhaltensmodells der Steuerung sind:

1) Anschaulichkeit (benutzerfreundlicher, hierarchisierter Aufbau),
2) Erweiterbarkeit (veränderte Verfahrensabläufe),
3) Übertragbarkeit (andere Lösungselemente).

Die Hauptfunktion des zu betrachtenden Systems *Wäsche reinigen* (vgl. Bild 3-11) lässt sich unter anderem mit Hilfe der drei Verfahrensabschnitte *Vorwäsche*, *Hauptwäsche* und *Spülen* realisieren. Diese Abschnitte werden im Verlauf als Phasen bezeichnet. Jede Phase enthält, je nach Benutzerwunsch (gewähltes Waschprogramm), unterschiedliche Prozesse. Jeder Prozess kann wiederum einen oder mehrere Aktoren ansteuern. Bild 3-13 zeigt eine Abbildung des ereignisdiskreten Verhaltens für das Umflut-Waschverfahren. Es sei an dieser Stelle erwähnt, dass parallele Zustände durch gestrichelte Linien kenntlich gemacht werden. Jeder Zustand wird verlassen, wenn ein bestimmtes Ereignis eintritt.

Bei der Abbildung des Verhaltens der Steuerung kann neben der Darstellung der Zustände eine Darstellung der jeweiligen Aktivitäten erfolgen. Bild 3-14 zeigt exemplarisch die Aktivitäten für die Prozesse während des Zustandes Hauptwäsche.

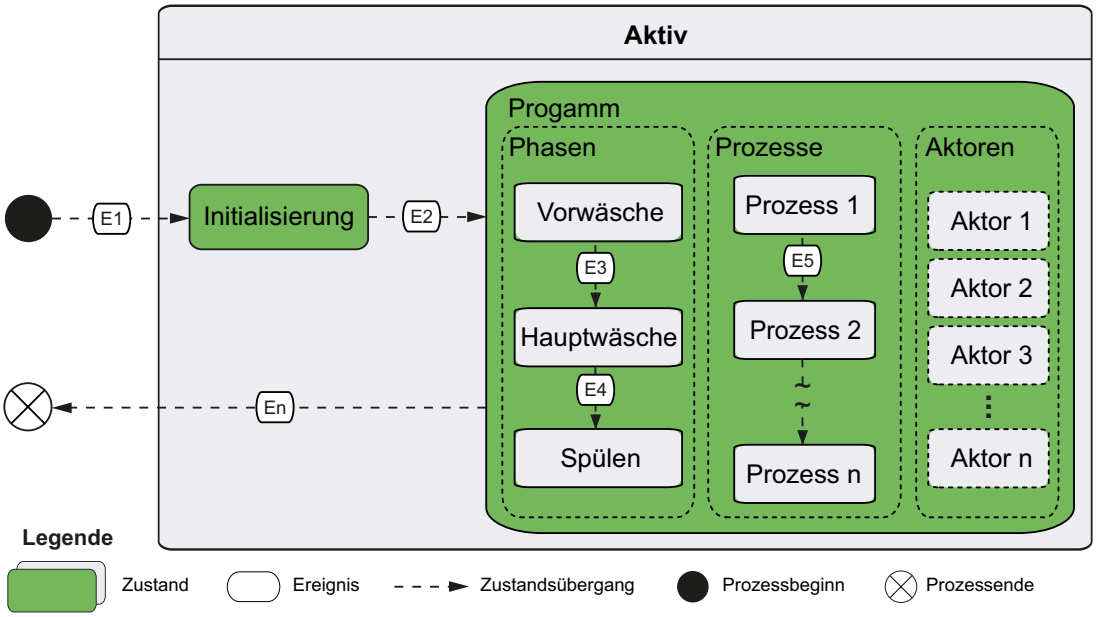

Bild 3-13: Abbildung des ereignisdiskreten Verhaltens der Steuerung für das Umflut-Waschverfahren

Je nach Benutzerwunsch werden unterschiedliche Aktivitäten für ein Waschverfahren verwendet. Der Benutzerwunsch wird durch die Programmwahl zu Beginn festgelegt.

Durch die Verwendung von Entscheidungspunkten kann das Waschprogramm flexibel gestaltet werden. Eine solche Darstellung ermöglicht durch eine passende Parametrierung sowohl ein klassisches (Standard-) Waschverfahren als auch ein Umflut-Waschverfahren. Die Aktivitäten-Struktur muss hierzu nicht verän-

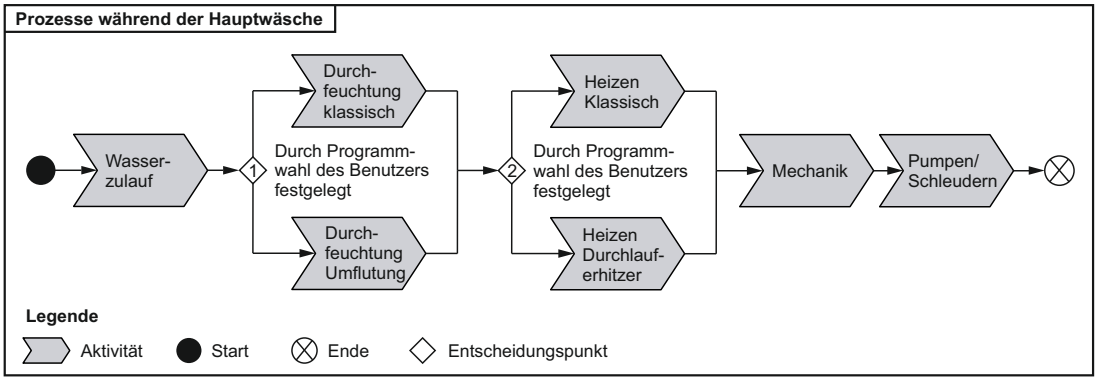

Bild 3-14: Exemplarisches Aktivitätendiagramm für Prozesse während der Hauptwäsche

dert werden. Um das ereignisdiskrete Verhalten der Steuerung zustandsbasiert zu modellieren, wird MATLAB/Simulink-Stateflow verwendet [KFS+12] (siehe Abschnitt 2.4.2). Die zuvor ausgearbeiteten Ideen bezüglich des Verhaltens der Steuerung können nahezu eins zu eins übernommen werden. Die Parametrierung erfolgt durch den MATLAB-Workspace, der durch ein entsprechendes *m-File* seine Daten erhält. Das *m-File* legt, in Anlehnung an das zuvor erstellte Aktivitätendiagramm, den gewünschten Verfahrensablauf fest. Definiert wird der Verfahrensablauf durch die Programmablaufmatrix (vgl. Gleichung 3-1). Jedem unter Stateflow modellierten Prozess ist eine eindeutige Nummer zugewiesen. Wird diese Nummer in der Programmablaufmatrix z. B. mithilfe der Prozessvariablen $P_{HW,1}$ gesetzt, so wird der entsprechende Prozess an erster Stelle in der Hauptwäsche ausgeführt. Die Programmablaufmatrix beinhaltet somit Nummern, welche die Abfolge der Prozesse für die jeweilige Phase der *Vorwäsche*, der *Hauptwäsche* und des *Spülens* festlegen.

$$M_{progAblauf} = \begin{bmatrix} P_{VW,1} & P_{VW,2} & \dots & P_{VW,n} \\ P_{HW,1} & P_{HW,2} & \dots & P_{HW,n} \\ P_{SP,1} & P_{SP,2} & \dots & P_{Sp,n} \end{bmatrix} \tag{3-1}$$

Die jeweils zu verwendenden Aktoren sind fest den Prozessen zugeordnet und werden daher nicht explizit durch ein Skript vorgegeben. Die Umsetzung des durch die Programmablaufmatrix definierten Verfahrens erfolgt im Stateflow-Modell durch eine Abfrage des entsprechenden Matrixelements vor dem anstehenden Prozess. Bild 3-15 zeigt schematisch einen Ausschnitt des erstellten Verhaltensmodells der Steuerung. Der parallele Unterzustand *Phasen* gibt den Wert der Zählervariablen *z_phase* vor. Im Unterzustand *Prozesse* wird mittels dieser und der aktuellen Prozessvariablen *z_prozess* das entsprechende Matrixelement aus der Programmablaufmatrix ausgelesen. Handelt es sich dabei etwa um die Nummer des Prozesses Wasserzulauf, so wird dieser in einem nächsten Schritt aktiviert. Der Unterunterzustand Wasserzulauf gibt dem Unterunterzustand *Zulaufventil* das Signal, das entsprechende Ventil solange zu öffnen, bis eine bestimmte Bedingung erfüllt ist. Bei dieser Bedingung kann es sich zum Beispiel um eine zeitliche Abfrage handeln (vgl. Bild 3-15 Bedingung *timeout*). Ist diese Bedingung erfüllt, schließt sich das Ventil, und der Prozessschritt endet. Die Prozesszählervariable *z_prozess* erhöht sich, und das nächste Matrixelement wird, falls vorhanden, ausgelesen; anderenfalls endet die Phase.

Das erstellte ereignisdiskrete Verfahrensmodell weist bereits während der Konzipierungsphase eine komplexe Struktur auf. Diese ist notwendig, um den zuvor gestellten Anforderungen an das zu untersuchende System gerecht zu werden. Die

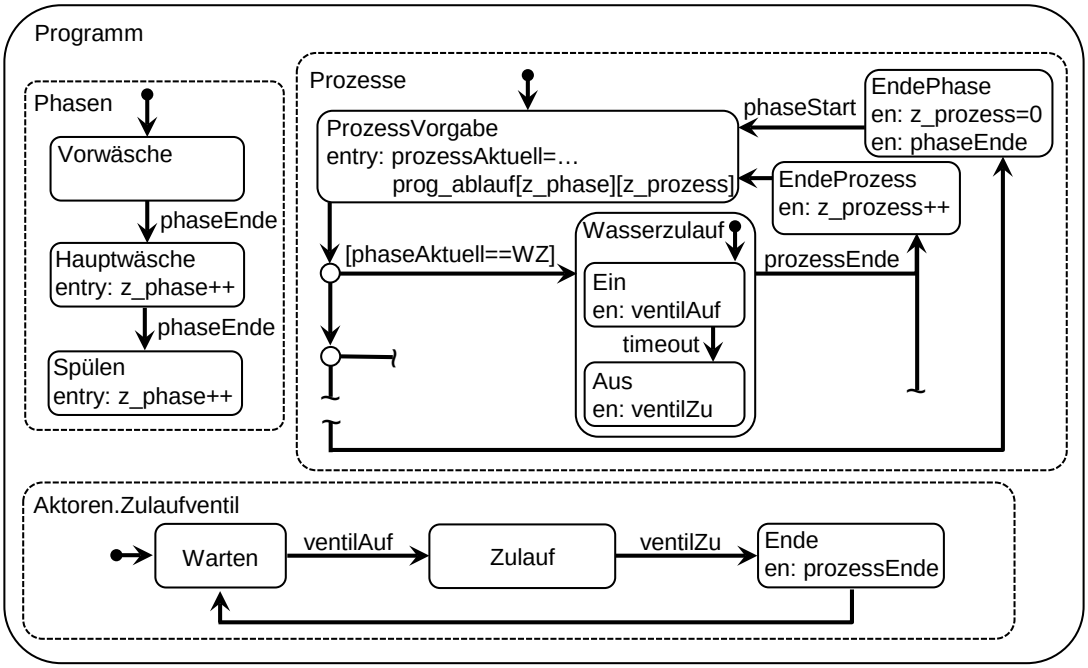

Bild 3-15: Schematische Darstellung des ereignisdiskreten Verhaltensmodells der Steuerung

im Stateflow-Modell hinterlegten Unterzustände sind zunächst noch sehr idealisiert abgebildet. Diese werden erst im disziplinspezifischen Entwurf detailliert und ggf. erweitert.

Für einen prinzipiellen Funktionsnachweis des zu betrachtenden Systems und für erste Analysen des Gesamtsystemverhaltens wird parallel (vgl. Bild 3-6, S. 47) ein `Verhaltensmodell der Dynamik` entwickelt, das anschließend mit dem Verhaltensmodell der Steuerung verkoppelt werden kann. Für die Modellierung des dynamischen Verhaltens wird die Entwicklungsumgebung Dymola [Dym13], basierend auf der objektorientierten Modellbeschreibungssprache Modelica [Mod09], verwendet. Die für die Modellierung benötigten Modellkomponenten können z. T. aus verfügbaren Modellbibliotheken entnommen werden. Zudem bietet Modelica die Möglichkeit, eigene Modellbibliotheken zu erstellen, falls ein benötigtes Element nicht vorhanden sein sollte, wodurch ein Höchstmaß an Flexibilität gegeben ist.

Die Basis für die Modellierung des dynamischen Modells bilden die Partialmodelle. Speziell die zuvor durch das Aufstellen der Wirkstrukturen erarbeiteten Einflüsse der Systemkomponenten aufeinander werden im Modell berücksichtigt. Ein Ansatz, diesen Schritt automatisiert durchzuführen, ist in [BGK+12] und [BGK+13] zu finden.

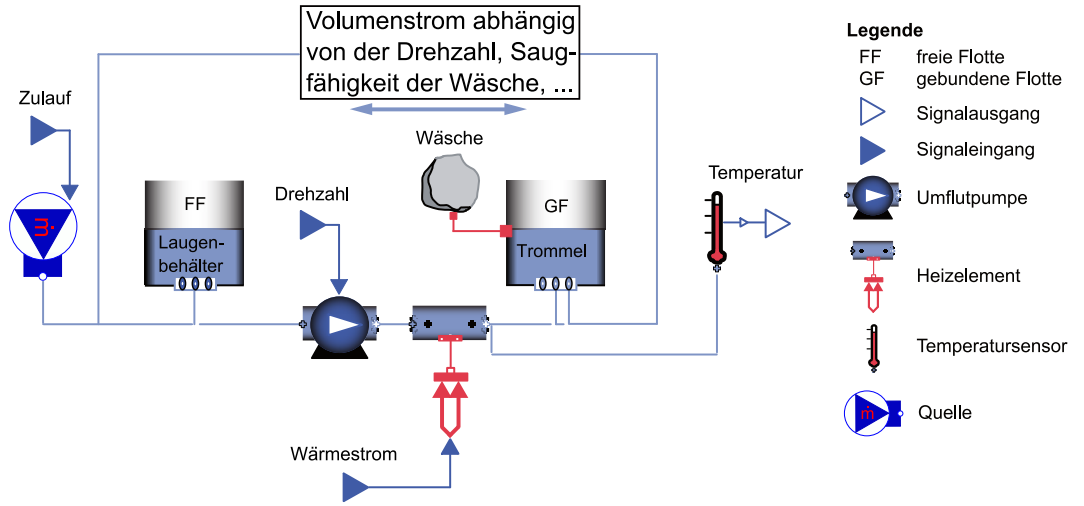

Bild 3-16: Schematisches, idealisiertes Verhaltensmodell der Dynamik in Dymola [KTH13]

Ein stark vereinfachter Aufbau eines idealisierten Modells ist in Bild 3-16 dargestellt. Die verwendeten Komponenten sind analog zu denen der Wirkstruktur (vgl. Bild 3-12). Die FF befindet sich in einem Tank, der den Laugenbehälter darstellt, und kann mittels einer Umflutpumpe über eine beheizbare Leitung (Durchlauferhitzer) in einen zweiten Tank befördert werden. Der Inhalt dieses Tanks stellt die in der Wäsche gebundenen Flotte (GF) dar. Um einen kontinuierlichen Austausch zwischen FF und GF zu realisieren, wird eine weitere Modellkomponente modelliert. Diese beschreibt sowohl das Absorptions- als auch das Desorptionsverhalten der Wäsche. Für einen ersten idealisierten Ansatz werden diese komplexen Zusammenhänge mittels vereinfachter Kennfelder mit einer geringen Anzahl an

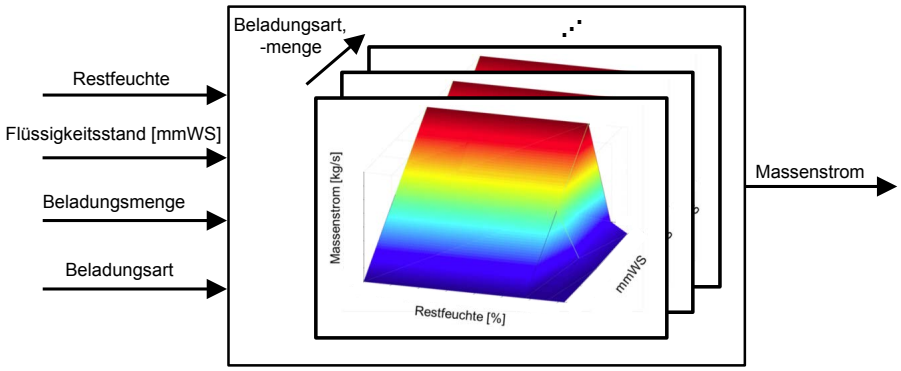

Bild 3-17: Schematische Darstellung des idealisierten Absorptionsmodells

Stützstellen modelliert. Diese bilden, in Abhängigkeit verschiedener Parameter, wie zum Beispiel der Trommeldrehzahl, des Flüssigkeitsstands, der Menge der Wäsche etc., das gewünschte Verhalten idealisiert ab (vgl. Absorptionskennfeld in Bild 3-17). Grundlage dieser Kennfelder bilden die in [LKS+11, LSK+11, Löf16] dargestellten Ergebnisse.

Nach dem Abbilden der idealisierten Verhaltensmodelle endet die `Synthese`, und es beginnt die `Analyse` als letzter Schritt der fachgebietsübergreifenden Systemkonzipierung (vgl. Bild 3-6, S. 47). Das Ziel der Analyse ist der prinzipielle Funktionsnachweis der ausgewählten potentiellen Prinziplösung. Zudem werden die Komponentenanforderungen konkretisiert, was die Basis für die nachfolgende Suche nach Lösungselementen bildet.

3.3.3 Analyse

Um das Gesamtsystem modellbasiert analysieren zu können, koppelt man das Verhaltensmodell der Steuerung mit dem Verhaltensmodell der Dynamik. Dies geschieht unter der Simulationsumgebung MATLAB/Simulink. Wie zuvor beschrieben, befindet sich das Modell des ereignisdiskreten Verhaltens bereits unter MATLAB/Simulink-Stateflow. Das dynamische Modell wird mittels einer System Function (S-Function) zu MATLAB/Simulink importiert. Hierbei handelt es sich um eine MATLAB/Simulink-spezifische Modellaustauschmethode [ABR+11]. Durch die Verbindung können das `Verhalten analysiert und das Systemkonzept bewertet` werden (vgl. Bild 3-6, S. 47).

Für das Verbinden der beiden Verhaltensmodelle wird ein entsprechendes BUS-Signal[2] verwendet. Durch diese Verbindung ist es möglich, erste Teilsystemabläufe zu simulieren. Gleichzeitig bietet die BUS-Kommunikation eine einfache Möglichkeit, die zu übertragenden Signale zu erweitern. Dies ist speziell für die nachfolgende Konkretisierung während des disziplinspezifischen Entwurfs nützlich. Für den weiteren Verlauf dieser Arbeit werden das Verhaltensmodell der Steuerung auch als `Steuerungsmodell`[3] und das Verhaltensmodell der Dynamik als `Streckenmodell` bezeichnet, da diese Begrifflichkeiten für die X-in-the-Loop (XiL)-Techniken gebräuchlich sind. Bild 3-18 zeigt die somit entstandene Model-in-the-Loop (MiL)-Umgebung unter MATLAB/Simulink mit den jeweils dahinter liegenden Modellen sowie die Simulationsergebnisse für einen exemplarisch gewählten Systemablauf.

Der Systemablauf setzt sich aus vier Phasen zusammen. Zunächst wird in Phase I durch das Einschalten des Wasserzulaufventils Wasser in das System gelassen.

[2] In einem BUS-Signal werden mehrere Signale zu einem Signal gebündelt.

[3] Dieser Begriff wird verwendet, da es sich um ein Modell der Gerätesteuerung handelt, es können ebenfalls Regelungen enthalten sein.

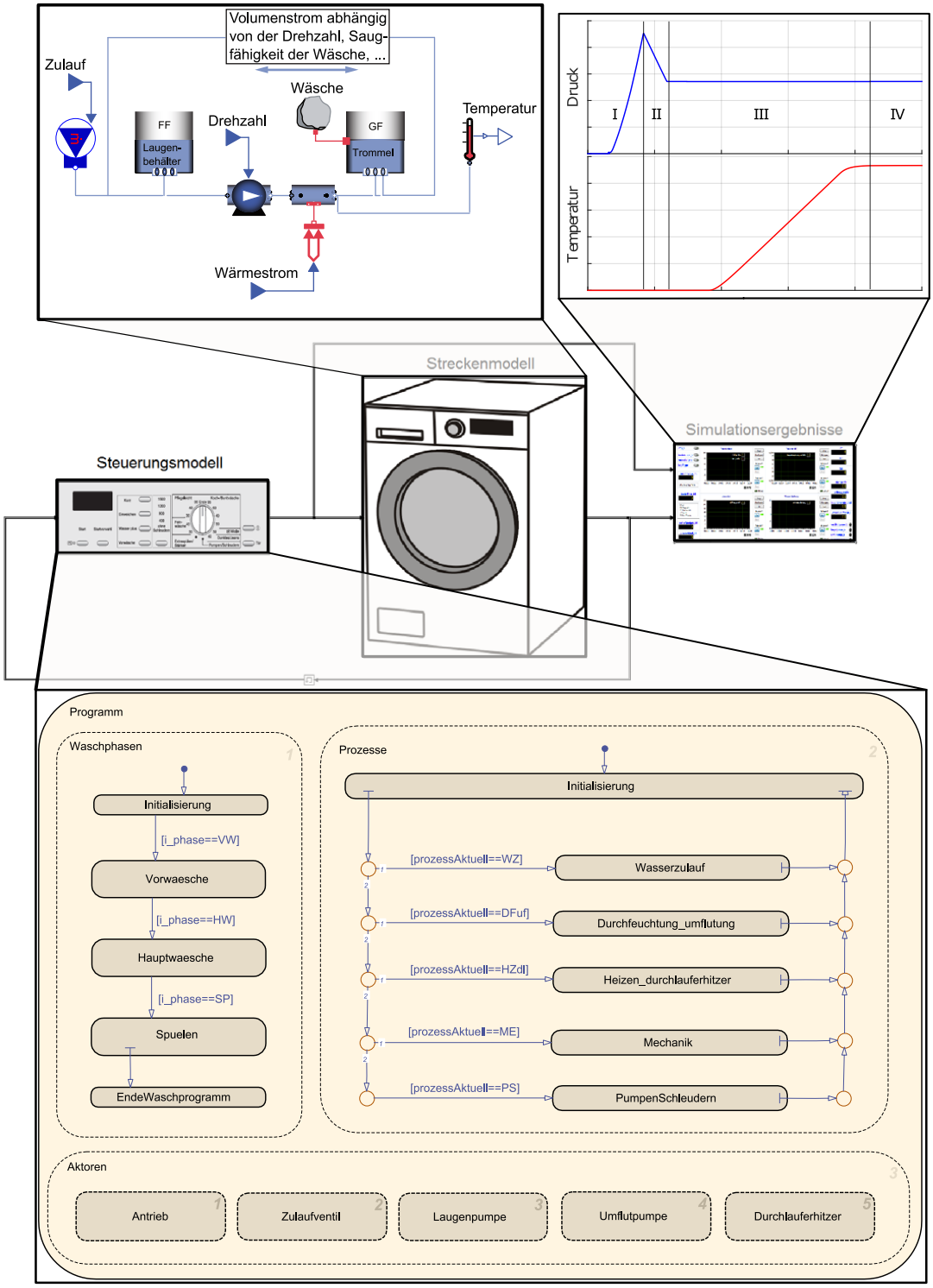

Bild 3-18: MiL-Umgebung für die idealisierten Waschautomaten-Teilmodelle unter MATLAB-Simulink

Durch das einlaufende Wasser steigen der Flüssigkeitsstand und somit das gemessene Drucksignal im Laugenbehälter. Nach dem Wasserzulauf findet eine Saugphase (Phase II) statt. Ein Teil der FF wird von der Beladung aufgenommen und geht zur GF über, wodurch das Drucksignal sinkt. Nach der Saugphase folgt die Heizphase (Phase III), beginnend mit dem Einschalten der Umflutpumpe. Durch die Zirkulation der Lauge im System kann über das Einschalten des Durchlauferhitzers die Temperatur erhöht werden. Nach der Heizphase werden die Umflutung und die Heizung ausgeschaltet (Phase IV).

Für den weiteren Verlauf wird eine schematische Darstellung der jeweiligen XiL-Umgebung gewählt, um eine gewisse Wiedererkennung zu gewährleisten. Bild 3-19 zeigt eine solche schematische Darstellung für eine MiL-Umgebung der idealisierten Modelle der Steuerung sowie der Strecke.

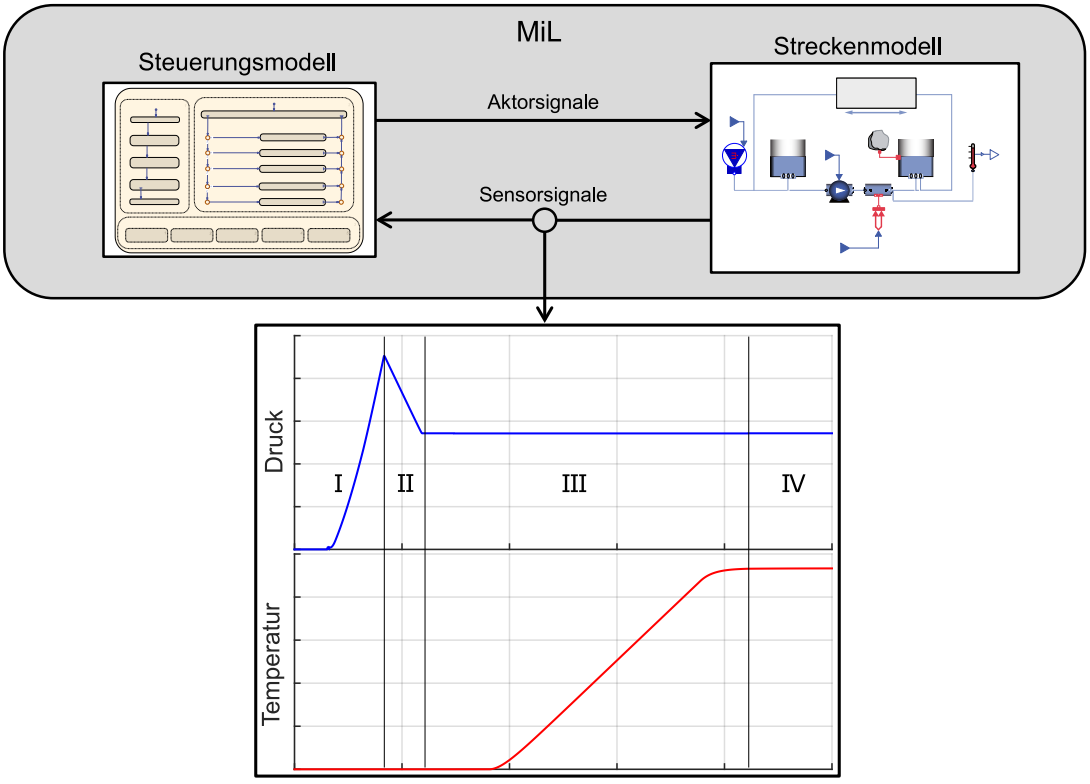

Bild 3-19: Schematische Darstellung der MiL-Umgebung für die idealisierten Teilmodelle

Durch die Verknüpfung der idealisierten Modelle kann der gezeigte Systemablauf modellbasiert entwickelt werden. Dabei kann die grundsätzliche Umsetzung eines Umflut-Waschverfahrens mithilfe eines Durchlauferhitzers demonstriert werden. Zudem lassen sich durch eine Analyse der Simulationsergebnisse erste Abschät-

zungen bezüglich möglicher Energieeinsparpotenziale treffen und die potenzielle Prinziplösung bewertet werden. Bei einer schlechten Bewertung erfolgen weitere Iterationen, um alternative Prinziplösungen zu konzipieren. Bei einer guten Bewertung endet die Phase der Systemkonzipierung, und man geht zum disziplinspezifischen Entwurf über, so wie im Falle der Prinziplösung Umflut-Waschen mittels Durchlauferhitzer.

3.4 Disziplinspezifischer Entwurf

Um die in der Systemkonzipierung entstandene Prinziplösung genauer zu untersuchen, konkretisiert man diese durch den disziplinspezifischen Entwurf. Nachfolgend wird dies für den Bereich Steuerungs- und Regelungstechnik genauer vorgestellt. Hierbei werden wiederum die Schritte `Zielbeschreibung`, `Synthese` und `Analyse` durchlaufen (vgl. Bild 3-20).

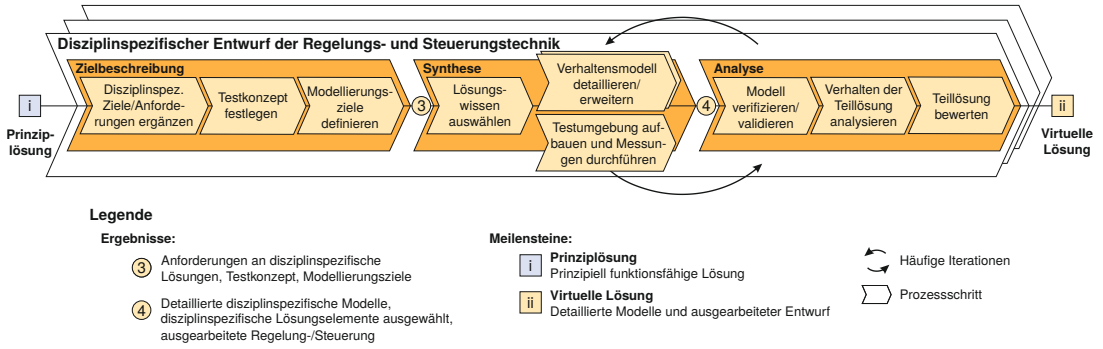

Bild 3-20: Vorgehensmodell für den disziplinspezifischen Entwurf am Beispiel der Regelungs- und Steuerungstechnik (vgl. [Loc, Oes18])

3.4.1 Zielbeschreibung des Verhaltensmodells der Steuerung

Die Zielbeschreibung für den disziplinspezifischen Entwurf Steuerungs- und Regelungstechnik definiert die Anforderungen an die jeweiligen Lösungselemente und legt Modellierungstiefe sowie -ziele fest. Sie gliedert sich in die drei Tätigkeiten `Disziplinspezifische Ziele und Anforderungen ergänzen`, `Testkonzept festlegen` und `Modellierungsziele definieren`.

Die Tätigkeit `Disziplinspezifische Ziele und Anforderungen ergänzen` dient vor allem der Abschätzung möglicher Risiken, die für die konzipierte Prinziplösung eine Rolle spielen. Bei der Verwendung der Komponente Durchlauferhitzer entsteht grundsätzlich kein Risiko, da diese bereits in unterschiedlichen Anwendungsgebieten eingesetzt wird und Funktionalität sowie Zuverlässigkeit nachweis-

lich gegeben sind. Ein wesentlicher Aspekt bei der Prinziplösung Durchlauferhitzer liegt auf dem benötigten neuen Waschverfahren. Die hierdurch verursachten Auswirkungen auf den gesamten Waschprozesses (Waschwirkung, Schaumbildung, Akustik ...) sind nicht bekannt und müssen genauer untersucht werden. Daher wird zunächst das modellierte Verhaltensmodell der Steuerung detaillierter betrachtet. Um dieses zu untersuchen, wird in dem Bereich `Testkonzept festlegen` entschieden, was an einem Modell und was real getestet werden soll, um die Risiken abschätzen zu können.

Da zu diesem Zeitpunkt eine modellbasierte Analyse von Effekten wie Schaumbildung, Waschwirkung, Akustik oder ähnlichen, falls möglich, mit erheblichem Aufwand verbunden ist, soll stattdessen ein reales Funktionsmuster verwendet werden. Des Weiteren wird festgelegt, welche Testhardware und -software eingesetzt wird. Durch das festgelegte Testkonzept werden anschließend die `Modellierungsziele definiert`. Für die Konkretisierung des Verhaltensmodells der Steuerung ist ein solches Modellierungsziel die Funktionsfähigkeit auf einer Prüfstandshardware sowie die Berücksichtigung entsprechender Sicherheitslogiken, die zum Schutze von Mensch und Maschine dienen. Sind diese Schritte abgeschlossen, endet der Bereich der Zielbeschreibung, und es schließt sich die Synthese an.

3.4.2 Synthese des Verhaltensmodells der Steuerung

Der Schritt `Synthese` dient zur Detaillierung der disziplinspezifischen Modelle hinsichtlich der zuvor definierten Anforderungen und Ziele. Die hierzu angewendeten Tätigkeiten werden nachfolgend beschrieben.

In einer ersten Tätigkeit der Synthese werden ausgehend von den zuvor definierten Modellierungszielen `Lösungswissen ausgewählt`. Durch die konkrete Auswahl von realen Lösungselementen können die zuvor modellierten idealisierten Teilmodelle durch detaillierte ersetzt werden. Bei der Suche nach detaillierten Lösungselementen soll langfristig eine entwickelte ENTIME-Suchmaschine verwendet werden können [GTS+14]. Für die Konkretisierung des Verhaltensmodells der Steuerung können detaillierte Lösungselemente, z. B. bestehende Funktionsblöcke, verwendet werden. Da zum derzeitigen Stand auf keine existierenden Lösungselemente für das zu behandelnde System zurückgegriffen werden kann, müssen diese herkömmlich, mit dem entsprechenden Fachwissen, modelliert werden.

Um das `Verhaltensmodell der Steuerung zu detaillieren`, bleibt die in der Systemkonzipierung festgelegte Modellstruktur (*Phasen, Prozesse, Aktoren*) erhalten und wird um den Unterzustand *Sicherheitsueberwachung* ergänzt (vgl. Bild 3-21). Dieser Bereich dient der Sicherheit für Mensch und Maschine, sobald die Steuerung für einen realen Prüfstand verwendet wird. In ihm werden Kriterien modelliert, die einen vollen Zugriff auf alle Aktoren besitzen. So schalten sich im

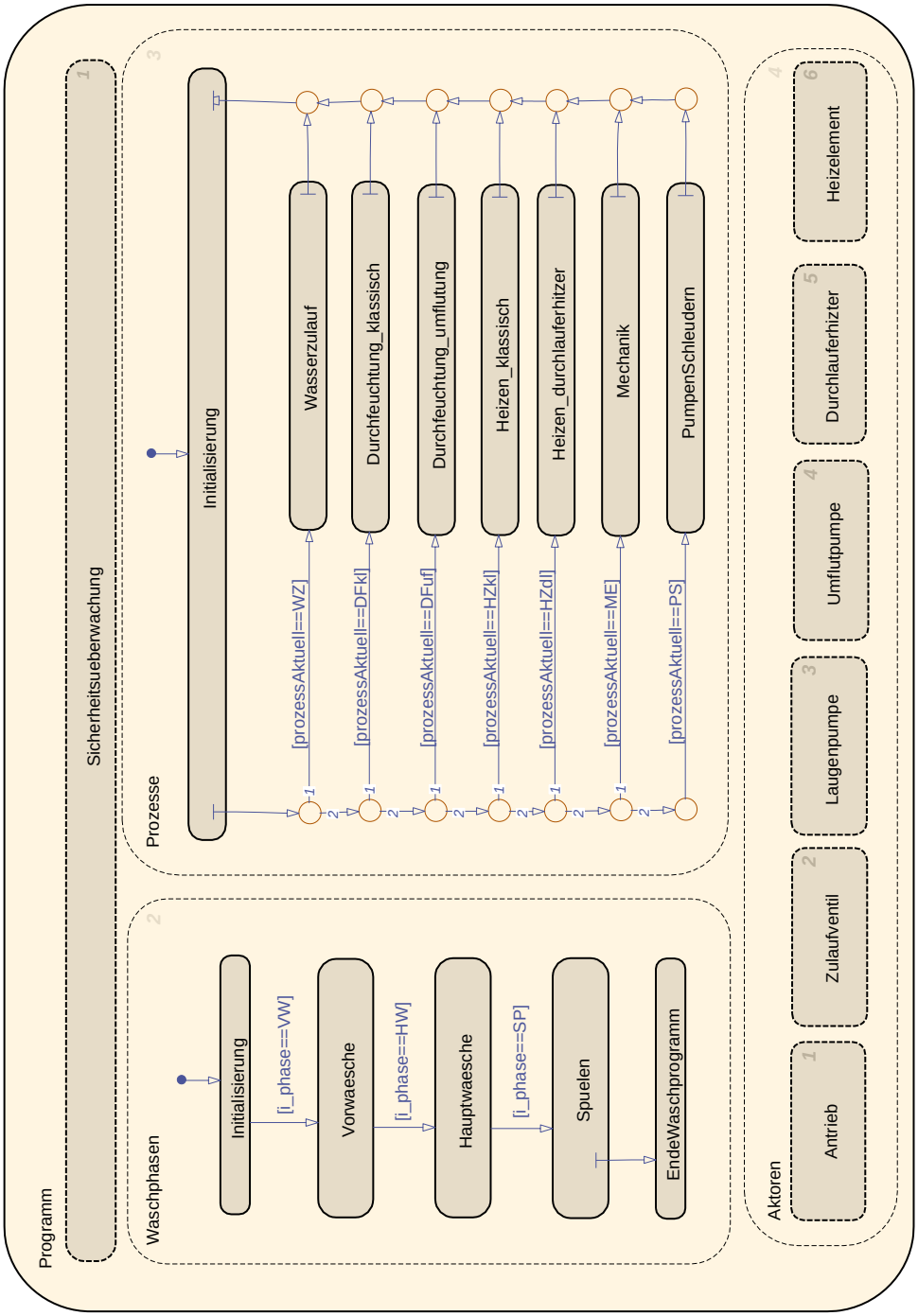

Bild 3-21: Screenshot des Verhaltensmodells der Steuerung unter Stateflow

Falle eines Fehlers oder einer Fehlparametrierung entsprechende Aktoren automatisch aus. Da es sich hierbei um sensible Daten aus der Vorentwicklung eines Industrieunternehmens handelt, wird dieser Block nicht weiter im Detail vorgestellt. Allgemein kann gesagt werden, dass der Unterzustand *Prozesse* um die Unterunterzustände *Durchfeuchtung_klassisch* und *Heizen_klassisch* sowie der Unterzustand *Aktoren* um den Unterunterzustand *Heizelement* erweitert werden. Genaueres hierzu ist unter den in diesem Zusammenhang erarbeiteten Ergebnissen von SCHREIER in [Sch12] zu finden.

Neben dem `Aufbau der realen Testumgebung`, die zuvor in der Zielbeschreibung definiert wurde, besteht ein weiterer wichtiger Aspekt bei der `Detaillierung des Verhaltensmodells der Steuerung` in der Schnittstellenerweiterung für die Anbindung an das reale Prüfgerät. Während bei den simulativen Untersuchungen die definierten physikalischen Ein- und Ausgänge des Steuerungmodells (Sensor- und Aktorsignale) direkt mit den entsprechenden physikalischen Ein- und Ausgängen des Streckenmodells verbunden werden können, müssen für die Anbindung an ein reales Prüfgerät Signalaufbereitungen stattfinden.

Die Signalverarbeitung der Sensor- bzw. Aktorsignale berechnet aus den zur Verfügung stehenden Mess- bzw. Stellsignalen der realen Komponenten die jeweiligen physikalischen Größen. Gekoppelt mit entsprechenden Hardwarekomponenten, bilden die Signalverarbeitungsblöcke die Schnittstelle zwischen der logischen und der physikalischen Ebene (siehe Bild 2-1, S. 8). Sie wandeln die Informationsflüsse in Energieflüsse und umgekehrt. Die realen Mess- und Stellsignale hängen von der verwendeten Hardware ab und können zum Beispiel in Form von Spannungen oder BUS-Protokollen vorliegen. Die entsprechende Signalverarbeitung findet in einem separaten Block und nicht im Verhaltensmodell der Steuerung statt (vgl. Bild 3-22). Sollte sich durch eine Iteration die Zielbeschreibung hinsichtlich des Prüfstandkonzeptes bzw. der Prüfhardware ändern, so müssen lediglich die Schnittstellenblöcke angepasst werden. Das Verhaltensmodell der Steuerung bleibt hingegen unverändert. Die für die Modellierung der Signalverarbeitungsblöcke benötigten Informationen können zum Beispiel aus entsprechenden Datenblättern oder aus Kalibrierungsversuchen entnommen werden.

3.4.3 Analyse des Verhaltensmodells der Steuerung

Die Schritte Synthese und Analyse werden besonders beim disziplinspezifischen Entwurf der Steuerungs- und Regelungstechnik mit häufigen Iterationen durchlaufen. Bevor die Messungen in der Synthese durchgeführt werden, sollte das `Verhaltensmodell der Steuerung` in der Analyse `verifiziert` werden (vgl. Bild 3-20). Hierbei wird geprüft, ob das modellierte Verhaltensmodell der Steuerung formal korrekt ist [Ger13, Har10]. Des Weiteren kann durch eine MiL-Simulation

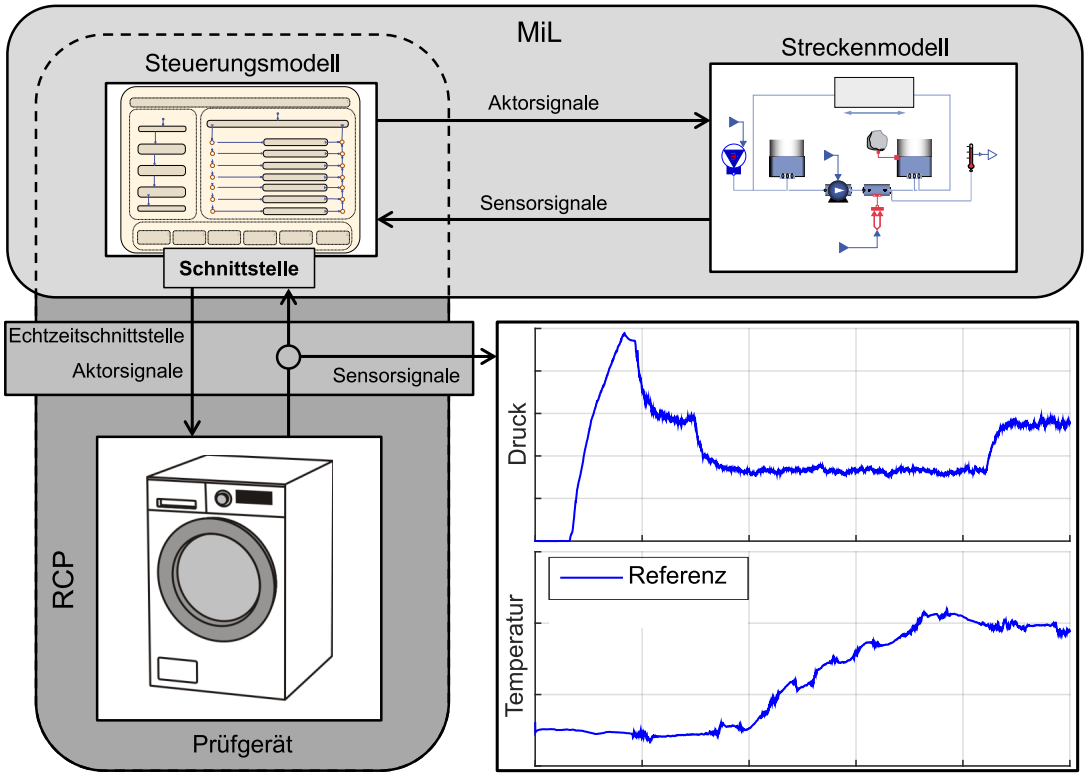

Bild 3-22: Schematische Darstellung der RCP-Umgebung für die Analyse des detaillierten Teilmodells des Steuerungsmodells

geprüft werden, ob das modellierte Steuerungsmodell auch funktional korrekt modelliert wurde, woraus eine signifikante Zeitersparnis und eine Ergebnisreproduzierbarkeit resultieren. Spätes Erkennen von Programmier- bzw. Verfahrensfehlern am realen Prüfstand und damit einhergehende zeitintensive Prüfstands- und -postenaufbereitung werden minimiert. Um das Verhalten der Steuerung im Falle eines auftretenden Fehlers zu untersuchen, bietet es sich an, modellierte Störszenarien in Form eines Fehlerautomaten-Moduls [FKS+13] mit in die MiL-Simulation einzubeziehen. Die somit bereitstehende fehlerreduzierte Ansteuerung kann im Anschluss an einem sogenannten Rapid Control Prototyping (RCP)-Prüfstand gegen nicht modellierte Effekte und Störgrößen im realen Prozess getestet werden.

Für die simulative funktionale Prüfung der Steuerung kann zunächst das bereits idealisiert vorliegende Streckenmodell (Verhaltensmodell der Dynamik) verwendet werden. Sollte eine detailliertere Prüfung als eine Funktionale gefordert sein, z. B. das simulative Prüfen von Waschwirkungen oder des genauen Energieverbrauchs eines neuen Verfahrens, so muss das Streckenmodell entsprechend detail-

liert und validiert werden. Für erste Untersuchungen der Prinziplösung Umflut-Waschverfahren mittels Durchlauferhitzer genügt zunächst eine funktionale Prüfung des Verhaltensmodells der Steuerung, bevor die Messungen durchgeführt werden. Diese werden nach der Verifikation des Steuerungsmodells durch eine Iteration in dem Abschnitt der Synthese mithilfe des RCP-Prüfstandes durchgeführt. Eine schematische Darstellung findet sich in Bild 3-22.

Die `Analyse des Verhaltens` und die anschließende `Bewertung der Systemlösung` (vgl. Bild 3-20, S. 63) des Steuerungsmodells, als Teillösung für das Umflut-Waschverfahren mittels Durchlauferhitzer, erfolgt durch entsprechende Auswertungen der durch den RCP-Prüfstand ermittelten Ergebnisse. Anhand der Erkenntnisse der Prüfversuche sowie der Untersuchung möglicher Risiken wird nach der positiven Bewertung des Umflut-Waschverfahrens mittels Durchlauferhitzers die modellbasierte Entwicklung intensiviert weiter betrieben. Die hierzu `festgelegten Analyse- und Testkonzepte` während der `disziplinübergreifenden Koordination` (vgl. Abschnitt 3.5.1) erfordern ein detailliertes Streckenmodell (Verhaltensmodell der Dynamik), das mithilfe einer Iteration zum Beginn des disziplinspezifischen Entwurfs erarbeitet wird.

3.4.4 Zielbeschreibung des Verhaltensmodells der Dynamik

Eine wesentliche Anforderung bzw. ein wesentliches Modellierungsziel, das in der Zielbeschreibung für die Detaillierung des Streckenmodells definiert wird, ist die Echtzeitfähigkeit des Modells. Ein Rechensystem wird als echtzeitfähig bezeichnet, wenn seine Programme zur Verarbeitung anfallender Daten ständig betriebsbereit sind. Das bedeutet, dass die Verarbeitungsergebnisse innerhalb einer vorgegebener Zeitspanne verfügbar sind [Sch06]. Diese Anforderung ergibt sich aus den unter Abschnitt 3.5.1 vom disziplinübergreifenden Koordinator festgelegten Testkonzepten Software-in-the-Loop (SiL) und Hardware-in-the-Loop (HiL), die später noch genauer erläutert werden. Eine weitere Anforderung an das Modell ist die Abbildung des Wassermanagements und der damit verbundenen Energieverteilung im System, damit simulative Untersuchungen des neuen Heizverfahrens durchgeführt werden können. Die Anforderungen an ein solches Modell können durch den disziplinübergreifenden Koordinator geändert bzw. ergänzt werden. Anforderungen an das Modell bezüglich Waschwirkungssimulationen sowie simulative Bauteiloptimierungen spielen hierbei zunächst keine Rolle. Entsprechende Modellierungsansätze werden daher an dieser Stelle nicht berücksichtigt.

3.4.5 Synthese des Verhaltensmodells der Dynamik

In der Synthese wird, basierend auf den zuvor getroffenen Anforderungen und den damit verbundenen Modellierungszielen, ein detailliertes disziplinspezifisches

Modell erstellt. Die `Detaillierung des Verhaltensmodells der Dynamik` wird nachfolgend genauer beschrieben.

Das detailliertere, erweiterte physikalische Multidomänen-Modell ist in Bild 3-23 dargestellt. Neben den modellierten Teilsystemen (Zulauf, Ablauf, Umflutsystem ...) zeigt es schematisch die Wechselwirkungen der Teilmodelle, die sich auf das Fluidlevel im Laugenbehälter auswirken. Durch die Multidomänenmodellierung, speziell durch die Verknüpfung der fluidtechnischen und der thermodynamischen Koppelpunkte, kann eine Aussage über den Temperaturverlauf und die damit zusammenhängende Energieverteilung getroffen werden.

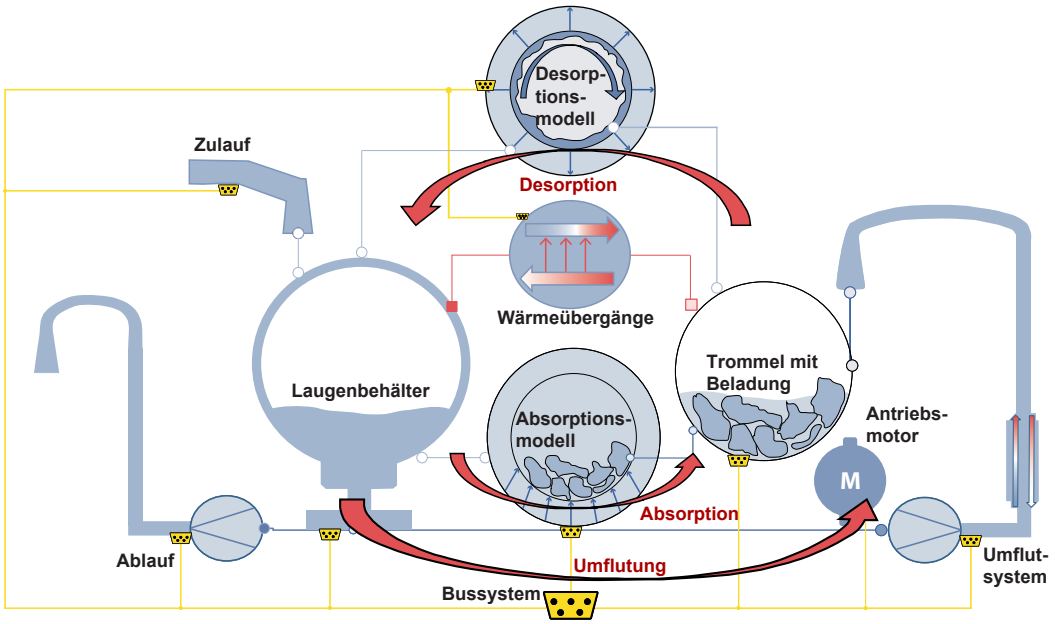

Bild 3-23: Schematische Darstellung der Wechselwirkungen des detaillierten Modells unter Dymola [KTH13]

Das Multidomänen-Modell soll die Flüssigkeitsverteilung im System und speziell die damit zusammenhängende Energieverteilung simulieren, damit alternative Heizverfahren, zum Beispiel mittels Durchlauferhitzers, untersucht werden können. Es dient nicht der Simulation von Waschwirkungen. Deshalb wird der Einfluss der Chemie in diesem Modell nicht berücksichtigt.

Das Multidomänen-Modell ist stark nichtlinear. Neben physikalischen Grundlagen sind vermehrt heuristische Ansätze zur Modellierung verwendet worden. Viele Parameter können nicht direkt aus Datenblättern oder anderen Quellen abgelesen werden. Um einen Einblick in die Komplexität des Gesamtmodells zu bieten, wird im Folgenden das Absorptionsmodell genauer beschrieben, was einer Detaillierung der unter Bild 3-17 (S. 59) gezeigten Kennfelder entspricht.

Absorptionsmodell

Einer der entscheidendsten Einflüsse für die gesamte Dynamik, speziell für die Energieverteilung, ist die Flüssigkeitsverteilung im System. Ein Teil der Flüssigkeit ist in der Wäsche gebunden, und der andere Teil befindet sich frei im Laugenbehälter. Der freie Teil kann durch den Flüssigkeitsstand ermittelt werden, der wiederum durch das Drucksignal messbar ist. Um den Flüssigkeitsstand bzw. das Drucksignal abzubilden, besitzt das Multidomänen-Modell unter anderem einen Teil für die Beschreibung des Absorptionsverhaltens. Bei der zuvor beschriebenen idealisierten Modellierung ist dieser Bereich durch Kennfelder abgebildet worden, die experimentell ermittelt wurden. Für jede Beladungsart und -menge muss jeweils ein Kennfeld erstellt werden; eine gesamtheitliche Abbildung der wesentlichen Effekte ist somit kaum möglich. Zudem entsteht durch die große Kombinationsvielfalt von Beladung und Menge ein entsprechender Aufwand bei den Messungen. Die Detaillierung des Absorptionsmodell soll diese Nachteile beheben. Langfristig soll an dieser Stelle Lösungswissen mithilfe des Semantic Webs wiederverwendet werden, so dass entsprechende `Lösungselemente` `ausgewählt` werden können (vgl. Schritt `Synthese` Bild 3-20 S. 63). Ein solches Lösungselement könnte in Form eines Absorptionsmodells vorliegen. Da zu diesem Zeitpunkt keine entsprechenden Lösungselemente zur Verfügung stehen, wird das Absorptionsmodell auf der Basis von physikalischen Grundlagen, kombiniert mit heuristischen Ansätzen, modelliert. Einen wesentlichen Beitrag hierzu liefern die in diesem Zusammenhang erarbeiteten Ergebnisse von SCHREIER in [Sch13]. Um ein Absorptionsmodell zu erstellen, wurden hierzu Analogien zu anderen Domänen recherchiert und die jeweiligen Ergebnisse transferiert.

Vor allem der Bereich der angewandten Geologie befasst sich intensiv mit der Modellierung und der Simulation von Strömungen in flüssigkeitsleitenden Materialien. In der Hydrogeologie werden Gesetzmäßigkeiten, unter anderem für Versickern, Verdunsten und Grundwasserströmung, entwickelt, die sich im Zusammenwirken zu einem Modell des gesamten Wasserkreislaufs ergänzen. Eine schematische Darstellung des hydrologischen Kreislaufs findet sich in Bild 3-24.

Sowohl Böden als auch Textilien werden gleichsam als poröses Material klassifiziert [Ost11, Ost13]. Ansätze für einzelne bei Böden auftretende Effekte lassen sich somit auf das Saugverhalten von Textilien übertragen. Dies betrifft vor allem Effekte, die aufgrund der Struktur des durchströmten Materials entstehen. Benötigte Materialparameter können den Ergebnissen aus Untersuchungen der Textiltechnik entnommen werden [BBF+93, CG02]. Weitere ergänzende Kennwerte für hydrogeologische Gesetzmäßigkeiten lassen sich aus der Bodenkunde herleiten, mit dem Resultat, dass das Absorptionsverhalten von Wäscheposten im Wesentlichen von den folgenden Effekten abhängt:

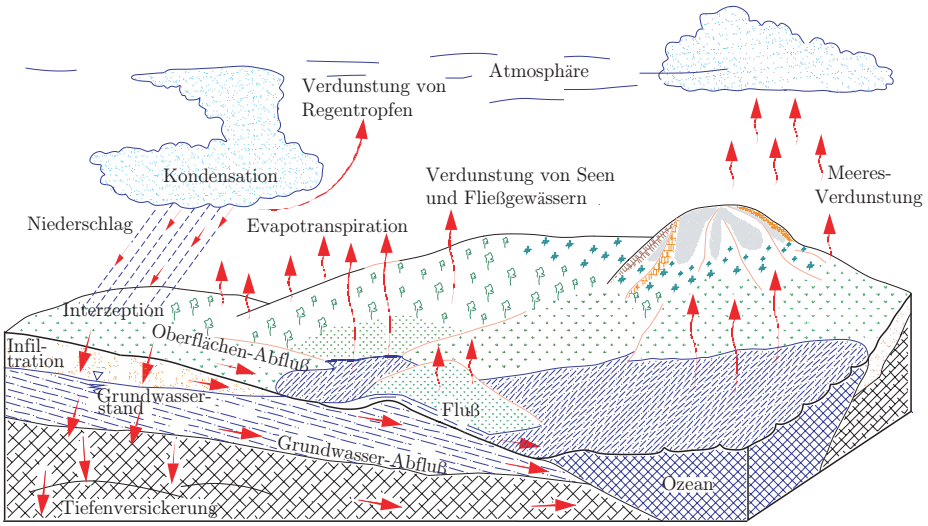

Bild 3-24: Der hydrologische Kreislauf [Ost11]

- Durchlässigkeitsersatzbeiwert $\chi(\rho_F,\eta_F,K,g)$ als Funktion der Dichte ρ_F [$^{kg}/_{m^3}$], der Viskosität η_F [$^{N\cdot s}/_{m^2}$], der Permeabilität[4] K [m^2] und der Erdbeschleunigung g [$^{m}/_{s^2}$],
- Kapillardruck p_C [$^{N}/_{m^2}$],
- Textilsättigung S [-],
- Kontaktfläche A_C [m^2].

Der resultierende Massenstrom zwischen der freien Flüssigkeit im Laugenbehälter und der gebundenen Flüssigkeit im Textil kann nach SCHREIER in [Sch13] durch folgende Gleichung beschrieben werden:

$$\dot{m}_{absorption} = \chi(\rho_F, \eta_F, K, g) \cdot p_C(S) \cdot A_C.$$

Die Werte für die Dichte und die Viskosität können Datenblättern entnommen werden. Die Erdbeschleunigung ist ebenfalls bekannt. Die Einheit der Permeabilität ist [m^2] und ihr Wertebereich für verschiedene Schüttungen und Böden liegt zwischen 10^{-2} und 10^{-12} [Sch74]. Auch für unterschiedliche Wäschearten sind verschiedene Werte der Permeabilität zu erwarten. Durch ihren verschiedenartigen inneren Aufbau werden sie unterschiedlich stark durchströmt. Hinzu kommt

[4] Die Permeabilität beschreibt die Durchlässigkeit eines durchströmten Mediums.

der Einfluss des Waschmittels. Konkrete Werte könnten durch aufwändige Versuche ermittelt werden. Allerdings besitzt das Modell mit dem Kapillardruck in Abhängigkeit der Sättigung $p_C(S)$ bereits eine unbekannte Kennlinie, die es zu identifizieren gilt (vgl. Bild 3-26). Dadurch kann nicht sichergestellt werden, dass der ermittelte Funktionsverlauf der Permeabilität ansatzweise realistische Verhältnisse widerspiegelt. Zudem sollen zu viele freie Parameter vermieden werden, insbesondere solche, die eine ähnliche Bedeutung für das Gesamtergebnis haben. Abweichungen der Permeabilität sollen deshalb durch Anpassung des Kapillardruckverlaufs kompensiert werden. An dieser Stelle wird sich dazu entschlossen, den Wert der Permeabilität auf 1 zu setzen, wodurch der Durchlässigkeitsersatzbeiwert χ als konstant angenommen wird. Die Durchlässigkeit unterschiedlicher Wäschearten wird für sämtliche Sättigungszustände durch die Kapillardruckkurve (vgl. Bild 3-26) abgebildet, sodass die Permeabilität dort berücksichtigt wird. Die weiteren Parameter für das Absorptionsmodell werden nachfolgend genauer beschrieben.

Insgesamt besitzt das Absorptionsmodell 11 Identifikationsparameter (vgl. Tabelle 3-1) für das Identifikationsverfahren, das in dem späteren Abschnitt 4.3 beschrieben wird.

Tabelle 3-1: Identifikationsparameter des Absorptionsmodells

Ident. Name	Untere Grenze	Obere Grenze	Einheit
pS_{Table}	0	10^{-5}	$[N/m^2]$
RD_{max}	50	500	[%]
KA_{Table}	10^{-4}	5000	[−]
$Tank_{Table}$	0	0,6	[m]
rp_{Table}	1	0,05	[m]
f_{r1dyn}	10^{-3}	2	[Hz]
f_{r2dyn}	10^{-3}	2	[Hz]
SV_{Table}	0	1	[−]
KV_{Table}	10^{-2}	1	[−]
f_{r1KV}	10^{-3}	2	[Hz]
f_{r2KV}	10^{-3}	2	[Hz]

Kapillardruck

Bild 3-25 zeigt schematisch die Flüssigkeitsverteilung in den Poren eines Textils [Sch74]. Ein wesentlicher Faktor bei der Flüssigkeitsverteilung ist der Kapillardruck. Er entsteht durch eine gekrümmte Grenzfläche zwischen einer Flüssigkeit und den Poren eines Textils. Der Kapillardruck erhöht sich bei kleineren Poren und verschwindet bei vollständiger Sättigung des Textils.

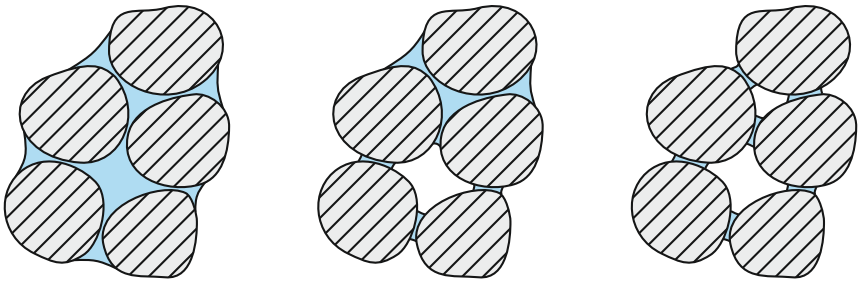

Bild 3-25: Schematiche Darstellung von Flüssigkeitsverteilungen in den Poren eines Textils [Sch13]

Nach derzeitigen Kenntnissen gibt es keinen einheitlichen theoretischen Ansatz zur Beschreibung des Kapillardruckes als Funktion der Sättigung. In diesem Modell wird der Kapillardruck daher als Kennlinie pS_{Table} modelliert. Als Eingang dient der Kennlinie der aktuelle Wert der Sättigung S. Als Ausgang erhält man den aktuellen Wert des Kapillardrucks p_C. Die Kennlinie ist exemplarisch in Bild 3-26 dargestellt. Es wird angenommen, dass sich der Kapillardruck mit steigender Sättigung verringert und in den Grenzen 0–10^{-5} N/m^2 liegt [Sch82]. Den genauen Wert des Kapillardrucks gilt es an jeder Stützstelle zu identifizieren.

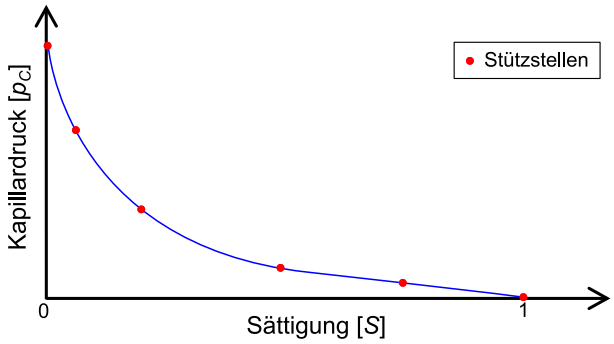

Bild 3-26: Kapillardruckkurve mit exemplarisch dargestellten Stützstellen

Sättigungsgrad

Die maximale Sättigung S_{max} hängt von den Materialeigenschaften des jeweiligen Textils ab. Der Wert der Sättigung liegt in den Grenzen von 0–1 und ist maximal, wenn die Restfeuchte maximal ist, d. h. wenn die Wäsche gesättigt und die GF somit maximal ist. Die aktuelle Restfeuchte RD lässt sich wie folgt beschreiben:

$$RD = \frac{m_F}{m_B} \cdot 100$$

mit:

m_F: Masse der gebundenen Flotte (GF),

m_B: Masse der trockenen Beladung.

Typische Werte für die hier zu betrachtenden Textilien liegen laut Experten in den unter Tabelle 3-1 gezeigten Grenzen.

Kontaktfläche

Zur Berechnung des Massenstroms der aufgesaugten Flotte wird die Querschnittsfläche der Beladung benötigt, die von der Flotte durchströmt werden kann. In der Mantelfläche der Trommel sind Löcher eingebracht, die von freier Flotte durchströmt werden können. Es wird angenommen, dass sich die Wäsche optimal der Trommelgeometrie anpasst. Der Übergang von freier Flotte zu gebundener Flotte (also von trockener zur nasser Wäsche) findet demnach direkt an der Trommelwand statt. Aufgrund dieser Annahme entspricht die Kontaktfläche A_C der Oberfläche der Trommel als Funktion des Flüssigkeitsstands h. Die Berechnung hierzu basiert auf der Geometrie eines Kreiszylinders (vgl. Bild 3-27).

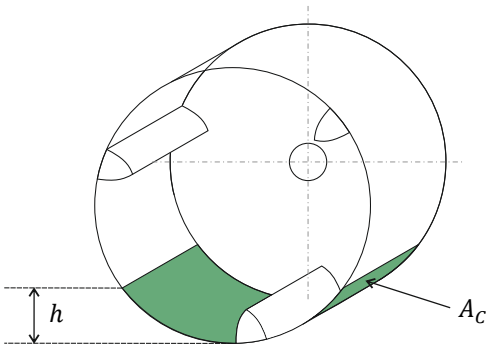

Bild 3-27: Schematische Darstellung der Kontaktfläche zwischen der Beladung und der FF bei ruhender Trommel

Die bisher betrachteten Effekte beschreiben einen langsam ablaufenden Vorgang durch die kapillare Strömung. Sinngemäß steht ein poröses Material in Kontakt zu einer Flüssigkeit, die durch das Porensystem transportiert wird. An der Kontaktstelle zur Flüssigkeit besteht im Material eine hohe Sättigung und an den weiter entfernten Stellen eine geringere. Hierdurch entsteht ein Sättigungsgefälle. Die in der Trommel befindliche Beladung besitzt in der Regel jedoch kein zusammenhängendes Porensystem, da sie meist aus einzelnen Wäschestücken besteht. Durch

die Grenzen der Wäschestücke wird ein durchgängiger Flüssigkeitstransport behindert. Nur die außenliegenden Stücke würden hinreichend durchströmt werden. Durch Drehrhythmen der Trommel wird eine kontinuierliche Umverteilung der Wäschestücke erreicht. In den Pausezeiten liegt dann ein anderer Teil der Wäsche in der freien Flotte im unteren Bereich der Trommel. Insgesamt wird somit eine gleichmäßigere Durchfeuchtung erzielt. Verstärkt wird dies durch die Mitnehmerrippen in der Trommel. Diese nehmen bei drehender Trommel einen Teil der freien Flotte auf und geben diese anschließend wieder ab, wodurch die Wäsche zusätzlich von oben befeuchtet wird. Des Weiteren sorgt die Trommeldrehung für einen kontinuierlichen Austausch von freier und gebundener Flotte, selbst bei vollständig gesättigter Wäsche. Der Kapillardruck kann dies nicht bewirken, da dieser bei vollständiger Sättigung nicht mehr vorliegt. Die beschriebenen Phänomene können in ihrer Gesamtheit nicht durch die zuvor hergeleiteten Gleichungen beschrieben werden. Die dort auftretenden dynamischen Effekte sind ohnehin schwierig zu identifizieren. Deshalb sollen sie zusammenfassend als eine Vergrößerung der durchströmten Querschnittsfläche beschrieben werden, was wiederum den Saugmassenstrom erhöht. Im Modell wird dies durch die Hinzunahme eines drehzahlabhängigen Verstärkungsfaktors $K_A(n)$ berücksichtigt. Die verwendete Kontaktfläche A_C^* berechnet sich daraufhin wie folgt:

$$A_C^*(h,n) = K_A(n) \cdot A_C(h).$$

Der Wertebereich des Verstärkungsfaktors K_A ist nicht genau bekannt. Es wird angenommen, dass der Wert bei einer bestimmten Drehzahl ein Maximum besitzt, bevor er sich verringert. Die Ursache liegt in der bereits angesprochenen Relativbewegung zwischen der Beladung und der Trommel, die ebenfalls ein Maximum bei einer gewissen Drehzahl besitzt, bevor durch die steigende Zentrifugalkraft die Beladung an die Trommel gepresst wird und ein Wäschering entsteht. Um diesen Effekt in dem Multidomänen-Modell abzubilden, wird der zu identifizierende Verstärkungsfaktor KA_{Table} als Kennlinie modelliert. Diese besitzt als Eingang die Drehzahl n und als Ausgang den jeweiligen Verstärkungsfaktor K_A (vgl. Tabelle 3-1).

Eine weitere Erkenntnis ist, dass das Drucksignal nicht allein von der Absorptionsfähigkeit der Beladung abhängt. Der nächste Abschnitt beschreibt die zusätzlich modellierten Effekte, um das Drucksignal so realistisch wie nötig abzubilden.

Hydrostatischer und hydrodynamischer Druck

Der hydrostatische Druck entsteht aus der Flüssigkeit, die nicht in der Beladung gebunden ist (FF), und dem daraus resultierenden Flüssigkeitsstand im Laugenbehälter (vgl. Bild 3-28).

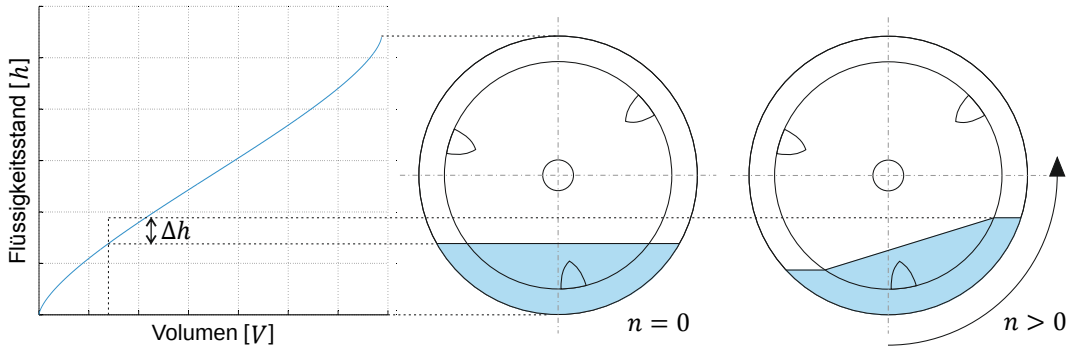

Bild 3-28: Exemplarische Darstellung des Flüssigkeitsstand-Volumen-Verhältnisses unter dem Einfluss der Trommeldrehung

Da die Geometrie des Laugenbehälters Abweichungen von einer zylindrischen Form hat, wird für die Modellierung des Verhältnisses Flüssigkeitsstand-Volumen eine entsprechende Kennlinie hinterlegt: $Tank_{Table}$, deren Werte es zu identifizieren gilt. Effekte aus der Trommeldrehung werden hiermit nicht abgebildet. Hierzu wird der hydrodynamische Druck separat modelliert. Er entsteht durch die Relativbewegung zwischen der Trommel und der Flüssigkeit und führt zu einer einseitigen Erhöhung des Flüssigkeitsstands und somit zu einer Erhöhung des gemessenen Drucksignals (vgl. Δh in Bild 3-28). Für die Berücksichtigung dieses Effektes wird wiederum eine zu identifizierende Kennlinie im Modell hinterlegt (rp_{Table}), die als Eingang die Trommeldrehung n und als Ausgang die Erhöhung des Flüssigkeitsstands Δh besitzt. Zudem ändert sich das Drucksignal mit einer Verzögerung bei Beschleunigung und Entschleunigung der Trommel. Um diese Effekte abzubilden, werden zwei Verzögerungsglieder 1. Ordnung modelliert. Neben den Kennlinien für den hydrostatischen und den hydrodynamischen Druck müssen auch die jeweiligen Zeitkonstanten (f_{r1dyn},f_{r2dyn}) identifiziert werden (vgl. Tabelle 3-1).

Flüssigkeitsverdrängung

Die Beladung stellt ein zusätzliches Volumen V_d in der Trommel dar (vgl. Bild 3-29). Die FF wird durch dieses Volumen verdrängt, wodurch der Flüssigkeitsstand um Δh ansteigt. Das verdrängte Volumen hängt zudem von der Sättigung der Beladung ab. Bei geringer Sättigung wird weniger verdrängt, da mehr freie Poren vorhanden sind. Dies lässt sich ausdrücken durch

$$V_d^*(S,h) = S_V(S) \cdot V_{Tr}(h).$$

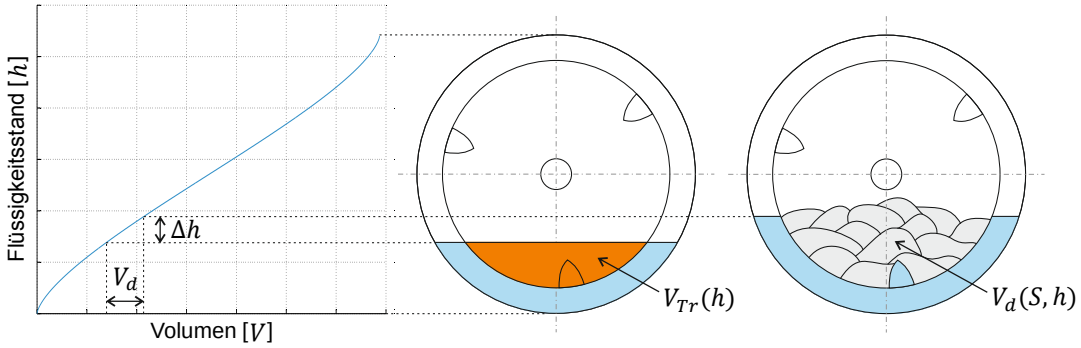

Bild 3-29: Flüssigkeitsverdrängung aufgrund der Beladung

Um die Abhängigkeiten der Sättigung zu simulieren, wird die Größe S_V (in den Grenzen von 0–1) modelliert. Wenn $S_V = 1$, entspricht das verdrängte Volumen dem Trommelvolumen V_{Tr} als Funktion des Flüssigkeitsstands h. Durch die Drehung der Trommel und die damit zusammenhängende Bewegung der Beladung wird das verdrängte Volumen verkleinert (vgl. Bild 3-30).

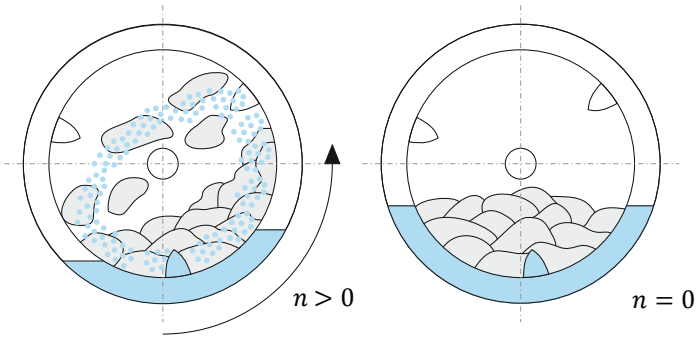

Bild 3-30: Schematische Darstellung des Einflusses der Trommeldrehung auf das verdrängte Volumen

Die zuvor vorgestellte Gleichung wird um einen drehzahlabhängigen Verstärkungsfaktor $K_V(n)$ erweitert:

$$V_d(S,h,n) = S_V(S) \cdot V_{Tr}(h) \cdot K_V(n).$$

Für die Modellierung der sättigungsabhängigen Größe $S_V(S)$ sowie des drehzahlabhängigen Verstärkungsfaktors $K_V(n)$ werden die beiden Kennlinien SV_{Table} und KV_{Table} verwendet. Die Verzögerungen der Volumenänderungen werden wie zuvor

durch zwei Verzögerungsglieder 1. Ordnung modelliert. Alle zu identifizierenden Parameter für die Flüssigkeitsverdrängung sind in Tabelle 3-1 abgebildet.

Die Modellierung des Absorptionsverhaltens, als exemplarisches Beispiel für die Detaillierung des Verhaltensmodells der Dynamik im disziplinspezifischen Entwurf, ist an dieser Stelle abgeschlossen, und es folgt der Schritt der `Analyse` (vgl. Bild 3-20 S. 63). Weitere Details für den Aufbau physikalischer Modelle im Kontext der Haushaltsgerätetechnik finden sich unter [LKS+11, LSK+11, Löf16].

3.4.6 Analyse des Verhaltensmodells der Dynamik

Durch die Konkretisierung der Verhaltensmodelle (Steuerung und Strecke) können detaillierte Analysen des Gesamtsystems durch MiL-Simulationen gewonnen werden. Die Ergebnisse des Multidomänen-Modells der Strecke sind jedoch erst aussagekräftig, wenn das Modell validiert ist. Die `Modellvalidierung` ist die erste Tätigkeit im Schritt der `Analyse` (vgl. Bild 3-20, S. 63). Hierzu kann eine im Rahmen dieser Arbeit entwickelte Parameteridentifikations- und Modellvalidierungsmethodik (vgl. Kapitel 4) verwendet werden. Voraussetzung für die Verwendung dieser Methodik ist das Vorhandensein von Referenzgrößen, mit deren Hilfe das Modell validiert werden kann. Bei den Referenzgrößen kann es sich zum Beispiel um Simulationsergebnisse eines bereits validierten Streckenmodells mit gegebenenfalls höherer Modellierungstiefe handeln oder, wie in diesem Fall, um reale Messsignale, die in Form von Prüfstandsversuchen während der Tätigkeit `Messungen durchführen` in der `Synthese` gewonnen werden (vgl. Bild 3-20, S. 63). Das genaue Vorgehen für die Validierung des Multidomänen-Waschautomaten-Modells wird in Abschnitt 4.3 dargestellt. Die Ergebnisse der Modellvalidierung sind exemplarisch in Bild 3-31 zu sehen. Zudem zeigt es schematisch die Verknüpfung des konkretisierten Verhaltensmodells der Steuerung mit dem konkretisierten Verhaltensmodell der Dynamik (Streckenmodell) sowie mit dem realen Prüfgerät unter Zuhilfenahme des Schnittstellenblocks.

Die Simulationsergebnisse stimmen sehr gut mit den Referenzgrößen überein. Die gezeigte Modellierung des Absorptionsverhaltens berücksichtigt somit alle nötigen Effekte im Multidomänen-Modell, um das reale Drucksignal so genau wie nötig abzubilden. Dass die Identifikation und die damit verbundene Validierung in den anderen Domänen ebenfalls erfolgreich verliefen, wird exemplarisch für die Domäne Thermodynamik durch den in Bild 3-31 gezeigten Temperaturverlauf veranschaulicht. Durch die sehr gute Abbildung des Wassermanagements und der damit verbundenen Energieverteilung im System können gezielt simulative Untersuchungen durchgeführt werden, um das Umflut-Waschverfahren mittels Durchlauferhitzers weiter zu optimieren. Eine andere wichtige Anforderung an das Streckenmodell ist die Echtzeitfähigkeit, da das Modell für die weitere

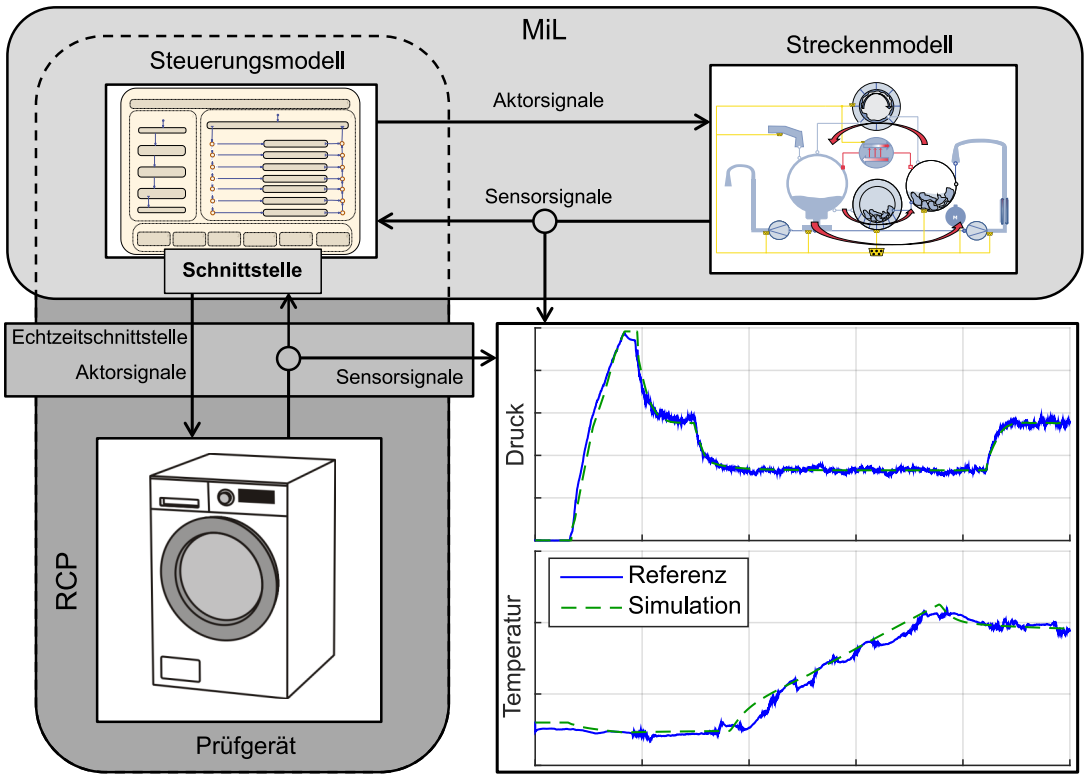

Bild 3-31: Schematische Darstellung der MiL- und der RCP-Umgebung für die detaillierten Teilmodelle

modellbasierte Entwicklung und speziell für die XiL-Techniken SiL und HiL verwendet werden soll. Letztlich kann die Echtzeitfähigkeit des Modells erst dann nachgewiesen werden, wenn es auf einer Echtzeithardware ausgeführt wird und entsprechende Echtzeitbedingungen erfüllt sind (vgl. [Löf16]). Jedoch kann bereits an dieser Stelle davon ausgegangen werden, dass das entwickelte Modell echtzeitfähig ist, da es mit einem Fixstep-Solver *Euler* und einer Schrittweite von 0,01 sek simulierbar ist.

Bevor an dieser Stelle mit den vorliegenden detaillierten Verhaltensmodellen der Steuerung und der Strecke die konkrete Beschreibung des modellbasierten Entwurfs mechatronischer Systeme am Beispiel des Umflut-Waschverfahrens endet, werden nachfolgend der Vollständigkeit halber weitere durchzuführende Phasen kurz beschrieben, welche jedoch nicht den Schwerpunkt dieser Arbeit darstellen.

3.5 Weitere durchzuführende Phasen

Um sicherzustellen, dass die in den jeweiligen Disziplinen erarbeiteten Ergebnisse nicht divergieren, findet parallel zu dem `disziplinspezifischen Entwurf` eine `disziplinübergreifende Koordination` statt, bevor die `modellgestützte Systemintegration` beginnt. Die jeweils durchzuführenden Schritte der beiden Phasen werden kurz vorgestellt.

3.5.1 Disziplinübergreifende Koordination

Während der disziplinübergreifenden Koordination sorgt ein Koordinator dafür, dass die parallel erarbeiteten Ergebnisse der unterschiedlichen Disziplinen nicht auseinanderlaufen. Diese Person sammelt und überwacht gesamtsystemrelevante Änderungen, analysiert diese und legt Strategien für die modellbasierte Integration sowie die anschließende Inbetriebnahme fest [Jus13]. Orientieren kann sich der disziplinübergreifende Koordinator ebenfalls an den drei Schritten `Zielbeschreibung`, `Synthese` und `Analyse` (vgl. Bild 3-32).

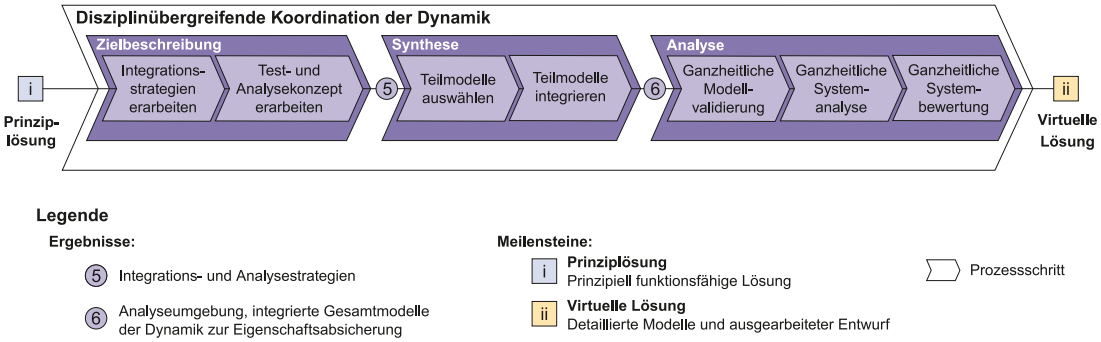

Bild 3-32: Vorgehensmodell für die disziplinübergreifende Koordination (vgl. [Loc, Oes18])

Die **Zielbeschreibung** startet mit der `Erarbeitung von Integrationsstrategien`. Diese hängen maßgeblich von den parallel erstellten Teillösungen und Anforderungen an das zu betrachtende System ab. Bei dem zuvor genannten Beispiel soll durch die Erkenntnisse aus den Prüfversuchen sowie der Untersuchung möglicher Risiken nach der positiven Bewertung des Umflut-Waschverfahrens mittels Durchlauferhitzers die modellbasierte Entwicklung intensiviert weiter betrieben werden. Hierzu werden in der Zielbeschreibung der modellgestützten Systemintegration die Techniken SiL sowie HiL als `Testkonzepte` für das zu entwickelnde Waschverfahren `festgelegt`. Hierzu gehört auch die Ausarbeitung entsprechender Zeitpläne. Durch diese Techniken wird die Testtiefe der Waschverfahrens-Logik, vorliegend als Steuergerätecode sowie später als reales Steuergerät, erhöht.

Bild 3-33 ist eine Erweiterung der Bilder 3-19 (S. 62), 3-22 (S. 67), 3-31 (S. 79) und zeigt schematisch die jeweilige XiL-Technik. Den Kern dieser Techniken bildet ein detailliertes, echtzeitfähiges Streckenmodell. Die besonderen Anforderungen und Modellierungsziele für dieses Modell werden durch den disziplinübergreifenden Koordinator definiert und durch den jeweiligen Domänenexperten im disziplinspezifischen Entwurf umgesetzt (vgl. Abschnitt 3.4.4).

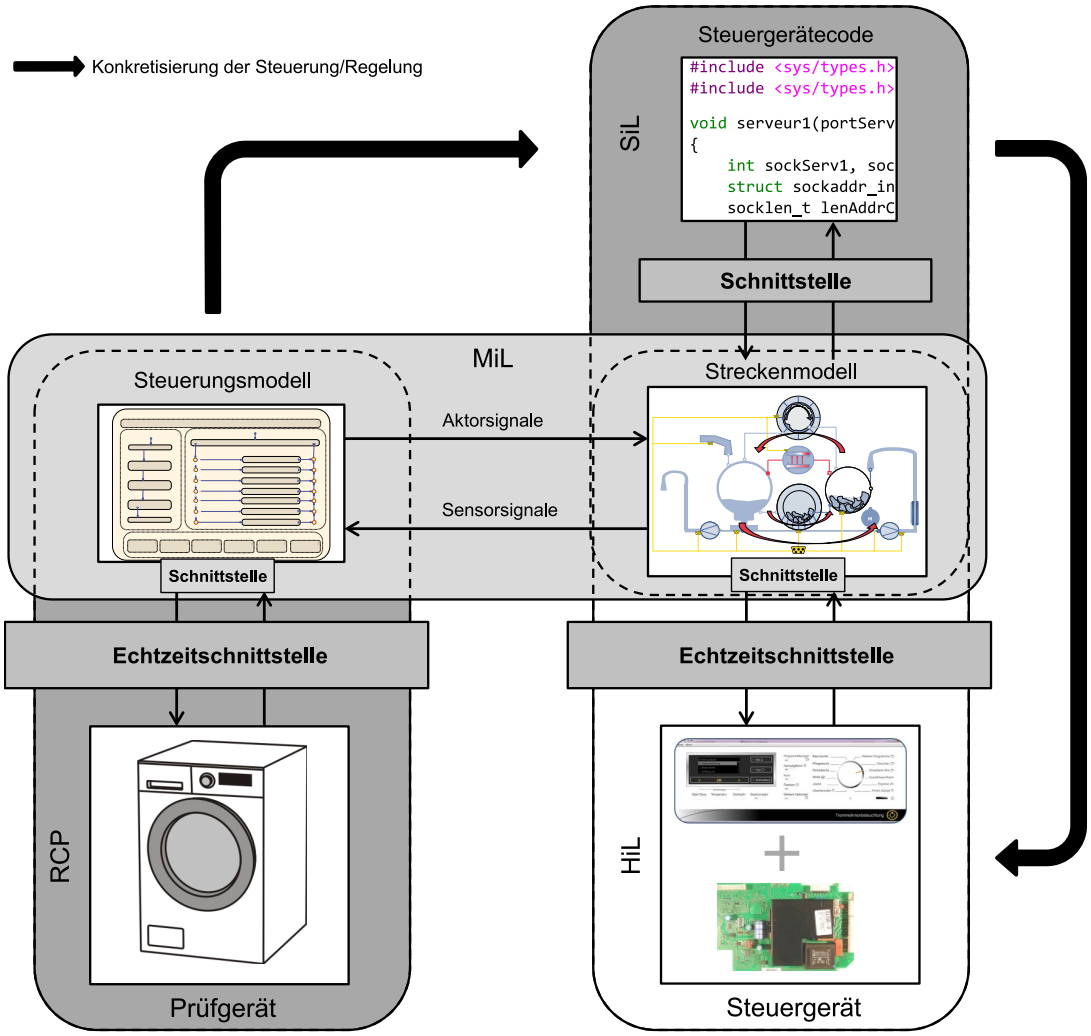

Bild 3-33: Schematische Darstellung der XiL-Techniken MiL, RCP, SiL und HiL (in Anlehnung an [KFS+12])

Nachdem Integrations- und Analysestrategie festgelegt sind, werden in der **Synthese** die entsprechenden Teilmodelle ausgewählt und zu einem Gesamtsystemmodell integriert (vgl. Bild 3-32). Bei dem Gesamtmodell der Strecke handelt es sich um das zuvor gezeigte Multidomänen-Modell des Waschautomaten. Neben

den Teillösungen für das Abbilden des Wassermanagements sowie der damit verbundenen Energieverteilung ist es außerdem denkbar, Teilmodelle für das Abbilden der Schwingungsdynamik mit in das Gesamtmodell einzubinden. Ein solches Teilmodell ist zum Beispiel in [Sch17] entwickelt worden. Aufgrund der unterschiedlichen Domänen, die in einem Gesamtmodell der Strecke zusammengeführt werden, kann dieses mechanische, elektrische, fluidtechnische sowie thermische Effekte abbilden. Zu der Tätigkeit `Teilmodelle auswählen und integrieren` (vgl. Bild 3-32) gehört neben der Integration der Teilmodelle zu einem Gesamtmodell auch die modelltechnische Integration von benötigten Schnittstellen. Diese können ebenfalls als parallel erarbeitete Teillösungen ausgewählt und integriert werden. Sollte eine benötigte Teillösung nicht vorliegen, muss der disziplinübergreifende Koordinator eine entsprechende Erarbeitung im `disziplinspezifischen Entwurf` veranlassen. Die Teillösung für die Schnittstelle des Bereiches SiL ist zum Beispiel in [DS14] entwickelt worden.

Im Schritt **Analyse** muss eine `ganzheitliche Modellvalidierung` durchgeführt werden, bevor die `ganzheitliche Systemanalyse` erfolgt (vgl. Bild 3-32). Die jeweiligen Teillösungen sind während der Analyse des disziplinspezifischen Entwurfs bereits validiert worden (vgl. Abschnitt 3.4.6). Es muss jedoch sichergestellt werden, dass durch das Verbinden der unterschiedlichen Teillösungen das Gesamtverhalten dem realen Verhalten noch hinreichend genau entspricht. Aufgrund starker Nichtlinearitäten, die solche Teillösungen aufweisen können, in Verbindung mit wechselnden Betriebspunkten, die zuvor evtl. nicht betrachtet wurden, ist dies ohne eine erneute Modellvalidierung nicht sichergestellt. Auch hierzu bietet sich die Parameteridentifikations- und Modellvalidierungsmethodik an, die in Kapitel 4 genauer vorgestellt wird.

Durch das integrierte, validierte Multidomänen-Modell der Strecke kann die Steuerung während der Systemanalyse ganzheitlich auf die unterschiedlichen Einflüsse der realen Strecke virtuell getestet und analysiert werden, bevor das reale Steuergerät aufgebaut wird. Die schwarzen Pfeile in Bild 3-33 zeigen hierbei die Konkretisierung der zuvor entwickelten virtuellen Steuerung bis hin zum realen Steuergerät. Letztlich liegt, durch die `ganzheitliche Systembewertung`, die *virtuelle Lösung* vor, und die Phase der `modellgestützten Integration` beginnt.

3.5.2 Modellgestützte Systemintegration

Die modellgestützte Systemintegration ist ein Teilbereich der im V-Modell etablierten Systemintegration. Sie konkretisiert die zuvor erarbeiteten Integrations- und Analysestrategien, erstellt die benötigte reale Testumgebung zur Eigenschaftsabsicherung und führt eine ganzheitliche Systembetrachtung und -bewertung der integrierten Lösung durch. Als Ergebnis endet die modellgestützte Systeminte-

gration mit der *konkreten Lösung* (vgl. Bild 3-34). Die hierzu zu durchlaufenden Schritte werden nachfolgend kurz beschrieben.

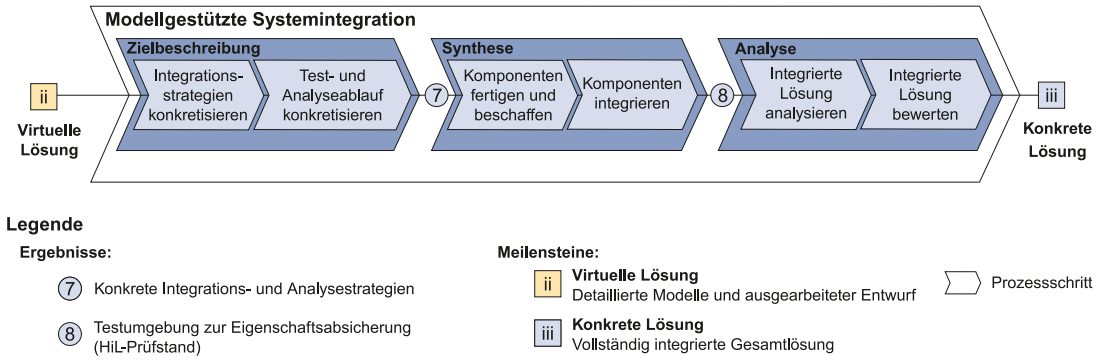

Bild 3-34: Vorgehensmodell für die modellgestützte Systemintegration (vgl. [Loc, Oes18])

Während der **Zielbeschreibung** werden die durch den disziplinübergreifenden Koordinator erarbeiteten `Integrationsstrategien` sowie der erarbeitete `Test- und Analyseablauf konkretisiert` (vgl. Bild 3-34). Anschließend werden während der **Synthese** die benötigten `Komponenten gefertigt` bzw. `beschafft`, so dass durch eine entsprechende `Integration der Komponenten` eine Testumgebung zur Eigenschaftsabsicherung vorhanden ist. Im Falle des Umflut-Waschverfahrens besteht eine solche Testumgebung in der Umsetzung des zuvor festgelegten HiL-Testkonzeptes. Hierzu wird ein Schaltkasten, in dem das reale Steuergerät integriert ist, aufgebaut und mittels entsprechender Schnittstellen mit dem virtuellen Gesamtmodell der Strecke verbunden. Der Aufbau eines solchen Testkonzeptes inklusive der benötigten Schnittstelle, sowohl software- als auch hardwareseitig, ist unter anderem in [LSK+11] zu finden. In der anschließenden **Analyse** erfolgt als letzter Schritt der modellgestützten Systemintegration die `Bewertung` der `integrierten Lösung`, so dass als Ergebnis die *konkrete Lösung* vorliegt und die Fertigung sowie die reale Inbetriebnahme erfolgen können.

4 Parameteridentifikations- und Modellvalidierungsmethodik

Der nachfolgende Abschnitt beschreibt eine Methodik zur teilautomatisierten Parameteridentifikation für die Validierung von Multidomänen-Modellen. Hierzu ist ein Parameteridentifikations-Tool, bestehend aus einer MATLAB-Identifikationsumgebung und einem Interface zur Simulation von FMU-Modellen, entwickelt worden. Die MATLAB-Identifikationsumgebung ermöglicht eine teilautomatisierte Parameteridentifikation unter Verwendung etablierter Verfahren für komplexe, stark nichtlineare Multidomänen-Modelle. Um solche Modelle in die erstellte Umgebung einzubinden, wird das entstandene Interface gemäß dem FMI-Standard verwendet. Das Interface stellt zudem die Identifikationsparameter der Identifikationsumgebung automatisiert zur Verfügung. Da es sich bei dem Interface um eine standardisierte Schnittstelle handelt, ist die Identifikationsumgebung unabhängig von der verwendeten Modellentwicklungslandschaft, was ein Höchstmaß an Flexibilität bietet.

Nachfolgend werden die MATLAB-Identifikationsumgebung und der Aufbau des FMU-Interfaces kurz erläutert. Anschließend wird zur Veranschaulichung die Funktionalität der Parameteridentifikations- und Modellvalidierungsmethodik an einem einfachen Anwendungsbeispiel vorgestellt, bevor sie für die Validierung des komplexen Multidomänen-Modells des Waschautomaten angewendet wird.

4.1 Parameteridentifikations-Tool

Bild 4-1 zeigt schematisch die Gesamtarchitektur des entwickelten Parameteridentifikations-Tools. Es besteht aus einem FMU-Interface und einer MATLAB-Identifikationsumgebung. Das FMU-Interface stellt die Schnittstelle zwischen der Identifikations- und der Modellierungsumgebung dar und besteht aus einem Initialisierungsskript und einem Simulationsskript. Es dient zur Einbindung von FMU-Modellen zur Nutzung von Identifikationsalgorithmen oder weiteren modellbasierten Techniken in MATLAB [Sch17]. Die Basis bildet ein Software Development Kit (SDK) von Qtronic [PC14], das wesentliche Funktionen (XML-Parsen, DLL-Laden ...) für die Verarbeitung von FMUs beinhaltet. Das SDK wurde für die Ansteuerung der FMU aus MATLAB heraus und für die Übergabe der benötigten Modellgrößen für die Parameteridentifikation entsprechend erweitert [SKO+13, KST14].

Die MATLAB-Identifikationsumgebung ist in zwei Hauptbereiche unterteilt (vgl. Abschnitt 4.1.1). In einem ersten Bereich wird überprüft, ob das Modell validiert

ist, das heißt, ob es ein gewünschtes Verhalten hinreichend genau abbildet. Ist dies nicht der Fall, findet in einem zweiten Bereich eine Parameteridentifikation statt. Das Vorgehensmodell für die Modellvalidierung ist schematisch im unteren Bereich in Bild 4-1 dargestellt. Detaillierter wird es in Abschnitt 4.2 erläutert. Nachfolgend wird zunächst die Funktionsweise sowohl des FMU-Interfaces als auch der MATLAB-Identifikationsumgebung genauer vorgestellt.

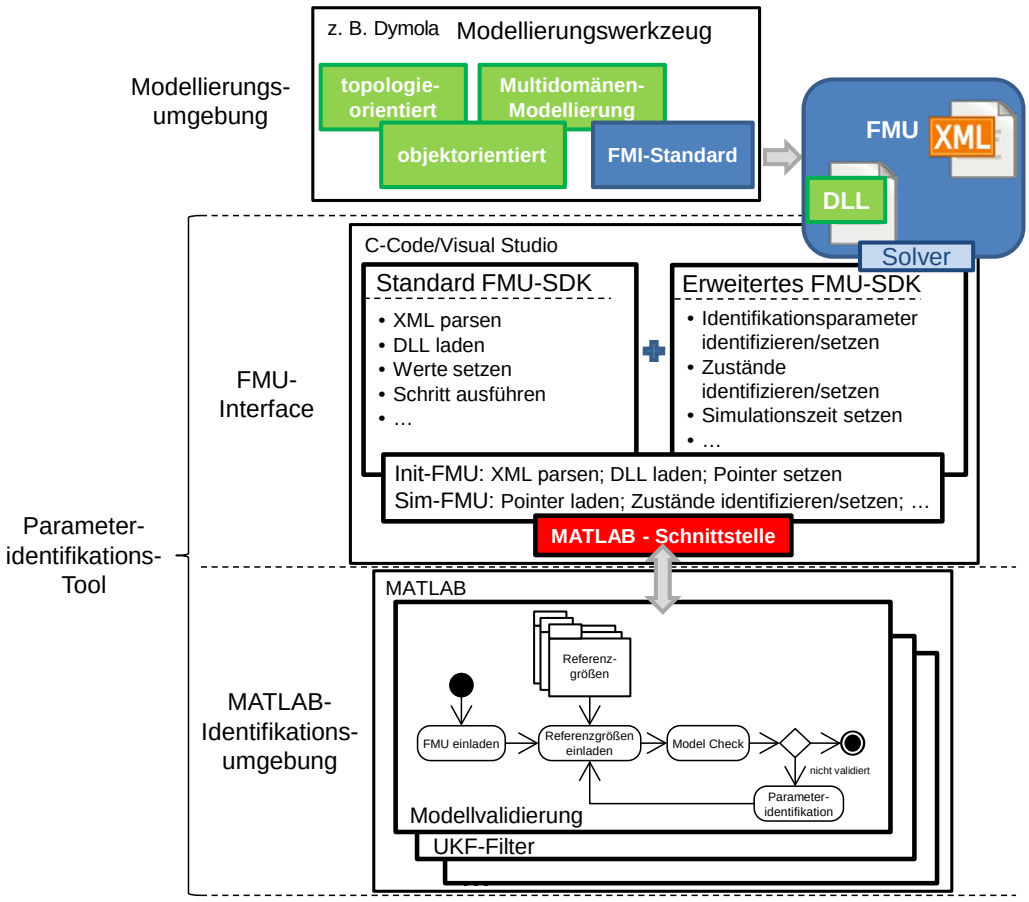

Bild 4-1: Schematische Darstellung des entwickelten Parameteridentifikations-Tools, bestehend aus dem FMU-Interface und der MATLAB-Identifikationsumgebung (in Anlehnung an [SKO+13, KST14])

4.1.1 MATLAB-Identifikationsumgebung

Für eine erleichterte Anwendung der Parameteridentifikations- und Modellvalidierungsmethodik beinhaltet die erstellte MATLAB-Identifikationsumgebung eine graphische Benutzeroberfläche (Graphical User Interface (GUI)). Die jeweils

durchzuführenden Schritte sind durch ein hinterlegtes Vorgehensmodell in der GUI definiert, was in Abschnitt 4.2 an einem Beispiel demonstriert wird. Das Zusammenwirken der MATLAB-Identifikationsumgebung mit dem FMU-Interface ist etwas detaillierter in Bild 4-2 dargestellt.

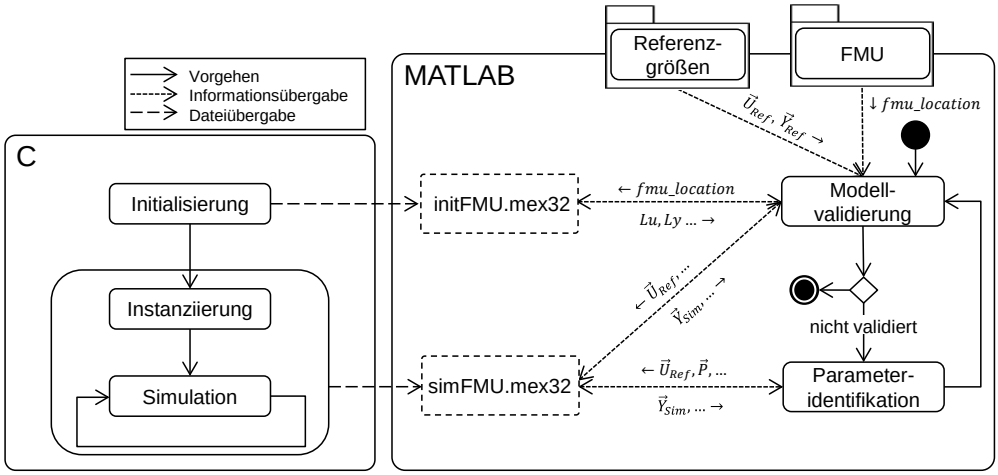

Bild 4-2: Schematische Darstellung des Zusammenwirkens des FMU-Interfaces und der MATLAB-Identifikationsumgebung

Um das als C-Code unter Visual Studio entwickelte FMU-Interface in der MATLAB-Identifikationsumgebung nutzbar zu machen, gibt es im C-Code einen fest definierten Methodenrumpf *mexFunction*. Hierdurch und mit den von MATLAB zur Verfügung gestellten Compiler-Funktionen werden eine *initFMU.mex32*-Datei und eine *simFMU.mex32*-Datei kompiliert, die wiederum von MATLAB aufrufbar sind (vgl. Bild 4-2). Der genaue Aufbau des Initialisierungsskriptes und des Simulationsskriptes wird in Abschnitt 4.1.2 beschrieben. In der MATLAB-Identifikationsumgebung wird während der Modellvalidierung zunächst der Pfad der FMU geladen (*fmu_location*) und an das Initialisierungsskript des FMU-Interfaces übergeben. Mithilfe der somit automatisch zur Verfügung stehenden Informationen bzgl. der Ein- und der Ausgänge der FMU (Lu und Ly) können im nächsten Schritt Referenzgrößen ($\vec{U}_{ref}$ und $\vec{Y}_{ref}$) eingeladen und mit den entsprechenden Schnittstellen der FMU verbunden werden. Durch das Ausführen des Simulationsskriptes des FMU-Interfaces über einen vorgegebenen Simulationszeitraum kann anschließend ein Vergleich zwischen den Simulationsgrößen $\vec{Y}_{sim}$ und den Referenzgrößen $\vec{Y}_{ref}$ gezogen werden. Hierdurch wird geprüft, ob das zu untersuchende Modell ein gewünschtes Verhalten hinreichend genau abbildet. Ist dies der Fall, so ist das Modell validiert; anderenfalls erfolgt die Parameteridentifikation.

Die Identifikation basiert hierbei auf den Parameterschätzmethoden für parametrische Modelle (vgl. Abschnitt 2.5). Es ist zu wählen zwischen einem Identifizieren durch *trial and error* oder durch das Verwenden von Optimierungsverfahren. Bei den auszuwählenden Optimierungsverfahren handelt es sich um etablierte Methoden und Algorithmen, die durch MATLAB zur Verfügung gestellt werden (vgl. Abschnitt 2.6.3). Für die Auswahl und für die entsprechend zu wählenden Einstellungen eines Optimierungsverfahrens sind Voreinstellungen hinterlegt. Des Weiteren wird die Zielfunktion automatisch gebildet. Hierbei handelt es sich derzeit um die quadratische Fehlerfläche zwischen dem ausgewählten Simulations- und dem Referenzbereich. Aufgrund der bereits hinterlegten Voreinstellungen kann die Identifikation zum Großteil automatisiert erfolgen. Zugleich ermöglicht die skriptbasierte Identifikationsumgebung unter MATLAB ein einfaches Ändern bzw. Ergänzen der Voreinstellungen. Aufgrund der Tatsache, dass durch die Nutzung des FMU-Interfaces der gesamte Zustandsvektor $\vec{X}$ zur Verfügung steht und dieser zu jedem Simulationsschritt neu gesetzt werden kann (vgl. Abschnitt 4.1.2), ist eine sequentielle Simulation des Gesamtmodells möglich. Kombiniert mit der Hinzunahme einer Sensitivitätsanalyse während des Bereiches der Parameteridentifikation, kann das jeweils ausgewählte Optimierungsverfahren noch effizienter eingesetzt werden, indem eine sequentielle Optimierung verschiedenster Abschnitte mit den jeweils sensitivsten Parametern stattfindet (vgl. Abschnitt 4.2.2). Eine genauere Beschreibung des Vorgehens wird in Abschnitt 4.2 an einem einfachen Beispiel gezeigt. Zunächst wird jedoch das der Simulation zugrundeliegende FMU-Interface vorgestellt.

4.1.2 FMU-Interface

Das als C-Code unter Visual Studio entwickelte FMU-Interface basiert auf zwei unterschiedlichen Skripten: einem Initialisierungs- und einem Simulationsskript (vgl. Bild 4-3). Die Initialisierung dient zum Laden der FMU und zum Parsen der XML-Datei. Des Weiteren werden hier alle benötigten Informationen (Anzahl der Zustände, Ausgänge, Eingänge und Identifikationsparameter) ausgelesen und die zugehörigen Referenzen, die als Value-References bezeichnet werden, gespeichert. Die für die MATLAB-Identifikationsumgebung benötigten Simulationsergebnisse werden durch das Simulationsskript bereitgestellt. Hierzu werden zunächst die Werte für die Startzeit, die Zustände, die Eingänge sowie die zu identifizierenden Parameter im Simulationsskript mithilfe der zuvor gespeicherten Informationen automatisch gesetzt. Anschließend erfolgt die Simulation und die entsprechenden Ergebnisse werden zurückgegeben. Für den Schritt der Simulation ist es unabhängig, ob die FMU als Co-Simulation (mit Solver) oder als Model Exchange (ohne Solver) exportiert vorliegt, da in das FMU-Interface ein Solver integriert wurde. Nachfolgend wird auf die einzelnen Skripte genauer eingegangen.

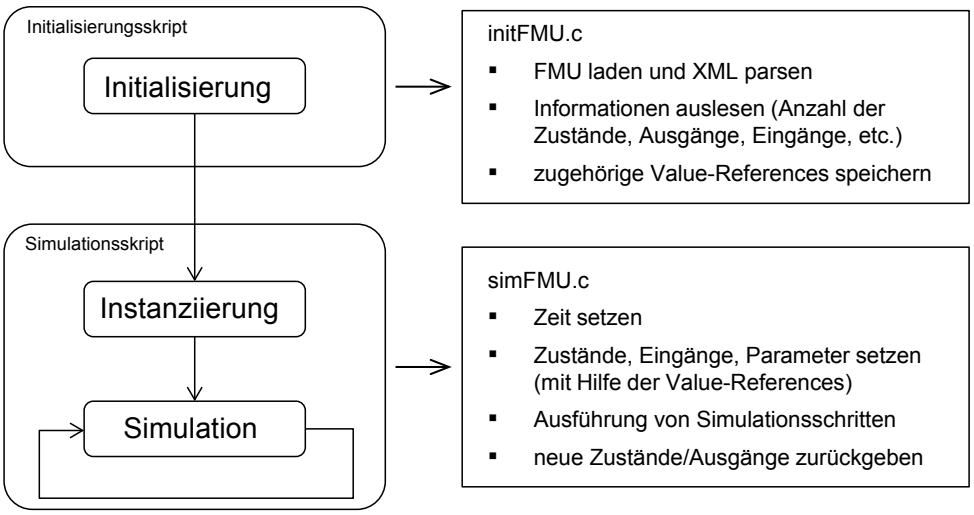

Bild 4-3: Schematische Darstellung des Initialisierungs- und des Simulationsskriptes des FMU-Interfaces

Bild 4-4 zeigt eine detailliertere Darstellung des Vorgehens während der Initialisierung des FMU-Interfaces. Zunächst wird mithilfe der XML-Datei, die alle wichtigen Informationen des Modells beinhaltet (vgl. Abschnitt 2.4.3), eine modellspezifische Klasse *modelDescription* erstellt (Schritt 1). Dies ermöglicht die

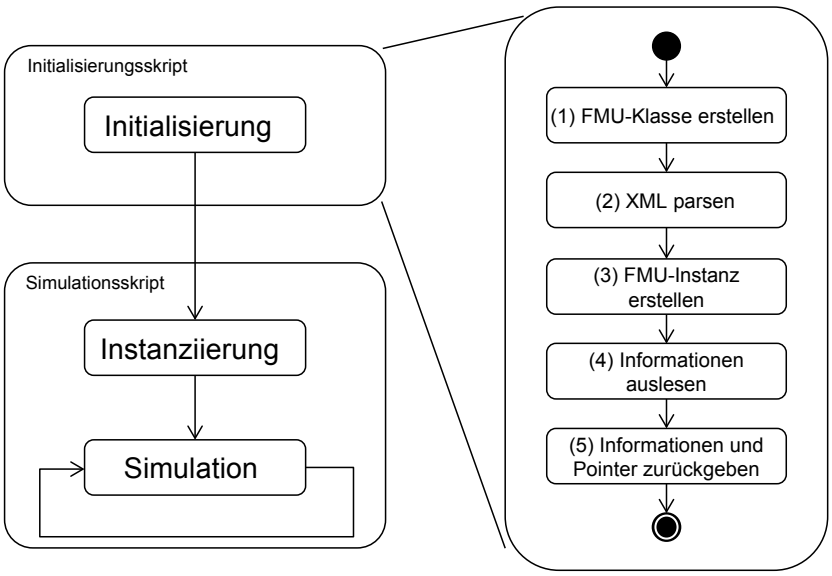

Bild 4-4: Vorgehensmodell für das Initialisierungsskript des FMU-Interfaces

später benötigte Erzeugung von Modell-Instanzen. Anschließend werden erste Daten aus der XML-Datei in die Klasse eingebunden (Schritt 2). Die hierzu benötigten Funktionen werden durch das SDK von Qtronic zur Verfügung gestellt. In Schritt 3 wird eine Instanz der Klasse *modelDescription* angelegt, durch die der Zugriff auf die Anfangswerte erfolgen kann. Für das Auslesen der benötigten Informationen (Schritt 4) sind die Funktionen des SDKs erweitert worden, um den besonderen Anforderungen an das Identifikations-Tool gerecht zu werden. So werden an dieser Stelle die während der Modellbildung markierten Identifikationsparameter sowie Eingänge, Zustände und Ausgänge des Modells ausgelesen. Hierzu werden zunächst alle in der XML-Datei enthaltenen *ScalarVariables* durchlaufen. Über die FMU-spezifischen Attribute *variability*, *description* und *causality* können die Identifikationsparameter sowie die Ein- und Ausgänge ermittelt werden. Quellcode 4.1 zeigt einen exemplarischen Auszug aus dem XML-Quellcode des einfachen Anwendungsbeispiels, die in Abschnitt 4.2.1 vorgestellt wird.

Quellcode 4.1: XML-Beispiel

```
...

<ScalarVariable
  name="Theta"
  valueReference="16777228"
  description="Identify;Value=15;LowerBound=5;UpperBound=100;Unit[kg.m^2];"
  variability="parameter">

...

<ScalarVariable
  name="z"
  valueReference="335544320"
  description="Absolute position vector frame_a.r_0 resolved in frame
  defined by resolveInFrame"
  causality="output">

...
```

Die benötigten Werte werden über die Funktionen *getVariability* und *getCausality* ermittelt. Durch die jeweilige Value-Reference (*getValueReference*) ist eine eindeutige Zuordnung möglich. Der Zugriff erfolgt mittels der Funktionen *getReal* und *setReal*. Für das Erfassen und das Setzen der Zustände werden die Funktionen *getContinuousStates* und *setContinuousStates* eingesetzt.

In Schritt 5 werden die zuvor ausgelesenen Informationen aus Schritt 4 gespeichert, da sie in dem Simulationsskript benötigt werden. Hierzu wird zunächst ein Speicherplatz unter MATLAB reserviert. Anschließend werden die Informationen in einen MATLAB-Datentyp konvertiert (C→MATLAB) und in den reservierten

Speicherplatz geschrieben (vgl. I in Bild 4-5). Das Simulationsskript des FMU-Interfaces kann somit die benötigten Informationen direkt aus dem angelegten Speicherbereich beziehen (vgl. II in Bild 4-5). Deshalb müssen das rechenaufwändige Laden und Parsen der XML nur einmalig ausgeführt werden, wodurch eine erhebliche Zeitersparnis gewährleistet ist.

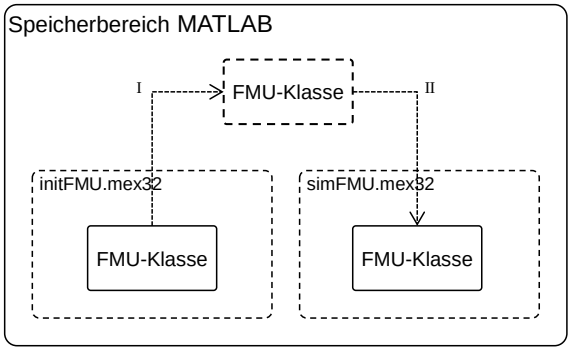

Bild 4-5: Schematische Darstellung der FMU-Informationen zwischen dem Initialisierungsskript und dem Simulationsskript des FMU-Interfaces

Nachdem die FMU geladen, das Parsen der XML abgeschlossen, die benötigten Informationen ausgelesen und die jeweiligen Value-References gespeichert wurden, kann die eigentliche Simulation erfolgen. Bild 4-6 zeigt das schematische Vorgehensmodell für das Simulationsskript des FMU-Interfaces.

Zu Beginn (Schritt 1) werden die von MATLAB übergebenen Parameter in ein C-Dateiformat konvertiert (MATLAB→C). In Schritt 2 wird der Pointer[1] verarbeitet, indem eine Kopie der FMU-Klasse erstellt wird (vgl. II in Bild 4-5). Um einen Zugriff zu ermöglichen, wird in Schritt 3 wiederum eine FMU-Instanz erstellt, die in Schritt 4 initialisiert wird. In Schritt 5 erfolgt die eigentliche Simulation über ein Zeitintervall der Länge t_{Int} mit einer Simulationsschrittweite $t_{StepSize}$[2]. Nach dem Simulationsintervall werden in Schritt 6 die Simulationsergebnisse an die MATLAB-Identifikationsumgebung zurückgegeben. Dieser Vorgang wird wiederholt, bis ein vorgegebenes Simulationsende t_{End} erreicht ist (vgl. Bild 4-7), wobei die Schritte 2–4 lediglich einmalig zu Beginn ausgeführt werden müssen. Nachfolgend wird auf die Ausführung eines Simulationsintervalls genauer eingegangen.

Das FMU-Interface ermöglicht die Einbindung einer FMU als Co-Simulation oder als Model Exchange. Daher gibt es zwei unterschiedliche Vorgehensweisen bei der

[1] Der Pointer bezieht sich auf die Adresse des angelegten Speichers.

[2] Der Wert der Schrittweite wird nur dann benötigt, wenn die FMU als Model Exchange mit einem Fixed-Step-Solver ausgeführt wird.

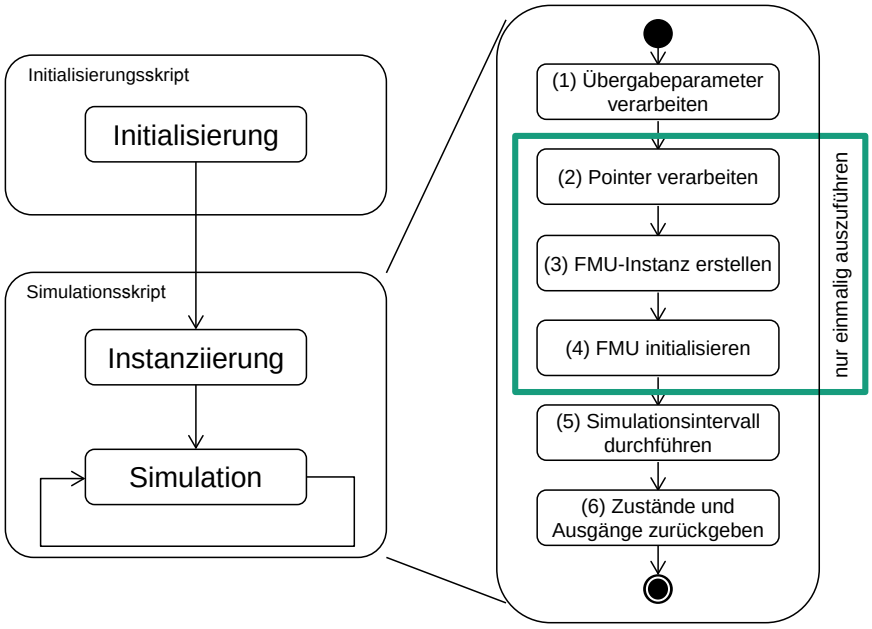

Bild 4-6: Vorgehensmodell für das Simulationsskript des FMU-Interfaces

Durchführung eines Simulationsintervalls: Beim Model Exchange erfolgt die numerische Integration derzeit mittels eines Runge-Kutta-Verfahrens 4. Ordnung, bei der die zu verwendende Simulationsschrittweite fest durch den Parameter $t_{StepSize}$ vorgegeben wird. Durch die Gleichung 4-1 wird ein Simulationsschritt realisiert, und man erhält aus Gleichung 4-2 den Zustandsvektor $\vec{X}_{k+1}$ zum nächsten Simulationsschritt. Gegeben sind hierbei die Schrittweite $t_{StepSize}$, der Zeitpunkt t_k sowie die Vektoren der Eingänge $\vec{U}_k$ und der Zustände $\vec{X}_k$. Das FMU-Interface ermöglicht eine beliebige Erweiterung von Integrationsverfahren unter C.

$$
\begin{aligned}
 & & \dot{\vec{X}}_k &= f(\vec{X}_k, \vec{U}_k, t_k) \\
\vec{X}_A &= \vec{X}_k + \frac{t_{StepSize}}{2} \cdot \dot{\vec{X}}_k & \dot{\vec{X}}_A &= f(\vec{X}_A, \vec{U}_k, t_k + \frac{t_{StepSize}}{2}) \\
\vec{X}_B &= \vec{X}_A + \frac{t_{StepSize}}{2} \cdot \dot{\vec{X}}_A & \dot{\vec{X}}_B &= f(\vec{X}_B, \vec{U}_k, t_k + \frac{t_{StepSize}}{2}) \\
\vec{X}_C &= \vec{X}_B + \frac{t_{StepSize}}{2} \cdot \dot{\vec{X}}_B & \dot{\vec{X}}_C &= f(\vec{X}_C, \vec{U}_k, t_k + t_{StepSize})
\end{aligned}
\tag{4-1}
$$

$$
\vec{X}_{k+1} = \vec{X}_k + \frac{t_{StepSize}}{6} \cdot (\dot{\vec{X}}_k + 2\dot{\vec{X}}_A + 2\dot{\vec{X}}_B + \dot{\vec{X}}_C) \tag{4-2}
$$

Bei der Einbindung einer FMU als Co-Simulation entfällt die Vorgabe der Simulationsschrittweite und des anzuwendenden Integrationsverfahrens, da diese bereits in der FMU integriert sind (vgl. Abschnitt 2.4.3). Es erfolgt lediglich eine Vorgabe über die Länge des zu simulierenden Zeitintervalls t_{Int}, das – wie bei dem Model Exchange – durch die übergebenen Parameter $t_{Int,Start}$ und $t_{Int,End}$ festgelegt wird. Bei der Verwendung von FMUs als Co-Simulation aus Dymola ist derzeit der *Sundials*-Solver CVODE eingebunden [Das13]. Dieser stellt im Gegensatz zum Runge-Kutta-Verfahren einen Variable-Step-Solver dar. Unabhängig vom Integrationsverfahren ist die Länge des auszuführenden Simulationsintervalls t_{Int} grundsätzlich frei wählbar. Bei der Verwendung eines Fixed-Step-Solvers muss jedoch gelten: $t_{StepSize} \leq t_{Int}$. Je größer t_{Int} gewählt wird, desto schneller ist das Simulationsende t_{End} erreicht, da der Kommunikationsaufwand zwischen der MATLAB-Identifikationsumgebung und dem FMU-Interface verringert wird. Dies hat jedoch Einfluss auf die übergebenen Simulationsergebnisse, was in Bild 4-7 dargestellt ist.

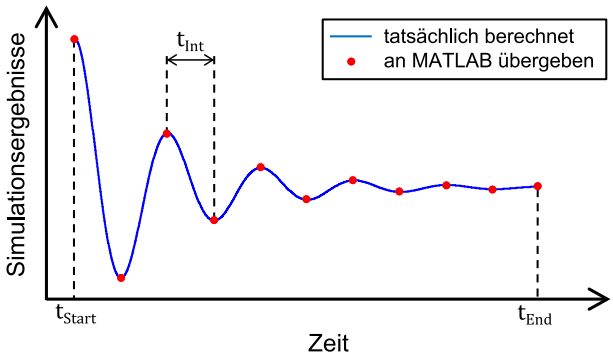

Bild 4-7: Einfluss von t_{Int} auf die in MATLAB vorliegenden Simulationsergebnisse (rot) im Vergleich zu den tatsächlich berechneten Ergebnissen (blau)

Damit das entwickelte Interface für weitere Anwendungen offen bleibt, ist ein solcher Datenaustausch nötig, etwa für die Auslegung von echtzeitfähigen Zustands- und Parameterschätzern an Black-Box-Systemen mittels Sigma-Punkte-Kalman-Filtern, was unter [Sch17] vorgestellt wurde. Je nach Anwendungsfall für die Parameteridentifikations- und Modellvalidierungsmethodik muss daher ein Kompromiss zwischen Schnelligkeit und Genauigkeit für die Simulation bzw. die Simulationsergebnisse getroffen werden.

Abschließend sind die Werte, die zwischen MATLAB und dem Initialisierungsskript sowie dem Simulationsskript ausgetauscht werden, in Tabelle 4-1 dargestellt. Bei den Value-References handelt es sich um eine Art Zeiger, wodurch die Referenzen auf die Daten übergeben werden. Einen genaueren Einblick in die Vor-

gehensweise bei der Parameteridentifikation mittels des entwickelten Tools zeigt der nachfolgende Abschnitt.

Tabelle 4-1: Zwischen MATLAB und dem FMU-Interface ausgetauschte Werte

Parameter	Bezeichnung
$fmu\_location$	Dateipfad der FMU
$fmu\_pointer$	Adresse des angelegten Speichers
Lu	Anzahl der Eingänge
$vrefInput$	Value-References der Eingänge
$namesInput$	Namen der Eingänge
Ly	Anzahl der Ausgänge
$vrefOutput$	Value-References der Ausgänge
$startValueOutput$	Startwerte der Ausgänge
$namesOutput$	Namen der Ausgänge
L	Anzahl der Zustände
$vrefStates$	Value-References der Zustände
$startValueStates$	Startwerte der Zustände
$namesStates$	Namen der Zustände
Lp	Anzahl der Identifikationsparameter
$vrefParam$	Value-References der Identifikationsparameter
$startValueParam$	Startwerte der Identifikationsparameter
$lowerBound$	Untere Grenzen der Identifikationsparameter
$upperBound$	Obere Grenzen der Identifikationsparameter
$unit$	Einheit der Identifikationsparameter
$namesParam$	Namen der Identifikationsparameter
$t_{Int,Start}$	Startzeit des auszuführenden Simulationsintervalls
$t_{Int,End}$	Endzeit des auszuführenden Simulationsintervalls
$t_{StepSize}$	Simulationsschrittweite
$\vec{U}_{Ref,k}$	Übergebene Eingangswerte
$\vec{X}_k$	Übergebene Werte der Zustände
$\vec{P}_k$	Übergebene Werte der Identifikationsparameter

4.2 Validierungsmethodik bei einem einfachen Anwendungsbeispiel

Um die Funktionalität des entwickelten Parameteridentifikations-Tools zu veranschaulichen, wird es an einem einfachen Beispiel vorgestellt. Bevor auf die Parameteridentifikations- und Modellvalidierungsmethodik für das Anwendungsbeispiel eingegangen wird, wird dieses zunächst vorgestellt.

4.2.1 Einfaches Anwendungsbeispiel

Bei dem einfachen Anwendungsbeispiel handelt es sich um ein idealisiert aufgebautes Segway-Modell unter Dymola (vgl. Bild 4-8). Das Modell hat keinen Anspruch, das dynamische Verhalten eines realen Segways detailliert nachzubilden. Es dient vielmehr der Veranschaulichung der Funktionsweise des erstellten Parameteridentifikations-Tools.

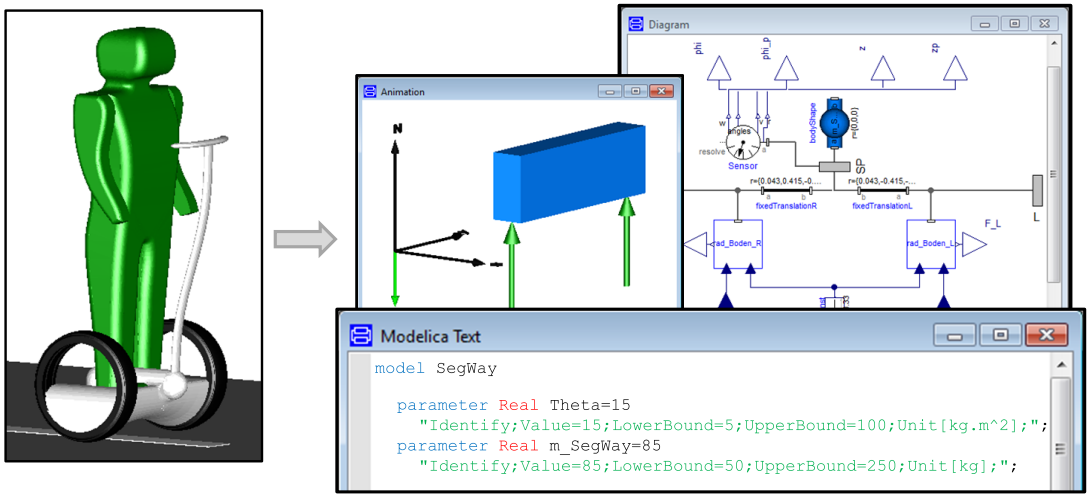

Bild 4-8: Einfaches Anwendungsbeispiel: links das Referenzsystem und rechts das zu identifizierende Modell

Das stark vereinfachte Segway-Modell bildet die vertikale Einfederung ab. Somit besitzt das Modell zwei Freiheitsgrade (Huben und Wanken), die über die Position z und den Winkel φ erfasst werden können. Das System besteht aus einem Starrkörper mit der Masse m_{Segway} und einem Trägheitsmoment Θ_{Segway} sowie zwei Koppelpunkten für die wirkenden Rad-Boden-Kräfte. Für das idealisierte Abbilden dieser Kräfte wird ein Kelvin-Voigt-Modell, bestehend aus der Parallelschaltung einer Feder und eines Dämpfers, verwendet. Die somit jeweils resultierende Kraft F_{Rad} ergibt sich aus der Federsteifigkeit c_{Rad} und der Dämpfung d_{Rad} des Rades:

$$F_{Rad}=c_{Rad}\cdot\Delta z_R+d_{Rad}\cdot\Delta\dot{z}_R \tag{4-3}$$

mit $\Delta z_R = z_R - r_R - z_{St}$ und $\Delta\dot{z}_R = \dot{z}_R - \dot{z}_{St}$.

z_R ist die aktuelle Position des Radmittelpunktes in vertikaler Richtung, r_R ist der Radius des Rades, und z_{St} ist die Position des Untergrundes.

Durch diese Kraftbedingung kann sich das Segway-Modell in der Umgebung bewegen. Es muss jedoch noch berücksichtigt werden, dass die Rad-Boden-Kräfte F_{Rad} nur bei Druckkräften wirken. Bei Zugkräften soll keine Kraft zwischen dem Boden und dem Rad wirken. Bild 4-9 zeigt eine vereinfachte Darstellung der wirkenden Rad-Boden-Kraft für unterschiedliche Situationen.

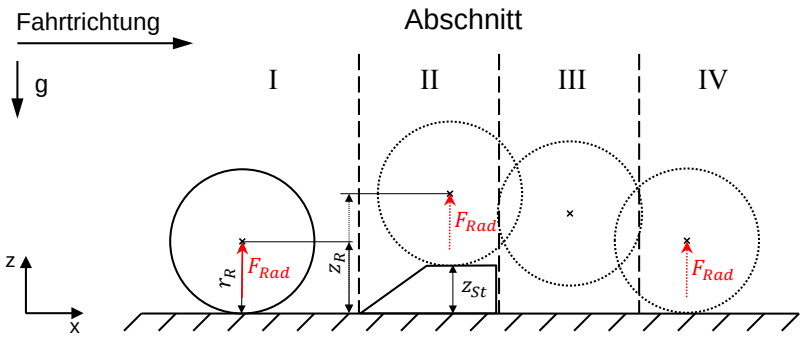

Bild 4-9: Vereinfachte Darstellung einer wirkenden Rad-Boden-Kraft in unterschiedlichen Situationen

In Abschnitt I gilt: $z_R \leq r_R$ mit $z_{St} = 0$. In Abschnitt II ist $z_{St} \neq 0$. In beiden Fällen gilt: $z_R \leq r_R + z_{St}$. Verlässt das Rad das Hindernis, so kann es vom Boden abheben (Abschnitt III), ohne dass eine Rad-Boden-Kraft wirkt. Landet das Rad (Abschnitt IV), so wirkt diese Kraft erneut. Durch folgende Fallunterscheidung lässt sich diese Bedingung realisieren:

$$F_{Rad} = \begin{cases} 0, & \text{für } z_R > r_R + z_{St}, \\ F_{Rad}, & \text{für } z_R \leq r_R + z_{St}. \end{cases}$$

Um die Funktionalität des Parameteridentifikations-Tools zu demonstrieren, werden bei dem Segway-Modell die Parameter für die Masse m_{Segway} sowie für die Trägheit Θ_{Segway} identifiziert. Hierzu werden die Parameter im Dymola-Modell von dem Modellierer entsprechend mit

Identify;Value;LowerBound;UpperBound;Unit;

markiert (vgl. Bild 4-8). Als Referenzwerte dienen simulierte Sensorwerte der Position z, der Geschwindigkeit $\dot{z}$, des Winkels φ und der Winkelgeschwindigkeit $\dot{\varphi}$ eines Referenz-Segway-Modells. Bei diesem sind die Parameter der Masse m_{Segway}

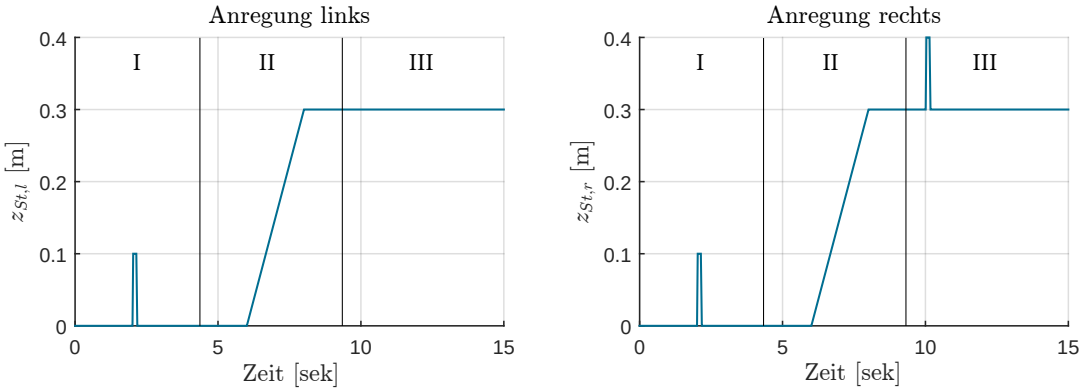

Bild 4-10: Anregungsgrößen z_{St} des Anwendungsbeispiels für das linke und das rechte Rad

und der Trägheit Θ_{Segway} gegenüber dem zu identifizierenden Segway-Modell variiert worden. Angeregt werden beide Modelle durch die Eingänge $z_{St,r}$ und $z_{St,l}$ (vgl. Bild 4-10).

Bei dem betrachteten Simulationsszenario handelt es sich um drei aufeinanderfolgende Hindernisse:

I. paralleles Hindernis,

II. Rampe,

III. einseitiges Hindernis.

Um dieses Szenario mit einem Modell mit nur zwei Freiheitsgraden realisierbar zu machen, werden die Hindernisse unter dem Modell hindurch bewegt, so dass das Segway-Modell keinen zusätzlichen translatorischen Freiheitsgrad benötigt.

Bild 4-11 zeigt die Abfolge des Messszenarios für das Anwendungsbeispiel. Beim ersten Bild (oben links) befindet sich das Segway in der Ausgangssituation, bei Bild 2 (oben rechts) kurz vor den parallelen Hindernissen, im Bild 3 kurz nach der Rampenauffahrt, und in Bild 4 befindet sich das Segway bei der Anregung durch das einseitige Hindernis. Die dabei aufgezeichneten Sensorgrößen sind ebenfalls in Bild 4-11 dargestellt. Diese dienen im nächsten Schritt als Referenzgrößen, um das Modell mit den variierten Parametern der Masse m_{Segway} und der Trägheit Θ_{Segway} zu identifizieren und somit zu validieren. Die auftretenden vertikalen Schwingungen während der Rampenauffahrt im Bereich II sind auf die vereinfachte Radmodellierung zurückzuführen. Bild 4-12 zeigt eine schematische Darstellung der MiL-Simulationen des Referenzmodells und des zu identifizierenden Modells sowie die Einbindung des letzteren als FMU in das Parameteridentifikations-Tool. Als Rückgabewerte erhält das Segway-Modell die identifizierten Parameter der Masse m_{Segway} und der Trägheit Θ_{Segway}.

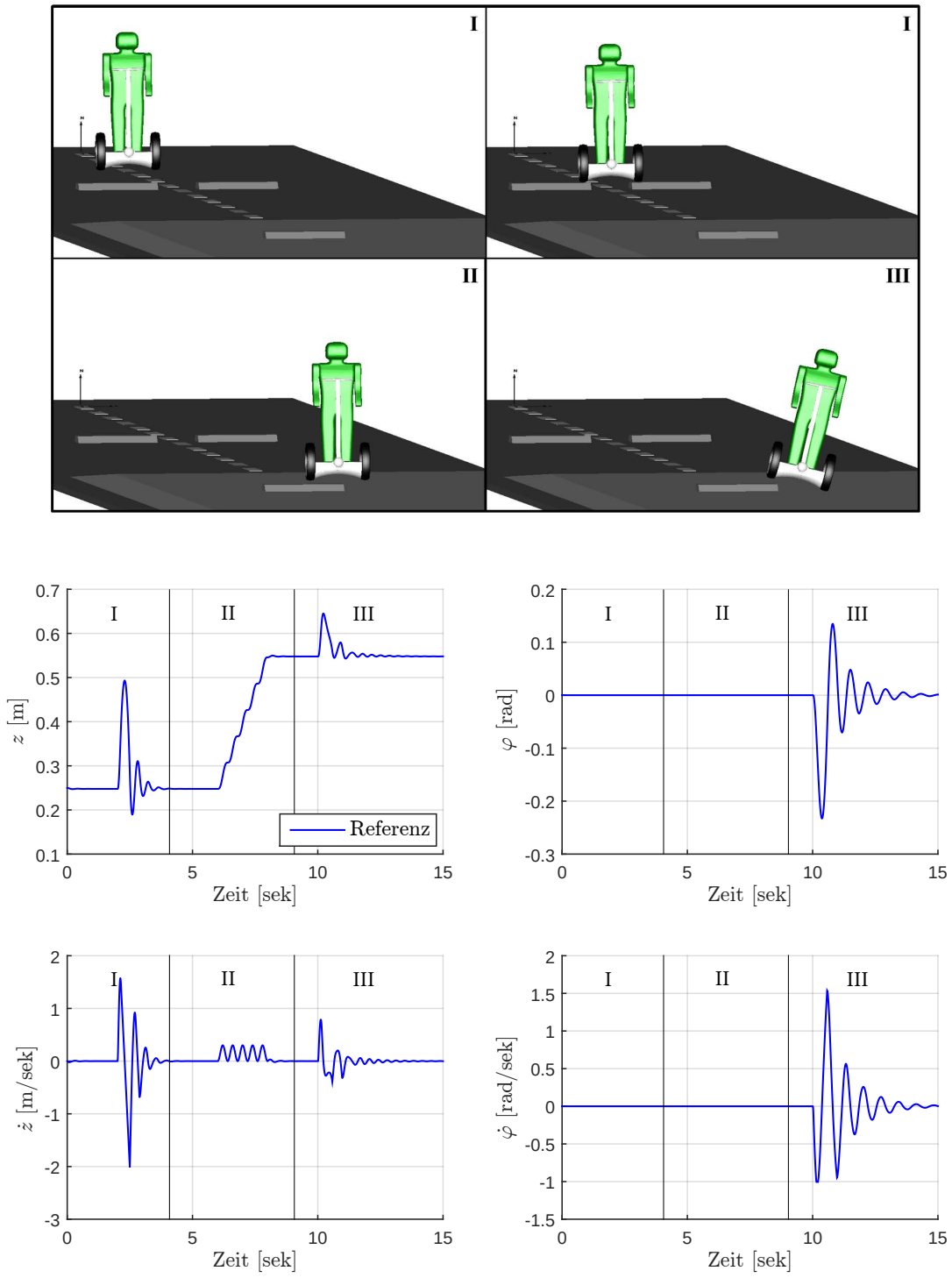

Bild 4-11: Messszenario des einfachen Anwendungsbeispiels

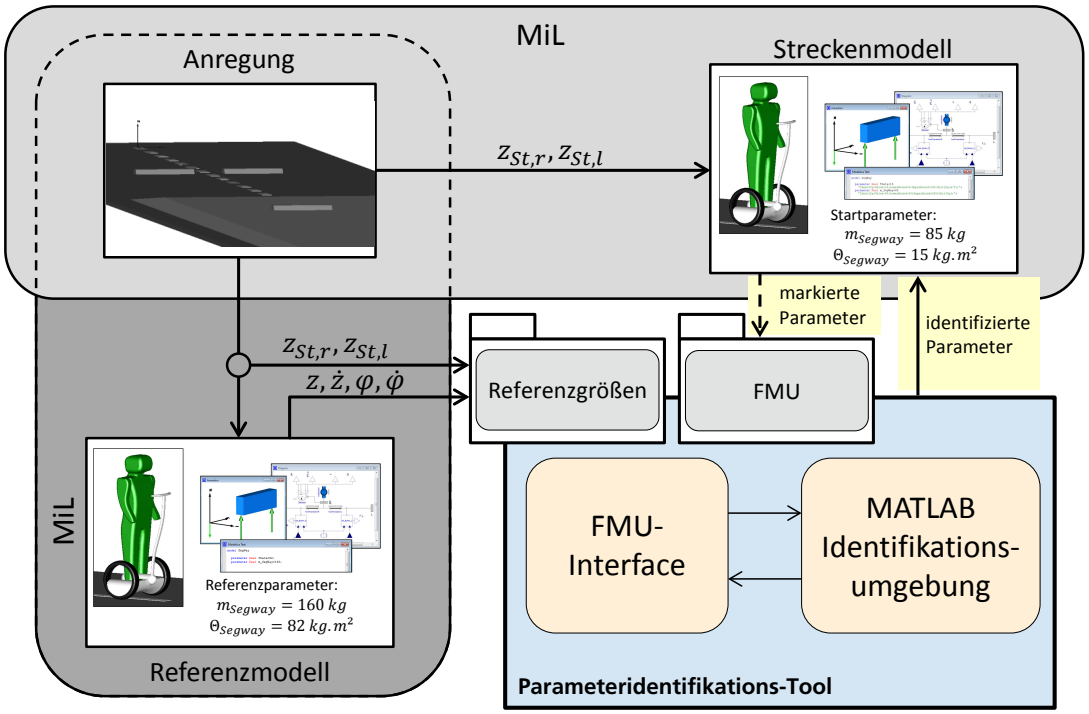

Bild 4-12: Schematische Darstellung der MiL-Simulationen des Referenzmodells und des zu identifizierenden Modells sowie die Einbindung des letzteren als FMU in das Parameteridentifikations-Tool

4.2.2 Modellvalidierung

Bild 4-13 zeigt das Vorgehen für den Bereich der Modellvalidierung inklusive einer schematischen Darstellung der enthaltenen Modellinformationen und der Referenzgrößen. Die einzelnen Schritte werden nachfolgend genauer beschrieben.

FMU einladen

Bei den unter Dymola zu wählenden Einstellungen ist der FMU-Export-Typ *All* ausgewählt worden. Hierdurch kann die erzeugte FMU sowohl für die Einbindung als Co-Simulation als auch als Model Exchange verwendet werden. Zu Beginn der Modellvalidierung wird der Pfad der FMU geladen. An dieser Stelle wird festgelegt, ob die FMU als Co-Simulation oder als Model Exchange geladen werden soll. Ist der Pfad ausgewählt, erfolgt automatisch eine Ausführung der *initFMU.mex32*-Datei. Hierdurch können die Informationen (Anzahl, Namen und Werte der Zustände sowie der Identifikationsparameter), die in der FMU gespei-

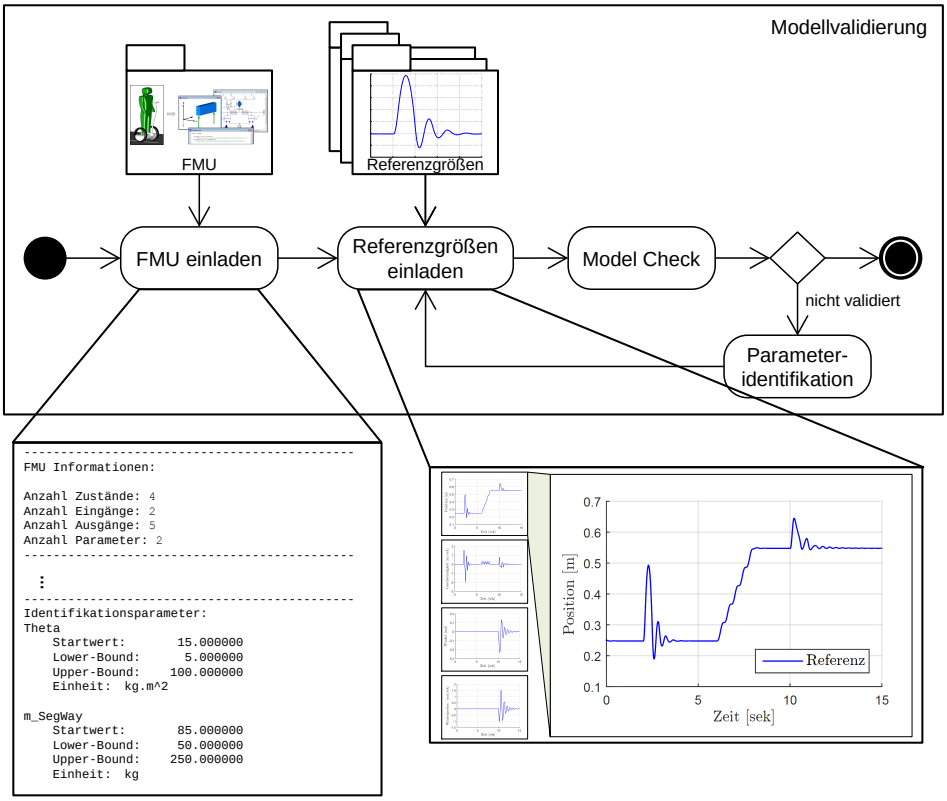

Bild 4-13: Vorgehensmodell für den Bereich der Modellvalidierung des MATLAB-Identifikationsbereiches inklusive einer schematischen Darstellung der enthaltenen Modellinformationen und der Referenzgrößen

chert sind, angezeigt werden. Einen Ausschnitt über die erhaltenen Informationen des Segway-Modells zeigt Bild 4-13.

Wie zuvor beschrieben, besitzt das Segway-Modell zwei Freiheitsgrade und somit vier Zustände. Als Eingänge sind die Radanregungen für die linke und die rechte Seite modelliert. Bei den Ausgängen handelt es sich um die vier zuvor vorgestellten Sensorwerte Position z, Geschwindigkeit $\dot{z}$, Winkel φ und Winkelgeschwindigkeit $\dot{\varphi}$. Neu hinzugekommen ist ein Ausgang der aktuellen Simulationszeit. Hierzu ist eine Uhr in das Modell hinzugefügt worden, deren Ausgangswert in Sekunden als fünfter Ausgang der FMU verwendet wird. Dieser Wert dient der Kontrolle, ob die von extern vorgegebene Simulationszeit $t_{Int,Start}$ in der FMU korrekt verarbeitet wird. Als weitere FMU-Informationen werden die Anzahl, die Werte, die oberen und die unteren Grenzen sowie die jeweilige Einheit der im Modell markierten Identifikationsparameter ausgegeben.

Referenzgrößen einladen

Im nächsten Schritt werden die Referenzgrößen geladen und den jeweiligen Ein- bzw. Ausgängen der FMU zugeordnet. Die Referenzgrößen müssen zuvor als Datentyp Array in einer **.mat*-Datei abgespeichert sein. Neben den Werten der Referenzgrößen muss der Zeitvektor in der **.mat*-Datei vorhanden sein. Ein separater Bereich der GUI ermöglicht das Anzeigen aller FMU Ein- und Ausgänge, das Laden der entsprechenden *Referenzgrößen.mat*-Datei sowie optional das Plotten der Referenzgrößen (vgl. Bild 4-14). Ein- und Ausgänge der FMU sowie die geladenen Referenzgrößen werden in der GUI einander gegenübergestellt. Das Mapping erfolgt teilautomatisiert, lediglich die korrekte Reihenfolge der zu verbindenden Größen muss gesichert sein. Hierzu kann ein entsprechendes Referenzsignal markiert und nach oben oder unten verschoben werden. Sollte die FMU Schnittstellen besitzen, zu denen keine Referenzgrößen vorliegen (gleiches gilt auch umgekehrt), so können sie aus der Auflistung gelöscht werden, so dass letztlich nur die zu verbindenden Signale in der richtigen Reihenfolge gegenübergestellt werden.

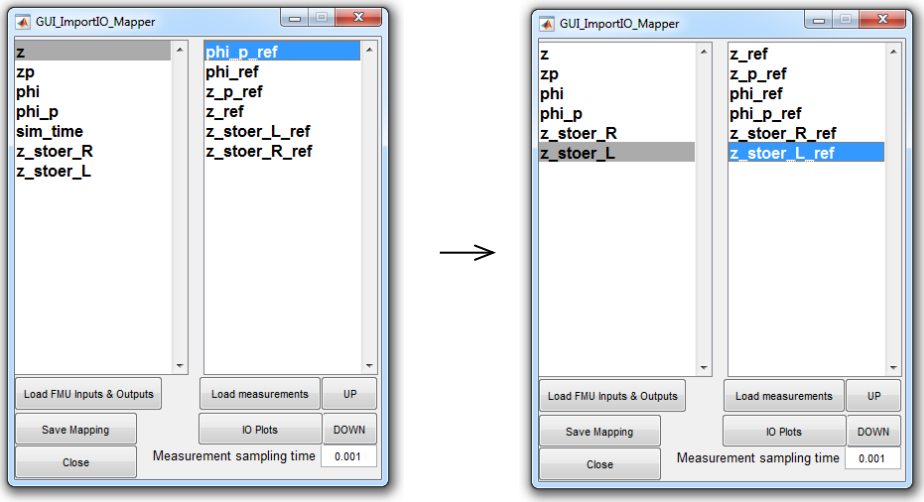

Bild 4-14: Screenshot der Mapping-GUI vor (links) und nach (rechts) Anordnung der Referenzsignale in der richtigen Reihenfolge

Bild 4-14 zeigt den Mapping-Bereich der erstellten GUI vor und nach der Anordnung der Signale in die richtige Reihenfolge. Die angezeigte Abtastzeit dient als zusätzliche Information und wird automatisiert den Referenzgrößen entnommen, sie muss nicht manuell eingegeben werden.

Model Check

Ist das Mapping erfolgt, so wird die FMU im nächsten Schritt simuliert. Hierzu werden die Parameter der Simulationszeit t_{End} (t_{Start} ist beim `Model Check` Null) und der Intervalllänge t_{Int} sowie, falls es sich um die Einbindung einer FMU als Model Exchange handelt, der Solver-Schrittweite $t_{StepSize}$ gesetzt. Mittels einer *for*-Schleife führt die *simFMU.mex32*-Datei Simulationsintervalle der Länge t_{Int} durch, bis die gewählte Simulations-Endzeit t_{End} erreicht ist. In einem letzten Schritt der Modellvalidierung können die Simulationswerte mit den Referenzgrößen verglichen werden.

Bild 4-15 zeigt den Vergleich zwischen den Simulations- und den Referenzgrößen für das Segway-Modell nach dem `Model Check`. Wie erwartet, führen die zuvor geänderten Werte der Masse m_{Segway} und der Trägheit Θ_{Segway} zu deutlichen Abweichungen in den Ergebnissen. Anhand dieses Vergleiches kann entschieden

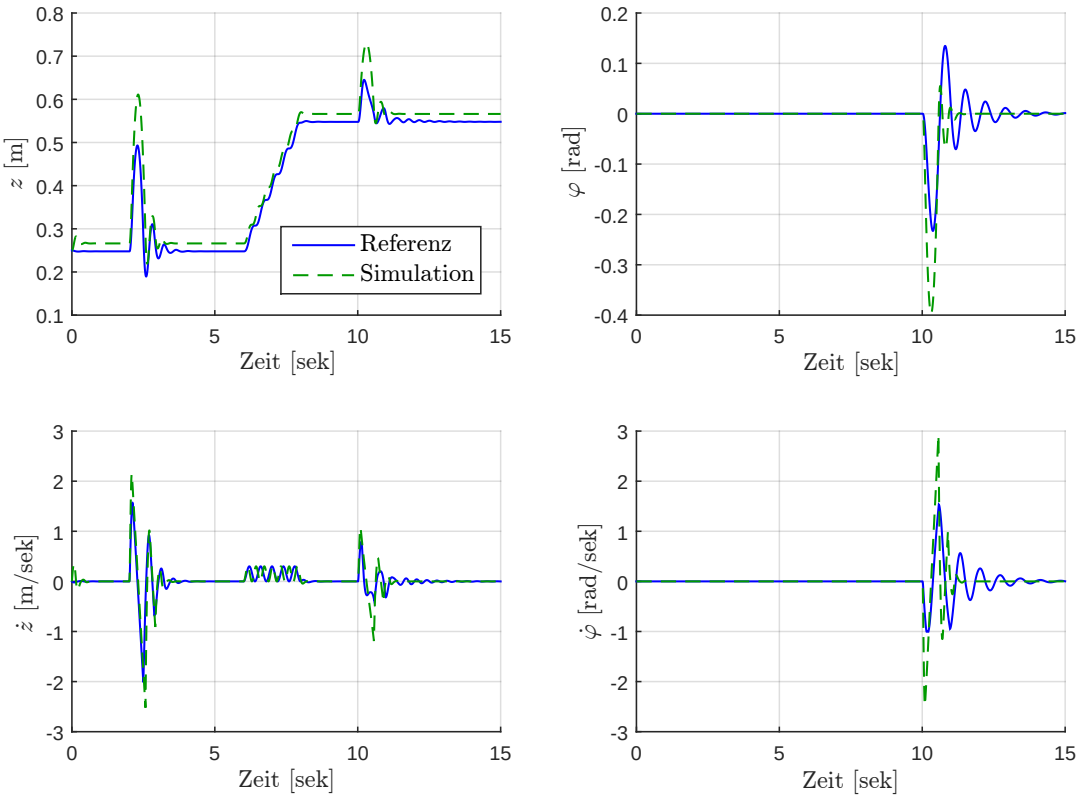

Bild 4-15: Vergleich zwischen den Referenzgrößen und der nicht validierten Simulation

werden, ob das Modell das gewünschte Verhalten hinreichend genau abbildet. Ist dies nicht der Fall, so folgt der Bereich der Parameteridentifikation.

Parameteridentifikation

Um die Abweichungen zwischen Simulation und Referenz zu minimieren, wird der Bereich der Parameteridentifikation verwendet. Bild 4-16 zeigt schematisch das Vorgehen für diesen Bereich. Er gliedert sich, wie auch der Bereich der Modellvalidierung, in mehrere Einzelschritte, die iterativ bearbeitet werden können. Nachfolgend werden diese anhand des Segway-Beispiels erläutert. Um nachvollziehbar zu machen, um welche Iteration es sich handelt, werden die gezeigten Schritte entsprechend markiert (I. II.).

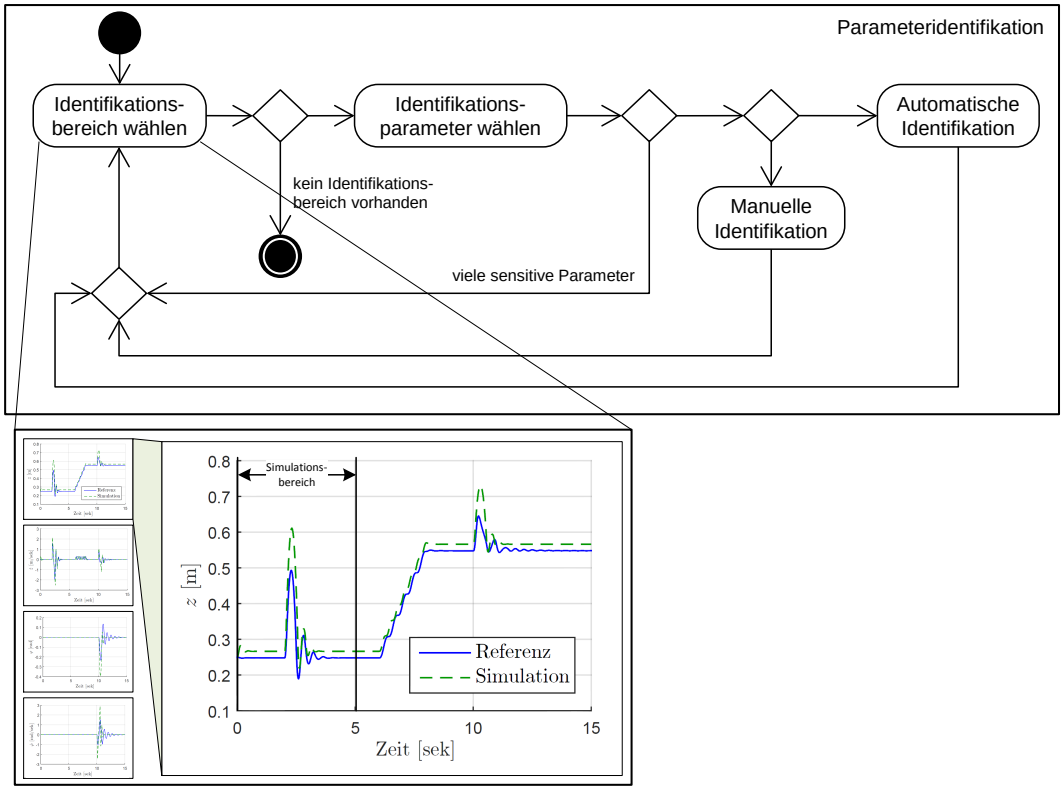

Bild 4-16: Vorgehensmodell für die Parameteridentifikation des MATLAB-Identifikationsbereiches inklusive einer schematischen Darstellung des ausgewählten Identifikationsbereiches

I. Identifikationsbereich wählen

Der erste Schritt bei der Parameteridentifikation befasst sich mit der Auswahl des zu identifizierenden Bereiches. Hierzu können zunächst die Ausgänge ausge-

wählt werden, die identifiziert werden sollen. Es können auch mehrere gleichzeitig gewählt werden. Der Simulationsbereich wird mittels der Größen t_{Start} und t_{End} festgelegt. Bild 4-16 zeigt schematisch die ausgewählte Referenzgröße der Position z mit dem Simulationsbereich von t_{Start}= 0 sek und t_{End}= 5 sek.

Neben der Auswahl des Simulationszeitraumes kann ein separater Zeitraum für die Identifikation eingestellt werden. Bei diesem handelt es sich um einen Bereich innerhalb des Simulationszeitraumes, der bei der Verwendung eines Optimierungsverfahrens (vgl. Abschnitt 4.2.2) oder der Sensitivitätsanalyse (vgl. Abschnitt 4.2.2) für die Berechnung der Zielfunktion eingesetzt wird. Der Identifikationszeitraum ist zu diesem Zeitpunkt als Rechteck-Fensterfunktion hinterlegt. Wird kein separater Identifikationsbereich angegeben, so wird für die Berechnung der Zielfunktion automatisch der gesamte Simulationsbereich gewählt.

I. Identifikationsparameter wählen

Ist der Identifikationsbereich ausgewählt, findet im zweiten Schritt die Auswahl der zu identifizierenden Parameter statt. Hierbei werden zunächst alle im Modell als Identifikationsparameter markierten Parameter aufgelistet. Sollte kein Wissen darüber vorliegen, welche Parameter für den ausgewählten Identifikationsbereich sensitiv sind, so kann eine Sensitivitätsanalyse angewendet werden, die sich wie folgt berechnet:

$$RS = \frac{f(P_i, \Delta P_i) - f(P_i)}{f(P_i)}. \tag{4-4}$$

Der Rückgabewert ist die Relative Sensitivität (RS). Sie gibt Auskunft darüber, inwieweit ein ausgewählter Identifikationsparameter P_i Einfluss auf die Zielfunktion $f(P_i)$ besitzt. Hierzu wird dieser um eine vorgegebene Differenz ΔP_i variiert und die Zielfunktion $f(P_i, \Delta P_i)$ neu gebildet. Je größer die relative Sensitivität, desto mehr Einfluss hat ein Parameter auf die Zielfunktion. Das verwendete ΔP_i ist frei wählbar und beträgt derzeit 15 % des Wertes des jeweiligen Identifikationsparameters. Bei der automatisch hinterlegten Zielfunktion handelt es sich um die quadratische Fehlerfläche zwischen den Simulations- und den Referenzgrößen innerhalb des ausgewählten Identifikationsbereiches. Diese kann beliebig geändert werden.

Bei dem Vergleich von sensitiven Parametern miteinander sei darauf hingewiesen, dass der jeweilige Wert der relativen Sensitivität maßgeblich vom Startwert des entsprechenden Parameters abhängt. Ist dieser ungünstig gewählt, wird ein vergleichsweise geringer Wert für die relative Sensitivität berechnet, obwohl der Parameter einen sehr großen Einfluss auf die Zielfunktion besitzen kann. Bei der

derzeitigen Betrachtung der relativen Sensitivität ist daher entscheidend, dass ein Parameter grundsätzlich Einfluss auf die Zielfunktion besitzt, also dass $RS > 0$ ist.

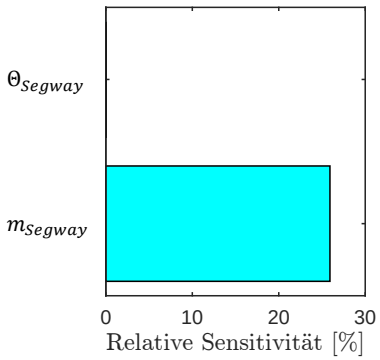

Bild 4-17: Relative Sensitivität der beiden Identifikationsparameter für den ersten Identifikationsbereich

Bild 4-17 zeigt die relative Sensitivität der beiden Identifikationsparameter der Masse m_{Segway} und der Trägheit Θ_{Segway} für den unter Bild 4-16 ausgewählten Identifikationsbereich. Da das Segway-Modell in diesem Bereich parallel angeregt wird, hat der Parameter der Trägheit Θ_{Segway} keinen Einfluss auf die Zielfunktion und wird für die weitere Identifikation nicht ausgewählt:

I. Identifikation starten

Sind die Identifikationsbereiche und die zu identifizierenden Parameter ausgewählt, findet die eigentliche Identifikation statt. Hierbei kann zwischen zwei Vorgehensweisen gewählt werden.

Manuelle Identifikation

Bei der ersten Vorgehensweise handelt es sich um eine manuelle Identifikation mittels *trial and error*. Hierbei werden die ausgewählten Identifikationsparameter manuell variiert und das Modell simuliert. Dieses Vorgehen wird wiederholt, bis sich die Fehlerfläche zwischen den Simulationsgrößen und den Referenzgrößen hinreichend genau minimiert hat. Das Abbruchkriterium legt hierbei subjektiv der Benutzer fest. Um die Entscheidung zu erleichtern, wird neben der graphischen Darstellung der Ergebnisse der aktuelle Wert der Zielfunktion, in diesem Fall der Wert der quadratischen Fehlerfläche, angezeigt.

Automatische Identifikation

Eine weitere, effizientere Identifikationsmethode ist das Verwenden von Optimierungsverfahren. Dies ist vor allem dann effizienter, wenn es eine große Anzahl

zu identifizierender Parameter gibt. Der erste Schritt bei der Anwendung der automatischen Identifikation besteht in der Auswahl der Optimierungsmethode. MATLAB bietet eine große Anzahl unterschiedlicher, etablierter Optimierungsverfahren. Jedes Verfahren hat spezielle Vorteile bezüglich des Umganges mit Nichtlinearitäten, Black-Box-Modellen, der Notwendigkeit von Nebenbedingungen etc. (vgl. Kapitel 2.6.3). Im Rahmen dieser Arbeit ist unter [Bru14] eine Entscheidungsmatrix erarbeitet und in das entwickelte Parameteridentifikations-Tool integriert worden, welche die Auswahl des zu verwendenden Optimierungsverfahrens erleichtert. Die Entscheidungsmatrix basiert auf Untersuchungen verschiedener Optimierungsverfahren bezüglich ihrer Schnelligkeit, Genauigkeit sowie Robustheit, bezogen auf verschiedene Optimierungsprobleme. Um Vergleichbarkeit und vor allem Reproduzierbarkeit zu gewährleisten, wurde die erarbeitete Entscheidungsmatrix auf die deterministischen, nichtlinearen, einkriteriellen Optimierungsverfahren beschränkt.

Nach der Auswahl des Optimierungsverfahrens kann die Identifikation gestartet werden. Alle für das jeweilige Optimierungsverfahren notwendigen Einstellungen, wie zum Beispiel die Zielfunktion und die benötigten Nebenbedingungen, werden automatisch gesetzt. Das automatische Setzen ist durch die zuvor getroffene Auswahl des Identifikationsbereiches und die in der FMU übergebenen Informationen möglich. Falls es gewünscht ist, können die automatisch gewählten Einstellungen geändert und/oder erweitert werden. Tabelle 4-2 zeigt die initialen Werte der Identifikationsparameter des Segway-Modells sowie die entsprechenden skalierten Werte, die in einem ersten Schritt an das ausgewählte Optimierungsverfahren übergeben werden, wobei zunächst nur der Parameter m_{Segway} identifiziert wird. Die Skalierungsfaktoren F_{scale} sowie die initialen Werte der skalierten Parameter $\vec{P}_{scale,init}$ werden automatisch mittels der unter Abschnitt 2.5 gezeigten Gleichungen 2-2 und 2-3 (S. 25) berechnet.

Tabelle 4-2: Wertebereiche der zu identifizierenden Segway-Parameter und die daraus resultierenden Skalierungsfaktoren

Parameter	P_{init}	Einheit	lb_P	ub_P	F_{scale}	$\vec{P}_{scale,init}$
m_{Segway}	85	kg	50	250	10	$-6{,}5$
Θ_{Segway}	15	$\mathrm{kg\,m^2}$	5	100	$4,75$	$-7,8947$

Bild 4-18 zeigt ähnlich wie Bild 2-14 (S. 26, Abschnitt 2.5) eine schematische Darstellung des Vorgehens für die Parameteridentifikation mit entsprechender Parameter-Skalierung und -Rückskalierung. Neu hinzugekommen ist ein *Preprocessing*-Bereich. In diesem wird überprüft, ob die durch das Optimierungsverfahren variierten Parameter $\vec{P}_{var}$ innerhalb des vorgegebenen Wertebereiches liegen.

Bei der Verwendung von Optimierungsverfahren mit Restringierung ist dies sichergestellt. Bei der Auswahl eines Verfahrens ohne Restringierung werden die vorgegebenen Grenzen nicht berücksichtigt. Sollten Parameter außerhalb des gültigen Wertebereiches für die Simulation an die *simFMU.mex32*-Datei übergeben werden, so kann dies zur Instabilität des Systems führen (z.B. Teilen durch Null). Um dies zu verhindern, wird im *Preprocessing*-Bereich der Simulationsschritt übersprungen und eine Straffunktion in die Auswertung eingeführt, sollte ein Parameter außerhalb seines Wertebereiches liegen. Als Ergebnis der Auswertung erhält das Optimierungsverfahren durch die Straffunktion einen Zielfunktionswert von $f = 10^6$ für die zuvor gewählten Parameter $\vec{P}_{scale,var}$, die in einem nächsten Schritt durch das Optimierungsverfahren neu bestimmt werden. Durch dieses Vorgehen kann sichergestellt werden, dass auch bei Optimierungsverfahren ohne Restringierung nur Parameter innerhalb der vorgegebenen Grenzen für die Simulation verwendet werden. Die Optimierung endet anschließend automatisch, wenn entsprechende Abbruchkriterien (ein Teil der Einstellungen) erreicht sind.

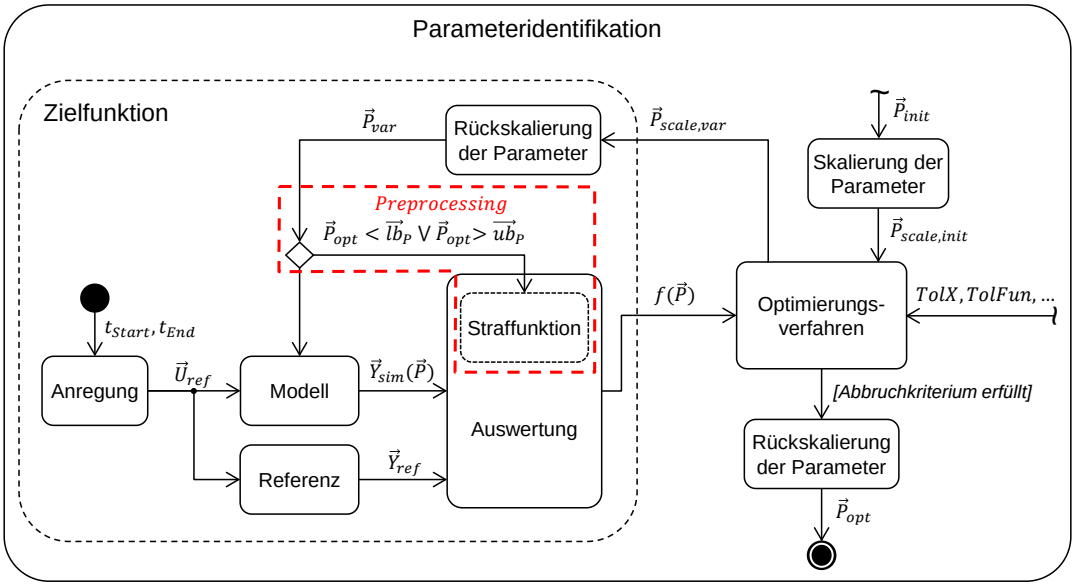

Bild 4-18: Schematische Darstellung des Vorgehens für die Parameteridentifikation mit entsprechender Parameter-Skalierung und -Rückskalierung sowie eines hinzugefügten Preprocessing-Schrittes

Um zu demonstrieren, welche Auswirkungen die Wahl des Optimierungsverfahrens auf das Ergebnis hat, zeigt Tabelle 4-3 die Ergebnisse der Parameteridentifikation des Parameters m_{Segway} für den zuvor ausgewählten Identifikationsbereich mithilfe unterschiedlicher Optimierungsverfahren und Solver. Um eine Vergleich-

barkeit zu gewährleisten, werden an dieser Stelle nur die Ergebnisse der deterministischen, nichtlinearen, einkriteriellen Optimierungsverfahren aufgeführt.

Tabelle 4-3: Ergebnisse der Algorithmen für die Identifikation des Parameters m_{Segway} für den ersten Identifikationsbereich

Algorithmus	$f(\vec{P})$	m_{Segway}	Zeit [sek]	F_{Count}
Startparameter	9,1589	85	-	-
Gradientenbasierte Methode				
fmincon (IP)	0,00025518	160,1518	36,47	22
fmincon (AS)	0,00025537	160,1651	28,58	17
fmincon (SQP)	0,00025518	160,1518	31,44	19
lsqnonlin	0,00025518	160,1518	43,04	26
fminunc	0,00025518	160,1518	33,25	20
Goldener Schnitt				
fminbnd	0,00025518	160,1518	19,90	12
Direktsuchmethode				
fminsearch	0,00025518	160,1518	87,37	54
PS (GPS2N)	0,00025518	160,1518	95,82	58
PS (GPSNp1)	0,00025518	160,1518	95,96	58
PS (MADS2N)	0,00025518	160,1518	95,76	58
PS (MADSNp1)	0,00025518	160,1518	95,61	58

Der Identifikationsbereich ist der zuvor vorgestellte Bereich der Position z von t_{Start}= 0 sek und t_{End}= 5 sek. Die FMU ist als Model Exchange eingebunden und wird mit einem Runge-Kutta-Verfahren 4. Ordnung sowie einer Abtastung und einer Intervalllänge von 0,001 sek durchgeführt. Für die unterschiedlichen Optimierungsverfahren sind die Parameter *TolX* und *TolFun* (vgl. Bild 2-20 S. 35) auf den Wert 10^6 gesetzt sowie die maximale Anzahl der Funktionsevaluationen *MaxFunEvals* auf 1500. Durch die Verwendung identischer Abbruchkriterien ist ein Vergleich der Ergebnisse möglich. Die angegebenen Werte für die benötigte Zeit eines Optimierungsverfahrens lassen sich aufgrund der Verwendung desselben Rechners vergleichen. Dieser Rechner[3] wird auch für den weiteren Verlauf dieser Arbeit verwendet. Bei den ausgewählten Optimierungsverfahren werden *fzero* und *fsolve* nicht betrachtet, da diese für quadratische Zielfunktionen nicht geeignet sind (vgl. Abschnitt 2.6.3). Bei den Algorithmen *fminsearch* und *fminunc*

[3]Bei dem verwendeten Rechner handelt es sich um ein Lenovo ThinkPad W530 mit einem Intel Core i7-3720QM (2,60 GHz, 6 MB Cache) und einem Arbeitsspeicher von 8 GB PC3-12800 1600MHz DDR3 SDRAM. Die Identifikationsumgebung ist unter MATLAB 2014b (32-Bit) erstellt.

werden grundsätzlich keine Nebenbedingungen berücksichtigt. Um sicherzustellen, dass die Simulation aufgrund eines falschen Parametersatzes nicht instabil wird, werden die Grenzen durch den zuvor beschriebenen *Preprocessing*-Bereich eingehalten (vgl. Bild 4-18).

Der Tabelle 4-3 ist zu entnehmen, dass alle Algorithmen sehr gute Ergebnisse liefern. Die minimalen Abweichungen sind auf numerische Ungenauigkeiten während der Simulation zurückzuführen und sind für diesen Anwendungsfall vernachlässigbar. Ein entscheidender Unterschied bei den verwendeten Optimierungsverfahren liegt in der Anzahl der Simulationsaufrufe F_{Count} und somit in der benötigten Zeit. Zwischen dem schnellsten und dem langsamsten Optimierungsverfahren liegt ungefähr ein Faktor 5. Grundsätzlich ist in diesem Fall der Algorithmus *fminbnd* am besten geeignet, da er das Optimum am schnellsten findet. Dies liegt an dem *Goldener-Schnitt-Verfahren mit parabolischer Interpolation*, das der Algorithmus verwendet (vgl. Abschnitt 2.6.2). Dieses Verfahren kann durch die Schrittweitensteuerung mittels des *Goldener-Schnitt-Verfahrens* ohne Berechnung eines Gradienten schnell in die Nähe des Optimums kommen. Dabei ist nach [Mat14] darauf zu achten, dass, wenn ein Optimum auf der Grenze eines Parameters liegt, die Konvergenzrate sehr langsam werden kann.

Das Ergebnis der automatischen Identifikation mittels des Optimierungsverfahrens *fminbnd* für den ersten Identifikationsbereich ist in Bild 4-19, zusammen mit der Referenzgröße und den Simulationsergebnissen vor der Identifikation, dargestellt. Simulation und Referenz liegen nahezu deckungsgleich übereinander.

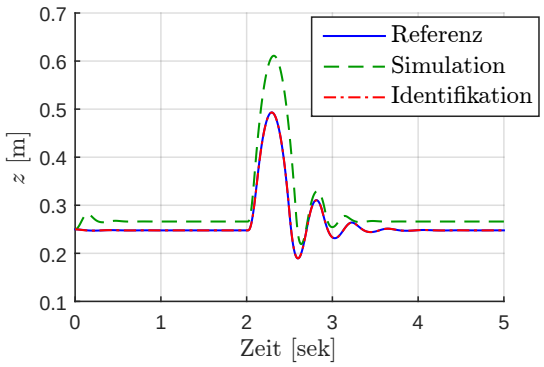

Bild 4-19: Vergleich der Referenzgrößen der Position z des ersten Identifikationsbereiches mit den Simulationsergebnissen vor und nach der Parameteridentifikation

Nach der Identifikation endet der Bereich der Parameteridentifikation, sofern alle Abweichungen der Identifikationsbereiche hinreichend genau minimiert sind.

Bild 4-20 zeigt alle Referenzgrößen über den gesamten Referenzbereich inklusive der Simulationsergebnisse vor und nach der Identifikation der Masse m_{Segway}.

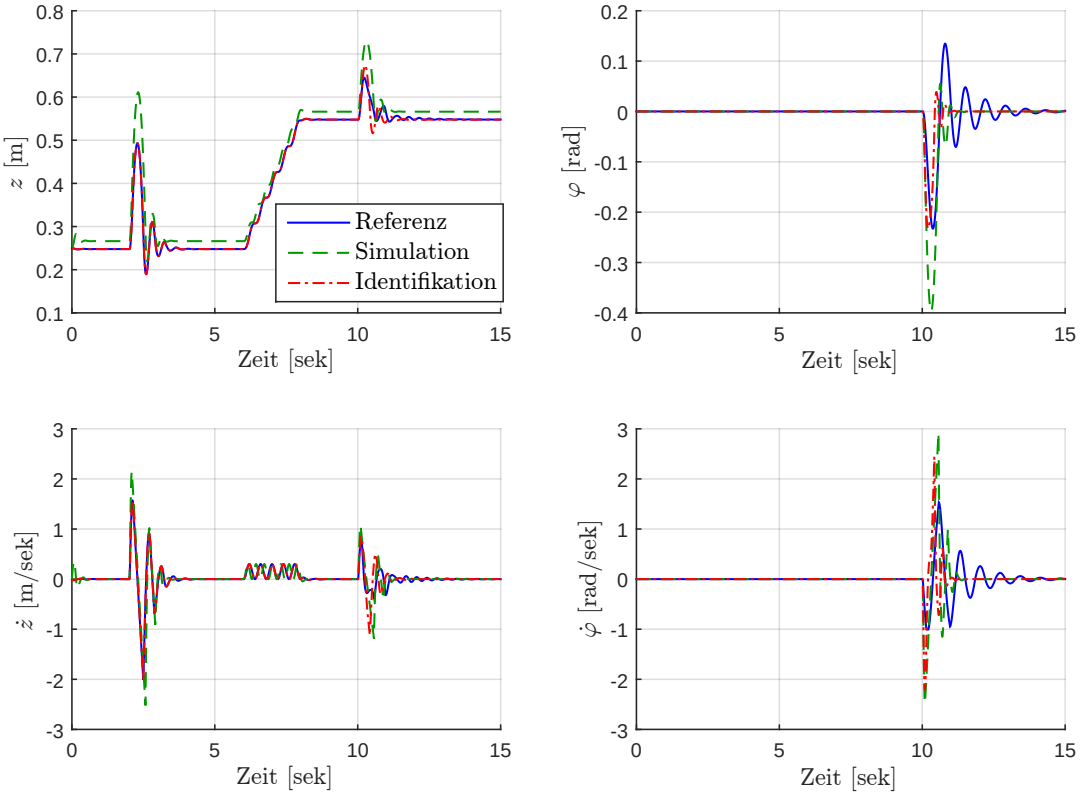

Bild 4-20: Vergleich aller Referenzgröße über den gesamten Referenzbereich inklusive der Simulationsergebnisse vor und nach der Identifikation der Masse m_{Segway}

Durch die Identifikation der Masse m_{Segway} ist die Differenz zwischen den Referenzgrößen und den Simulationsergebnissen deutlich verringert worden. Die Abweichungen sind jedoch noch zu groß, um den Bereich der Parameteridentifikation zu verlassen. Daher erfolgt an dieser Stelle eine Iteration innerhalb des Bereiches der Parameteridentifikation, und ein weiterer Identifikationsbereich wird gewählt.

II. Identifikationsbereich wählen

Die Wahl des zweiten Identifikationsbereiches ist in Bild 4-21 dargestellt. Der Simulationsbereich ist für die Referenzgröße des Winkels φ mit einem Zeitintervall von t_{Start}= 9 sek und t_{End}= 15 sek festgelegt.

Um die Zeit für die anschließende Identifikation zu verkürzen, bietet es sich an, nicht den gesamten Referenzbereich zu simulieren, sondern nur den zu identifizie-

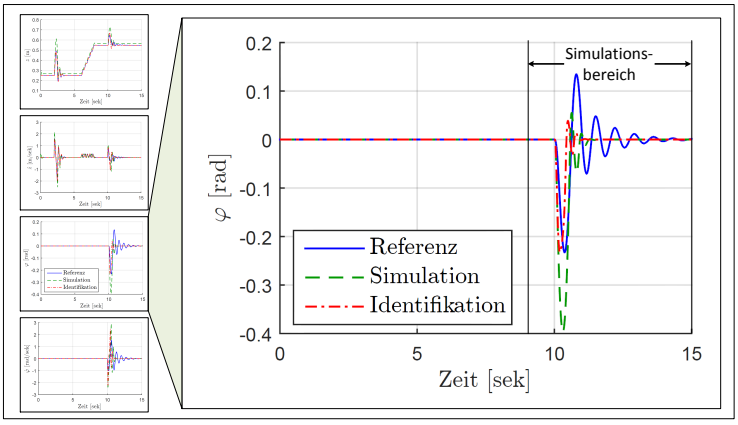

Bild 4-21: Schematische Darstellung der Auswahl des zweiten Identifikationsbereiches für das Segway-Modell

renden. Um dies zu realisieren, bietet das entwickelte Tool die Möglichkeit einer sequentiellen Simulation. Hierzu wird automatisch die *simFMU.mex32*-Datei vom Zeitpunkt $t_{init} = 0\,\text{sek}$ bis t_{Start} des ausgewählten Bereiches ausgeführt. Der am Ende übergebene Zustandsvektor $\vec{X}$ wird zwischengespeichert und dient zur Initialisierung des ausgewählten Simulationszeitraumes. Durch die Möglichkeit, den Gesamtbereich sequentiell zu simulieren, ergibt sich eine erhebliche Zeitersparnis, da nicht zu betrachtende Simulationsbereiche lediglich einmal zur Initialisierung simuliert werden müssen.

Die Initialisierung des Zustandsvektors $\vec{X}$ ist zwingend notwendig, damit das Modell in dem korrekten Betriebspunkt startet. Bild 4-22 zeigt schematisch die Auswirkung auf das zu simulierende Segway-Modell bei einer sequentiellen Simulation bei t_{Start}= 9 sek mit und ohne Zustandsneubesetzung. Das Segway-Modell besitzt

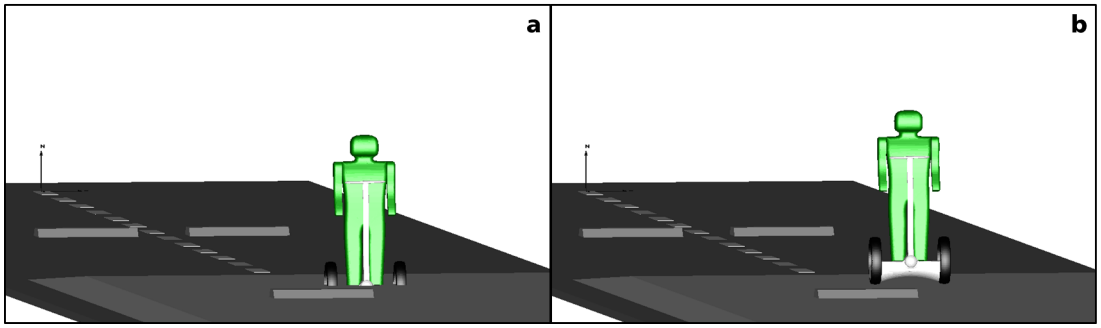

Bild 4-22: Start einer sequentiellen Simulation (a) ohne und (b) mit Zustandsvektorvorgabe

vier Zustände, die in diesem Fall gleichzeitig die Ausgänge des Modells darstellen. Wird lediglich die Startzeit für die Simulation auf t_{Start}= 9 sek gesetzt, ohne dass der Zustandsvektor neu initialisiert wird, so befindet sich das Segway-Modell zu Beginn der Simulation in der Rampe, da der initiale Zustandswert der Position bei z_{init}= 0,25 m liegt, was einer Höhe vor der Rampe entspricht. Aufgrund der daraus resultierenden Rad-Boden-Kräfte (vgl. Gleichung 4-3, S. 95) wird das Segway-Modell nach oben katapultiert. Die Simulationsergebnisse sind für diesen Fall nicht aussagekräftig. Wird der Zustandsvektor wie zuvor beschrieben neu initialisiert, so wird der Wert der Startposition automatisch auf z^*_{init}= 0,5478 m gesetzt, was einer Position auf der Rampe entspricht. Bei der sequentiellen Identifikation sollte kein Bereich übersprungen werden, in dem zu identifizierende Parameter sensitiv sind. Werden diese erst in späteren Bereichen identifiziert, so kann nicht sichergestellt werden, dass der für die Initialisierung verwendete Zustandsvektor bei einer Gesamtsimulation erreicht wird. Eine sequentielle Identifikation sollte daher – wie hier gezeigt – von vorn nach hinten stattfinden. Sind in einem ersten Bereich alle bzw. sehr viele Parameter sensitiv, so sollte ein anderes Messszenario eingeladen werden, das für die Identifikation besser geeignet ist. Nach der Auswahl des Identifikationsbereiches und der Neuinitialisierung des Zustandsvektors wird die Aktivität Identifikationsparameter wählen durchgeführt.

II. Identifikationsparameter wählen

Bei der Auswahl der Parameter kann wiederum die Sensitivitätsanalyse herangezogen werden. Bild 4-23 zeigt die relative Sensitivität der beiden Identifikationsparameter für den zweiten Identifikationsbereich. Neben dem Parameter der Masse m_{Segway} ist für diesen Bereich auch der Parameter der Trägheit Θ_{Segway} sensitiv.

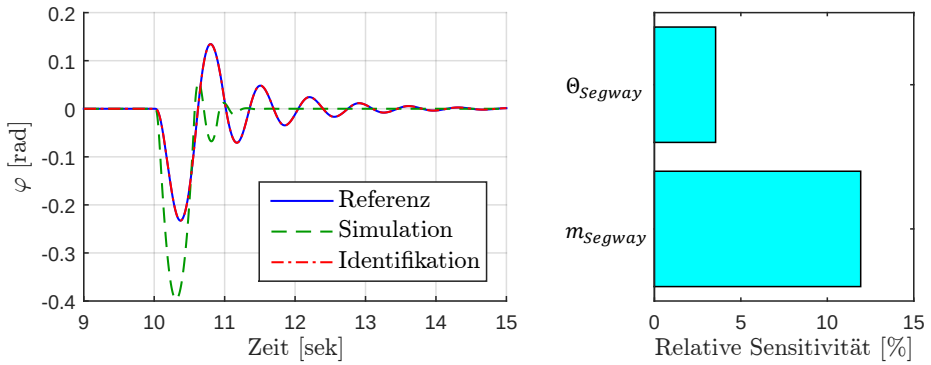

Bild 4-23: Vergleich der Referenzgröße des Winkels für den zweiten Identifikationsbereich mit den Simulationsergebnissen vor und nach der Parameteridentifikation

Dies ist auf die Wankanregung des Systems zurückzuführen. Da der Parameter der Masse bereits identifiziert wurde, wird für die nachfolgende Parameteridentifikation nur der Parameter der Trägheit Θ_{Segway} verwendet.

II. Identifikation starten

Für die Identifikation wird das für diesen Anwendungsfall am besten geeignete Optimierungsverfahren *fminbnd* verwendet. Bild 4-23 zeigt die entsprechenden Ergebnisse. Der identifizierte Wert für die Trägheit liegt bei Θ_{Segway}= 81,97 kg m^2 bei einer quadratischen Fehlerfläche von $f(\vec{P})$= 2,0429 · 10^{-4}. Die minimalen Abweichungen sind wiederum auf numerische Ungenauigkeiten zurückzuführen und können hierbei vernachlässigt werden. Bild 4-24 zeigt die Ergebnisse der Simulation vor und nach der Identifikation der Parameter der Masse m_{Segway} und der Trägheit Θ_{Segway} sowie die dazugehörigen Referenzgrößen für den gesamten Referenzbereich. Durch diese Identifikation sind alle Abweichungen in den zu

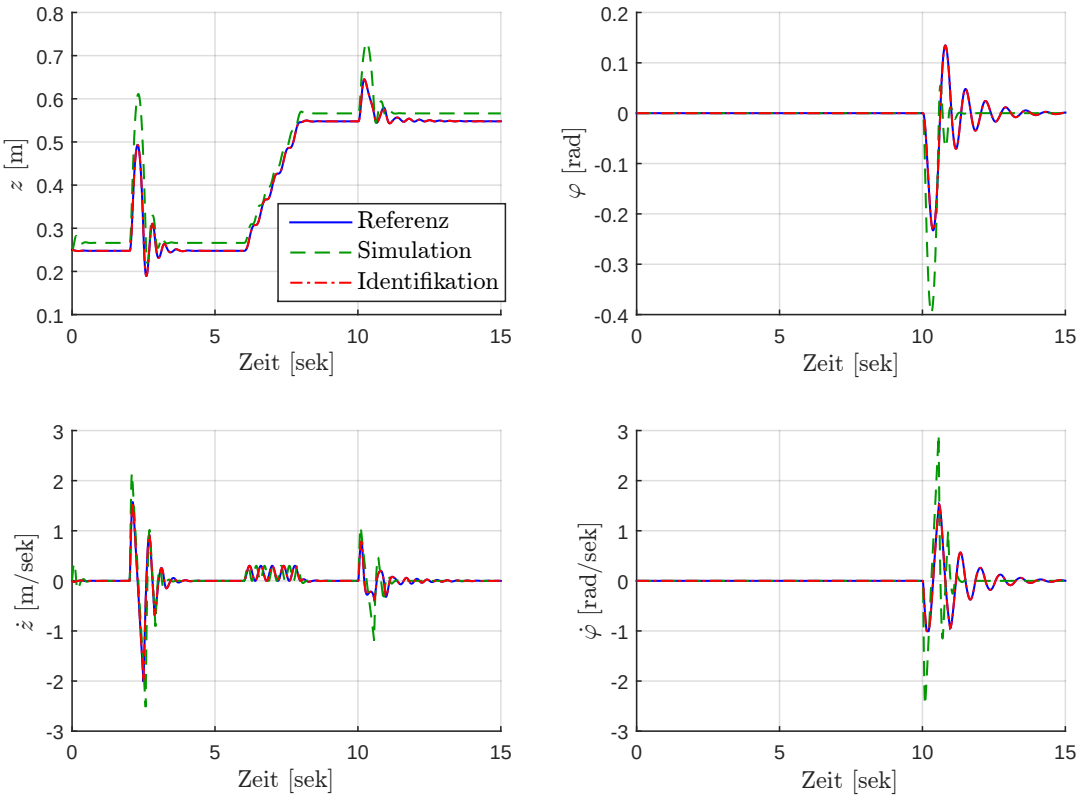

Bild 4-24: Vergleich aller Referenzgröße über den gesamten Referenzbereich inklusive der Simulationsergebnisse vor und nach der Identifikation der Masse m_{Segway} und der Trägheit Θ_{Segway}

betrachtenden Bereichen minimiert, und der Bereich der Parameteridentifikation endet.

Um zu demonstrieren, wie sich die unterschiedlichen Optimierungsverfahren bei der Identifikation mehrerer Parameter gleichzeitig verhalten, werden die Ergebnisse der zuvor verwendeten Algorithmen für eine parallele Identifikation der Parameter m_{Segway} und Θ_{Segway} in Tabelle 4-4 gezeigt. Für die Identifikation werden die zuvor sequentiell identifizierten Parameter auf ihre jeweiligen Initialwerte von m_{Segway}= 85 kg und Θ_{Segway}= 15 kg m^2 zurückgesetzt. Als Identifikationsbereich werden hierzu alle zur Verfügung stehenden Ausgangsgrößen (Position z, Geschwindigkeit $\dot{z}$, Winkel φ, Winkelgeschwindigkeit $\dot{\varphi}$) über den gesamten Referenzbereich von t_{Start}= 0 sek und t_{End}= 15 sek verwendet. Die vorgegebenen Werte für die Abtastzeit und die Intervalllänge liegen wie zuvor bei $t_{StepSize}$=t_{Int}= 0,001 sek.

Tabelle 4-4: Ergebnisse der Algorithmen für die parallele Identifikation der Parameter m_{Segway} und Θ_{Segway}

Algorithmus	$f(\vec{P})$	m_{Segway}	Θ_{Segway}	Zeit [sek]	F_{Count}
Startparameter	1833,0662	85	15	-	-
Gradientenbasierte Methode					
fmincon (IP)	0,066986	160,1511	81,9428	297,53	53
fmincon (AS)	0,066986	160,1509	81,9426	227,72	42
fmincon (SQP)	0,066986	160,1511	81,9428	270,68	50
lsqnonlin	0,052862	160,0278	81,9061	722,45	131
fminunc	0,066353	160,1499	81,9484	628,09	117
Goldener Schnitt					
fminbnd	-	-	-	-	-
Direktsuchmethode					
fminsearch	0,052644	160,0296	81,9192	1151,63	214
PS (GPS2N)	0,052648	160,0294	81,9174	957,78	176
PS (GPSNp1)	0,053355	160,0330	81,9429	990,01	184
PS (MADS2N)	0,052648	160,0294	81,9174	1756,17	326
PS (MADSNp1)	0,052648	160,0294	81,9174	1536,63	284

Zunächst einmal ist zu erkennen, dass die initiale Fehlerfläche von $f(\vec{P})$= 1833,07 deutlich höher ist als bei der zuvor betrachteten Identifikation eines Parameters (vgl. Tabelle 4-3: $f(P)$= 9,1589). Die Ursache liegt im ausgewählten Identifikationsbereich. Dieser setzt sich aktuell aus allen Referenzgrößen über den gesamten Referenzzeitraum zusammen. Des Weiteren kann festgehalten werden, dass der Algorithmus *fminbnd* für diesen Anwendungsfall nicht geeignet ist, da er nur für die Identifikation eines Parameters verwendet werden kann. Die ver-

wendeten Algorithmen zeigen sehr gute Ergebnisse. Ein wesentlicher Unterschied liegt jedoch wie schon zuvor in der benötigten Zeit. Der langsamste Algorithmus $PS(MADS2N)$ benötigt im Vergleich zum schnellsten Algorithmus $Fmincon(AS)$ mehr als das Siebenfache der Simulationsaufrufe und somit der benötigten Zeit, um auf ein vergleichsweise ähnliches Ergebnis zu kommen. Damit ein Entwickler ohne detaillierte Vorkenntnisse bezüglich Optimierungsverfahren dennoch in der Lage ist, den jeweils geeignetsten Algorithmus zu wählen, kann auf die eingebundene Entscheidungsmatrix [Bru14] zurückgegriffen werden. Diese ordnet die jeweils zur Verfügung stehenden Algorithmen im Bezug auf Schnelligkeit, Genauigkeit und Robustheit dynamisch dem entsprechenden Optimierungsproblem zu.

II. Referenzgrößen einladen

Nach der erfolgreich durchgeführten Parameteridentifikation muss eine Iteration für die Modellvalidierung stattfinden (vgl. Bild 4-25). Hierzu werden in einem nächsten Schritt weitere Referenzgrößen eingeladen und der `Model Check` durchgeführt. Sollten trotz der zuvor durchgeführten Parameteridentifikation die Abweichungen zwischen den neuen Referenzgrößen und den Simulationsergebnissen zu groß sein, so ist das Modell nicht validiert. Falls bereits alle Parameter durch die zuvor eingebundenen Referenzgrößen identifiziert wurden, wird eine erneute Parameteridentifikation zwar die Abweichungen minimieren, die Ergebnisse sind jedoch nicht allgemeingültig. An dieser Stelle ist zu prüfen, ob für die jeweils identifizierten Parameter ein Parametersatz als Kompromiss gefunden werden kann, durch den das Modell alle Referenzgrößen hinreichend genau abbildet. Ist dies nicht möglich, so ist eine Strukturidentifikation durchzuführen. Das heißt, dass

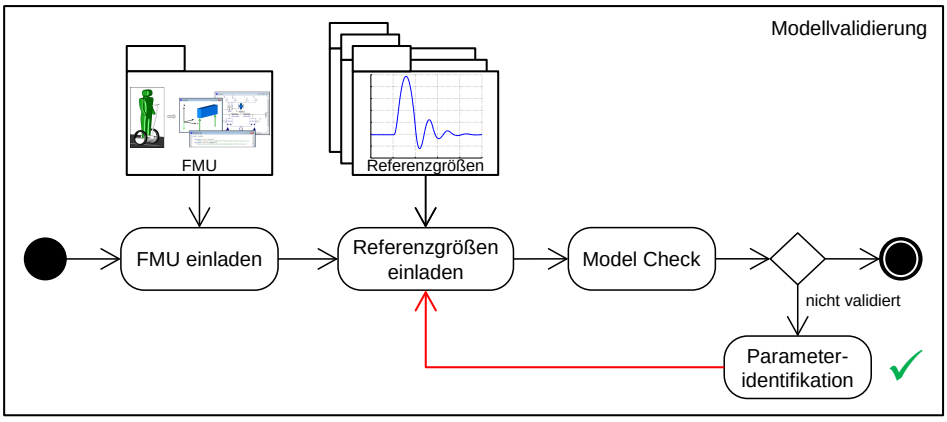

Bild 4-25: Iteration der Modellvalidierung nach erfolgreicher Parameteridentifikation

das Modell gegebenenfalls erweitert oder eine andere Modellierungstiefe gewählt werden muss [LSB+12, Loc]. Anschließend wird der Bereich der Modellvalidierung erneut durchgeführt. Die Iteration zwischen Parameter- und Strukturidentifikation erfolgt so lange, bis ein gewünschtes Verhalten erreicht wird. Im Falle des Anwendungsbeispiels kann auf das Einladen weiterer Referenzgrößen verzichtet werden, da die identifizierten Parameter nahezu identisch mit den Parametern des Referenzmodells sind. Weitere Vergleiche mit Referenzgrößen sehen entsprechend identisch aus. Erforderlich wird das erneute Einladen bei komplexeren Modellen, deren Identifikationsparameter nicht bekannt sind. An dieser Stelle zeigt sich, ob das Modell validiert und somit für Analysen zu betrachtender Betriebsbereiche geeignet ist oder nicht. Der nachfolgende Abschnitt beschreibt das Vorgehen bei einem entsprechend komplexen Modell am Beispiel des zuvor vorgestellten Umflut-Waschverfahren-Modells.

4.3 Validierungsmethodik, angewendet für das Umflut-Waschverfahren-Modell

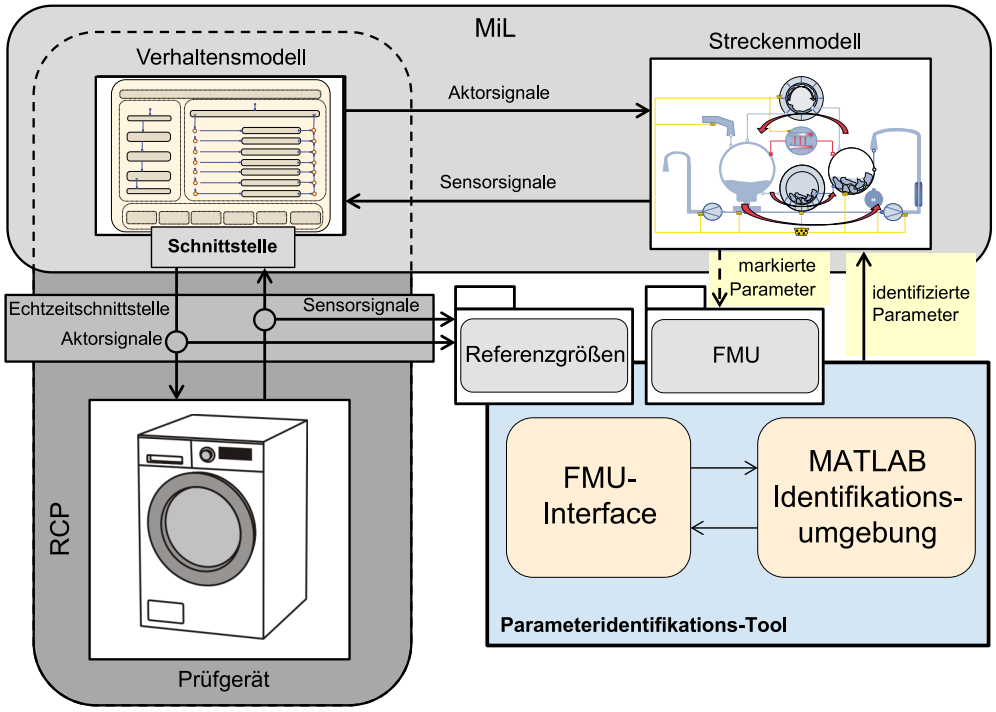

Bild 4-26: Schematische Darstellung der Einbindung des Parameteridentifikations-Tools für die MiL- und RCP-Umgebung des Umflut-Waschverfahren-Modells

Während der in Abschnitt 3.4 gezeigten Synthese des disziplinspezifischen Entwurfes sind komplexe Verhaltensmodelle der Steuerung und der Strecke entstanden. Des Weiteren ist eine Prüfumgebung in Form eines RCP-Prüfstandes aufgebaut worden. Die für die Analyse und Bewertung erforderliche Modellvalidierung des Streckenmodells erfolgt mithilfe der zuvor vorgestellten Parameteridentifikations- und Modellvalidierungsmethodik. Bild 4-26 zeigt schematisch die Einbindung des Parameteridentifikations-Tools für die MiL- und RCP-Umgebung des Umflut-Waschverfahren-Modells. Zur Validierung des Modells, wird dieses als FMU in das Parameteridentifikations-Tool eingebunden. Bild 4-27 zeigt diesen Vorgang. Des Weiteren zeigt das Bild exemplarisch einige durch das Tool automatisch zur Verfügung gestellte Modellinformationen. Neben der Anzahl, den Namen und den Werten von Eingängen, Ausgängen, Zuständen und Identifikationsparametern kann der FMU des Weiteren entnommen werden, in welchem Tool, wann und in welcher Version sie erstellt wurde.

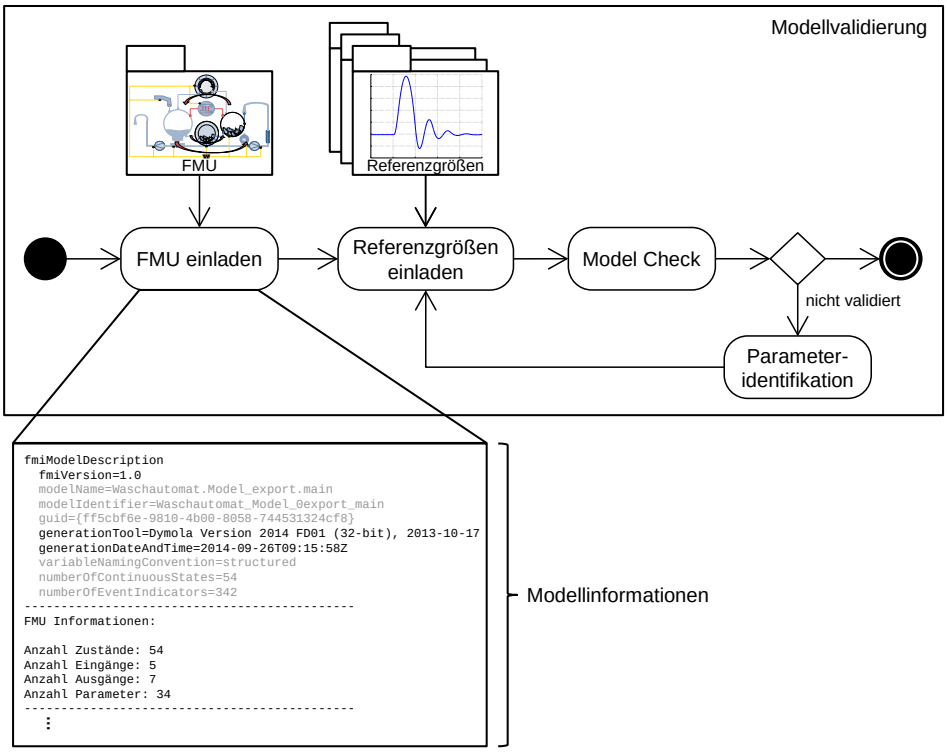

Bild 4-27: Schematische Darstellung des eingeladenen Waschautomatenmodells in das Parameteridentifikations-Tool

Aufgrund vieler Annahmen, die während der Modellierung getroffen wurden, ist es nachvollziehbar, warum das Modell ohne eine Parameteridentifikation nicht validiert ist. Um die jeweiligen Parameter zu identifizieren, ist es ratsam, spezielle

Messszenarien zu verwenden. Das bedeutet, dass nur ein paar Identifikationsparameter für bestimmte Messszenarien sensitiv sind und genügend Messszenarien zur Verfügung stehen, um alle Parameter zu identifizieren. Die gezielten Messszenarien können durch entsprechendes Vorwissen über die jeweiligen Identifikationsparameter gewählt werden. Falls dieses Wissen nicht vorhanden ist (vgl. Motivation, Kapitel 1 – keine Experten mehr vorhanden), sollte, wenn möglich, Rücksprache mit dem jeweiligen Modellierer gehalten werden. Ist dies nicht möglich und sind viele Parameter für ein Messszenario sensitiv, so kann es dennoch möglich sein, diese zu identifizieren. Anderenfalls kann über das Messszenario ausgesagt werden, dass es nicht zur Identifikation geeignet ist, und es muss ein anderes ausgewählt werden. Das hier betrachtete Multidomänen-Modell des Waschautomaten benötigt entsprechend viele gezielte Messszenarien, damit die unterschiedlichen Parameter der jeweiligen Domäne sequentiell identifiziert werden können. Um einen Überblick über das Vorgehen für ein solch komplexes Modell zu liefern, wird nachfolgend auf die Identifikation der unter Abschnitt 3.4.5 vorgestellten Absorptionsparameter eingegangen.

Teilautomatisierte Identifikation der Absorptionsparameter

Um die markierten Parameter des Absorptionsmodells sequentiell zu identifizieren, werden drei gezielte Messszenarien verwendet (siehe Bild 4-28). Die Messszenarien sind wie folgt durchgeführt worden:

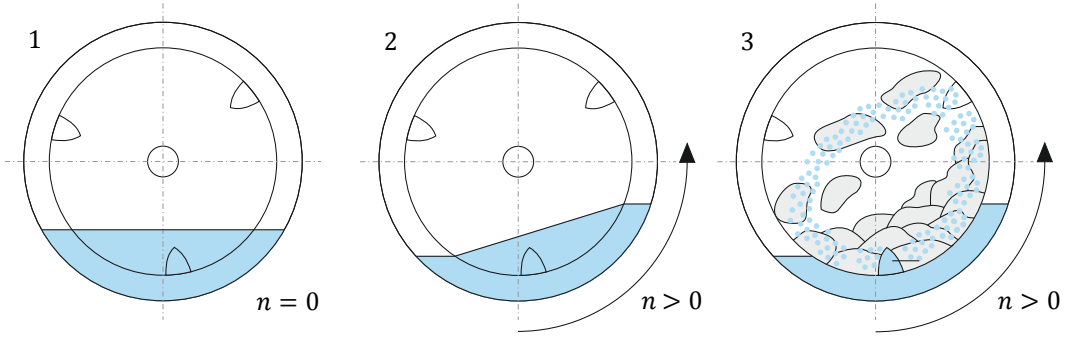

Bild 4-28: Durchgeführte Messszenarien für die Identifikation der Absorptionsparameter

1) Definierte Wasserzulaufmenge, keine Trommeldrehung, keine Beladung,
2) definierte Wasserzulaufmenge, variable Trommeldrehung, keine Beladung,
3) definierte Wasserzulaufmenge, variable Trommeldrehung, definierte Beladung.

Das erste Messszenario wird im Schritt `Referenzgrößen laden` (vgl. Bild 4-27) eingebunden, und die Signale werden mit den entsprechenden Schnittstellen des Modells verknüpft (vgl. Abschnitt 4.2.2). Bild 4-29 zeigt die Ergebnisse des `Model Checks` für dieses Messszenario. Hierbei handelt es sich um das gemessene und das simulierte Drucksignal, die dem Flüssigkeitsstand der FF in dem Laugenbehälter entsprechen.

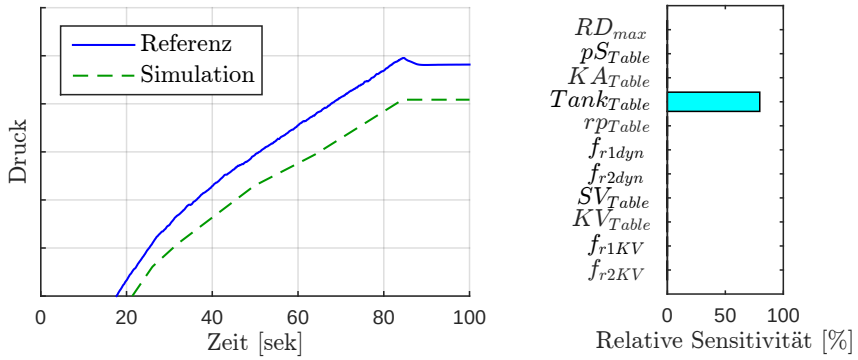

Bild 4-29: Vergleich zwischen Messung und Simulation des nicht validierten Multidomänen-Modells und den dazugehörigen Sensitivitätswerten

Um die deutlichen Abweichungen zu minimieren, legt man im nächsten Schritt den Identifikationsbereich fest, bevor die entsprechenden zu identifizierenden Parameter ausgewählt werden. Wie zuvor beschrieben, stellt der Identifikationsbereich den Bereich dar, in dem die Abweichungen minimiert werden sollen. In diesem Fall ist dies der gesamte Simulationsbereich. Um herauszufinden, welche Parameter für diesen Bereich sensitiv sind, verwendet man die unter Gleichung 4-4, (S. 104) vorgestellte Sensitivitätsanalyse. Die entsprechenden Ergebnisse sind ebenfalls in Bild 4-29 abgebildet. Für die Darstellung der relativen Sensitivität von Kennlinien wird der entsprechend maximale Wert angezeigt. Für das erste Messszenario sind nur die hydrostatischen Parameter in der $Tank_{Table}$-Kennlinie sensitiv. Dies liegt daran, dass alle anderen Identifikationsparameter des Absorptionsmodells, wie unter Kapitel 3.4.5 beschrieben, von der Drehzahl oder der Beladung abhängen. Da diese in dem ersten Messszenario nicht vorhanden sind, werden die anderen Werte der Sensitivitätsanalyse zu Null. Für die Ermittlung der Kennlinienparameter wird die in das Identifikations-Tool integrierte automatische Identifikation verwendet (vgl. Abschnitt 4.2.2). Mithilfe der integrierten Voreinstellungen kann das Identifikationsverfahren direkt gestartet werden.

Die resultierenden Ergebnisse für das erste Messszenario sind in Bild 4-30 dargestellt. Die Ergebnisse der Simulation stimmen sehr gut mit den Referenzgrößen

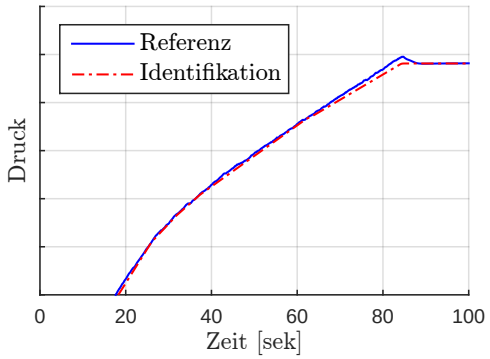

Bild 4-30: Vergleich zwischen dem ersten Messszenario und den entsprechenden Simulationsergebnissen des identifizierten Multidomänen-Modells

überein. Da kein weiterer Identifikationsbereich vorhanden ist, endet der Bereich der Parameteridentifikation für das erste Messszenario.

Im nächsten Schritt erfolgt erneut die Aktivität `Referenzgrößen laden` des Bereiches der Modellvalidierung (vgl. Bild 4-31). Die Klammer um den grünen Pfeil des Bereiches der Parameteridentifikation soll andeuten, dass dieser zwar erfolgreich war, jedoch noch nicht alle Parameter identifiziert wurden. Um die nächsten Parameter zu identifizieren, wird das zweite Messszenario (definierte Wasserzulaufmenge, variable Trommeldrehung, keine Beladung) eingeladen und mit den entsprechenden Ein- und Ausgängen der FMU verknüpft. Bild 4-32 zeigt die Ergebnisse des `Model Checks` für das zweite Messszenario inklusive einer Darstellung der dazugehörigen Sensitivitätswerte.

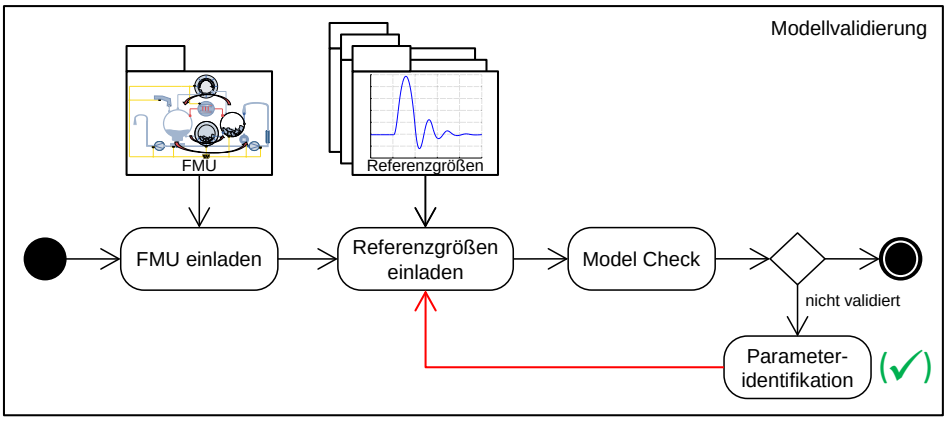

Bild 4-31: Iteration der Modellvalidierung nach erfolgreicher Parameteridentifikation

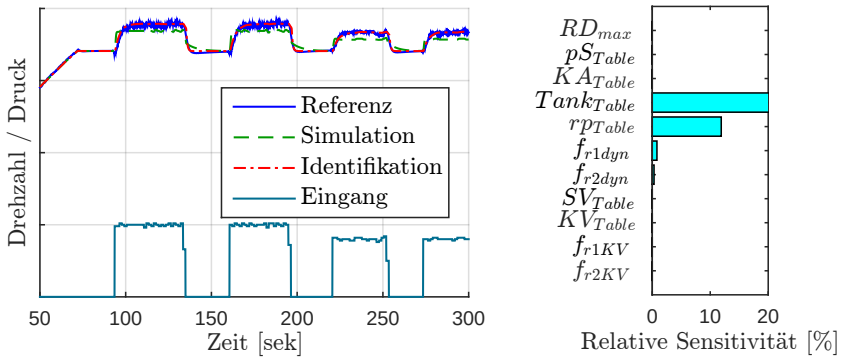

Bild 4-32: Vergleich des zweiten Messszenarios mit den jeweiligen Simulationsergebnissen und den dazugehörigen Sensitivitätswerten

In diesem Fall stimmen die Ergebnisse der Simulation bis zum Beginn der Trommeldrehung mit der Referenzgröße sehr gut überein. Dieser Bereich ist vergleichbar mit dem ersten Messszenario. Die hierfür sensitiven Parameter wurden bereits identifiziert, was die Ergebnisse erklärt. Diese werden bei der weiteren Identifikation nicht weiter betrachtet. Ab Beginn der Trommeldrehung werden weitere, zuvor nicht betrachtete Parameter sensitiv (vgl. Bild 4-32). Wie zuvor gezeigt (vgl. Abschnitt 4.2.2), ist mithilfe des Parameteridentifikations-Tools eine sequentielle Identifikation möglich, was an dieser Stelle genutzt wird. Ein erster Identifikationsbereich (I) wird gewählt, kurz bevor die Trommel zu drehen beginnt, und ein zweiter (II), kurz bevor diese wieder still steht (vgl. Bild 4-33). Im ersten Bereich sind der Parameter f_{r1dyn} und im zweiten Bereich der Parameter f_{r2dyn} nicht sensitiv (vgl. Tabelle 3-1, S. 72). Dadurch, dass die Optimierungsverfahren mit steigender Anzahl an Optimierungsparametern deutlich mehr Zeit für das Auffinden eines Optimums benötigen, bietet die sequentielle Identifikation eine erhebliche Zeitersparnis. Zudem wird durch die gezielte Wahl von möglichst kurzen Identifikationsbereichen die benötigte Simulationszeit ebenfalls verkürzt, was wiederum zu einer Zeitersparnis führt.

Um die Bereiche mit den jeweiligen Parametern zu identifizieren, wählt man zunächst den ersten Identifikationsbereich im Zeitintervall von t_{Start}= 93 sek bis t_{End}= 110 sek. Der Zustandsvektor $\vec{X}$ wird daraufhin automatisch neu initialisiert. Durch die gezeigte Sensitivitätsanalyse können die sensitiven Parameter ausgewählt werden. Anschließend erfolgt mithilfe der Voreinstellungen eine automatische Identifikation. Ist diese erfolgreich durchlaufen, wird der zweite Identifikationsbereich im Zeitintervall von t_{Start}= 130 sek bis t_{End}= 150 sek gewählt. Das Vorgehen ist, wie zuvor beschrieben, die automatische Neuinitialisierung des Zustandsvektors $\vec{X}$, die Auswahl der sensitiven Parameter und das Starten des

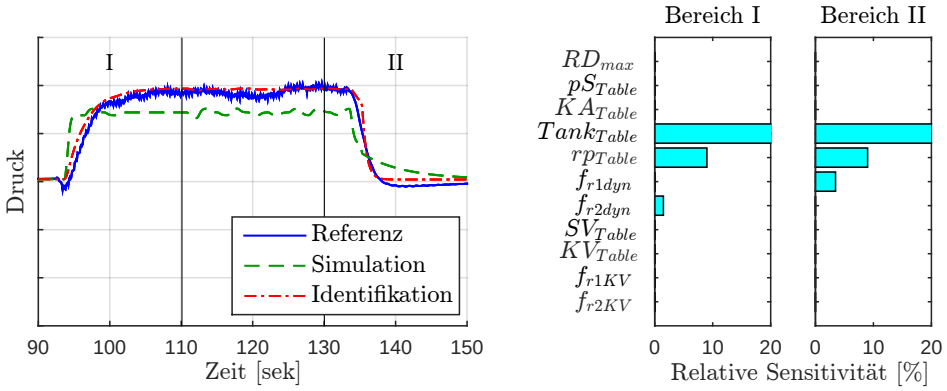

Bild 4-33: Vergrößerung des zweiten Messszenarios und Unterteilung in zwei Identifikationsbereiche

Optimierungsverfahrens, das automatisiert die ausgewählten Parameter variiert, bis die entsprechenden Abbruchkriterien die Optimierung automatisch beenden. Bild 4-33 zeigt die Ergebnisse der sequentiellen Identifikation für die beiden Identifikationsbereiche. Auch für das zweite Messszenario bilden die Ergebnisse der Simulation, nach der Parameteridentifikation, die Referenzgröße sehr gut nach.

Für die Identifikation der noch ausgebliebenen Parameter wird das dritte Messszenario (definierte Wasserzulaufmenge, variable Trommeldrehung, definierte Beladung) durch eine weitere Iteration geladen und mit dem Modell verknüpft. Bild 4-34 zeigt die Ergebnisse des `Model Check`, die sensitiven Parameter und die Simulationsergebnisse nach der Parameteridentifikation. Durch die in diesem

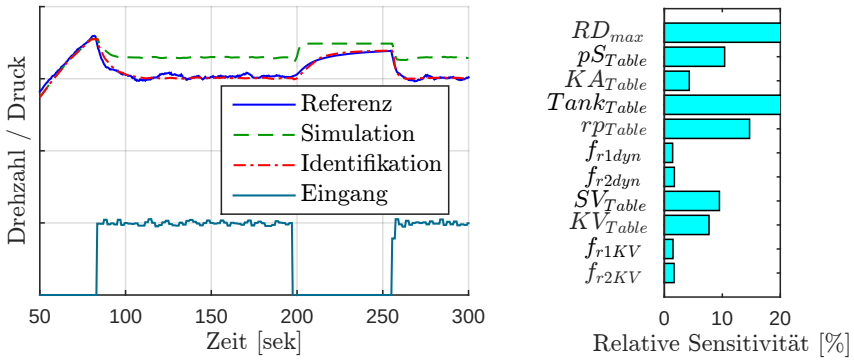

Bild 4-34: Vergleich des dritten Messszenarios mit der Simulation ohne Parameteridentifikation, mit Parameteridentifikation und den dazugehörigen Sensitivitätswerten

Szenario zum ersten Mal vorhandene Beladung sind nun auch die noch verbliebenen Parameter sensitiv. Die zuvor identifizierten Parameter werden nicht weiter berücksichtigt. Eine erneute Identifikation könnte negative Auswirkungen auf die bereits identifizierten Messszenarien haben. Nur die neu hinzugekommenen sensitiven Parameter werden im dritten Messszenario identifiziert. Erst durch die somit reduzierte Zahl an sensitiven Parametern ist das dritte Messszenario für die Identifikation geeignet. Durch das Vorgehen der sequentiellen Simulation kann die Anzahl nochmals reduziert werden. Des Weiteren wird somit nur der Zeitraum simuliert, in dem entsprechende Parameter sensitiv sind. Ein verkürzter Simulationszeitraum bedeutet wiederum eine Zeitersparnis für die Identifikation. Die in Bild 4-34 gezeigten Ergebnisse des identifizierten Modells zeigen, dass die Durchführung des Parameteridentifikationsbereiches für das dritte Messszenario erfolgreich war.

Nachdem durch die drei Messszenarien alle markierten Parameter identifiziert sind, werden weitere Referenzgrößen in einer weiteren Iteration eingebunden, um sicherzustellen, dass das Modell validiert und auch außerhalb der für die Identifikation verwendeten Messszenarien gültig ist. Bild 4-35 bestätigt, dass das Modell nach der erfolgreichen Parameteridentifikation für die Abbildung des Drucksignals validiert ist. Die Ergebnisse der Simulation stimmen hinreichend genau mit den Referenzgrößen überein. Das eingebundene Szenario ist geeignet, um eine Aussage zu treffen, ob das Modell validiert ist, da in diesem alle zuvor identifizierten Parameter sensitiv sind.

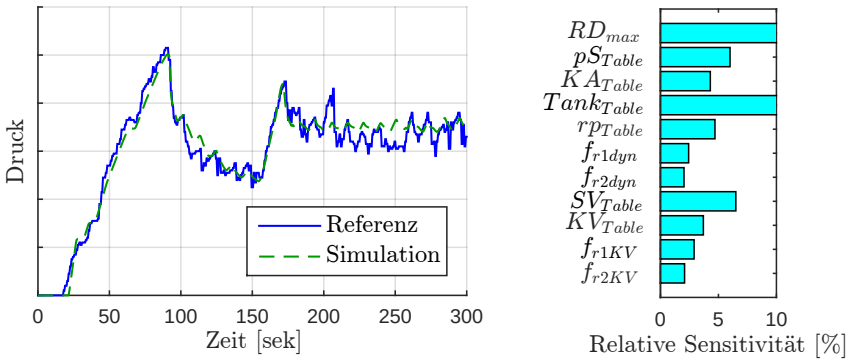

Bild 4-35: Darstellung der Ergebnisse der Modellvalidierung für die Abbildung des Drucksignals des Umflut-Waschverfahren-Modells inklusive der zugehörigen Sensitivitäten

Da es sich beim betrachteten Waschautomatenmodell um ein Multidomänen-Modell handelt, gehört zu einer ganzheitlichen Validierung mehr als das Abbilden des Drucksignals. Die Identifikation der anderen Domänen erfolgt analog zu dem

gezeigten Vorgehen für das Absorptionsmodell. Auf die Identifikation der anderen Domänen wird an dieser Stelle nicht näher eingegangen. Bild 4-36 zeigt die Simulationsergebnisse des zuvor unter Abschnitt 3.4.6 gezeigten Verfahrensablaufes des validierten Modells. Neben der Abbildung des Drucksignals ist hier zudem der Temperaturverlauf gezeigt. Die Abweichungen des Temperaturverlaufes sind etwas größer als die des Drucksignals. Dies ist darauf zurückzuführen, dass in dem Modell die gesamte Temperatur der FF als Sensorwert ausgegeben wird und in der Realität nur der Bereich der FF, an der sich der Sensor befindet. Da sich die reale FF im System nicht so ideal homogen verhält wie im Modell, gibt es während des Aufheizens kleinere Abweichungen. Wichtig ist hierbei, dass die gemessene und die simulierte Temperatur der FF nach der Heizphase hinreichend genau übereinstimmen, da hier von einer gleichmäßigen Temperaturverteilung im System ausgegangen werden kann. Dies ist der Fall, und es kann gezeigt werden, dass das Modell für die Abbildung des Temperaturverlaufes validiert ist.

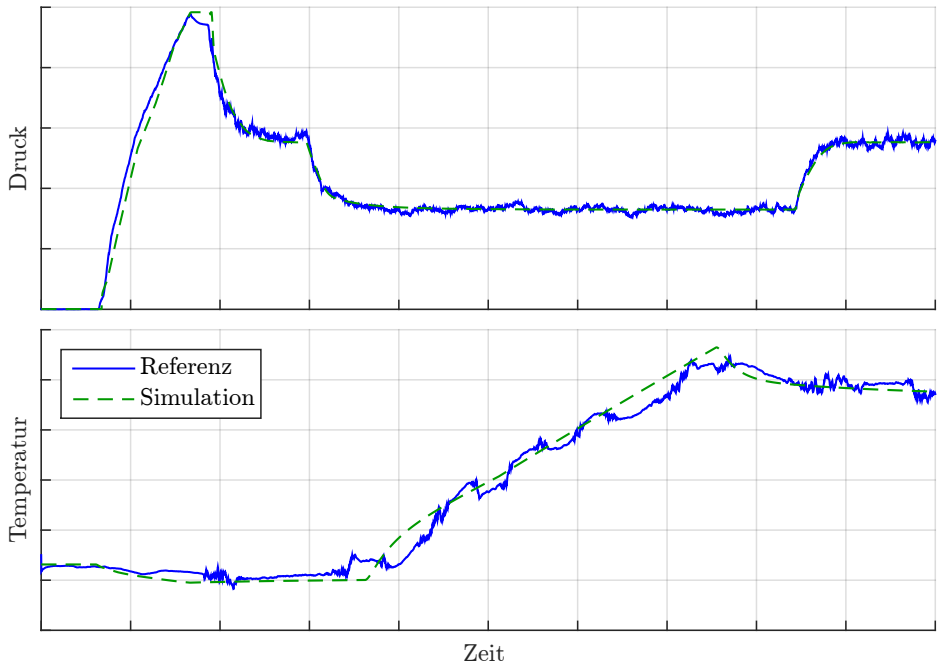

Bild 4-36: Darstellung der Ergebnisse der Modellvalidierung für die Abbildung des Drucksignals sowie des Temperaturverlaufes des Umflut-Waschverfahren-Modells

Anhand der vorgestellten Parameteridentifikations- und Modellvalidierungsmethodik wurde gezeigt, wie dynamische Modelle aus modernen Modellierungswerkzeugen wie Dymola teilautomatisiert mithilfe eines entwickelten Identifikations-

Tools und der darin eingebundenen etablierten Optimierungsverfahren identifiziert werden können. Schon während der Modellierung in der Konkretisierung werden hierzu die benötigten Identifikationsparameter von den jeweiligen Experten markiert. Die markierten Parameter können für die Validierung automatisch durch das erarbeitete Interface der Identifikationsumgebung zur Verfügung gestellt werden. Da es sich beim Interface um eine standardisierte Schnittstelle handelt, ist die Identifikationsumgebung unabhängig von der verwendeten Modellentwicklungslandschaft, was ein Höchstmaß an Flexibilität bietet. Um bei der Parameteridentifikation eine Auswahl des geeignetsten Optimierungsverfahrens treffen zu können, ohne dass detailliertes Vorwissen über diese vorliegen muss, kann man auf eine implementierte Entscheidungsmatrix zurückgreifen [Bru14]. Die Vorteile einer sequentiellen Identifikation, die das entwickelte Tool ermöglicht, liegen vor allem in einer erheblichen Zeitersparnis. Dass die entwickelte Parameteridentifikations- und Modellvalidierungsmethodik auch auf andere Modelle übertragbar ist, wird im nachfolgenden Kapitel anhand ausgewählter Beispiele demonstriert.

5 Übertragbarkeit der Modellvalidierungsmethodik

Dieses Kapitel zeigt die Übertragbarkeit der zuvor vorgestellten Parameteridentifikations- und Modellvalidierungsmethodik. Hierzu werden exemplarisch ausgewählte Dynamikmodelle, die sich jeweils in unterschiedlichen Entwicklungszuständen befinden und die aus unterschiedlichen Dömanen bestehen, in das Tool eingebunden und mittels der teilautomatisierten Parameteridentifikation validiert. Hierbei handelt es sich zum einen um ein Mehrkörpersystem (MKS)-Modell eines leichten Kettenfahrzeugs Wiesel 1 sowie zum anderem um ein Multidomänen-Modell eines Teigkneters. Eine weitere Besonderheit der ausgewählten Modelle, die die Übertragbarkeit der Parameteridentifikations- und Modellvalidierungsmethodik zeigen soll, besteht in dem unterschiedlich vorliegenden Expertenwissen über die Modelle. Das MKS-Modell eines leichten Kettenfahrzeugs Wiesel 1 wird vom Modellierer selbst mithilfe der Parameteridentifikations- und Modellvalidierungsmethodik identifiziert. Hierdurch soll gezeigt werden, dass das entwickelte Parameteridentifikations-Tool für die Modellvalidierung besser geeignet ist, als die zeitaufwändige Vorgehensweise mittels *trial and error*, wodurch die Parameter des Modells zuvor identifiziert wurden. Das Multidomänen-Modell eines Teigkneters wird hingegen vom Modellierer für die Parameteridentifikation und Modellvalidierung übergeben. Hierzu hat der Modellierer die erforderlichen Identifikationsparameter zuvor entsprechend im Modell markiert. Hierdurch soll demonstriert werden, dass es das entwickelte Parameteridentifikations-Tool ermöglicht, Modelle zu validieren, ohne dass Expertenwissen über das vorliegende Modell benötigt wird (vgl. Abschnitt 2.7).

Die jeweiligen Modelle werden im entsprechenden Unterkapitel kurz dargestellt. Zunächst wird die Übertragbarkeit der zuvor vorgestellten Parameteridentifikations- und Modellvalidierungsmethodik anhand des MKS-Modells eines Wiesel 1 beschrieben.

5.1 Wiesel 1

Das Ziel eines früheren Projektes zwischen der Universität Paderborn und der Bundeswehr bestand in dem mechatronischen Entwurf und der Implementierung einer semiaktiven Federung in ein leichtes Kettenfahrzeug [GIT11]. Einer der wichtigsten Meilensteine in diesem Projekt bestand in der Erstellung eines geeigneten Modells für die Vertikaldynamik des Fahrzeugs [Kru09]. Das Modell diente als Grundlage für den nachfolgenden modellbasierten Entwurf und die Erprobung von Regelstrategien.

Um die Übertragbarkeit der vorgestellten Ergebnisse nachzuweisen, wird das bereits entwickelte Modell des leichten Kettenfahrzeugs aufgegriffen und in das zuvor beschriebene Parameteridentifikations-Tool implementiert. Bevor dieses Vorgehen genauer beschrieben wird, wird das hier zu betrachtende leichte Kettenfahrzeug Wiesel 1 vorgestellt.

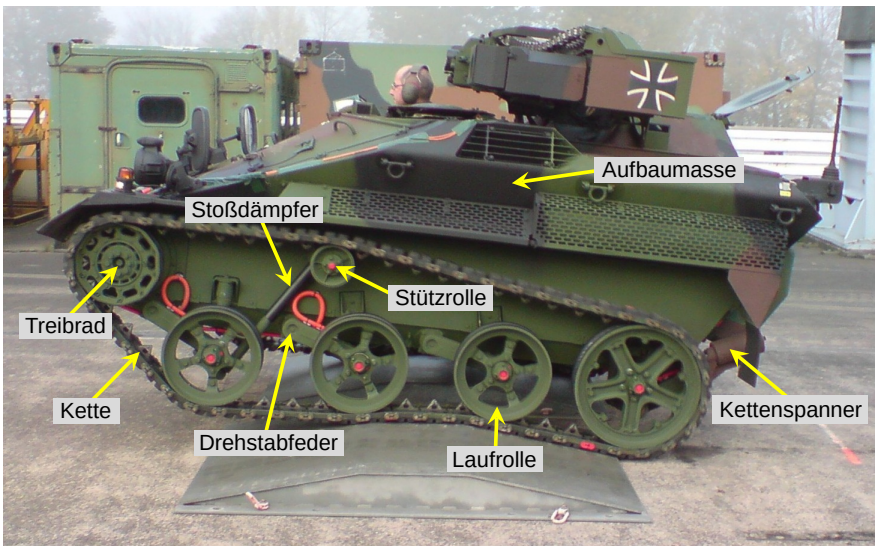

Bild 5-1: Leichtes Kettenfahrzeug Wiesel 1 (in Anlehnung an [Kru09])

Der Wiesel 1 (vgl. Bild 5-1) gehört mit einem Gewicht von 2,75 t zu den leichten Kettenfahrzeugen. Seine Abmaße betragen 3,545 m in der Länge, 1,82 m in der Breite und 1,825 m in der Höhe. Das Fahrwerk des Wiesel 1 besitzt auf jeder Seite vier drehstabgefederte Laufrollen, ein Treibrad und eine Stützrolle. Die vorderen und die hinteren Laufrollen haben zusätzlich hydraulische Stoßdämpfer. Die Kette wird über einen Kettenspanner vorgespannt. Als Motor besitzt der Wiesel 1 einen 5-Zylinder-Dieselmotor mit 64 kW und erreicht damit eine Höchstgeschwindigkeit von 70 km/h. Alle technischen Daten sind in Tabelle 5-1 aufgelistet.

Aufgrund seines geringen Gewichts zeichnet sich der Wiesel 1 durch ein hohes Maß an Agilität aus. Wegen der kleinen Fahrzeuglänge ist es jedoch schwierig, die Nickbewegung der Aufbaumasse ausreichend zu dämpfen. Eine Geländefahrt mit erhöhter Geschwindigkeit bewirkt somit hohe Belastungen auf Insassen und Material. Die fehlende Primärfederung führt außerdem zu einer erhöhten Belastung bei höheren Anregungsfrequenzen. Mit einem aktiven bzw. semiaktiven Fahrwerk lassen sich diese Probleme minimieren [Ill14].

Das für die modellbasierte Auslegung eines aktiven bzw. semiaktiven Fahrwerks benötigte Dynamikmodell ist unter [Kru09] auf Basis der Modellbeschreibungs-

Tabelle 5-1: Technische Daten vom Wiesel 1

Bedeutung	Name/Wert	Einheit
Baujahr	1990-1992	[-]
Motor	5-Zylinder-VW-Diesel	[-]
Hubraum	2000	cm^3
Leistung nach DIN	64 (87)	kW (Ps)
Leistungsgewicht	23 (31)	kW/t (Ps/t)
Höchstgeschwindigkeit	70 (44)	km/h (mph)
Federung	Drehstäbe	[-]
Länge über alles	3,545	m
Breite über alles	1,820	m
Höhe über alles	1,825	m
Gewicht	2,75	t

sprache Modelica [Mod09] unter Dymola [Dym13] erstellt (vgl. Bild 5-2) und mittels zur Verfügung stehender Referenzgrößen validiert worden. Auf die Entwicklung des Modells sowie auf die Besonderheiten zum Beispiel bei der Modellierung der Kettenkräfte wird an dieser Stelle nicht genauer eingegangen. Eine detaillierte Beschreibung ist unter [Kru09] zu finden. Die Validierung erfolgte durch *trial and error* und lieferte hinreichend genaue Ergebnisse, um mit der modellbasierten Auslegung des aktiven bzw. semiaktiven Fahrwerks fortzufahren. Dennoch soll das entstandene, bereits validierte Modell verwendet werden, um zu demonstrieren, dass das entwickelte Parameteridentifikations-Tool für die Modellvalidierung besser geeignet ist, damit in zukünftigen Projekten auf die zeitaufwändige Vorgehensweise mittels *trial and error* bei solch komplexen Modellen verzichtet werden kann.

Hierzu wird aus dem unter Dymola erstellten Modell eine FMU-Datei generiert und in das entwickelte Tool eingebunden. Für die Parameteridentifikation werden die Werte der bereits identifizierten Parameter im Modell variiert und wie zuvor beschrieben markiert. Bei den ausgewählten Parametern handelt es sich um einen Trägheitstensor Θ_{yy} der Aufbaumasse sowie die Federsteifigkeit $c_{Torsion}$ und die Dämpfung $d_{Torsion}$ der eingebauten Drehstabfedern (vgl. Bild 5-1). Über diese Parameter lagen keine genauen Werte vor. Um Referenzgrößen zu erhalten, hat man mehrere Messfahrten durchgeführt. An dieser Stelle wird exemplarisch die Messfahrt über drei parallel angeordnete Hindernisse gewählt, bei einer Geschwindigkeit von 10 km/h. Anordnung und Abmaße der Hindernisse sind in Bild 5-3 schematisch dargestellt.

Bild 5-4 zeigt exemplarisch die Einbindung des Wiesel-Modells als FMU in das entwickelte Parameteridentifikations-Tool. Die FMU wird als Model Exchange

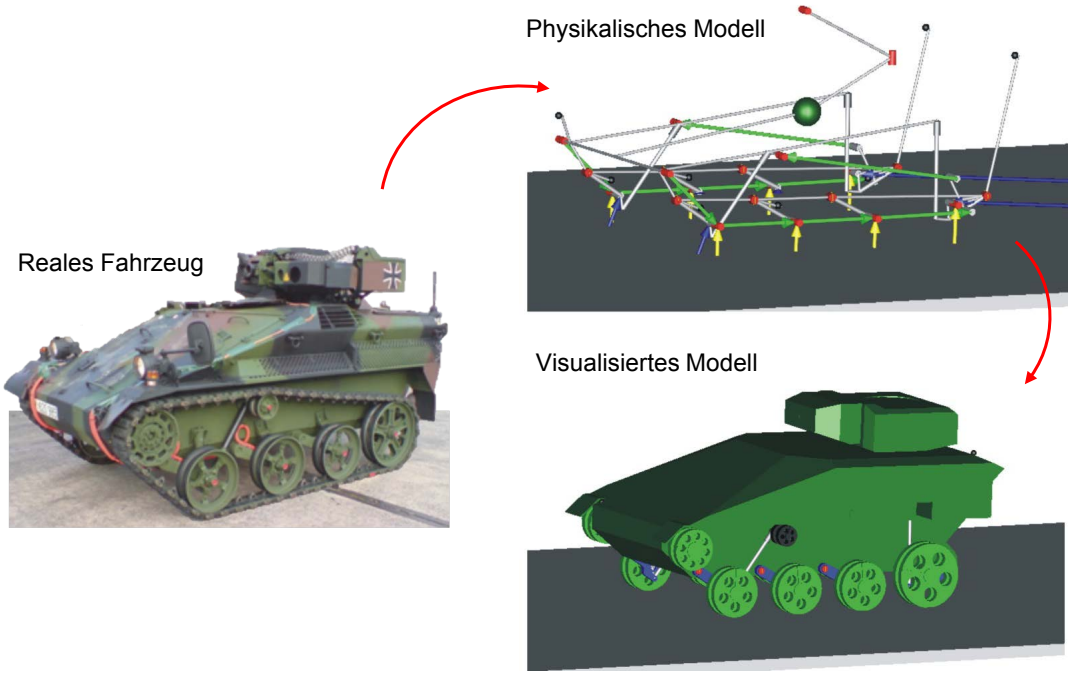

Bild 5-2: Abbildung des Wiesel-1-Fahrzeuges und des entsprechenden Mehrkörpersimulations-Modells [Kru09]

mit einer Simulationsschrittweite $t_{StepSize} = 0{,}001\,\text{sek}$ und einer Intervalllänge $t_{Int} = 0{,}01\,\text{sek}$ eingebunden.

Den Modellinformationen ist zu entnehmen, dass das Modell 68 Zustände, 5 markierte Parameter und 5 Ausgänge besitzt. Eingänge werden bei diesem Modell nicht benötigt, da die Anregung durch die Teststrecke, inklusive der erwähnten Hindernisse, als Teilmodell in dem Gesamtmodell integriert ist. Für das Nachstellen entsprechender Messszenarien können dem Modell die Werte der Startposition und der Geschwindigkeit als jeweilige Parameter übergeben werden.

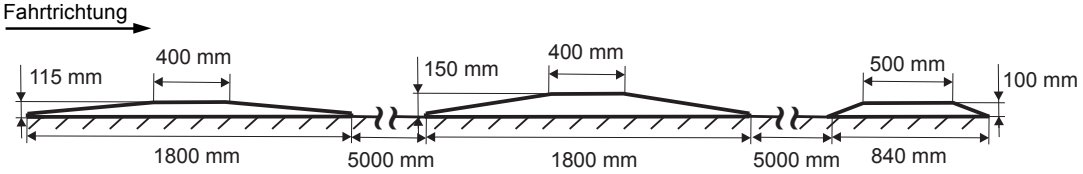

Bild 5-3: Schematische Darstellung der Anordnung und der Abmaße der verwendeten Hindernisse [Kru09]

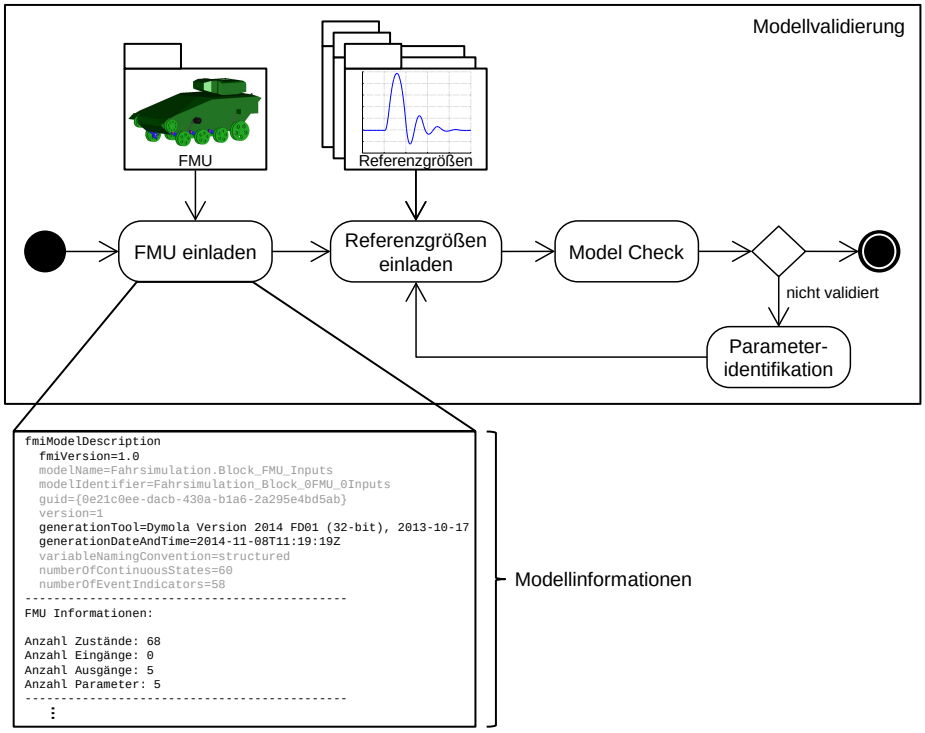

Bild 5-4: Schematische Darstellung des in das Parameteridentifikations-Tool eingebundenen Wiesel-Modells

Das Vorgehen erfolgt wie zuvor beschrieben mittels des Parameteridentifikations-Tools (vgl. Kapitel 4). Nachdem die FMU importiert ist, werden die entsprechenden Referenzgrößen geladen und mithilfe des `Model Check` mit den Ergebnissen des Modells verglichen. Bild 5-5 zeigt diesen Vergleich für die gemessenen und die simulierten Beschleunigungssignale des Fahrzeugs. Die Beschleunigungssensoren befinden sich vorn rechts und hinten links am Fahrzeugaufbau.

Wegen der Variation der bereits identifizierten Parameter spiegelt das Modell den Verlauf der Referenzgrößen nicht hinreichend genau wieder. Das Modell ist somit nicht mehr validiert, und es folgt der Bereich der Parameteridentifikation. Der Simulationsbereich wird von 0–12 sek gewählt. Damit sich das initiale Einschwingen des Fahrzeuges bei der Identifikation der Parameter nicht auf die Zielfunktion auswirkt, wird ein Identifikationsbereich von t_{Start}= 2 sek bis t_{End}= 12 sek für die in Bild 5-5 gezeigten Sensorsignale gewählt. Des Weiteren zeigt das Bild 5-5 die Ergebnisse der dazugehörigen Sensitivitätsanalyse. Nachdem sichergestellt ist, dass die drei ausgewählten Parameter sensitiv für den eingestellten Identifikationsbereich sind, startet die automatische Identifikation. Das hierzu benötigte Optimierungsverfahren kann, wie zuvor beschrieben, frei gewählt werden.

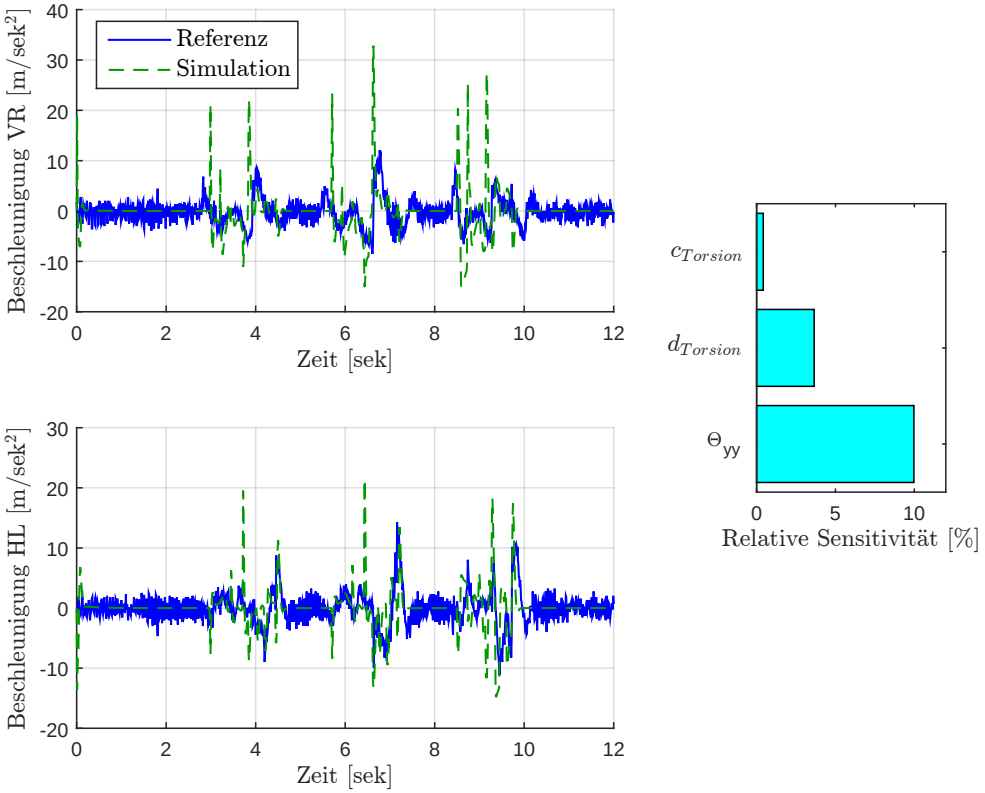

Bild 5-5: Vergleich der gemessenen und der simulierten Beschleunigungssignale des Aufbaus sowie die entsprechende relative Sensitivität der drei Identifikationsparameter des Wiesel-Modells

Bild 5-6 zeigt die gemessenen und die simulierten Beschleunigungssignale des Aufbaus vor und nach der Identifikation mittels des Optimierungsverfahren *PS* und des Solvers *GPS2N*. Die Identifikation der drei Parameter endet nach etwas mehr als einer Stunde. Bei den simulierten Signalen handelt es sich um die Ergebnisse mit variierten und neu identifizierten Parametern. Des Weiteren sind die Simulationsergebnisse des ursprünglich validierten Modells dargestellt, bevor die Parameter nachträglich variiert wurden. Die quadratische Fehlerfläche ist von $f_{init}(\vec{P})= 36\,726{,}88$ auf $f_{ident(neu)}(\vec{P})= 12\,904{,}98$ reduziert worden. Das entspricht einer Verbesserung um ca. 65 %. Erfolgt eine Simulation mit den bereits zuvor identifizierten Parametern, so beträgt der Wert der quadratischen Fehlerfläche $f_{ident(alt)}(\vec{P})= 18\,156{,}88$, was einer Verbesserung um ca. 50 % im Vergleich zu den variierten Parametern entspricht. Die Werte der verwendeten Parameter sind in Tabelle 5-2 aufgelistet.

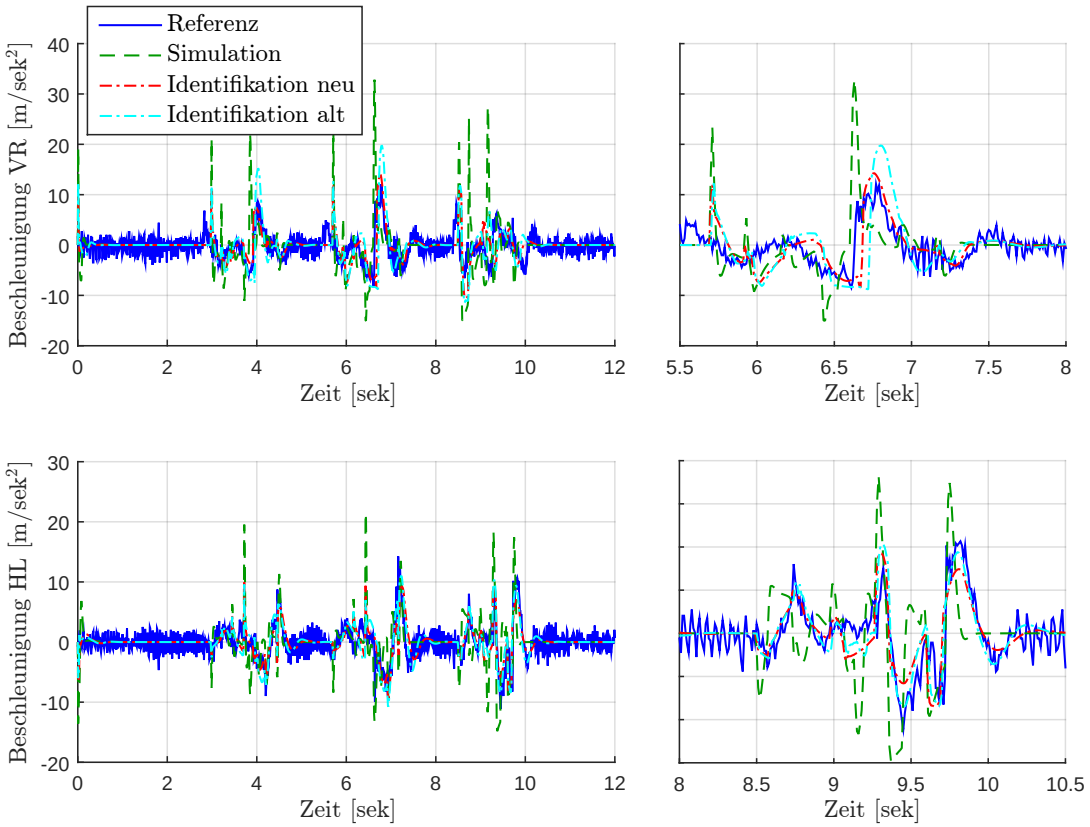

Bild 5-6: Vergleich der gemessenen (blau) und simulierten Beschleunigungssignale des Aufbaus

Die Ergebnisse des Wiesel-Modells konnten durch das Parameteridentifikations-Tool noch einmal verbessert werden. Vor allem die Einbindung des Modells in das Identifikations-Tool mit der automatischen Darstellung aller wichtigen Informationen sowie aller zuvor markierten Parameter ermöglicht eine schnelle, zuverlässige Parameteridentifikation. Des Weiteren besteht ein großer Vorteil in der einfachen Einbindung von Referenzgrößen. Die vorliegenden Messgrößen wurden bei dem damaligen Projekt bereits unter MATLAB entsprechend aufbereitet. Anschließend wurden diese in Dymola eingebunden, damit das Modell mittels *trial and error* identifiziert werden konnte. Dieser Aufwand entfällt bei der neu entwickelten Parameteridentifikations- und Modellvalidierungsmethodik, da sich das daraus entwickelte Identifikations-Tool ebenfalls unter MATLAB befindet und somit eine einfache Einbindung der Referenzgrößen ermöglicht wird. Es muss lediglich eine FMU des Modells erstellt werden, die sich anschließend in das Identifikations-Tool einbinden lässt. Für die Abbildung der komplexen und nichtlinearen Eigenschaften des Wiesel 1 im MKS-Modell besitzt dieses viele unbekannte

Tabelle 5-2: Unterschiedliche Werte der markierten Parameter und deren Auswirkungen auf die Fehlerfläche

Name	$c_{Torsion}$	$d_{Torsion}$	Θ_{yy}	$f(\vec{P})$
Startwert	3000	900	1400	36726,8844
ident. Wert alt	5607	100	1800	18156,8813
ident. Wert neu	4432,5	127,41	1986,1	12904,9751
Einheit	Nm/rad	Nms/rad	$\mathrm{kg\,m^2}$	

Größen. Es sei darauf hingewiesen, dass es sich bei den identifizierten Werten der markierten Parameter um optimale Werte für diese Referenzmessung handelt. Gezielte Messungen sind besser geeignet, um die Parameter sequentiell zu identifizieren; diese liegen zu diesem Zeitpunkt jedoch nicht vor. Für die anschließende Validierung des Modells ist die oben gezeigte Messung entsprechend gut geeignet. Letztlich bleibt das Vorgehen bei der Validierung bzw. bei der Parameteridentifikation identisch, unabhängig von den gewählten Referenzgrößen. Daher kann an dieser Stelle gesagt werden, dass die Übertragbarkeit der Modellvalidierungsmethodik für dieses Anwendungsbeispiel erfolgreich war.

5.2 Teigkneter

In diesem Abschnitt wird abschließend die Übertragbarkeit der erarbeiteten Parameteridentifikations- und Modellvalidierungsmethodik inklusive des entwickelten Identifikations-Tools auf ein Modell eines Teigkneters demonstriert. Bei dem Modell handelt es sich um ein unter Dymola entwickeltes Multidomänen-Modell. Es ist im Rahmen des Spitzenclusters Intelligente Technische Systeme OstWestfalenLippe (it's OWL) in dem Innovationsprojekt InoTeK (**In**telligenter und **o**ptimierter **Te**ig-**K**netprozess) mit der Firma WP Kemper GmbH erstellt worden. Bevor auf die Identifikation des Modells eingegangen wird, werden zunächst die Aufgaben eines Teigkneters sowie die aktuelle Problemlage und der daraus resultierende Handlungsbedarf vorgestellt.

Die Aufgabe von Knetmaschinen (vgl. Bild 5-7) besteht darin, Zutaten (Mehl, Wasser etc.) zu vermischen und zu kneten. Die Masse der zu mischenden Zutaten reicht von 25 kg bis 400 kg Teig für die Brot- und Brötchenproduktion. Dabei muss immer ein gleichmäßiger Teig, unabhängig vom Füllstand, gewährleistet werden, um den hohen Anforderungen der Lebensmittelindustrie gerecht zu werden. Hierzu werden zum einen der Teigbehälter und zum anderen der Knetarm gedreht. Die Steuerung des Kneters erfolgt rein zeitbasiert durch einen geschulten Bäcker, dessen Expertenwissen erforderlich ist. Im Rahmen des Spitzenclusterprojektes soll daraufhin unter anderem eine intelligente Informationsverarbeitung zur

Führung des Knetprozesses entwickelt werden, um den hohen Ansprüchen dauerhaft gerecht zu werden. Die Entwicklung dieser Informationsverarbeitung soll modellbasiert erfolgen. Das hierzu benötigte Modell des Kneters ist in [Kir13] beschrieben. Hierbei handelt es sich um eine erste Abbildung wesentlicher Effekte, die noch nicht validiert wurden. Zudem befindet sich die Struktur des Modells weiterhin in der Entwicklung.

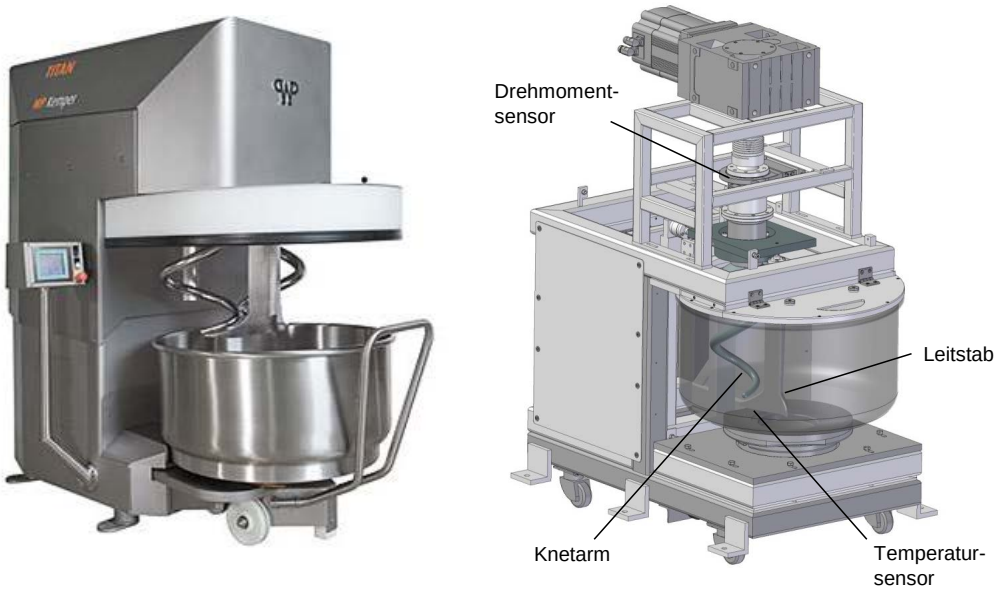

Bild 5-7: Abbildung eines Teigkneters (links real und rechts als Modell) [Tra14]

Dieser Abschnitt demonstriert, dass die entwickelte Parameteridentifikations- und Modellvalidierungsmethodik auch in einer frühen Phase der Modellbildung eingesetzt werden kann, um entsprechende Parameter zu identifizieren und gleichzeitig Rückschlüsse auf die getroffene Wahl der Modellierungstiefe sowie -komplexität zu erlauben. Die Identifikation des Knetermodells wird hierbei in enger Abstimmung mit dem jeweiligen Modellentwickler durchgeführt. Bei dem erstellten Modell handelt es sich um die Abbildung eines aufgebauten Labormusters zur Untersuchung von Teigeigenschaften. Durch das Labormuster können gezielte Messszenarien durchgeführt werden, die für die Parameteridentifikation benötigt werden. Des Weiteren können somit frühzeitig Rückschlüsse bzgl. der getroffenen Wahl der Modellierungstiefe und der -komplexität gewonnen werden. Um Informationen über die zeitlich stark veränderlichen Teigeigenschaften (Temperatur und Konsistenz) zu erhalten, hat man Sensoren für die Erfassung des Drehwinkels und des Drehmomentes des Knetwerkzeugs sowie Sensoren für die Temperaturerfassung des Teiges im Labormuster verbaut. Das Ziel der verbauten Sensoren besteht in der Identifizierung von Zusammenhängen und charakteristischen Merkmalen

von Teigeigenschaften. Für die modellbasierte Entwicklung gilt es zunächst, die jeweiligen Effekte im Modell abzubilden. Hierzu werden gezielte Messszenarien durchgeführt, anhand derer die entsprechenden Parameter im Modell mithilfe des vorgestellten Parameteridentifikations-Tools identifiziert werden.

Bild 5-8 zeigt schematisch die Einbindung des Kneter-Modells in das Parameteridentifikations-Tool. Das Modell besitzt 2 Eingänge, 2 Ausgänge, 15 Zustände und 9 markierte Identifikationsparameter. Bei den Eingängen handelt es sich zum einen um die Soll-Drehzahl des Knetwerkzeugs und zum anderen um die Umgebungstemperatur. Die Ausgänge sind die Teigtemperatur, die über einen Temperatursensor an dem Leitstab (siehe Bild 5-7) erfasst wird und das an dem Knetwerkzeug anliegende Drehmoment. Letzteres wird über einen Drehmomentmessflansch ermittelt. Dieser befindet sich zwischen dem Knetwerkzeug und der am Antriebsmotor befestigten Kupplung. Auf die physikalische Bedeutung der 9 Identifikationsparameter wird an dieser Stelle nicht genauer eingegangen. Sie werden im folgenden als Identifikationsparameter P_1 bis P_9 bezeichnet.

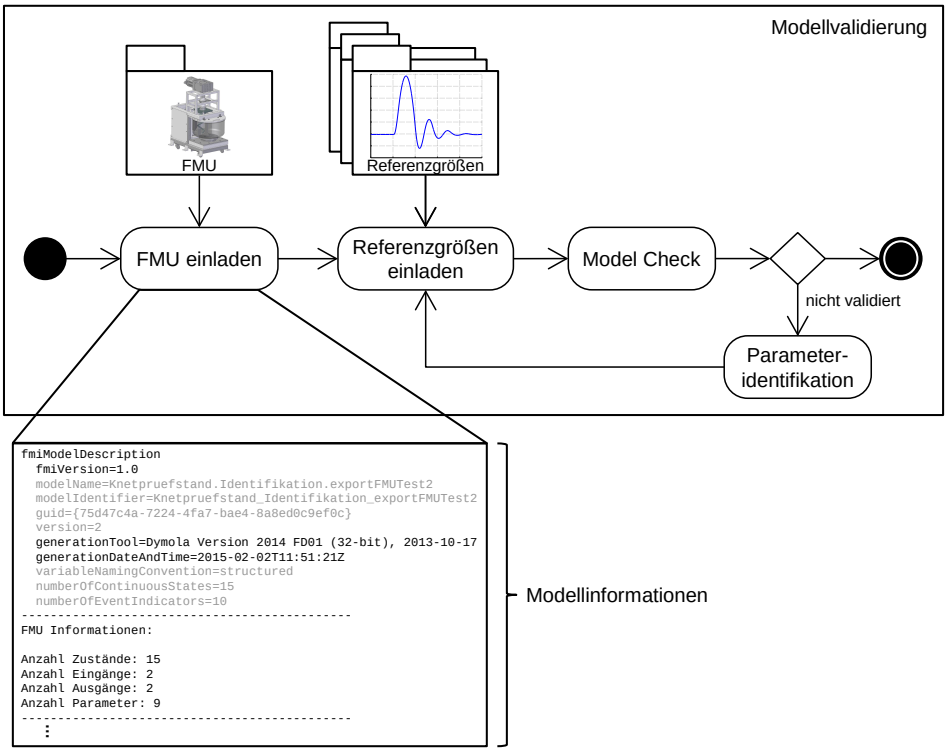

Bild 5-8: Schematische Darstellung des in das Parameteridentifikations-Tool eingebundenen Kneter-Modells

Um P_1 bis P_9 sequentiell zu identifizieren, lädt man das erste Messszenario ein. Bei diesem handelt es sich um eine Leerlaufmessung. Hierdurch können diejenigen

Parameter im Modell identifiziert werden, die nicht das Teigverhalten beschreiben. Nach dem Verbinden der Referenzgrößen mit den entsprechenden Ein- und Ausgängen des Modells erfolgt der `Model Check`. Bild 5-9 zeigt die daraus resultierenden Ergebnisse. Da sich bei diesem Messszenario kein Teig im Kneter befindet, wird das Temperatursignal, das die Teigtemperatur messen soll, an dieser Stelle nicht betrachtet.

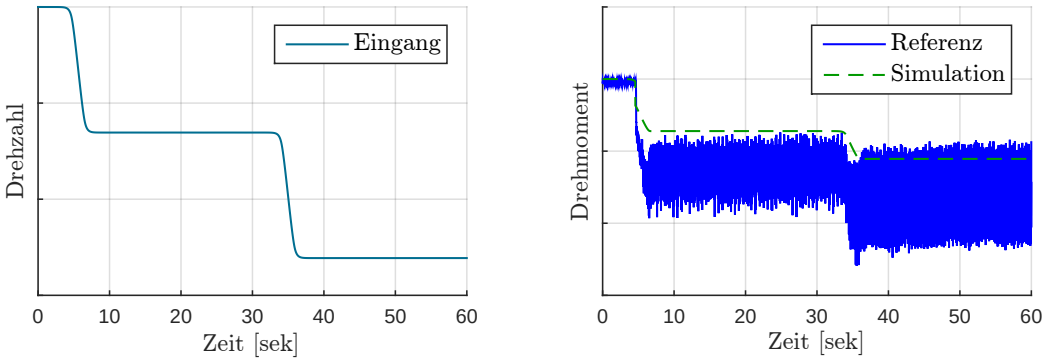

Bild 5-9: Gemessenes Signal der Drehzahl und des Drehmoments des ersten Messszenarios (Leerlaufmessung ohne Teig)

Das Drehmoment spiegelt bei diesem Messszenario die Reibung im System wieder. Tendenziell weist das Modell schon ein richtiges Verhalten bzgl. des simulierten Drehmomentes auf. Zur Minimierung der Abweichungen wird der Bereich der Parameteridentifikation im Identifikations-Tool verwendet (vgl. 5-8). Als Identifikationsbereich wird hierzu das Signal des gemessenen Drehmomentes über den gesamten Messbereich ausgewählt. In Bild 5-9 ist zu erkennen, dass das aufgezeichnete Signal des Drehmomentes verrauscht ist. Um dieses Rauschen für die Parameteridentifikation zu entfernen, wird die im Parameteridentifikations-Tool hinterlegte Auswertung der Zielfunktion um eine Signalfilterung erweitert (siehe Bild 5-10).

Bei den hinzugefügten Filtern handelt es sich jeweils um ein Verzögerungsglied 1. Ordnung (PT1-Glied). Da es sich hierbei um keine phasenfreie Filterung handelt, werden sowohl die Messwerte als auch die Simulationswerte gefiltert. Hierdurch wird eine Phasenverschiebung zwischen den Messwerten und den Simulationswerten verhindert. Die verwendete Eckfrequenz wird über den Parameter f_{TP} übergeben. Die somit berechnete Fehlerfläche $f(\vec{P}, t, f_{TP})$ der Zielfunktion ist abhängig von den Werten der Identifikationsparameter $\vec{P}$, dem ausgewählten Zeitintervall t sowie der gewählten Filtereckfrequenz f_{TP}.

Nach der Anpassung der Zielfunktion werden die Identifikationsparameter ausgewählt, bevor die automatische Parameteridentifikation durchgeführt wird. Für

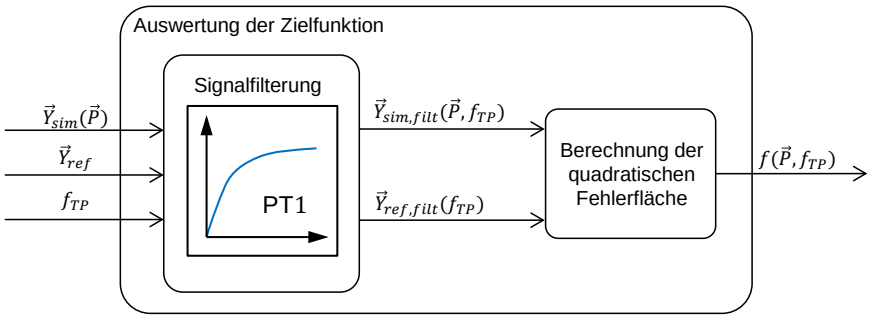

Bild 5-10: Schematische Darstellung der erweiterten Zielfunktionsbestimmung

die Auswahl der zu wählenden Identifikationsparameter kann die in dem Parameteridentifikations-Tool integrierte Sensitivitätsanalyse verwendet werden. Die entsprechenden Ergebnisse sind in Bild 5-11 dargestellt.

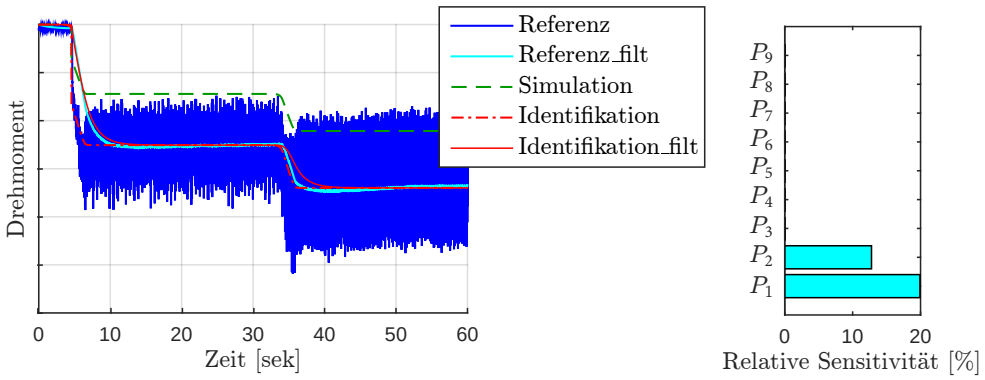

Bild 5-11: Darstellung der Ergebnisse der Parameteridentifikation des Kneter-Modells für das erste Messszenario inklusive der dazugehörigen sensitiven Parameter

Da sich bei diesem Messszenario kein Teig in dem Kneter befindet, sind nur die Identifikationsparameter P_1 und P_2 sensitiv. Alle anderen Identifikationsparameter beschreiben das Teigverhalten. Diese werden im Anschluss mithilfe weiterer Messszenarien identifiziert. Weiterhin zeigt Bild 5-11 die Ergebnisse der Parameteridentifikation. Es ist zu erkennen, dass die Fehlerfläche zwischen dem simulierten und dem gemessenen Drehmoment des Knetwerkzeuges durch die Parameteridentifikation deutlich reduziert werden konnte. Dies gilt sowohl für die gefilterten Signale, die für die Berechnung der quadratischen Fehlerfläche verwendet werden, als auch für die nichtgefilterten Signale, die ohne Aufbereitung gemessen bzw. simuliert werden.

Um das Modell bzgl. des Abbildens einer Leerlaufmessung zu validieren, wird der Bereich der Parameteridentifikation verlassen, und es werden neue Referenzgrößen für durchgeführte Leerlaufmessungen eingeladen. Durch einen anschließenden `Model Check` kann eine Aussage getroffen werden, ob das Modell für Leerlaufmessungen validiert ist oder ob durch die Parameteridentifikation lediglich eine Messung nachgestellt werden kann. Bild 5-12 zeigt verschiedene Vergleiche von aufgezeichneten Drehmomentsignalen bei Leerlaufmessungen sowie die dazugehörigen Simulationsergebnisse, ohne dass die zuvor identifizierten Parameter verändert wurden.

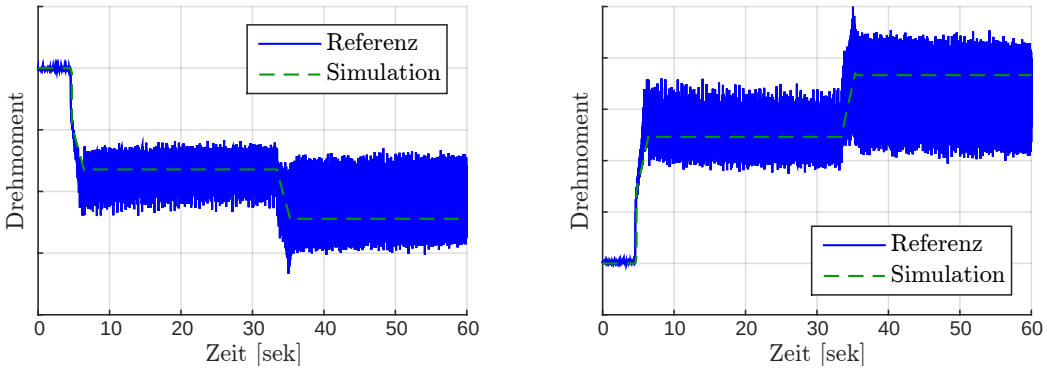

Bild 5-12: Vergleich unterschiedlich durchgeführter Leerlaufmessungen zur Modellvalidierung

Die Ergebnisse stimmen hinreichend genau überein, und das Modell ist für das Abbilden des Drehmomentes bei Leerlaufversuchen validiert. Das somit bekannte Reibmoment im System bildet die Grundlage für die Bestimmung des anliegenden Teigmomentes, das wiederum entscheidend für die Berechnung der Teigtemperatur ist. Das Teigmoment wird nicht durch eine Parameteridentifikation ermittelt, sondern mithilfe von Referenzgrößen berechnet. Für genauere Informationen diesbezüglich sei auf [Oes18] verwiesen.

Für die Validierung des Modells hinsichtlich des Abbildens der Teigtemperatur wird das zweite Messszenario (unter Berücksichtigung des Teiges) durch das Parameteridentifikations-Tool eingeladen und entsprechend mit dem Modell verknüpft. Die Ergebnisse des `Model Check` sind in Bild 5-13 abgebildet und werden nachfolgend kurz erläutert.

Der Temperaturverlauf lässt sich in zwei Phasen einteilen: eine Mischphase und eine Knetphase. Während der Mischphase werden die verschiedenen Zutaten in den Kneter gegeben und miteinander vermischt. Durch die Hinzufügung vom Wasser, dessen Temperatur unterhalb der Umgebungstemperatur liegt, fällt das gemessene Temperatursignal ab, bis eine gleichmäßige Vermischung der Zutaten

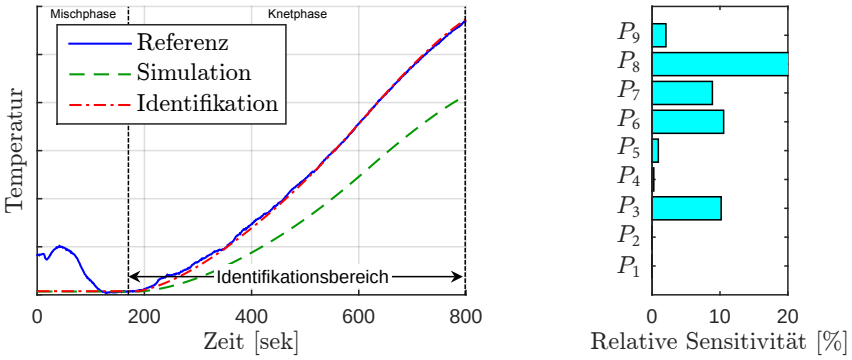

Bild 5-13: Darstellung der Ergebnisse der Parameteridentifikation des Kneter-Modells für das zweite Messszenario inklusive der dazugehörigen sensitiven Parameter

erfolgt. Dieser Effekt ist im Modell nicht abgebildet und wird daher für die Modellvalidierung außer acht gelassen. In der anschließenden Knetphase wird dem Teig über das Knetwerkzeug Energie zugeführt, und die Temperatur steigt. Dies soll mit dem Modell abgebildet werden. Damit der Startwert der simulierten Temperatur mit dem Startwert der gemessenen Temperatur zu Beginn der Knetphase übereinstimmt, wird dieser durch eine Änderung des initialen Zustandsvektors entsprechend eingestellt. Bild 5-13 zeigt, dass die Abweichungen zwischen dem simulierten und dem gemessenen Temperatursignal zu groß sind, so dass an dieser Stelle kein validiertes Modell vorliegt. Um das zu ändern, wird erneut die Parameteridentifikation angewendet.

Der Identifikationsbereich wird in dem Parameteridentifikations-Tool mithilfe von t_{Start}= 182 sek und t_{End}= 800 sek auf die Knetphase begrenzt. Im Gegensatz zu der für das erste Messszenario angewendeten Parameteridentifikation ist eine Filterung der Signale an dieser Stelle nicht notwendig und wird daher nicht vorgenommen. Nach der Auswahl der noch nicht identifizierten Parameter P_3 bis P_9 startet die automatische Identifikation mittels des ausgewählten Optimierungsverfahrens *PS* und des Solvers *GPS2N*. Nach erfolgreicher Optimierung der Parameter stimmt der simulierte Temperaturvelauf des Teiges sehr gut mit der Messgröße überein (vgl. Bild 5-13), und die Parameteridentifikation endet.

Für eine anschließende Modellvalidierung werden weitere Messszenarien eingeladen und mittels des `Model Check` mit dem Modell verglichen. Bild 5-14 zeigt exemplarisch einen solchen Vergleich. Die jeweils initiale Starttemperatur des Modells wurde, wie zuvor beschrieben, vor der Simulation entsprechend eingestellt. Wegen der hinreichenden Übereinstimmungen der Ergebnisse ist das Modell für das Abbilden der Teigtemperatur während des Knetprozesses validiert. Die Über-

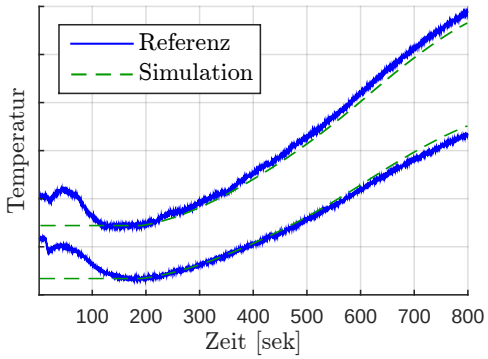

Bild 5-14: Vergleich unterschiedlich durchgeführter Messungen mit Teig zur Modellvalidierung

tragbarkeit der Parameteridentifikations- und Modellvalidierungsmethodik unter Verwendung des Parameteridentikations-Tools endet an dieser Stelle. Ein weiteres Anwendungsbeispiel für die Übertragbarkeit zeigt [KWK+16].

6 Resümee und Perspektiven

In dieser Arbeit wurde der modellbasierte Entwurf mechatronischer Systeme, in Anlehnung an die VDI-Richtlinie 2206 [VDI04], angewendet und signifikant erweitert, um die Lücke zwischen der steigenden Komplexität mechatronischer Systeme und der Leistungsfähigkeit der Entwicklungsmethoden zu verkleinern bzw. zu schließen (vgl. Bild 1-1 S. 1). Ein besonderes Merkmal dieser Vorgehensmethodik ist die ganzheitliche Betrachtung des Systems, die man fachgebietsübergreifend und funktionsorientiert gestaltet hat, um innovative Prinziplösungen zu erarbeiten und um auftretende Wechselwirkungen zwischen den Systemelementen der unterschiedlichen Fachdisziplinen frühestmöglich erkennen und berücksichtigen zu können. Besonders in Kombination mit der Wiederverwendung von Lösungswissen, z. B. mithilfe moderner Entwicklungswerkzeuge wie des Semantic Web, bietet die vorgestellte Entwurfsmethodik großes Innovationspotenzial für Erstellung und Ausarbeitung neuer Prinziplösungen für ein zu untersuchendes System.

Die Notwendigkeit der entworfenen Methodik und ihre Anwendung wurden an einem Praxisbeispiel aus der Industrie aufgezeigt. Hierbei sind anhand der in der Systemkonzipierung angewendeten Spezifikationstechnik zur Beschreibung der Prinziplösung eines mechatronischen Systems verschiedenste Prinziplösungen eines alternativen Heizprinzips für ein Umflut-Waschverfahren erarbeitet worden, wobei hier exemplarisch auf das Prinzip des Durchlauferhitzers eingegangen wurde. In der anschließenden Konkretisierung sind detaillierte, validierte Modelle der Strecke sowie der Steuerung entstanden. Durch die Integration der Teilmodelle zu einem Gesamtmodell können somit Aussagen über das Wassermanagement und die Energieverteilung im System zu den einzelnen Prozessphasen getroffen werden. Des Weiteren können die Modelle in Anlehnung an das V-Modell [VDI04] für den weiteren Entwicklungsprozess mechatronischer Systeme genutzt werden.

Ein besonderer Fokus lag außerdem auf dem Bereich der Validierung. Es wurde eine Parameteridentifikations- und Modellvalidierungsmethodik erarbeitet, die eine teilautomatisierte Parameteridentifikation für die Validierung von komplexen, nichtlinearen Multidomänen-Modellen ermöglicht, ohne dass detailliertes Expertenwissen über das zu identifizierende Modell oder das anzuwendende Optimierungsverfahren vorliegen muss. Das hierzu entstandene Parameteridentifikations-Tool, bestehend aus einer im Verbund mit der Dissertation von Schweers [Sch17] entwickelten FMU-Schnittstelle sowie einer MATLAB-Identifikationsumgebung, zeichnet sich besonders durch die leichte Einbindung von Dynamikmodellen aus, unabhängig von deren Modellentwicklungslandschaften, wodurch ein Höchstmaß an Flexibilität gegeben ist. Dies bietet einen besonderen Vorteil gegenüber anderen Ansätzen, Dynamikmodelle mittels Optimierungsverfahren zu identifi-

zieren [KOG03]. Ein weiteres besonderes Merkmal besteht in der Identifikationsumgebung, mit deren Hilfe komplexe, nichtlineare Multidomänen-Modelle mittels etablierter Verfahren teilautomatisiert identifiziert werden können. Hierdurch wird die Anwendung für einen großen Bereich der modellbasierten Entwicklung ermöglicht. Gezeigt wurde dies anhand der Übertragbarkeit der Methodik auf exemplarisch ausgewählte Dynamikmodelle, die sich jeweils in unterschiedlichen Entwicklungszuständen befinden. Bild 6-1 zeigt schematisch den Durchlauf mehrerer Makrozyklen mit zunehmender Produktreife. Es deutet zudem an, in welchen Phasen des Entwicklungszyklus das entwickelte Tool zur Modellvalidierung und zur Parameteridentifikation zum Einsatz kommen kann.

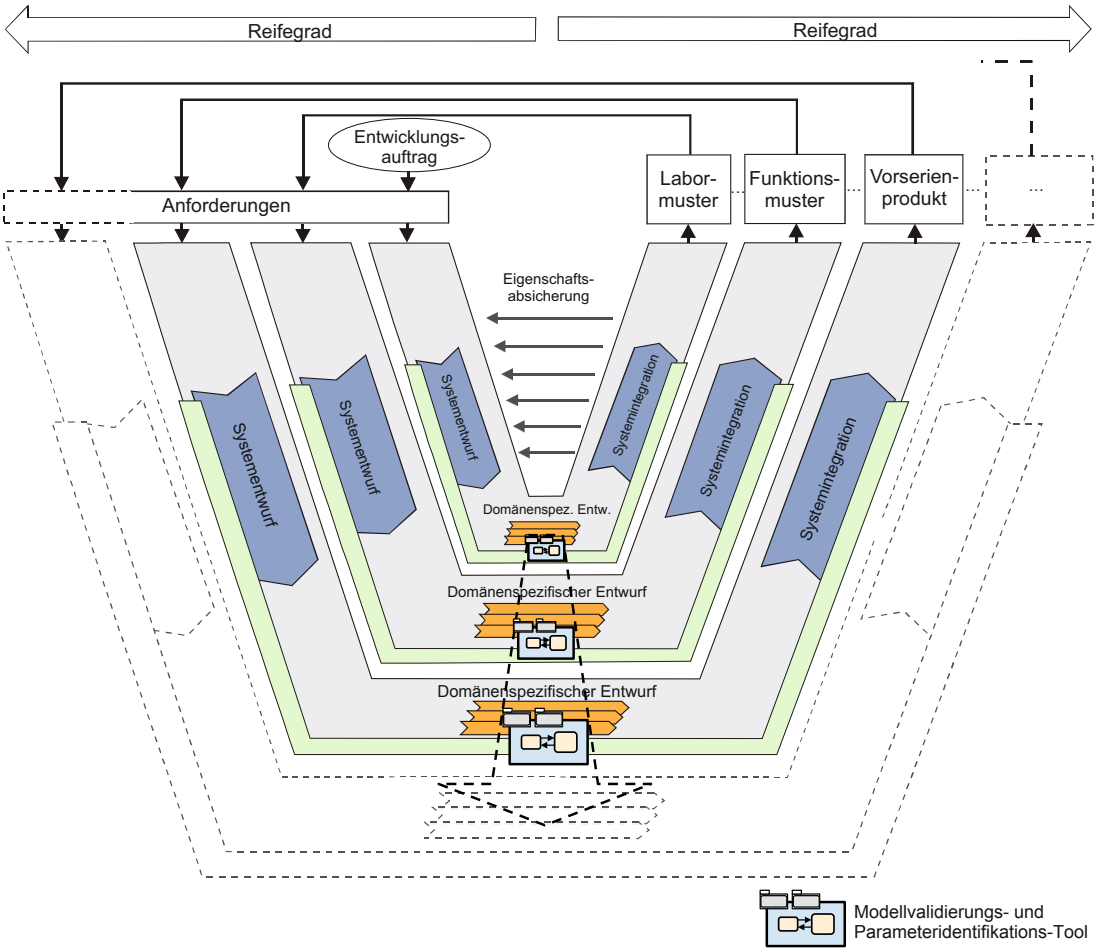

Bild 6-1: Durchlauf mehrerer Makrozyklen mit zunehmender Produktreife und Einordnung des entwickelten Modellvalidierungs- und Parameteridentifikations-Tools (in Anlehnung an [VDI04])

Neben dem Einsatz in einem großen Anwendungsgebiet ermöglicht das Parameteridentifikations-Tool durch die Hinzunahme einer graphischen Benutzeroberfläche zudem ein Höchstmaß an Automatisierung für die durchzuführenden Schritte und legt zudem eine Reihenfolge fest. Dem Benutzer wird somit die Anwendung der hier erarbeiteten Parameteridentifikations- und Modellvalidierungsmethodik erleichtert. Zudem können so Fehler vermieden und Zeit gespart werden. Bild 6-2 zeigt exemplarisch einige der entstandenen graphischen Benutzeroberflächen für die Parameteridentifikations- und die Modellvalidierungsmethodik. Die hinterlegten skriptbasierten Funktionen sind hierbei nahezu beliebig erweiterbar, was durch die Einbindung der Filterfunktion in Abschnitt 5.2 angedeutet wurde.

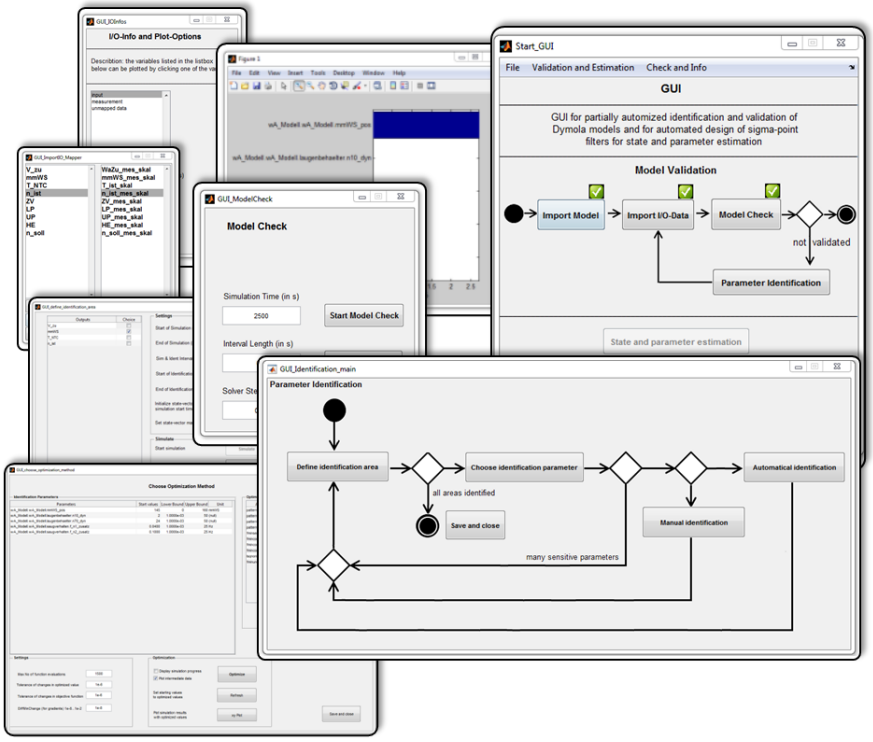

Bild 6-2: Screenshots diverser GUI-Ebenen des Parameteridentifikations-Tools

Durch die Möglichkeiten, die sich durch das FMU-Interface in Kombination mit der MATLAB-Umgebung ergeben, weist das entstandene Tool großes Potenzial auch für weitere regelungstechnische Funktionen auf. Als Beispiel sei hier die Auslegung von echtzeitfähigen Zustands- und Parameterschätzern an Black-Box-Systemen mittels Sigma-Punkte-Kalman-Filtern sowie deren Implementierung auf einer Echtzeithardware zu nennen, was z. T. in [SKT13, SKF+13, Sch17] gezeigt wurde.

Um das Potenzial der hier erarbeiteten Methodik zur teilautomatisierten Parameteridentifikation für die Validierung von Dynamikmodellen in dem modellbasierten Entwurf mechatronischer Systeme noch besser ausnutzen zu können, sollten in künftigen Projekten alternative IT-Werkzeuge für die Bearbeitung der CONSENS-Modelle während der Systemkonzipierung untersucht werden. Das hier verwendete IT-Werkzeug *Mechatronic Modeller* (vgl. Abschnitt 3.3) beinhaltet derzeit lediglich eine prototypische Umsetzung der eingebundenen CONSENS-Modelle [GTS+14]. Als vielversprechend seien hier *Enterprise Architect*, *Papyros* sowie die *3DEXPERIENCE-Plattform* von Dassault zu nennen [IKD+13, PHM14, KDH+13, KKA+13]. Des Weiteren sollte die Weiterentwicklung des Semantic WEB speziell für die Wiederverwendung von Lösungswissen berücksichtigt werden. Wenn mehr oder überhaupt Modelle von validierten Lösungselementen zur Verfügung stünden, könnten Neuentwicklungen noch effizienter durch Simulationen analysiert werden, bevor aufwändige Validierungen am Prüfstand notwendig werden. Die Idee des Wissensspeichers ist an dieser Stelle sehr vielversprechend. So kann z. B. das hier entwickelte Parameteridentifikations-Tool verwendet werden, um neue Lösungselemente zu validieren, bevor diese in einem Wissensspeicher abgelegt werden [Oes18].

Für die künftige Verwendung der standardisierten FMU-Schnittstelle sollte das FMU-Interface im nächsten Schritt auf die Einbindung von FMUs 2.0 erweitert und der Nachweis für die Einbindung von FMUs aus anderen Modellierungstools als Dymola erbracht werden (vgl. Abschnitt 4.2.2). Durch eine Erweiterung der auszuwählenden Solver kann das FMU-Interface zudem noch besser für unterschiedliche Modelle verwendet werden. Des Weiteren können die gezeigten Vorteile der Modellvalidierungs- und der Parameteridentifikationsmethodik durch eine Erweiterung der eingebundenen Sensitivitätsanalyse, welche die Querverbindungen der Identifikationsparameter untereinander berücksichtigt, sowie durch die Integration weiterer unter MATLAB zur Verfügung stehender Optimierungsverfahren noch erhöht werden. Um das im Verbund mit Schweers [Sch17] entwickelte Tool langfristig für die Entwicklung mechatronischer Systeme einsetzbar zu machen, wird ein Name für das Tool festgelegt. Da es sich bei den derzeitigen Funktionen um eine teilautomatisierte Parameteridentifikation und um eine Auslegung von echtzeitfähigen Zustands- und Parameterschätzern handelt, lautet dieser *PIET* (**P**arameter **I**dentification and **E**stimation **T**ool).

7 Literaturverzeichnis

[ABR+11] Angermann, A.; Beuschel, M.; Rau, M.; Wohlfahrt, U.: *MATLAB-Simulink-Stateflow. Grundlagen, Toolboxen, Beispiele.* 6. Auflage. München: Oldenbourg Wissenschaftsverlag GmbH, 2011.

[Ada14] Adamski, D.: *Simulation in der Fahrwerktechnik.* Wiesbaden: Springer Vieweg, 2014.

[Alt12] Alt, O.: *Modellbasierte Systementwicklung mit SysML.* München: Carl Hanser Verlag, 2012.

[BBB+14] Brian, B. M.; Bauman, L. E.; Bohnhoff, W. J.; Dalbey, K. R.; Eddy, J. P.; Ebeida, M. S.; Eldred, M. S.; Hough, P. D.; Hu, K. T.; Jakeman, J. D.; Swiler, L. P.; Stephens, J. A.; Vigil, D. M.: *Dakota Reference Manual.* `https://dakota.sandia.gov/sites/default/files/docs/5.4/html-ref/MethodCommands.html`. Version: 2014.

[BBF+93] Bobeth, W.; Berger, W.; Faulstich, H.; Fischer, P.; Heger, A.; Jacobasch, H.-J.; Mally, A.; Mikut, I.: *Textile Faserstoffe: Beschaffenheit und Eigenschaften; mit 120 Tabellen.* Berlin: Springer-Verlag, 1993.

[BD93] Bröhle, A.-P.; Dröhschl, W.: *Das V-Modell – Der Standard für die Softwareentwicklung mit Praxisleitfaden.* München: Oldenbourg Verlag, 1993.

[BF99] Berners-Lee, T.; Fischetti, M.: *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor.* San Francisco: Harper, 1999.

[BGK+12] Bauer, F.; Gausemeier, J.; Köchling, D.; Oestersötebier, F.: Simulative Absicherung mechatronischer Systeme in der frühen Phase der Produktentstehung. In: *Tag des Systems Engineering.* Paderborn, 2012.

[BGK+13] Bauer, F.; Gausemeier, J.; Köchling, D.; Oestersötebier, F.: Approach for an Early Validation of Mechatronic Systems using Idealized Simulation Models within the Conceptual Design. In: *Smart Product Engineering - Proceedings of the 23rd CIRP Design Conference.* Bochum, 2013.

[BHL01] Berners-Lee, T.; Hendler, J.; Lassila, O.: *The Semantic Web.* Scientific American, 2001.

[BOA+11] Blochwitz, T.; Otter, M; Arnold, M; Bausch, C; Clauss, C; Elmqvist, H; Junghanns, A; Mauss, J; Monteiro, M; Neidhold, T u. a.: The functional mockup interface for tool independent exchange of simulation models. In: *Modelica'2011 Conference*. München, 2011.

[Bru14] Brunstein, F. L.: *Teilautomatisierte Parameteridentifikation zur Validierung von Multidomänenmodellen*, Studienarbeit, Universität Paderborn, 2014.

[CG02] Chatterjee, P. K.; Gupta, B. S.: In: *Absorbent technology. Elsevier Science*. Amsterdam Oxford, 2002.

[Cho13] Chombart, P.: MODELISAR Innovation Report. 2013. – Forschungsbericht

[Das13] Dassault Systèmes AB: *Dymola: Dynamic Modeling Laboratory – User Manual*. Lund, 2013.

[DS14] Drüke, S.; Scheering, C.: Software-in-the-Loop-Simulation mit NI VeriStand und Visual Studio unter Verwendung der Execution API. In: *Kongress Virtuelle Instrumente in der Praxis (VIP)*. München, 2014.

[Dym13] *Dymola – Multi-Engineering Modeling and Simulation*. `http://www.dymola.com/`. Version: 2013.

[Ehr09] Ehrlenspiel, K. (Hrsg.): *Integrierte Produktentwicklung – Denkabläufe, Methodeneinsatz, Zusammenarbeit*. München: Carl Hanser Verlag, 2009.

[Eyk74] Eykhoff, P.: *System identification: parameter and state estimation*. `https://katalog.ub.uni-paderborn.de/records/PAD_ALEPH000101021`. Version: 1974.

[FKS+13] Fast, V.; Kruse, D.; Schweers, C.; Gehrmann, T.; Trächtler, A.: Methoden zur Einbindung von Fehlersimulationen in die XiL-Techniken. In: *Kongress Virtuelle Instrumente in der Praxis (VIP)*. München, 2013.

[FMI12] MODELISAR Consortium: *Functional Mock-up Interface for Model Exchange*. `https://www.fmi-standard.org/`. Version: 2012.

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

[GDS+13] Gausemeier, J.; Dumitrescu, R.; Steffen, D.; Czaja, A.; Wiederkehr, O.; Tschirner, C.: Systems Engineering in der

industriellen Praxis. In: *Heinz Nixdorf Institut, Universität Paderborn, Lehrstuhl für Produktentstehung; Fraunhofer-Institut für Produktionstechnologie IPT - Projektgruppe Entwurfstechnik Mechatronik; Unity AG.* Paderborn, 2013.

[GEK01] Gausemeier, J.; Ebbesmeyer, P.; Kallmeyer, F.: *Produktinnovation: Strategische Planung und Entwicklung der Produkte von morgen.* München, Wien: Carl Hanser Verlag, 2001.

[Ger13] Gerking, C.: *Transparent Uppaal-based Verification of MechatronicUML Models*, Masterarbeit, Universität Paderborn, 2013.

[Geu98] Geuther, A.: *Wie man eine weiße Weste bekommt!* Bamberg: Gastvortrag, Otto-Friedrich-Universität Bamberg, Institut für Didaktik für Chemie, 1998.

[GFD+08] Gausemeier, Jürgen; Frank, Ursula; Donoth, Jörg; Kahl, Sascha: Spezifikationstechnik zur Beschreibung der Prinziplösung selbstoptimierender Systeme des Maschinenbaus. In: *Konstruktion*, 2008.

[GIT11] Gense, A.; Illg, I.; Trächtler, A.: Model-Based Design of the Controlled Suspension Of Tracked Vehicles. In: *27th Applied Vehicle Technology Panel.* Sofia, 2011.

[GLL12] Gausemeier, J.; Lanza, G.; Lindemann, U.: *Produkte und Produktionssysteme integrativ konzipieren – Modellbildung und Analyse in der frühen Phase der Produktentstehung.* München: Carl Hanser Verlag, 2012.

[GSA+11] Gausemeier, J.; Schäfer, W.; Anacker, H.; Bauer, F.; Dziwok, S.: Einsatz semantischer Technologien im Entwurf mechatronischer Systeme. In: *8. Paderborner Workshop Entwurf mechatronischer Systeme* HNI-Verlagsschriftenreihe, Band 294, Paderborn, 2011.

[GTS+14] Gausemeier, J.; Trächtler, A.; Schäfer, W.; Anacker, H.; Bauer, F.; Borcherding, H.; Dziwok, S.; Frank, U.; Herden, R.; Hoppe, G.; Just, V.; Kiele-Dunsche, M.; Kruse, D.; Oestersötebier, F.; Papenfort, J.; Pohlmann, U.; Reddehase, H.; Rieke, J.; Schierbaum, T.; Seifert, L.; Stichweh, H.; Teichrieb, H.; Wagner, R.; Wessels, S.: *Semantische Technologien im Entwurf mechatronischer Systeme: Effektiver Austausch von Lösungswissen in Branchenwertschöpfungsketten.* München: Carl Hanser Verlag, 2014.

[Har87] Harel, D.: Statecharts: A visual formalism for complex systems. In: *Science of computer programming* 8, 3 (1987), S. 231–274.

[Har10] Harchenko, J.: *Mechatronischer Entwurf eines neuartigen aktiven Fahrzeugfederungssystems für PKW unter Verwendung einer reversierbaren Flügelzellenpumpe*, Dissertation, Universität Paderborn, 2010.

[IKD+13] Iwanek, P.; Kaiser, L.; Dumitrescu, R.; Nyssen, A.; Maurer, M.; Schulze, S.-O.: Fachdisziplinübergreifende Systemmodellierung mechatronischer Systeme mit SysML und CONSENS. In: *Tag des Systems Engineering 2013 (TdSE 2013)*. München: Carl Hanser Verlag, 2013, S. 337–346.

[Ill14] Illg, I.: *Mechatronischer Entwurf und Erprobung einer regelbaren Federung für ein leichtes geländegängiges Kettenfahrzeug*, Dissertation, Universität Paderborn, 2014.

[Ise99] Isermann, R.: *Mechatronische Systeme – Grundlagen*. 1. Auflage. Berlin: Springer-Verlag, 1999.

[Ise08] Isermann, R.: *Mechatronische Systeme – Grundlagen*. 2. Auflage. Berlin: Springer-Verlag, 2008.

[ISO05] Norm ISO/IEC 19501:2005 2005. *Information technology – Open Distributed Processing – Unified Modeling Language (UML) Version 1.4.2.*

[Jan09] Janschek, K.: *Systementwurf mechatronischer Systeme: Methoden – Modelle – Konzepte*. Berlin, Heidelberg: Springer, 2009.

[Jus13] Just, V.: *Modellbasierter Entwurf mechatronischer Systeme*. Skript zur Vorlesung, Heinz Nixdorf Institut, Universität Paderborn, Fakultät Maschinenbau, Paderborn, 2013.

[KDH+13] Kaiser, L.; Dumitrescu, R.; Holtmann, J.; Meyer, M.: Automatic Verification of Modeling Rules in Systems Engineering for Mechatronic Systems. In: *Proceedings of the ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*. New York, NY: American Society of Mechanical Engineers (ASME), 2013.

[KFS+12] Kruse, D.; Fast, V.; Schweers, C.; Trächtler, A.: Einheitliche Testumgebung für MiL und RCP mittels NI VeriStand am Beispiel eines Waschautomaten. In: *Kongress Virtuelle Instrumente in der Praxis (VIP)*. München, 2012.

[Kir13] Kirchhoff, S.: *Modellierung eines Teigkneters in Dymola/Modelica*, Bachelorarbeit, Universität Paderborn, 2013.

[KKA+13] Kleiner, S.; Kramer, C.; Abramovici, M.; Stark, R.: *Model Based Design with Systems Engineering Based on RFLP Using V6.* Berlin, Heidelberg: Springer-Verlag, 2013.

[Klu02] Kluge, F.: *Etymologisches Wörterbuch der deutschen Sprache.* 24. Auflage. Bearbeitet v. Elmar Seebold, Berlin, 2002.

[KM02] Kiessling, A.; Matthes, M.: *Textil-Fachwörterbuch.* Berlin: Schiele und Söhne Fachverlag, 2002.

[KOG03] Klotzbach, S.; Oedekoven, S.; Grassmann, O.: Optimierung im mechatronischen Entwicklungsprozess. In: *Tagungsband VDI Mechatronik.* Fulda, 2003.

[Kru09] Kruse, D.: *Modellierung der Vertikaldynamik eines Kettenfahrzeugs*, Studienarbeit, Universität Paderborn, 2009.

[KST14] Kruse, D.; Schweers, C.; Trächtler, A.: Methodology for a partly automated parameter identification for the validation of multi-domain models. In: *ASME International Mechanical Engineering Congress and Exposition*, Montreal, 2014.

[KTH13] Kruse, D.; Trächtler, A.; Herden, R.: Modellbasierte Entwicklung eines neuartigen Heizverfahrens für Waschautomaten. In: *9. Paderborner Workshop Entwurf Mechatronischer Systeme (EMS)*, Band 310, HNI-Verlagsschriftenreihe, Paderborn, 2013.

[Kun11] Kunath, J.: Optimierung mit Matlab / Brandenburgische Technische Universität Cottbus. 2011. – Forschungsbericht

[KWK+16] Kruse, D.; Warkentin, A.; Krüger, M.; Trächtler, A.; Rackow, S.: Multidomänenmodell zur Optimierung der Hydraulik eines Raupenlaufwerks für Landmaschinen. In: *Proc. 4. Internationales Commercial Vehicle Technology Symposium*, Kaiserslautern, 2016.

[LD10] Le Digabel, S.: Algorithm xxx: NOMAD: Nonlinear Optimization with the MADS algorithm, 2010.

[Leo73] Leonhard, W.: *Statistische Analyse linearer Regelsysteme.* `https://katalog.ub.uni-paderborn.de/records/PAD_ALEPH000114375`. Version: 1973.

[Lew98] Lewis, R. M.: Why Pattern Search Works / Institute for Computer Application in Science and Engineering (ICASE). Hampton, VA: NASA, 1998. – Forschungsbericht

[Löf16] Löffler, A.: *Entwicklung einer modellbasierten In-the-Loop-Testumgebung für Waschautomaten*, Dissertation, Universität Paderborn, 2016.

[Lju87] Ljung, L.: *System Identification: Theory for the user*. Englerwood Cliffs, NJ: Prentice-Hall, 1987.

[Lju11] Ljung, L.: Approaches to identification of nonlinear systems. Linköping University, 2011. – Forschungsbericht

[LKS+11] Löffler, A.; Koert, D.; Schweers, C.; Trächtler, A.: Einführung in die modellbasierte Entwicklung im Bereich der Haushaltsgerätetechnik. In: *Tagung Mechatronik*. Dresden, 2011.

[Loc] Lochbichler, M.: *Systematische Wahl einer Modellierungstiefe im Entwurfsprozess mechatronischer Systeme*, Dissertation (noch nicht erschienen), Universität Paderborn.

[LS86] Ljung, L.; Söderström, T.: *Theory and practice of recursive identification*. `https://katalog.ub.uni-paderborn.de/records/PAD_ALEPH000503907`. Version: 3. Auflage, 1986.

[LSB+12] Lochbichler, M.; Schmuedderrich, T.; Brökelmann, J.; Trächtler, A.: Methodology for Selecting the Modeling Depth of Object-Oriented Behavioral Models. In: *World Academy of Science, Engineering and Technology*. Zürich, 2012, S. 327–331.

[LSK+11] Löffler, A.; Schweers, C.; Kruse, D.; Fast, V.; Trächtler, A.: Multidomänen-Modell eines Waschvollautomaten für einen Hardware-in-the-Loop-Prüfstand. In: *8. Paderborner Workshop Entwurf mechatronischer Systeme (EMS)*, HNI-Verlagsschriftenreihe, Paderborn, 2011.

[Mar63] Marquardt, D. W.: An algorithm for least-squares estimation of nonlinear parameters. In: *Journal of the Society for Industrial & Applied Mathematics*, 1963, S. 431–441.

[Mar09] Marinescu, D. C.: Engineering Analysis ENG 3420, University of Central Florida, 2009.

[Mat14] Mathworks: *Optimization Toolbox - Users Guide*. Natrick, MA: The Mathworks, Inc., 2014.

[Mod09] Modelica Association: *Modelica – A Unified Object-Oriented Language for Physical Systems Modeling – Language Specification*. Linköping, 2009.

[Mor95] Moreira, D. A.: *Agents: A Distributed Client/Server System for Leaf Cell Generation*, Dissertation, University of Kent at Canterbury, 1995.

[NM65] Nelder, J.; Mead, R.: A simplex method for function minimization, Oxford Journal, 1965.

[NW99] Nocedal, J.; Wright, S. J.: *Numerical optimization.* New York, Berlin, Heidelberg: Springer-Verlag, 1999.

[Nye06] Nyenhuis, M.: *Strukturierter mechatronischer Entwurf einer SbW-Lenkung*, Dissertation, Universität Paderborn, 2006.

[Oes18] Oestersötebier, F.: *Modellbasierter Entwurf intelligenter mechatronischer Systeme mithilfe semantischer Technologien*, Dissertation, Universität Paderborn, 2018.

[Ost11] Ostrowski, M. W.: *Ingenieurhydrologie I.* Skript zur Vorlesung, Technische Universität Darmstadt, Institut für Wasserbau und Wasserwirtschaft, Fachgebiet Ingenieurhydrologie und Wasserbewirtschaftung, Darmstadt, 2011.

[Ost13] Ostrowski, M. W.: *Ingenieurhydrologie II.* Skript zur Vorlesung, Technische Universität Darmstadt, Institut für Wasserbau und Wasserwirtschaft, Fachgebiet Ingenieurhydrologie und Wasserbewirtschaftung, Darmstadt, 2013.

[PB97] Pahl, G.; Beitz, W.: *Konstruktionslehre: Methoden und Anwendungen.* Berlin, Heidelberg: Springer-Verlag, 1997.

[PC14] Pavlov, I.; Clark, J.: *FMU SDK 1.0.2.* Online. `https://resources.qtronic.de/fmusdk/FmuSdk_reference.html`. Version: 2014.

[PHM14] Priesterjahn, C.; Holtmann, J.; Meyer, M.: Smarte Entwicklung für smarte Systeme: Softwareentwicklung im Kontext des Gesamtsystems. In: *Tagungsband Embedded Software Engineering Kongress 2014.* Sindelfingen, 2014, S. 619–627.

[Rob94] Robinson, S.: *Successful simulation – a practical approach to simulation projects.* Maidenhead: McGraw-Hill, 1994.

[Rod06] Roddeck, W.: *Einführung in die Mechatronik.* 3. Auflage. Wiesbaden: B.G. Teubner Verlag / GWV Fachverlage, 2006.

[Sal93] Salt, J. D.: Simulation Should Be Easy and Fun! In: *Proceedings of the 25th Conference on Winter Simulation.* New York, NY, 1993, S. 1–5.

[Sch74] Scheidegger, A. E.: *The physics of flow through porous media.* Toronto: University of Toronto Press, 1974.

[Sch82] Schubert, H.: *Kapillarität in porösen Feststoffsystemen.* Berlin, Heidelberg: Springer-Verlag, 1982.

[Sch06] Scholz, P.: *Softwareentwicklung Eingebetteter Systeme.* Berlin: Springer-Verlag, 2006.

[Sch10] SCHRÖDER, D.: *Intelligente Verfahren - Identifikation und Regelung nichtlinearer Systeme.* Berlin, Heidelberg: Springer-Verlag, 2010.

[Sch12] SCHREIER, V.: *Entwicklung einer ereignisdiskreten Steuerung für den Waschvollautomaten unter MATLAB/Simulink-Stateflow*, Bachelorarbeit, Universität Paderborn, 2012.

[Sch13] SCHREIER, V.: *Modellierung des Saugverhaltens unterschiedlicher Beladungsarten im Waschautomaten*, Studienarbeit, Universität Paderborn, 2013.

[Sch17] SCHWEERS, C.: *Adaptive Sigma-Punkte-Filter-Auslegung zur Zustands- und Parameterschätzung an Black-Box-Modellen*, Dissertation, Universität Paderborn, 2017.

[SKF+13] SCHWEERS, C.; KRUSE, D.; FAST, V.; TRÄCHTLER, A.: Online-Zustands- und Parameterschätzung an Dymola-Modellen auf NI-Echtzeithardware. In: *Kongress Virtuelle Instrumente in der Praxis (VIP)*, 2013.

[SKO+13] SCHWEERS, C.; KRUSE, D.; OESTERWINTER, T.; TRÄCHTLER, A.: Automated Design of an Unscented Kalman Filter for State- and Parameter Estimation on unknown Models. In: *IEEE International Conference on Control, Automation, Robotics & Embedded Systems.* Jabalpur, 2013.

[SKT13] SCHWEERS, C.; KRUSE, D.; TRÄCHTLER, A.: Entwurf eines Unscented-Kalman Filters zur Zustands- und Parameterschätzung an Dymola-Modellen. In: *Tagungsband VDI Mechatronik.* Aachen, 2013.

[Str75] STROBEL, H.: *Experimentelle Systemanalyse.* `https://katalog.ub.uni-paderborn.de/records/PAD_ALEPH000142401`. Version: 1975.

[Toe02] TOEPPER, S.: *Die mechatronische Entwicklung des Parallelroboters TRIPLANAR*, Dissertation, Universität Paderborn, 2002.

[Tra14] TRAPHÖNER, P.: *Sensorauswertung zur Erkennung des Teigzustands in einem Teigkneter*, Studienarbeit, Universität Paderborn, 2014.

[VDI04] VDI-RICHTLINIE 2206: *Entwicklungsmethodik für mechatronische Systeme.* Beuth Verlag, Berlin, 2004.

[Wal95] WALLASCHEK, J.: Modellierung und Simulation als Beitrag zur Verkürzung der Entwicklungszeiten mechatronischer Produkte. In: *VDI Reports*, 1995.

[Wei08] Weilkiens, T.: *Systems Engineering mit SysML/UML: Modellierung, Analyse, Design.* 2. Auflage. Heidelberg: dpunkt.verlag, 2008.

[Wor13] World Wide Web Consortium (W3C): *Semantic Web Development Tools.* `http://www.w3.org/2001/sw/wiki/Tools`. Version: 2013.

[WWH14] *What When How. Nelder-Mead Evolutionary Hybrid-Algorithms (Artificial Intelligence).* `http://what-when-how.com/artificial-intelligence/`. Version: 2014.

[You84] Young, P.: *Recursive estimation and time series analysis: an introduction.* `https://katalog.ub.uni-paderborn.de/records/PAD_ALEPH000402627`. Version: 1984.

Das Heinz Nixdorf Institut – Interdisziplinäres Forschungszentrum für Informatik und Technik

Das Heinz Nixdorf Institut ist ein Forschungszentrum der Universität Paderborn. Es entstand 1987 aus der Initiative und mit Förderung von Heinz Nixdorf. Damit wollte er Ingenieurwissenschaften und Informatik zusammenführen, um wesentliche Impulse für neue Produkte und Dienstleistungen zu erzeugen. Dies schließt auch die Wechselwirkungen mit dem gesellschaftlichen Umfeld ein.

Die Forschungsarbeit orientiert sich an dem Programm „Dynamik, Mobilität, Vernetzung: Eine neue Schule des Entwurfs der technischen Systeme von morgen". In der Lehre engagiert sich das Heinz Nixdorf Institut in Studiengängen der Informatik, der Ingenieurwissenschaften und der Wirtschaftswissenschaften.

Heute wirken am Heinz Nixdorf Institut neun Professoren mit insgesamt 150 Mitarbeiterinnen und Mitarbeitern. Pro Jahr promovieren hier etwa 20 Nachwuchswissenschaftlerinnen und Nachwuchswissenschaftler.

Heinz Nixdorf Institute – Interdisciplinary Research Centre for Computer Science and Technology

The Heinz Nixdorf Institute is a research centre within the University of Paderborn. It was founded in 1987 initiated and supported by Heinz Nixdorf. By doing so he wanted to create a symbiosis of computer science and engineering in order to provide critical impetus for new products and services. This includes interactions with the social environment.

Our research is aligned with the program "Dynamics, Mobility, Integration: Enroute to the technical systems of tomorrow". In training and education the Heinz Nixdorf Institute is involved in many programs of study at the University of Paderborn. The superior goal in education and training is to communicate competencies that are critical in tomorrows economy.

Today nine Professors and 150 researchers work at the Heinz Nixdorf Institute. Per year approximately 20 young researchers receive a doctorate.

Zuletzt erschienene Bände der Verlagsschriftenreihe des Heinz Nixdorf Instituts

Bd. 361 PETER, S.: Systematik zur Antizipation von Stakeholder-Reaktionen. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 361, Paderborn, 2016 – ISBN 978-3-942647-80-9

Bd. 362 ECHTERHOFF, O.: Systematik zur Erarbeitung modellbasierter Entwicklungsaufträge. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 362, Paderborn, 2016 – ISBN 978-3-942647-81-6

Bd. 363 TSCHIRNER, C.: Rahmenwerk zur Integration des modellbasierten Systems Engineering in die Produktentstehung mechatronischer Systeme. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 363, Paderborn, 2016 – ISBN 978-3-942647-82-3

Bd. 364 KNOOP, S.: Flachheitsbasierte Positionsregelungen für Parallelkinematiken am Beispiel eines hochdynamischen hydraulischen Hexapoden. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 364, Paderborn, 2016 – ISBN 978-3-942647-83-0

Bd. 365 KLIEWE, D.: Entwurfssystematik für den präventiven Schutz Intelligenter Technischer Systeme vor Produktpiraterie. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 365, Paderborn, 2017 – ISBN 978-3- 942647-84-7

Bd. 366 IWANEK, P.: Systematik zur Steigerung der Intelligenz mechatronischer Systeme im Maschinen- und Anlagenbau. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 366, Paderborn, 2017 – ISBN 978-3-942647-85-4

Bd. 367 SCHWEERS, C.: Adaptive Sigma-Punkte-Filter-Auslegung zur Zustands- und Parameterschätzung an Black-Box-Modellen. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 367, Paderborn, 2017 – ISBN 978-3-942647-86-1

Bd. 368 SCHIERBAUM, T.: Systematik zur Kostenbewertung im Systementwurf mechatronischer Systeme in der Technologie Molded Interconnect Devices (MID). Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 368, Paderborn, 2017 – ISBN 978-3-942647-87-8

Bd. 369 BODDEN, E.; DRESSLER, F.; DUMITRESCU, R.; GAUSEMEIER, J.; MEYER AUF DER HEIDE, F.; SCHEYTT, C.; TRÄCHTLER, A. (Hrsg.): Intelligente technische Systeme. Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 369, Paderborn, 2017 – ISBN 978-3-942647-88-58

Bd. 370 KÜHN, A.: Systematik zur Release-Planung intelligenter technischer Systeme. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 370, Paderborn, 2017 – ISBN 978-3-942647-89-2

Bd. 371 REINOLD, P.: Integrierte, selbstoptimierende Fahrdynamikregelung mit Einzelradaktorik. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 371, Paderborn, 2017 – ISBN 978-3-942647-90-8

Bd. 372 BÄUMER, F. S.: Indikatorbasierte Erkennung und Kompensation von ungenauen und unvollständig beschriebenen Softwareanforderungen. Dissertation, Fakultät für Wirtschaftswissenschaften, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 372, Paderborn, 2017 – ISBN 978-3-942647-91-5

Bd. 373 ECKELT, D.: Systematik zum innovationsorientierten Intellectual Property Management. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 373, Paderborn, 2017 – ISBN 978-3-942647-92-2

Bd. 374 GAUSEMEIER, J. (Hrsg.): Vorausschau und Technologieplanung. 13. Symposium für Vorausschau und Technologieplanung, Heinz Nixdorf Institut, 23. und 24. November 2017, Berlin-Brandenburgische Akademie der Wissenschaften, Berlin, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 374, Paderborn, 2017 – ISBN 978-3-942647-93-9

Bezugsadresse:
Heinz Nixdorf Institut
Universität Paderborn
Fürstenallee 11
33102 Paderborn

Zuletzt erschienene Bände der Verlagsschriftenreihe des Heinz Nixdorf Instituts

Bd. 375 WESTERMANN, T.: Systematik zur Reifegradmodell-basierten Planung von Cyber-Physical Systems des Maschinenund Anlagenbaus. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 375, Paderborn, 2017 – ISBN 978-3-942647-94-6

Bd. 376 JÜRGENHAKE, C.: Systematik für eine prototypenbasierte Entwicklung mechatronischer Systeme in der Technologie MID (Molded Interconnect Devices). Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 376, Paderborn, 2017 – ISBN 978-3-942647-95-3

Bd. 377 WEBER, J.: Modellbasierte Werkstück und Werkzeugpositionierung zur Reduzierung der Zykluszeit in NC-Programmen. Dissertation, Fakultät für Wirtschaftswissenschaften, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 377, Paderborn, 2018 – ISBN 978-3-942647-96-0

Bd. 378 OESTERSÖTEBIER, F.: Modellbasierter Entwurf intelligenter mechatronischer Systeme mithilfe semantischer Technologien. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 378, Paderborn, 2018 – ISBN 978-3-942647-97-7

Bd. 379 ABELDGAWAD, K.: A System-Level Design Framework for Networked Driving Simulation. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 379, Paderborn, 2018 – ISBN 978-3-942647-98-4

Bd. 380 JUNG, D.: Local Strategies for Swarm Formations on a Grid. Dissertation, Fakultät für Elektrotechnik, Informatik und Mathematik, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 380, Paderborn, 2018 – ISBN 978-3-942647-99-1

Bd. 381 PLACZEK, M.: Systematik zur geschäftsmodellorientierten Technologiefrühaufklärung. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 381, Paderborn, 2018 – ISBN 978-3-947647-00-2

Bd. 382 KÖCHLING, D.: Systematik zur integrativen Planung des Verhaltens selbstoptimierender Produktionssysteme. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 382, Paderborn, 2018 – ISBN 978-3-947647-01-9

Bd. 383 KAGE, M.: Systematik zur Positionierung in technologieinduzierten Wertschöpfungsnetzwerken. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 383, Paderborn, 2018 – ISBN 978-3-947647-02-6

Bd. 384 DÜLME, C. (Hrsg.): Systematik zur zukunftsorientierten Konsolidierung variantenreicher Produktprogramme. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 384, Paderborn, 2018 – ISBN 978-3- 947647-03-3

Bd. 385 GAUSEMEIER, J.: Vorausschau und Technologieplanung. 14. Symposium für Vorausschau und Technologieplanung, Heinz Nixdorf Institut, 8. und 9. November 2018, Berlin-Brandenburgische Akademie der Wissenschaften, Berlin, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 385, Paderborn, 2018 – ISBN 978-3- 947647-04-0

Bd. 386 SCHNEIDER, M.: Spezifikationstechnik zur Beschreibung und Analyse von Wertschöpfungssystemen. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 386, Paderborn, 2018 – ISBN 978-3-947647-05-7

Bd. 387 ECHTERHOFF, B.: Methodik zur Einführung innovativer Geschäftsmodelle in etablierten Unternehmen. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 387, Paderborn, 2018 – ISBN 978-3-947647-06-4

Bezugsadresse:
Heinz Nixdorf Institut
Universität Paderborn
Fürstenallee 11
33102 Paderborn