Strukturierter Entwurf selbstoptimierender mechatronischer Systeme

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

genehmigte
DISSERTATION

von
Oliver Oberschelp
aus Bad Oeynhausen

Tag des Kolloquiums:	07. November 2008
Referent:	Prof. Dr.-Ing. Joachim Lückel,
Korreferent:	Prof. Dr. rer. nat. Wilhelm Schäfer

Vorwort

Die Ergebnisse dieser Arbeit resultieren aus den Erkenntnissen, die ich während der Jahre am Mechatronik Laboratorium Paderborn (MLaP) der Universität Paderborn unter Leitung von Prof. Dr.-Ing. Joachim Lückel gewinnen konnte. Ihm gilt auch mein Dank als Doktorvater, der die Anregung zu diesem Thema gab und die Fertigstellung der Arbeit in vielfältiger Weise gefördert hat. In meiner Zeit als sein Mitarbeiter unterstütze er mich durch konstruktive Anregungen und stete Gesprächsbereitschaft. Er gab mir die Chance an zahlreichen Forschungsprojekten mitzuarbeiten, die diese Arbeit erst ermöglichten. Seine Zuversicht war mir stets Ansporn, sein Vertrauen erfüllte mich mit Stolz.

Herrn Prof. Dr.-Ing. habil. Ansger Trächtler danke ich für die freundliche Übernahme der Rolle des Erstgutachters im Kolloquium nach Ausscheiden von Prof. Lückel.

Weiterhin möchte ich Prof. Dr. Wilhelm Schäfer für die Übernahme des Zweitgutachtens danken. Ohne die ausgezeichnete Zusammenarbeit mit ihm und den Mitarbeitern seiner Fachgruppe wäre diese Arbeit nicht zustande gekommen.

Allen meinen lieben Kolleginnen und Kollegen aus dem MLaP möchte ich darüber hinaus Danken. Das besondere Arbeitsklima der gegenseitigen Hilfsbereitschaft und die immer offenen Türen, waren der Saatboden für viele Ideen und Erkenntnisse und gaben mir Kraft und Freude.

Besonders möchte ich Herrn Dr.-Ing. Thorsten Hestermeyer für seine Begeisterung über meine Ideen und für seinen fachlichen Rat bedanken. Herrn Dipl.-Ing. Alfonso Gambuzza und Herrn Dipl.-Math. Henner Vöcking möchte ich doppelt Danken. Zum einen für ihre Mitarbeit als studentische Hilfskräfte und zum anderen für ihre Zusammenarbeit in vielen Projekten als Kollegen. Prof. Dr. rer. nat. Holger Giese und Herrn Dr. rer. nat. Sven Burmester danke ich für die vertrauensvolle Zusammenarbeit und die vielen Diskussionen die zum Fundament dieser Arbeit wurden.

Frau Bökamp-Gros danke ich für die sorgfältige Durchsicht des Manuskripts.

Besonders herzlicher Dank gebührt meiner Familie, insbesondere meiner lieben Frau Evelyn, die mich über die Jahre begleitet und unterstützt hat und meinen Eltern, die mir diese Arbeit erst ermöglichten.

Löhne, im Dezember 2008 *Oliver Oberschelp*

Meinen Eltern

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

ABS	Antiblockiersystem
AMS	Autonomes Mechatronisches System
ANN	Artificial Neuronal Network
ASR	Antischlupfregelung
BB	Basisblock
BDI	Belief-Desires-Intentions
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAMeL	Computer Aided Mechatronik Laboratory
DIN	Deutsche Industrienorm
DSC	Dynamik System Code
ESP	Elektronisches Stabilitätsprogramm
HE	Hierarchieelement
HHE	Hybrides Hierarchieelement
IPANEMA	Integration Platform for Networked Mechatronic Applications
KI	Künstliche Intelligenz
MFM	Mechatronisches Funktionsmodul
MLaP	Mechatronik Laboratorium Paderborn
MLP	Multilayer Perception
MOPO	Multi Objective Parameter Optimization
MRK	Multirate-Runge-Kutta(-Verfahren)
NBP	Neue Bahntechnik Paderborn
OCM	Operator-Controller-Modul
O-DSL	Objective-Dynamic System Language
O-DSS	Objective-Dynamic System Structure
OMM	Objektorientiertes Mechatronikmodell
RBF	Radial-Basis-Funktions-Netz
RCOS	Real-time Communication System
RTOS	Real-time Operation System
SFB	Sonderforschungsbereich
VDI	Verein Deutscher Ingenieure
VMS	Vernetztes Mechatronisches System

Wichtige Formelzeichen

c	Federkonstante
d	Dämpfungskonstante
D	Dämpfungsmaß
F	Kraft
g	Erdbeschleunigung
i	Stromstärke
J	Massenträgheit
K	Drehsteifigkeit
L	Induktivität
m	Masse
M	Drehmoment
N	Übersetzung
R	Ohmscher Widerstand
t	Zeit
u	Spannung
w	Windungszahl
x	x-Richtung
y	y-Richtung
z	z-Richtung
μ	Permeabilitätskonstante
ϑ	Drehwinkel
ω	(Eck-) Frequenz

Kapitel 1

Einführung

> Ratlosigkeit und Unzufriedenheit sind die ersten Vorbedingungen des Fortschritts (Thomas Alva Edison)

1.1 Herausforderung Selbstoptimierung

Die Anforderungen an industrielle Erzeugnisse waren noch nie so hoch wie in der heutigen Zeit. Dies gilt im besonderen Maße für Systeme des Maschinenbaus. Die globalen Märkte fordern immer leistungsfähigere Produkte in immer kürzerer Folge; gleichzeitig erlaubt der Marktdruck keine weitere Steigerung der Kosten bzw. der Endpreise. Mit jeder Generation werden die Produkte immer komplexer – ein Beispiel hierfür ist die Automobilindustrie. Jede Fahrzeuggeneration weist immer mehr technische Funktionen auf. Während in den achtziger Jahren noch Funktionen wie ABS, Airbag oder Klimaanlage der Oberklasse vorbehalten waren, sind dies mittlerweile Ausstattungsmerkmale jeder Fahrzeugklasse bis hin zum Kleinwagen. Die Wertschöpfung aus der Vital- und Komfortelektronik macht im PKW-Bereich inzwischen bis zu 40 % der Gesamtwertschöpfung (Neuwagen) aus und wird nach einigen Studien bis zum Jahre 2010 auf 60 % steigen [Ver04].

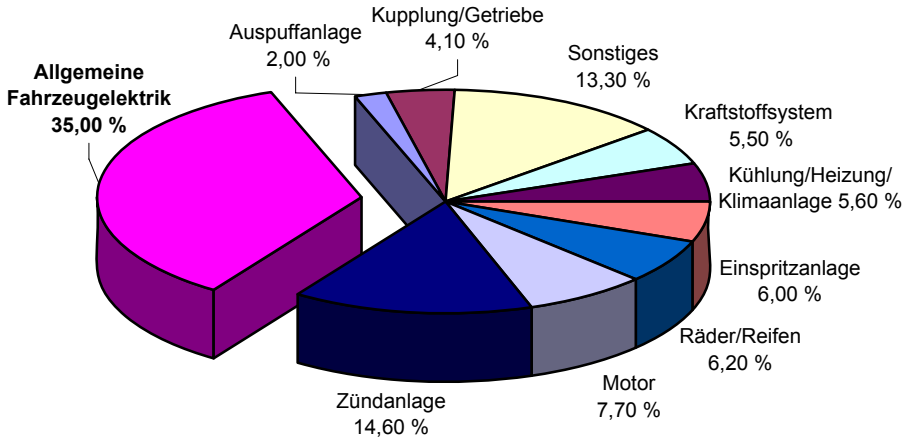

Abbildung 1.1: ADAC-Pannenstatistik 2005: Verteilung der Pannen (alle Baujahre)

Die Pannenstatistik des Allgemeinen Deutschen Automobilclubs (ADAC)[1] über das Jahr 2005 zeigt (Abbildung 1.1), dass mittlerweile mehr als ein Drittel aller

[1] http://www.adac.de

Pannen bei Fahrzeugen der Baujahre 2000-2005 auf fehlerhafte Elektronik zurückzuführen sind – Tendenz steigend. Diese Entwicklung führt zu immer größeren Problemen für die Fahrzeughersteller. Die Fehlerrate in neuen Fahrzeugen, insbesondere in der Karosserieelektronik, und die Kosten für die Produktpflege nehmen zu, was sich unter anderem in zahlreichen Rückrufaktionen ausdrückt.

Offensichtlich ist es nicht mehr ohne weiteres möglich, für solche komplexen Systeme alle Anforderungen und Randbedingungen schon im Entwurf exakt festzulegen. Daraus folgt, dass die klassische 0-Fehler-Strategie bei der Entwicklung technischer Systeme nicht länger tragbar ist; außerdem wird bei der üblichen Komponentenentwicklung das Zusammenspiel der Teilkomponenten nicht hinreichend berücksichtigt [MG04]. Zwar war das Prinzip der Bildung (unabhängiger) Aggregate der Schlüssel zur industriellen Massenfertigung [Lev02]. Auf heutige komplexe Systeme ist dieser Ansatz jedoch nur bedingt anwendbar. Die Dynamik des Gesamtsystems und der Einfluss der Teilkomponenten untereinander nehmen immer weiter zu. Deshalb müssen moderne Systeme schon beim Entwurf in ihrer Funktion ganzheitlich betrachtet werden.

Erste Antworten auf diese neuen Herausforderungen gibt bereits die *Mechatronik*. Die dort angestrebte ganzheitliche Betrachtung von technischen Systemen führt zu einer Aufweichung der Grenzen zwischen den verschiedenen Entwicklungsdomänen wie Maschinenbau, Elektrotechnik und Informatik. Vom Entwickler wird erwartet, das Zusammenspiel dieser Domänen systematisch zu nutzen.

Die Struktur technischer Systeme spielt eine große Rolle bei den heutigen Problemen, aber auch bei der Entwicklung von Lösungsansätzen. So kann die Struktur des technischen Systems wichtige Hinweise auf die Strukturierung der Informationsverarbeitung geben. Betrachtet man technisches System und Informationsverarbeitung ganzheitlich, ergibt sich aus der Wechselwirkung zwischen Entwurf und Strukturierung die Reduktion der Gesamtkomplexität, die in konventionellen Systemen durch Aggregation erreichbar war.

Vor diesem Hintergrund gilt es nun, Produkte zu entwickeln, die in der Lage sind, ihre *Funktionsqualität* nicht nur aufrecht zu erhalten, sondern selbsttätig zu verbessern und Probleme wie Fehler soweit wie möglich auch im laufenden Betrieb zu lösen oder abzumildern. Systeme, die diese Eigenschaft aufweisen, sollen im Folgenden als *selbstoptimierende Systeme* bezeichnet werden.

Selbstoptimierung kann helfen, komplexe mechatronische Systeme noch besser auf zukünftige Herausforderungen des Marktes vorzubereiten. Diese Arbeit soll die Entwicklung eines Paradigmas für Entwurfs- und Realisierungsansätze von selbstoptimierenden mechatronischen Systemen anregen und unterstützen. Darüber hinaus zeigt sie die Notwendigkeit einer ganzheitlichen Betrachtung der Selbstoptimierung als eines Teilaspekts mechatronischer Systeme.

1.2 Umfeld dieser Arbeit

Diese Arbeit entstand im Rahmen verschiedener Forschungsarbeiten am Mechatronik Laboratorium Paderborn unter der Leitung von Prof. Dr.-Ing. J. Lückel. Die Schwerpunkte dieser Arbeiten lagen in folgenden Bereichen: systematischer Entwurf

mechatronischer Systeme [GL00], Laufzeitplattform und Codegenerierung zur Simulation mechatronischer Systeme [Uni06], [ZRL+01], [Uni98] und Entwurf agentenbasierter Reglersysteme im Rahmen des Sonderforschungsbereichs 614: *Selbstoptimierende Systeme des Maschinenbaus* [GLR01]. Dem zuletzt genannten Projekt kommt besondere Bedeutung zu, da diese Arbeit auf Forschungsergebnissen basiert, die in diesem Projekt erarbeitet wurden. Ein weiteres Projekt, das eine Grundlage dieser Arbeit bildet, ist die *Neue Bahntechnik Paderborn* [Neu06]. Es handelt sich dabei um die Entwicklung eines neuartigen Bahnsystems, basierend auf autonomen, selbstfahrenden Schienentaxis, den sogenannten *RailCabs*. Das Projekt *Neue Bahntechnik Paderborn* dient wiederum als Anwendungsszenario für den Sonderforschungsbereich 614.

Im Folgenden werden die Schwerpunkte der wichtigsten Projekte skizziert und eingeordnet. Hierbei stehen der inhaltliche Rahmen und die Struktur im Vordergrund, da sie die Grundlage zur Strukturierung der vorliegenden Arbeit darstellen.

1.2.1 Sonderforschungsbereich 614

Wesentliche Teile der vorliegenden Arbeit entstanden im Rahmen des Sonderforschungsbereichs 614: *Selbstoptimierende Systeme des Maschinenbaus*. Zielsetzung dieses von der Deutschen Forschungsgemeinschaft geförderten Projektes ist die Entwicklung von Methoden für den Entwurf und die Realisierung selbstoptimierender mechatronischer Systeme. Schwerpunkte sind hierbei die Entwicklung und die Anwendung von Methoden der modellbasierten, aber auch verhaltensbasierten Optimierung, sowie die Untersuchung von Wirkprinzipien der Selbstoptimierung. Eines der zentralen Teilprojekte innerhalb des SFB 614 im ersten Antragszeitraum ist das Teilprojekt C3: *Agentenbasierte Regler* unter der Leitung von Prof. Dr.-Ing. J. Lückel. In diesem Teilprojekt werden Methoden des Maschinenbaus und der Regelungstechnik, wie die modellbasierte Optimierung zur System- und Reglerauslegung, mit Methoden der Informatik, wie verhaltensbasierte Systeme, verknüpft [GLR01].

Das Projekt hat sich das ehrgeizige Ziel gesetzt, Grundlagen für Entwicklung und Realisierung von selbstoptimierenden mechatronischen Systemen zu entwickeln, die in der Lage sind, sich nicht nur an veränderliche Umgebungsbedingungen anzupassen, sondern darüber hinaus auch selbsttätig ihr Verhalten gemäß expliziten Zielkriterien zu verbessern. So sollen solche Systeme auch mit unvorhergesehenen Situationen zurechtkommen, indem sie Parameter und innere Struktur durch ständige Optimierung an die neue Situation anpassen. Dies schließt auch das Lösen von Problemen bei Störungen ein, um einen sicheren und stetigen Betrieb zu gewährleisten. Das wird erst möglich durch eine komplexe, leistungsfähige Rechentechnik, welche die mechatronischen Systeme kontrolliert, steuert und regelt. Der Entwurf solcher Systeme unterscheidet sich wesentlich von dem konventioneller technischer Systeme, da eine Vielzahl von Methoden angewendet werden müssen, um die Gesamtfunktion zu realisieren.

Wie in der Mechatronik üblich, sind verschiedene Domänen an der Umsetzung beteiligt. Es gilt darüber hinaus, neue Strukturen für die Verknüpfung dieser Methoden zu entwerfen – die Gesamtkomplexität wird dadurch beherrschbar.

Für die Realisierung dieser Vision werden verschiedene Aufgabenbereiche bearbeitet. Dabei sollen Grundlagen der Selbstoptimierung untersucht werden, aus denen sich Methoden ableiten lassen. Entwurfsverfahren und -werkzeuge werden benötigt,

um solche neuartigen Systeme zu entwickeln. Die Realisierung solcher Systeme erfordert neue Ansätze in Regelungstechnik und Informationsverarbeitung, aber auch für die Hardware. Ziel ist die Realisierung solcher selbstoptimierender Systeme, um Methoden und Werkzeuge zu validieren und die theoretischen Arbeiten auf ein realistisches Fundament zu stellen.

Zentraler Demonstrator ist die Neue Bahntechnik Paderborn [Neu06], die für die notwendigen Tests und Realisierungen Prüfstände und Modelle zur Verfügung stellt. Die Anwendung Bahntechnik ist deshalb so interessant, da sie Selbstoptimierung vom kleinsten geregelten Ventil über verschiedenste Aggregate, autonome Systeme bis hin zur Logistik möglich macht. Ein so umfassendes Szenario ist notwendig, um dem geforderten Anspruch zu genügen. Das Projekt wird in Abschnitt 1.2.2 beschrieben.

Gemäß diesen Anforderungen wurde folgende Projekt- und Aufgabenstruktur abgeleitet, die sich in verschiedene Projektbereiche gliedert [GLR01]:

Projektbereich A: Grundlagen und Potentiale der Selbstoptimierung

Wissenschaftliche Durchdringung und ingenieurgerechte Aufarbeitung des Wirkparadigmas der Selbstoptimierung.

Projektbereich B: Entwurfsmethoden und -werkzeuge

Schaffung der methodischen und instrumentellen Voraussetzungen für die Entwicklung von innovativen Systemen, die auf dem Wirkparadigma der Selbstoptimierung beruhen.

Projektbereich C: Implementierungsmethoden

Realisierung der Selbstoptimierung auf Hardware-, System-, Software- und Reglersoftwareebene.

Projektbereich D: Selbstoptimierende Produkte und Systeme

Entwurf und prototypische Realisierung neuer Baugruppen, Erzeugnisse und Systeme, um das erarbeitete Instrumentarium zu validieren und der Sache der Produktinnovation sichtbare Impulse zu geben.

Diese Struktur bildet die Grundlage für die Projektstruktur, die sich in verschiedene Teilprojekte gliedert, wie in Abbildung 1.2 dargestellt. Die Aufgaben der Teilprojekte sind wie folgt definiert:

Im **Teilprojekt A1** (Modellbasierte Selbstoptimierung) werden Prinzipien der Selbstoptimierung, basierend auf Modellen eines gegebenen mechatronischen Systems, unterstützt.

Das **Teilprojekt A2** befasst sich mit der Bereitstellung von Selbstoptimierungswissen für Situationen, die sich nicht — bzw. nicht mit vertretbarem Aufwand — durch ein physikalisches Modell abbilden lassen.

Teilprojekt B1 entwickelt ingenieurgerechte Ausdrucksmittel für den Entwurf verteilter intelligenter Systeme. Die vernetzten Module, aus denen die Software selbstoptimierender Systeme aufgebaut ist, werden dabei als Multiagentensystem aufgefasst. Ein wichtiger Aspekt ist auch die Entwicklung von Spezifikations- und Modellierungstechniken, die eine Verifikation der Software erlauben.

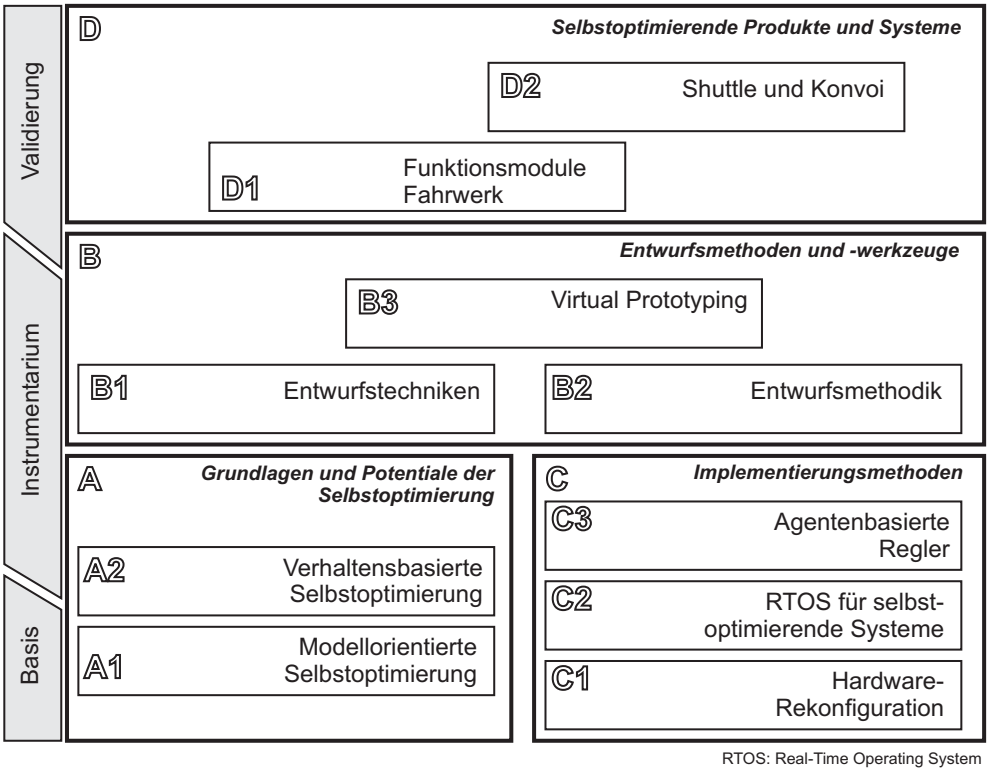

Abbildung 1.2: Projektstruktur des SFB 614

Im **Teilprojekt B2** sollen die Ergebnisse der anderen Teilprojekte zu einem Entwurfsinstrumentarium zusammengefasst werden.

Teilprojekt B3 entwickelt und erprobt neue Interaktions-, Darstellungs- und Analysetechniken zur Konstruktion und Verifikation von s.o. mechatronischen Systemen. Zur Verwendung kommen hier Methoden der virtuellen Realität.

Im **Teilprojekt C1** wird die Möglichkeit der Hardware-Rekonfiguration als besonders vielversprechende Realisierungstechnik für selbstoptimierende mechatronische Systeme betrachtet.

Teilprojekt C2 hat ein verteiltes Realzeitbetriebssystem (RTOS) in Verbindung mit einem Realzeitkommunikationssystem (RCOS) zum Ziel. Der verfolgte Ansatz betrachtet ein RTOS/RCOS selbst als Multiagentensystem, das dem Prinzip der Selbstoptimierung unterliegt.

Im **Teilprojekt C3** wird ein völlig neuer Weg eingeschlagen, indem symbiotisch die über den Bereich der Agententheorie einfließenden, sog. verhaltensbasierten Ansätze mit der aus der Regelungstechnik kommenden Modellbasierung verzahnt werden.

Im **Teilprojekt D1** werden die in den Bereichen A, B und C entwickelten selbstoptimierenden Verfahren und Methoden am aktiven Fahrwerk des Shuttles der Neuen Bahntechnik Paderborn untersucht und verifiziert.

Im **Teilprojekt D2** werden auf der Basis der in D1 untersuchten MFM-Module des Fahrwerks selbstoptimierende Systeme höherer Ordnung untersucht. Neben der im Teilprojekt D1 vorrangig behandelten modellgestützten Optimierung wird hier auch die verhaltensorientierte Optimierung, basierend auf der Agententechnik, angewandt.

1.2.2 Neue Bahntechnik Paderborn

Das Mechatronik Laboratorium Paderborn unter der Leitung von Prof. Dr.-Ing. J. Lückel beschäftigte sich seit seiner Gründung gegen Ende der 70er Jahre mit dem dynamischen Verhalten von Fahrzeugen. Ein besonderer Schwerpunkt bildete hierbei die Kontrolle des Bewegungsverhaltens des Aufbaus. Aus dieser Kernkompetenz heraus erkannte Prof. Lückel nach einem Besuch der Teststrecke des Transrapid im Emsland, dass heutige Bahnsysteme grundsätzliche konzeptionelle Mängel aufweisen: Im Hintergrund auch neuer Entwicklungen, wie des Transrapid, steht nach wie vor das Prinzip des Zuges – also eines Massenverkehrsmittels, das, durch eine Zugmaschine angetrieben, eine große Anzahl Waggons von einem Punkt zum anderen befördert.

Betrachtet man die Geschichte der Eisenbahn, so stellt man fest, dass die Entwicklung Strecke und Fahrzeug nach diesem Prinzip weitergeführt worden ist. Ja, man kann sogar soweit gehen, zu behaupten, dass mit der Erfindung des D-Zugs und der Einführung fester Fahrpläne die konzeptionelle Entwicklung der Eisenbahn abgeschlossen war. Seitdem hat es nur noch Verbesserungen der bestehenden Technik gegeben. Sogar der Transrapid macht da keinen entscheidenden Schritt nach vorn, auch wenn bei der Entwicklung von der klassischen Rad-Schiene-Technik abgewichen und ein neues Antriebskonzept entwickelt wurde.

Mit dem Automobil bekam die Bahn vor mehr als hundert Jahren ernsthafte Konkurrenz. Durch die industrielle Massenproduktion von Autos durch Henry Ford zu Beginn des 19. Jahrhunderts begann das Zeitalter des Individualverkehrs. Der Vorteil, zu einem beliebigen Zeitpunkt direkt ohne Umsteigen von A nach B zu fahren, hat das Auto zu *dem* Verkehrsmittel des 20. Jahrhunderts gemacht. Trotz dieser Entwicklung setzt die Bahn[2] nach wie vor auf feste Fahrpläne. Zudem zeigt die Reduzierung der Haltepunkte in den letzten Jahren zunehmend den Rückzug der Bahn aus der Fläche an. ICE und besonders Transrapid, als letzte Entwicklungen der Bahn, verstehen sich eher als Konkurrenz zum Nahverkehrsflugzeug denn als Alternative zum Automobil. Das wird besonders deutlich bei der Betrachtung der Investitionen der letzten Jahre bei der Deutschen Bahn AG. Die Neubaustrecke Köln-Frankfurt ist nur für den Transport von Personen geeignet, nicht für den von Gütern [Clö02], [DK07].

Jedoch gerade in der heutigen Zeit wird eine Entlastung der Straße auch im regionalen Verkehr dringend gebraucht. Der Individualverkehr stößt immer mehr an seine Grenzen. Das tägliche Verkehrschaos, gerade in Ballungszentren wie dem Ruhrgebiet, zeigt, dass neue Wege gegangen werden müssen, und zwar im regionalen Verkehr und auf Mittelstrecken. Das Beispiel des Transrapid zeigt außerdem, dass völlig neue Trassen politisch nur schwer durchzusetzen sind. Somit bietet es sich an, das vorhandene Schienensystem zu nutzen.

[2] An dieser Stelle wird nicht zwischen einzelnen Betreibergesellschaften differenziert.

Bei einem genaueren Vergleich zwischen Auto und Bahn fällt auf, dass es auch grundsätzliche Unterschiede in Fertigung und Lebensdauer gibt. Während Züge immer noch weitgehend in Einzelfertigung hergestellt werden, sind Automobile der Massenartikel schlechthin. Aufgrund der Herstellungskosten müssen Züge immer noch 30 und mehr Jahre halten. Die verbaute Elektronik muss aber schon viel früher ersetzt werden. Autos haben im Ansatz das gleiche Problem, jedoch ist die Lebensdauer deutlich kürzer und somit das Problem der Veralterung kleiner.

Fasst man all diese Erkenntnisse zusammen, lassen sich folgende Forderungen ableiten:

- Nutzung des vorhandenen Schienennetzes
- Bedarfsgerechtes Bereitstellen von Transportkapazitäten
- Kleinere, in Massen gefertigte Fahrzeuge
- Transport von Gütern und Personen direkt zwischen zwei Endpunkten (zielrein)
- Weitgehende Automatisierung
- Geringer Wartungsaufwand
- Emissionsfreier Antrieb

Eine Antwort auf diese Forderungen liefert das Konzept der *Neuen Bahntechnik Paderborn*. Hier werden kleine autonome Fahrzeuge, sogenannte Shuttles (im Folgenden auch als *RailCabs* bezeichnet), zum Transport von Personen und Gütern eingesetzt. Die Fahrzeuge nutzen dabei das bisherige Schienennetz, setzen beim Antrieb aber nicht auf Rad-Schiene-Technik, sondern auf den neuartigen Linearantrieb, wie er auch im Transrapid im Einsatz ist. In Kombination mit moderner aktiver Feder-Neigetechnik ergibt sich ein Verkehrsmittel, das in Komfort, Kosten, Schnelligkeit und Verfügbarkeit weder die Konkurrenz mit dem Auto, noch die mit der Bahn scheuen muss.

Das Konzept sieht selbstfahrende Fahrzeuge vor, die ihre Wege selbst finden können – ähnlich wie bei der Paketvermittlung im Internet. Deshalb wurde auch der Begriff „Internet auf Rädern“ für die Neue Bahntechnik geprägt. Neben dem Logistikkonzept, das auf der Planungsebene der wesentliche neue Aspekt ist, sind auf der Ebene der Technik zwei Dinge hervorzuheben:

1. Der Linearmotor als Antrieb wie auch als Energieversorger
2. Ein integriertes Konzept zur Kontrolle der Aufbaubewegung und -dynamik

Der Linearmotor ermöglicht durch eine direkte Kopplung zwischen Antrieb und Vortrieb eine sehr kompakte und wartungsfreundliche Bauweise der RailCabs. Die vollständige Kontrolle der Aufbaudynamik erlaubt einen unerreichten Komfortgewinn und eine deutliche Erhöhung der Sicherheit.

Die Neue Bahntechnik Paderborn liefert ein weites Feld für neue Entwicklungen. Selbstoptimierende Systeme können hier auf verschiedenen Ebenen des Systems zum Einsatz kommen. Angefangen von adaptiven Regelungen auf unterster Ebene,

über Funktionen wie Energiemanagement und Komfortoptimierung, bis hin zu Logistik und Ressourcenverwaltung. Durch das Zusammenspiel verschiedener aktiver Komponenten, zu denen neben den Shuttles auch die Schienenstrecke und die Energieversorgung zählen, sind Optimierungen des Gesamtsystems möglich. Es kommt darauf an, die Chancen zu sehen und zu nutzen. Systeme wie das RailCab werden in Zukunft immer häufiger Fragestellungen aufwerfen, die nur durch selbstoptimierende mechatronische Systeme zu lösen sind. Die Entwicklung von Grundlagen muss deshalb vorangetrieben werden.

1.3 Zielsetzung

Ziel dieser Arbeit ist die Darstellung von Grundlagen für die Entwicklung von selbstoptimierenden mechatronischen Systemen. Dabei sollen folgende Teilbereiche angesprochen werden:

- Ansätze für eine strukturierte Entwicklung
- Methoden der Selbstoptimierung
- Selbstoptimierung durch Rekonfiguration
- Numerische Aspekte der Simulation und der Realisierung
- Grundlagen zum Entwurf der Informationsverarbeitung
- Ausblick auf künftige selbstoptimierende Systeme

Ein übergeordnetes Ziel ist die Darstellung von Ansätzen zur *strukturierten Entwicklung* selbstoptimierender mechatronischer Systeme unter Berücksichtigung verschiedener Methoden. Dabei sollen Möglichkeiten für Entwurf und Realisierung aufgezeigt werden.

Aus der Struktur mechatronischer Systeme leitet sich die *Rekonfiguration* als ein weiteres wichtiges Thema dieser Arbeit ab. Im Mittelpunkt stehen dabei Ansätze zur Modellierung der Rekonfiguration in regelnder Informationsverarbeitung. Der damit verbundene Ansatz zur Modellierung ist auf weitere Domänen der Mechatronik übertragbar.

Wegen der strukturellen Veränderung der regelnden Informationsverarbeitung ist eine Untersuchung modularer Systeme unvermeidbar. Die Arbeit soll zeigen, welche Probleme bei der numerischen Verarbeitung modularer Systeme auftreten können, die wiederum Voraussetzung für eine Rekonfiguration sind. Eng damit verbunden ist der Entwurf einer Informationsverarbeitung, die in der Lage ist, solche Veränderungen zu modellieren und umzusetzen. Diese Arbeit bildet dabei auch Anknüpfungspunkte an weitere Arbeiten, die sich mit dem Thema selbstoptimierende Systeme befassen. Abschließend sollen mit Hilfe von einfachen Beispielen erste Ideen für die Anwendung von Selbstoptimierung gegeben werden.

1.4 Gliederung der Arbeit

Abbildung 1.3 zeigt schematisch die Struktur:

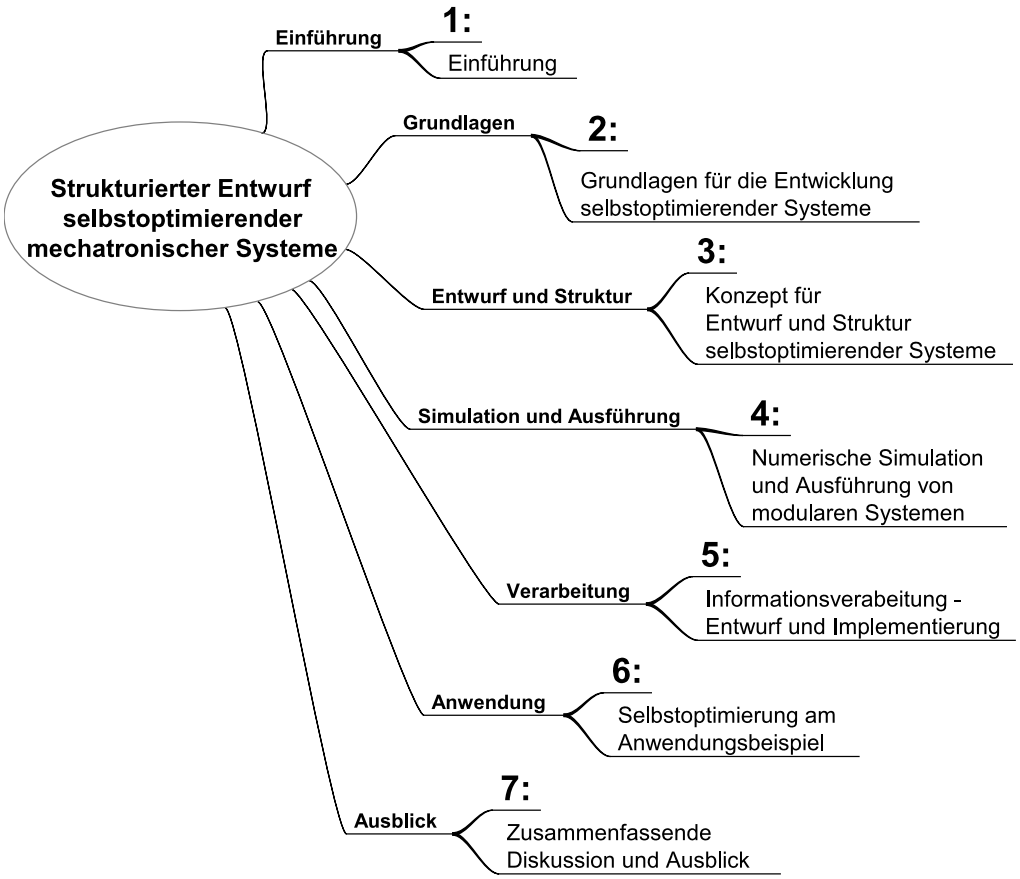

Abbildung 1.3: Schematische Darstellung der Gliederung der vorliegenden Arbeit

In der *Einführung* (1) wird das Thema der Arbeit motiviert und ein inhaltlicher Überblick gegeben. Im Kapitel *Grundlagen für die Entwicklung selbstoptimierender Systeme* (2) werden theoretische Grundlagen angesprochen, die wesentlich für das Verständnis von Selbstoptimierung in mechatronischen Systemen sind. Grundlagen für den Entwurf werden im Kapitel *Konzept für Entwurf und Struktur selbstoptimierender Systeme* (3) dargestellt. Dabei geht es vor allem um Ansätze zur Modellierung solcher Systeme. Neben der Parametrierung, also den freien Systemparametern eines Systems, kann eine Optimierung auch auf struktureller Ebene erfolgen. Dies führt zwangsläufig zu modularen Teilsystemen. Die besonderen Aspekte der numerischen Berechnung solcher modularen Systeme werden im Kapitel *Numerische Simulation und Ausführung von modularen Systemen* (4) betrachtet. Für die Simulation und den Betrieb von selbstoptimierenden mechatronischen Systemen ist neben der mathematischen Darstellung und der numerischen Behandlung auch eine besondere Informationsverarbeitung notwendig. Die Grundlagen werden im Kapitel *Informationsverarbeitung – Entwurf und Implementierung* (5) behandelt. Im Kapitel *Anwendungsbeispiele für Selbstoptimierung* (6) werden Beispiele für die Realisierung von Selbstoptimierung an verschiedenen einfachen Anwendungen gezeigt. Eine Zusammenfassung sowie ein Ausblick werden im Kapitel *Zusammenfassende Diskussion und Ausblick* (7) gegeben.

Kapitel 2

Grundlagen für die Entwicklung selbstoptimierender Systeme

> Die Grenzen des Möglichen lassen sich nur dadurch bestimmen, dass man sich ein wenig über sie hinaus ins Unmögliche wagt (Sir Arthur C. Clarke)

2.1 Bestimmung des Begriffs Selbstoptimierung

Der Begriff Selbstoptimierung taucht im Zusammenhang mit vielen technischen Systemen und Problemlösungen auf. Jedoch wird der Begriff selbst unterschiedlich aufgefasst und ausgelegt. Dieser Abschnitt stellt zunächst den Begriff Selbstoptimierung vor und versucht eine Begriffsbestimmung in Anlehnung an die Ergebnisse des SFB 614 und speziell an ein veröffentlichtes Teilergebnis aus [FGK+04].

2.1.1 Adaptive Regler und Selbstoptimierung

Der Begriff Selbstoptimierung ist im Zusammenhang mit geregelten technischen Systemen nicht neu. Er findet sich vor allem bei adaptiven Reglern. Nach [Web71] ist ein adaptives Regelsystem wie folgt definiert:

> *„Unter einem adaptiven Regelungssystem möge im Folgenden ein solches verstanden sein, bei dem sich bestimmte Eigenschaften des Systems (zumeist Struktur oder Parameter der Strecke) oder seiner Eingangssignale in nicht vorhersagbarer Weise ändern und sich andere, gezielt beeinflussbare Systemeigenschaften (zumeist Eigenschaften des Reglers, also z. B. dessen Struktur und Parameter) selbsttätig darauf einstellen, so dass ein gewünschter Systemzustand erhalten bleibt.“* [Web71]

Diese Definition kann auch als Basis für die Definition selbstoptimierender Reglersysteme verwendet werden, wobei hier der Unterschied im letzten Satz der Definition liegt: Bei adaptiven Systemen ist das Ziel, den gewünschten Systemzustand zu *erhalten*. Selbstoptimierende Reglersysteme gehen darüber hinaus. Ziel ist es hier, den Systemzustand nach bestimmten Kriterien zu *verbessern* und gegebenenfalls neu auszulegen.

Die in [Web71] verwendete allgemeine Struktur eines adaptiven Regelungssystems ist trotz dieses genannten Unterschieds auch auf selbstoptimierende Reglersysteme übertragbar.

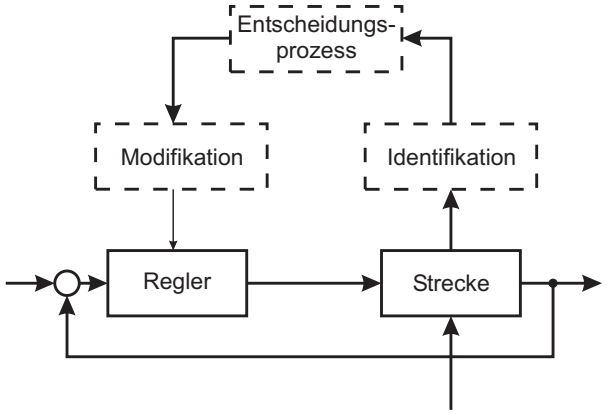

Abbildung 2.1: Grundfunktion eines adaptiven Regelungssystems

In Abbildung 2.1 [Web71] ist der grundsätzliche Aufbau eines adaptiven Regelungssystems dargestellt. Er besteht aus dem eigentlichen klassischen Regelkreis mit Strecke und Regler und den zusätzlichen Komponenten für die Anpassung, die aus *Identifikation*, *Entscheidungsprozess* und *Modifikation* bestehen.

In [ILM92] findet sich bereits eine Definition eines sogenannten selbstoptimierenden Reglers. Dieser Reglertyp gehört zu den adaptiven Reglern und wird als *Model Identification Adaptive Controller* (MIAC) bezeichnet, der auch *self-optimizing controller* genannt wird (siehe Abbildung 2.2). Er hat folgende Struktur:

Über eine Prozessidentifikation wird mit Hilfe eines Prozess-Modells eine Parameterbestimmung durchgeführt, die zu neuen Reglerparametern führt. Dieses Modell kommt den Forderungen an ein selbstoptimierendes System schon sehr nah.

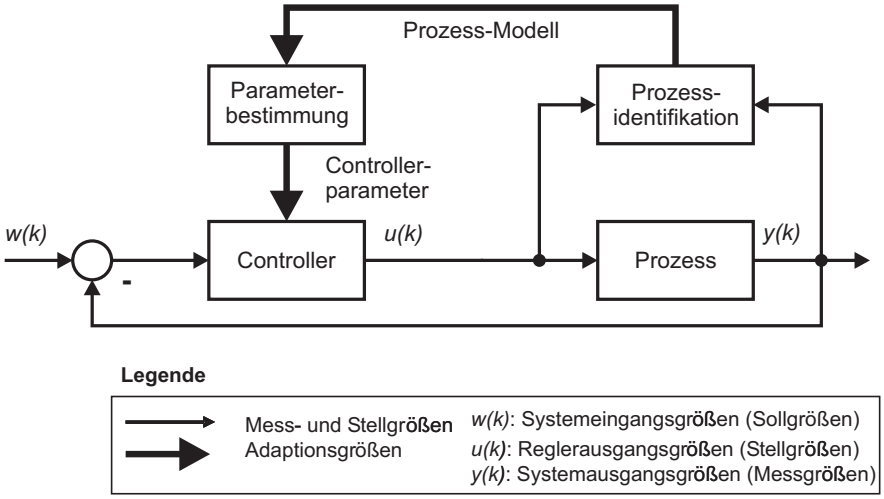

Abbildung 2.2: *Modell Identification Adaptive Controller* (MIAC) oder *self-optimizing controller*

In der Struktur unterscheidet sich der adaptive Regler nicht von einem s. o. Reglersystem. Der Unterschied verbirgt sich in den Komponenten. Beim adaptiven Regler sind sie definiert als:

1. Die *Identifikation* (identification) oder Erkennung, durch die der zeitvariable Zustand des Systems fortlaufend erfasst wird.

2. Der *Entscheidungsprozess* (decision), in dem die durch die Identifikation gewonnene Information über den realen Zustand des Systems mit einem erwünschten idealen Zustand verglichen und entschieden wird, welche Maßnahmen angebracht sind, um sich auf den idealen Zustand hinzubewegen.

3. Die *Modifikation* (modification, actuation) des Reglers, der sich aufgrund des Entscheidungsprozesses in vorgeschriebener Weise ändern muss [Web71].

Im Gegensatz hierzu erweitert sich der Entscheidungsprozess (2) bei s. o. Systemen um die *Auswahl des erwünschten idealen Zustands*, so dass je nach Gesamtzustand von System und Umwelt auch ein anderes Reglerverhalten als ideal angesehen, bzw. definiert wird. Hierdurch unterscheiden sich s. o. Systeme entscheidend von adaptiven Regelungssystemen.

Die Bewertung der erreichten Reglerqualität erfolgt bei adaptiven Reglern häufig durch die Betrachtung von so genannten Güteindizes (vgl. [Web71]). Diese korrespondieren mit einem vorbestimmten, fiktiven idealen Reglerzustand. Die Bewertung dieser Größe(n) führt über Entscheidungsprozess und Modifikation zu einem angepassten Regler. Der Erfolg dieses Vorgangs kann durch erneutes Bewerten des oder der Güteindizes bestimmt werden. Im Allgemeinen ist die Bestimmung und Bewertung dieser Gütemaße fest mit dem Adaptionsalgorithmus verbunden, so dass keine automatisierte Anpassung der Bewertung erfolgt. Dies ergibt aus Sicht der Adaption auch keinen Sinn, da das oberste Ziel der Adaption, nämlich die Einhaltung eines bestimmten idealen Reglerzustands, fest vorgegeben ist.

Verschiedene Verfahren zur Adaption basieren auf der Vorgabe eines idealen Regelungsverhaltens durch ein Modell. Dabei wird das Verhalten des realen geregelten Systems an das Verhalten des Modells durch Variation der Reglerparameter angepasst. Andere Verfahren verwenden Modelle der Strecke zur Identifikation. Hierbei werden die Parameter eines Modells so angepasst, dass das Modellverhalten mit dem der realen geregelten Strecke übereinstimmt. Dieses identifizierte Modell dient als Grundlage für den Entscheidungsprozess und die Modifikation.

Verfahren, die ein Modell der Strecke oder der geregelten Strecke zur Adaption benutzen, werden in diesem Zusammenhang oft zusammenfassend als *modellbasierte Verfahren* bezeichnet. Häufig wird dabei von sogenannten *self-tuning*-Reglern gesprochen. In [Lau99] wird eine allgemeine Struktur für adaptive bzw. self-tuning-Regler vorgestellt, jedoch auch darauf hingewiesen, dass in der Praxis die Verfahren schwer handhabbar sind und die Verschmelzung der Regel- und der Steuerkreismodelle die Strukturen noch komplexer machen.

Die Abbildung 2.3 zeigt die Struktur eines modellbasierten, parameteradaptiven Reglersystems nach [Lau99]. Hierbei werden grundsätzlich eine Regelungs- und eine Kontroll-/ Überwachungsebene unterschieden. Im Einzelnen existieren Elemente zur Regelung, zur Änderung der Regelung bzw. der Reglerparameter (Regler-Gesetz), zur Umschaltung zwischen verschiedenen Reglern, zur Identifikation der Strecke und zur Überwachung, Koordination und Fehlerkorrektur. Das technische System selbst ist auch Teil der Reglerstruktur, jedoch liegt es real und nicht als mathematische Beschreibung vor.

Die beschriebenen Reglermodelle zeigen verschiedene Lösungsansätze für das Problem der Umgebungsanpassung von Regelungen. Allen gemeinsam ist das Einwirken auf einen klassischen Regelungskreislauf mit Hilfe von nebenläufigen Kontrollprozessen. Sie können als Grundlage für selbstoptimierende mechatronische Systeme angesehen werden. Jedoch beschränken sich die Ansätze und Konzepte auf lokale

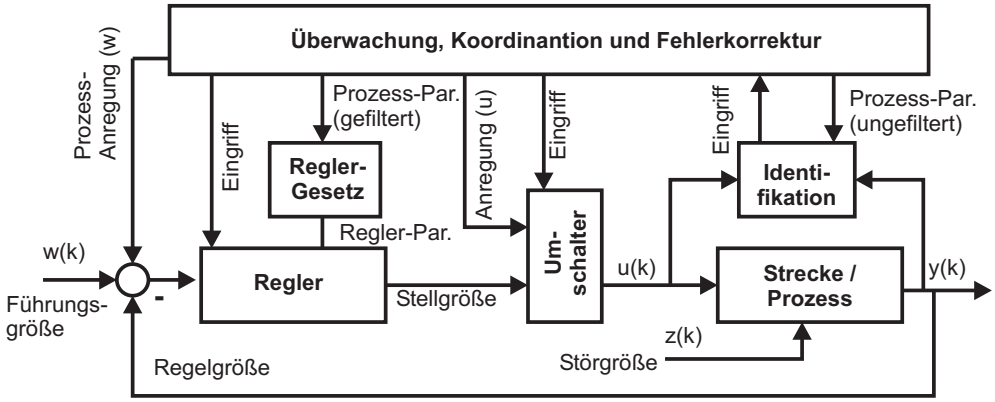

Abbildung 2.3: Struktur eines Regelkreises mit prozessabhängiger automatischer Wahl der Reglerparameter

Lösungen. Ein Hinweis darauf, wie *selbstoptimierende Reglersysteme* zusammenwirken können, wird nicht gegeben. Darüber hinaus muss berücksichtigt werden, dass Selbstoptimierung im Allgemeinen nicht auf die Optimierung von Regelungsvorgängen beschränkt ist, auch wenn das ein zentraler Punkt ist. Somit genügt es nicht, allein adaptive Reglersysteme zu selbstoptimierenden Reglersystemen zu erweitern.

2.1.2 Definition der Selbstoptimierung

In [Ras05] wird Selbstoptimierung als ein Prozess aufgefasst, der durch die Realisierung verschiedener Prinzipien definiert wird. Dies sind:

- Die Suche nach optimalen Lösungen, sowohl im Entwurf als auch im Betrieb, durch Online-Modellbildung, Vorabsimulation (einer Situation) und Vorabermittlung neuer Entwurfsparameter (Pareto-Punkte).
- Selektion oder Auswahl einer Lösung oder eines Entwurfspunktes.
- Erweiterung des Wissens durch vorausschauende Simulation (intelligente Vorausschau).

Im Mittelpunkt dieser Betrachtungen steht die Simulation. Für modellbasierte Optimierungssysteme ist diese Definition treffend, sie grenzt jedoch viele Methoden der Informatik aus. Deshalb wird eine weiter greifende Definition benötigt. Nach [FGK+04] wird Selbstoptimierung wie folgt definiert:

> *„In einem System findet genau dann Selbstoptimierung (self-optimization) statt, wenn durch das Zusammenwirken der enthaltenen Elemente die folgenden drei Aktionen wiederkehrend ausgeführt werden:“*

1. Analyse der Ist-Situation
2. Bestimmung der Systemziele
3. Anpassung des Systemverhaltens

Die *Ist-Situation* umfasst den Zustand des Systems selbst, sowie alle möglichen Beobachtungen[3] über seine Umgebung. Beobachtungen können direkt oder indirekt erfolgen. Direkte Beobachtungen erfolgen durch Erfassen von Parametern der Umgebung (in der Regel durch Messung mit Hilfe eines Sensors). Indirekte Beobachtungen erfolgen durch Kommunikation mit anderen Systemen. Dabei spielt es zunächst keine Rolle, auf welche Art und Weise das andere System die Parameter ermittelt hat. Weiterhin enthält der Zustand des Systems auch Informationen über zurückliegende Ereignisse, die als *Erfahrungen* des Systems aufgefasst werden können.

Systemziele bezeichnen Zustände, auf die das System hinarbeitet. Im einfachsten Fall könnte dies eine zu erreichende Führungsgröße sein. Die Ermittlung eines solchen Ziels kann durch eine Auswahl aus einer Liste von Zielen erfolgen oder durch kontinuierliche Anpassung einer Größe. Werden Ziele unabhängig von bisherigen Zielen neu erzeugt, wird von *Generierung von Zielen* gesprochen.

Das *Systemverhalten* kann auf verschiedene Art und Weise angepasst werden. Es kann durch die drei Aspekte *Parameter*, *Struktur* und *Verhalten* beschrieben werden. Parameter sind einstellbare freie Parameter des Systems. Mit Struktur wird der innere Aufbau eines Systems bezeichnet. Das gerichtete Ein- Ausgangsverhalten eines Systems wird als Verhalten bezeichnet.

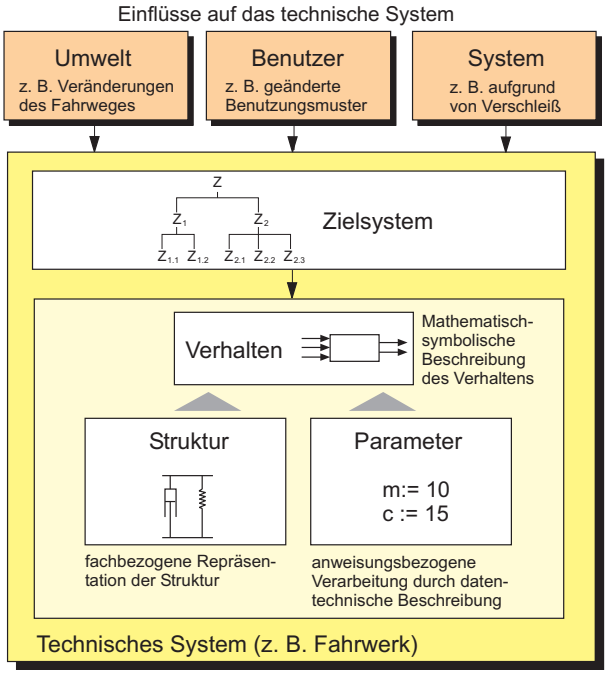

Abbildung 2.4: Die Aspekte Einflüsse, Ziele, Verhalten, Struktur und Parameter eines selbstoptimierenden Systems

Diese Begriffe dürfen nicht unabhängig voneinander betrachtet werden. Eine Parameteränderung führt auch zu einer Veränderung von Verhalten, und eine Veränderung der Struktur führt unter Umständen zu neuen Parametern. Aus Sicht eines überlagerten Systems ist aber ggf. nur die Vorgabe eines bestimmten Verhaltens (im

[3] In der Regelungstechnik werden häufig Größen durch sogenannte *Beobachtermodelle* rekonstruiert. Im Kern sind dies Aufbereitungen von gemessenen Werten, mit deren Hilfe auf weitere Werte durch spezielle Berechnungen geschlossen werden kann. An dieser Stelle sind Beobachtungen im Sinne von *Werteerfassung* gemeint.

Sinne eines vorgegebenen Teilziels) von Bedeutung, während Parameter und Struktur verborgen sind. In anderen Fällen, wie kaskadierten Regelungssystemen, ist die Vorgabe oder Optimierung von Parametern und Struktur für die Gesamtregelung zu berücksichtigen. Das Zusammenwirken verschiedener Einflussgrößen auf und in einem selbstoptimierenden System sind in Abbildung 2.4 [FGK+04] dargestellt.

Insgesamt kann festgestellt werden, dass Selbstoptimierung einem Prozess entspricht, der in einem technischen System oder Prozess so wirkt, dass er eine Verbesserung des Verhaltens des Systems bewirkt. Die *Richtung der Verbesserung*, also welches Verhalten als optimal angesehen wird, kann dabei wechseln. Im weitesten Sinn wird dieses optimale Verhalten definiert, z. B. durch den Entwickler vorgegeben. In hierarchisch verkoppelten Systemen können sich jedoch Teilziele für die unterlagerten Systeme ergeben, die durch das Wechselspiel der Komponenten entstehen. So kann es sinnvoll sein, eine Komponente in ihrem Verhalten zugunsten einer anderen Komponente zu verschlechtern (z. B. Abschaltung der Regelung der Klimaanlage zugunsten der Motorregelung in einem PKW).

2.2 Optimierungs- und Lernverfahren

Eine *klassische* Definition der Optimierung findet sich beispielsweise in der Brockhaus-Enzyklopädie:

> *„Aufsuchen des kleinsten (Minimierung) oder größten (Maximierung) Wertes einer Funktion (Zielfunktion, Objektfunktion) in einem bestimmten, durch Nebenbedingungen, oft in Form von Gleichungen oder Ungleichungen beschriebenen (zulässigem) Bereich.“* [Bro71]

Diese Definition bezieht sich auf mathematische Verfahren zum Auffinden eines Optimums in Funktionen. Bezieht man jedoch die Informatik mit ein, so muss der Begriff Optimierung allgemeiner gefasst werden.

Umgangssprachlich versteht man meist unter Optimierung die

> *„Verbesserung eines Vorgangs oder Zustands bezüglich eines Gesichtspunktes wie zum Beispiel der Qualität, Kosten, Geschwindigkeit, Effizienz und Effektivität, manchmal auch zu Lasten eines anderen Aspektes“* [Wik06]

Wird diese Auffassung auf die Domänen der Mechatronik übertragen, so ist *die Optimierung eines mechatronischen Systems, die Suche nach einem optimalen Zustand oder Verhalten.* Dabei ist eine Unterteilung in (1) Optimierung der kinematischen Funktion, (2) Optimierung der dynamischen Funktion und (3) Optimierung der mechatronischen Funktion möglich. Für die Selbstoptimierung steht vor allem die Optimierung der dynamischen und der mechatronischen Funktion im Vordergrund, da diese durch Informationsverarbeitung und Regelungstechnik im Betrieb beeinflusst werden können.

Ein wesentliches Unterscheidungsmerkmal von Verfahren zur Optimierung ist der Einsatz von Modellen zur Optimierung. Grundsätzlich wird zwischen modellbasierten und verhaltensbasierten Verfahren unterschieden.

Modellbasierte Verfahren benutzen ein Modell des zu optimierenden Systems zur Bewertung eines Suchschrittes. Dies hat den Vorteil, dass die Optimierung

unabhängig vom realen System erfolgen kann. Auch können potentiell unsichere Optimierungsergebnisse „ausprobiert“ werden, ohne den technischen Prozess zu gefährden. Nachteil ist jedoch, dass ein Modell benötigt wird, das die reale Strecke oder die Umwelt allgemein, hinreichend genau abbilden muss. Des Weiteren ist bei einer veränderlichen Strecke oder Umwelt das Modell permanent mit der Realität abzugleichen, was im Normalbetrieb nicht trivial ist, da nicht mit speziellen Testfunktionen gearbeitet werden kann [DOM01]. Dies gilt besonders dann, wenn stark nichtlineare Systeme betrachtet werden. Außerdem muss bei der modellbasierten Optimierung die Art des Modells unterschieden werden. Es wird zwischen physikalisch deutbaren Modellen und universellen Approximatoren unterschieden [Kno01], [GLR01].

- *Physikalisch deutbare Modelle* beschreiben mit Hilfe von Differentialgleichungssystemen, die aus der Physik abgeleitet werden können, ein dynamisches Verhalten. Auch wird häufig von parametrierbaren Modellen gesprochen, was die Eigenschaft dieser Modelle unterstreicht, dass sie physikalisch deutbare Parameter wie Masse, Federkonstante, elektrischer Widerstand, Fluss usw. aufweisen. Diese Modelle entstammen der *theoretischen Modellbildung*, die *Systemkenntnis voraussetzt* [HGP98]. In vielen Fällen werden diese Modelle noch durch experimentelle Modellbildung ergänzt oder bestätigt (Parameter, Plausibilität).
- Zu den *universellen Approximatoren*, die durch Trainingsfunktionen an das reale System angepasst werden, zählen Verfahren wie neuronale Netze, Fuzzy-Systeme oder Kolmogorov-Grabor-Polynome[4] (siehe z. B. [Fer02]). Diese Verfahren gehören auch zur experimentellen Modellbildung, setzen aber keine physikalische Deutbarkeit voraus. Auch sind die mathematischen Darstellungsformen für klassische Verfahren der Reglersynthese nicht ohne weiteres anwendbar.

Bei der Modellierung nichtlinearer Systeme ist eine Kombination von physikalisch deutbaren Modellen und universellen Approximatoren üblich [Kno01].

Verhaltensbasierte Verfahren optimieren direkt am technischen Prozess – ein Modell ist deshalb nicht nötig. Das Verhalten des technischen Prozesses wird nach einer Veränderung (Optimierungsschritt) direkt bewertet. Viele verhaltensbasierte Ansätze verwenden, ähnlich modellbasierten Ansätzen zur Optimierung, auch Suchverfahren. Ein häufig verwendetes Verfahren ist das Verfahren des steilsten Abstiegs oder einfach Gradientenverfahren (vgl. z. B. [WJ95], [RN03]).

Der Vorteil der verhaltensbasierten Optimierung ohne Modell liegt in der Direktheit der Anwendung. Es wird weder ein Modell benötigt, noch eine Systemidentifikation. Fehler, die beispielsweise durch falsche Vereinfachungen oder Annahmen bei der Modellbildung auftreten können, sind ausgeschlossen. Außerdem werden Veränderungen des Systems im Betrieb direkt wahrgenommen. Besonders geeignet ist die direkte Online-Optimierung als Kombination

[4] Andrei Nikolajewitsch Kolmogorow, russischer Wissenschaftler und Mathematiker (* 12./25. April 1903 in Tambow; † 20. Oktober 1987 in Moskau.)

aus modellbasierter und verhaltensbasierter Optimierung. Bei diesem Verfahren wird nach einer modellbasierten Optimierung das Modell durch ein reales (Teil-)System ersetzt. Gute Voraussetzungen bieten Hardware-in-the-Loop-Prüfstände. Ein Beispiel für solch eine Optimierung am Prüfstand findet sich in [DOM01].

Optimierung ohne Modell hat jedoch auch wesentliche Nachteile. Zum Einen muss jeder Schritt am realen System *getestet* werden. Dabei darf der technische Prozess nicht so stark gestört werden, dass der Betrieb gefährdet ist. Auch können aus dem gleichen Grund nicht beliebige Anregungsfunktionen aufgeschaltet werden. Bei einer ungünstigen Anregungsumgebung können somit nicht alle Störeinflüsse berücksichtigt werden [Lau99].

2.2.1 Modellbasierte Verfahren

In vielen technischen Entwicklungen werden mathematisch-physikalische Modelle zur Analyse des Systemverhaltens genutzt. Solche Modelle spiegeln das Wissen über das dynamische Verhalten eines Systems wieder. Verfahren zur Steuerung und Regelung, die solche abstrakten Modelle benutzen, werden häufig auch als *modellbasierte Verfahren* bezeichnet. Das Modell spiegelt dabei ein bestimmtes gerichtetes Ein-/Ausgangsverhalten wieder, das dem des realen Systems angenähert ist. Die Verfahren zur Abbildung dieses Ein-/Ausgangsverhalten sind dabei sehr unterschiedlich. Hinsichtlich des mathematischen Ansatzes unterscheidet beispielsweise [Lau99] zwischen Signalmodellen, Prozessmodellen und wissensbasierten Ansätzen bei der automatischen Identifikation des technischen Prozesses.

[Kno01] unterscheidet bei der Unterteilung von adaptiven Reglern zwischen adaptiven Reglersystemen, basierend auf modellbasierten, nichtlinearen Modellen und solchen, die auf universellen Approximatoren beruhen. Unter universellen Approximatoren werden hierbei alle Verfahren verstanden, die ein Modellverhalten durch eine allgemeine mathematische Methode annähern, wie beispielsweise neuronale Netze oder Fuzzy-Logik. Diese Einteilung ist jedoch unpraktisch, wenn vorausgesetzt wird, dass ein Modell die abstrakte Darstellung eines realen Systems ist und dazu dient,

> *„... die als wichtig angesehenen Eigenschaften des Vorbilds auszudrücken und nebensächliche Eigenschaften außer Acht zu lassen, um durch diese Vereinfachung zu einem übersehbaren oder mathematisch berechenbaren oder zu experimentellen Untersuchungen geeigneten Modell zu kommen. ...“* [Bro71]

Diese Definition schließt zweifellos die universellen Approximatoren als Grundlage von Modellen ein. Wichtigster Unterschied ist jedoch, dass im Gegensatz zu den aus der physikalischen Sicht abgeleiteten Modellen universelle Approximatoren *nicht oder nur beschränkt* interpretier- oder analysierbare Parameter aufweisen. Dadurch ist eine mathematische Analyse, beispielsweise für den Nachweis der Stabilität, aufwändig oder unmöglich.

2.2.2 Modellbasierte Optimierung

Neben der Ausnutzung für die Regelung technischer Systeme ist es auch möglich, mathematisch-physikalische Modelle für die Optimierung zu nutzen. Dabei werden

im Allgemeinen die Parameter des Modell-Systems so verändert, dass ein gewünschtes Systemverhalten erzeugt wird. Die bestimmten Parameter werden anschließend für den Entwurf des realen Systems genutzt.

In vielen Fällen läuft die Optimierung auf das Aufsuchen eines lokalen oder globalen Minima bzw. Maxima hinaus. Bei nichtlinearen Problemen ist eine rein analytische Lösung nicht möglich. Werden gleichzeitig mehrere Zielgrößen optimiert, so kann es zu widersprechenden Zielen kommen. Eine Zielgröße kann dann nicht mehr verbessert werden, wenn nicht gleichzeitig eine andere verschlechtert wird. Alle Lösungen zwischen den Optima bilden dann eine paretooptimale Lösungsmenge [Hil01].

Grob lassen sich die wesentlichen Verfahren in (1) nichtlineare Optimierung und (2) stochastische Optimierung unterteilen. Zu der nichtlinearen Optimierung zählen die ein- und die mehrdimensionale Suche. Zu den stochastischen Verfahren zählen beispielsweise Simulated Annealing, Particle Swarm Optimization, evolutionäre Algorithmen und genetische Algorithmen. Dabei ist die nichtlineare Optimierung wesentlich für die Online-Optimierung in mechatronischen Systemen.

2.2.2.1 Nichtlineare Optimierung

Bei der unbeschränkten nichtlinearen Optimierung wird das Minimum oder das Maximum einer skalaren Funktion $f(\underline{x})$ des Vektors $\underline{x}$ gesucht.

$$\min_{\underline{x}} \{f(\underline{x})\} \tag{2.2.1}$$

Für die Lösung dieses Problems existieren eine Vielzahl von Verfahren. Ein häufig verwendetes Verfahren arbeitet nach der Iterationsvorschrift :

$$\underline{x}_{k+1} = \underline{x}_k + \alpha_k \underline{d}_k \tag{2.2.2}$$

Ausgehend von einem Punkt $\underline{x}_k$ wird ein neuer Punkt $\underline{x}_{k+1}$ bestimmt, wobei $\underline{d}_k$ die Richtung der Suche angibt und α_k die Schrittweite [BSMM97].

Das Gesamtproblem der Optimierung besteht hier für jeden Schritt aus den Teilproblemen:

1. Finde eine Suchrichtung $\underline{d}_k$ und
2. finde eine geeignete Schrittweite α_k.

Ziel ist die möglichst schnelle und möglichst genaue Annäherung an das gesuchte Minimum oder Maximum. Die Schrittweite kann durch eine Liniensuche bestimmt werden. Die Richtung wird durch die Untersuchung der Umgebung des Punktes $\underline{x}_k$ bestimmt. Dabei benutzen viele Verfahren, wie beispielsweise das Newton-Verfahren, die erste oder auch die zweite Ableitung der Zielfunktion f an der Stelle $\underline{x}_k$ (vgl. z. B. [Mün03]).

Aus den ersten partiellen Ableitungen $\frac{\partial f}{\partial x_i}$ bildet sich der Gradientenvektor oder Gradient ∇f:

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \tag{2.2.3}$$

Aus den zweiten Ableitungen lässt sich analog die so genannte Hesse-Matrix $\nabla^2 f$ bilden. Die Hesse-Matrix[5] bildet sich aus den partiellen zweiten Ableitungen einer mehrdimensionalen Funktion $f(x_1, ..x_n)$:

$$\mathrm{H}(f) = \left(\frac{\partial^2 f}{\partial x_i \partial x_j} \right) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{pmatrix} \tag{2.2.4}$$

Anmerkung: Die Hesse-Matrix kann auch zur Bestimmung von lokalen Minima oder Maxima dienen. Dazu müssen zunächst die kritischen Punkte einer Abbildung in $\mathbb{R}^n$ und die Definitheit der Hesse-Matrix H an diesen Punkten bestimmt werden. Ist die Hesse-Matrix H an einer Stelle positiv definit, so liegt dort ein lokales Minimum der Funktion vor. Falls H dort negativ definit ist, so handelt es sich bei diesem Punkt um ein lokales Maximum [Hör07].

2.2.2.2 Gradientenverfahren

Das so genannte Gradientenverfahren sucht den steilsten Abstieg mit Hilfe der Anweisung [BSMM97]:

$$\underline{d}_k = -\nabla f\left(\underline{x}_k\right) \tag{2.2.5}$$

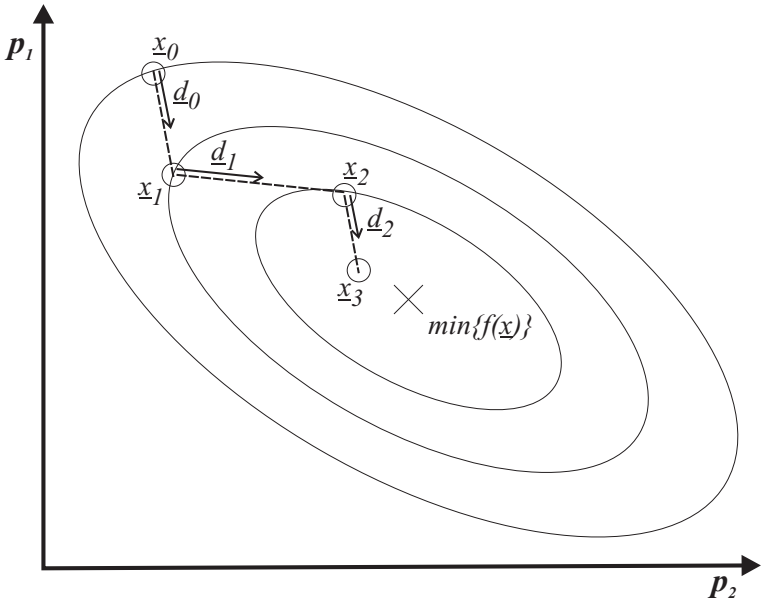

Abbildung 2.5: Prinzip des Optimierungsverlaufs eines Gradientenverfahrens

Der prinzipielle Verlauf einer Optimierung mit Hilfe des Gradientenverfahrens ist in Abbildung 2.5 zu sehen. Ausgehend von einem Punkt $\underline{x}_0$, wird mit Hilfe des Richtungsgradienten $\underline{d}_0$ der nächste Punkt $\underline{x}_1$ bestimmt. An diesem neuen Ort wird dann eine neue Richtung $\underline{d}_1$ bestimmt und so fort. Die Ellipsen stellen Höhenlinien der quadratischen Zielfunktion dar [Mün03].

[5] nach Otto Hesse (* 22. April 1811, † 4. August 1874), deutscher Mathematiker.

Die Nachteile des Verfahrens liegen vor allem in seiner langsamen Konvergenz. Weiterführende Verfahren berücksichtigen die Krümmung der Zielfunktion. Eine nähere Betrachtung dieser verbesserten Verfahren findet sich z. B. in Bezug auf die Optimierung von mechatronischen Systemen in [Mün01], [Mün03] und allgemein bei [Opt07].

2.2.2.3 Mehrgrößen-Optimierung

Bei Mehrgrößenproblemen geht es, wie bereits erwähnt, um die gleichzeitige Optimierung verschiedener Zielgrößen. Aus der Zielgrößenfunktion $f(\underline{x})$ bei der nichtlinearen Optimierung wird ein Zielgrößenvektor $\underline{f}(\underline{x})$. Das Optimierungsproblem lässt sich analog beschreiben mit:

$$\min_{\underline{x}} \left\{ \underline{f}(\underline{x}) \right\} \tag{2.2.6}$$

Die Optimierung läuft jedoch nicht auf einen Punkt hinaus, sondern auf eine Lösungs*menge*, die sogenannte Paretofront oder Paretomenge. Im Falle zweier Zielgrößenfunktionen ist die Paretofront eine eindimensionale Funktion. Alle Lösungen auf dieser Funktion sind paretooptimale Lösungen des Optimierungsproblems. Viele technische Anwendungen benötigen jedoch nur eine Lösung: die Lösungsmenge. Somit wird zusätzlich zu der Mehrgrößenoptimierung noch ein entsprechendes Selektionsverfahren benötigt, das einen bestimmten Punkt auswählt. Die Bestimmung von Paretomengen ist im Allgemeinen rechenintensiv. Eine vorab bestimmte Paretomenge kann jedoch beispielsweise durch ein Online-Selektionsverfahren für die Selbstoptimierung genutzt werden. Die Vielzahl von Verfahren zur Bestimmung einer Paretomenge soll an dieser Stelle nicht diskutiert werden.

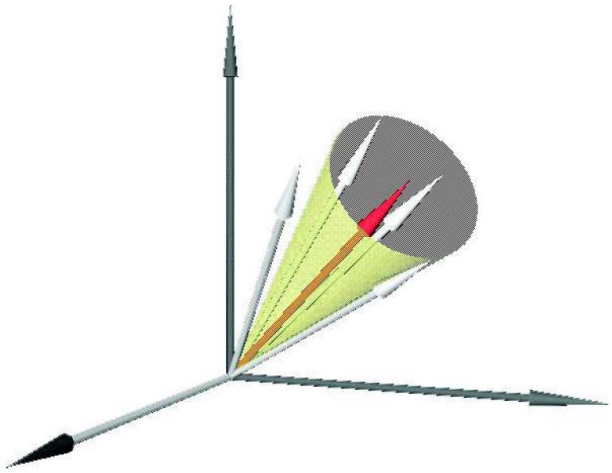

Abbildung 2.6: Suchrichtungsbestimmung durch Kegel um Gradienten

Wird von vornherein nur ein bestimmter Punkt auf der Paretomenge gesucht, können andere Punkte während der Annäherung ausgeschlossen werden. Ziel ist es hier, möglichst nahe an den durch Selektion bzw. Gewichtung angestrebten Zielbereich der Paretomenge heranzukommen. Auf diese Art und Weise reduziert sich der Berechnungsaufwand erheblich. Neben der Gewichtung der Zielgrößen als ein Kriterium für die Auswahl eines paretooptimalen Punkts existieren noch andere Verfahren, die von vornherein nur die Berechnung eines bestimmten Paretopunktes anstreben. Die Diskussion der verschiedenen Verfahren soll in dieser Arbeit nicht

erfolgen, jedoch ist ein Verfahren besonders zu beachten, da es sich aus dem Gradientenverfahren für nichtlineare Optimierung ableiten lässt. Dieses Verfahren wurde am Mechatronik Laboratorium Paderborn für das Werkzeug MOPO (Multi-Objective Parameter Optimization) weiterentwickelt [Kas85], [KLJS90] und [Mün01].

Kerngedanke ist die Ermittlung einer Abstiegsrichtung für *alle* Zielgrößen. Ist diese gefunden, wird in dieser Richtung gesucht. Als geometrisch anschauliches Modell dient ein Kegel, der möglichst eng alle Gradienten einschließt. Die Suchrichtung $\underline{d}_k$ liegt dann genau in entgegen gesetzter Richtung der Kegelachse. Abbildung 2.6 [Mün01] stellt diesen Zusammenhang grafisch dar. Für eine weitergehende Betrachtung des Verfahrens wird insbesondere auf [Mün01] und [Mün03] verwiesen.

2.2.3 Verhaltensbasierte Verfahren

Während modellbasierte Verfahren aus der Regelungstechnik abgeleitet wurden, stammen verhaltensbasierte Verfahren aus der Informatik. Verschiedene Ansätze der künstlichen Intelligenz (KI) erwiesen sich in der Praxis als nur schwer anwendbar. Insbesondere bei autonomen Robotern zeigen komplexe Planungsverfahren ihre Schwächen. Der Durchgriff zwischen Umweltreiz und Reaktion der Maschine war nicht befriedigend. Außerdem war die Formulierung einfacher Verhaltensweisen nur schwer umzusetzen [Ark98]. Für die Robotik wurden neue Ansätze für Abbildung und Umsetzung von Verhalten benötigt. Betrachtet man das Verhalten von einfachen Lebewesen, so lassen sich viele Verhaltensweisen durch direkte Zusammenhänge zwischen Reiz und Reaktion erklären. Grundlagen hierzu stammen aus Biologie, Verhaltensforschung und Psychologie, wie die *klassische Konditionierung* nach Pawlow[6] [Paw55] oder die *operante Konditionierung* nach Skinner[7] [Ski53] (vgl. insb. Abschnitt 2.2.5). Werden solche einfachen Verhalten mehrfach kombiniert und die Reize gewichtet oder durch zeitabhängige Parameter verstärkt, lassen sich *scheinbar* komplexe Verhalten beschreiben. Diese Grundidee liegt verhaltensbasierten Systemen zugrunde. Dabei werden nach dem beschriebenen Modell Reize, die durch Sensoren erfasst werden, mit Aktoren verbunden. Ein sehr anschauliches Beispiel ist das sog. *Braitenberg Vehicle* [Ark98].

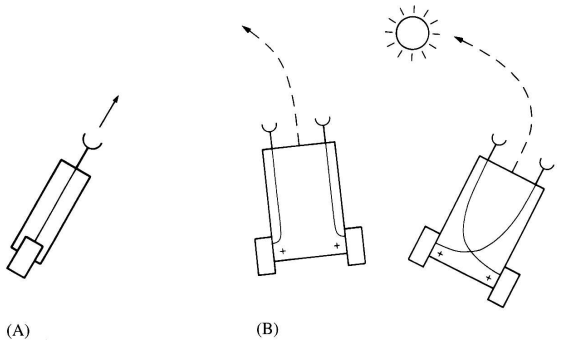

Abbildung 2.7: Braitenberg Vehicle

[6]Iwan Petrowitsch Pawlow (* 14. September/26. September 1849 in Rjasan; † 27. Februar 1936 in Leningrad) russischer Mediziner und Physiologe.

[7]Burrhus Frederic Skinner (* 20. März 1904 in Susquehanna, Pennsylvania; † 18. August 1990 in Cambridge, Massachusetts), amerikanischer Psychologe.

Das Fahrzeug (A) besteht aus einem Sensor und einem Motor. Die Bewegung dieses Fahrzeugs geht immer in der Richtung des Reizes (Licht), jedoch können Einflüsse durch die Umwelt, die auf das Fahrzeug wirken, nicht kompensiert werden – schon ein leichtes Rutschen oder eine raue Fahrbahn würden das Fahrzeug außer Kurs bringen. Das Fahrzeug (B) besteht aus zwei Motoren und zwei Sensoren. Werden die Sensoren direkt mit dem Motoren verbunden, so weicht das Fahrzeug dem Reiz aus, da die dem Licht zugewandte Seite ein stärkeres Signal produziert als die abgewandte Seite. Werden die Sensoren über Kreuz mit den Sensoren verbunden, so fährt das Fahrzeug auf das Licht zu. Bewegt man die Lichtquelle im Raum, folgt der Roboter *wie eine Katze einem Wollknäuel*. Dieses einfache Beispiel zeigt anschaulich, dass durch einen einfachen Zusammenhang ein (scheinbar) komplexes Verhalten erzeugt werden kann.

Das Problem ist allerdings nicht vollständig gelöst, da zwischen Reiz und Reaktion eine Verstärkung liegt, die im Allgemeinen die Dynamik des Systems berücksichtigen muss. Somit ist dieses System keineswegs der Ersatz einer Regelung, sondern erzeugt vielmehr nur eine Sollgröße. Bleibt dies unberücksichtigt, so stellt sich unter Umständen ein völlig anderes Verhalten ein, wie ursprünglich angenommen, wie die Abbildung 2.8 [Ark98] zeigt.

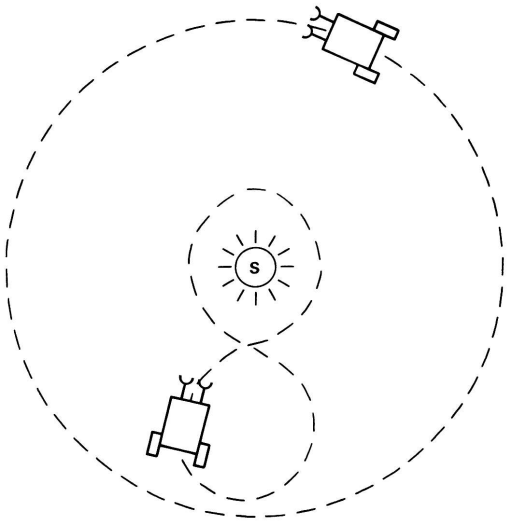

Abbildung 2.8: Braitenberg Vehicle mit unbestimmten Verhalten

Das Verhalten der Roboter in Abbildung 2.8 überrascht aus regelungstechnischer Sicht kaum. Der innere Roboter *übersteuert*, d. h. der Lenkwinkeleinschlag passt nicht zur Dynamik des Systems. Ähnlich ist es beim äußeren Fahrzeug, bei dem der Lenkwinkeleinschlag nicht ausreicht. Das Problem wird in der Praxis durch Überlagerung von verschiedenen Verhalten gelöst. Meistens ist dazu eine weitere Sensorgröße oder eine abgeleitete Größe nötig.

Ein weitergehender Ansatz überlagert die eingehenden Reize zu Potentialfeldern ([Ark98] aus [Lat90]). Potentialfelder werden auch als *virtuelle Reize* genutzt, z. B. bei einer bekannten relativen Position zwischen zu steuerndem System und Ziel bzw. Hindernis.

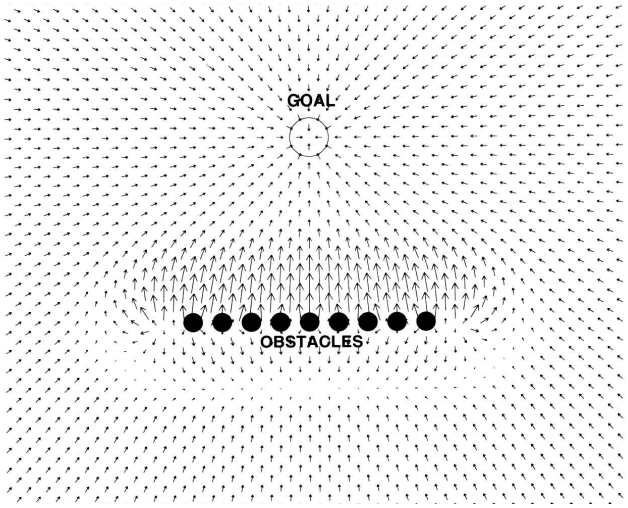

Abbildung 2.9: Potentialfeld zur Vorgabe von Steuergrößen

Abbildung 2.9 [Ark98] zeigt beispielhaft ein solches Potentialfeld als Steuervorgabe. Die Pfeile geben für die jeweilige Position eines Fahrzeugs eine Richtung und eine Geschwindigkeit vor.

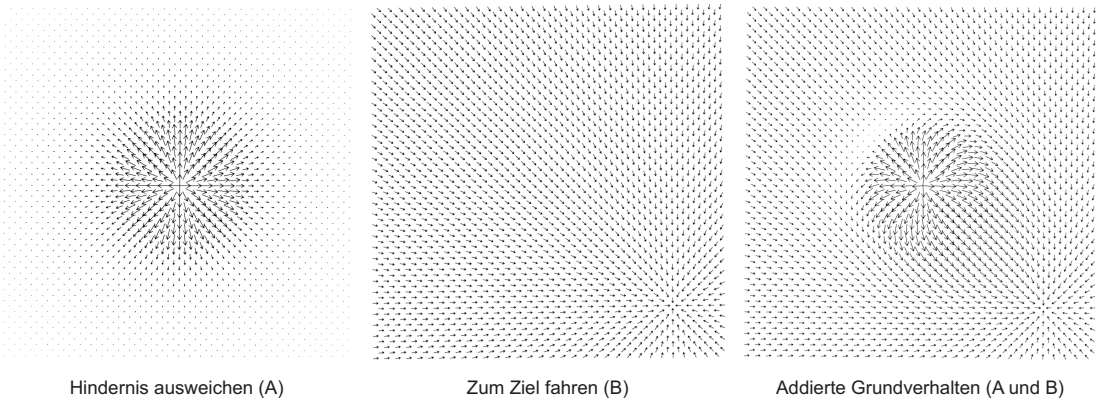

Abbildung 2.10: Addition von Potentialfeldern

Besonders interessant ist die Bildung von solchen Potentialfeldern. Sie werden aus einfachen Grundverhalten zusammengesetzt. In den Abbildungen 2.10 [Ark98] ist zu sehen, wie ein komplexeres Verhalten (2.10, rechts) durch die Addition zweier einfacher Verhalten (2.10, links und 2.10, Mitte) erzeugt wird.

Das Ergebnis ist eine Trajektorie (siehe Abbildung 2.11 [Ark98]), die als Vorgabe für die Bahnsteuerung eines Fahrzeugs, wie z. B. eines Roboters, dienen kann[8].

[8]Das dargestellte Problem wird in [Ark98] noch näher diskutiert und weiter verfeinert. An dieser Stelle soll nur das Grundprinzip als Grundlage für ein allgemeines Verständnis dargestellt werden.

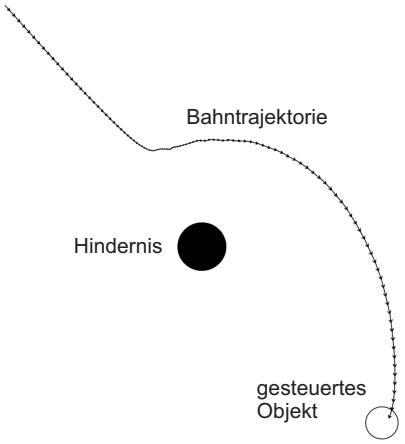

Abbildung 2.11: Bahntrajektorie durch Überlagerung von Verhalten

Aus dem Beispiel wird zum einen das mächtige Potential der verhaltensbasierten Programmierung deutlich, das es erlaubt, durch die Superposition einfacher Grundverhalten zu komplexen Verhalten zu gelangen. Zum anderen wird aber auch deutlich, dass zwar eine Rückkopplung zur Umgebung des technischen Systems vorhanden ist, aber die Dynamik unberücksichtigt bleibt. Das Zusammenspiel zwischen Umgebung und technischen System wird *quasistatisch* behandelt. Die Regelung der Dynamik bleibt unterlagerten Systemen vorbehalten. Wie die Dynamik der Umgebung in dem vorgestellten Ansatz mitberücksichtigt werden kann, ist noch zu untersuchen.

2.2.4 Lernverfahren – Grundlagen

Der Begriff Lernverfahren trifft im Zusammenhang mit Selbstoptimierung zwei wichtige Bereiche: den Bereich der *Lerntheorien* und den der *künstlichen Intelligenz* (KI). Beide sehr umfangreiche Bereiche können hier nur in Ansätzen dargestellt werden. Für die strukturierte Entwicklung von selbstoptimierenden mechatronischen Systemen sind jedoch einige Theorien von grundlegender Bedeutung. Dabei ist es notwendig, die Grundlagen des Lernens an sich zu betrachten. Eine Brücke zwischen dem Lernen beim Menschen, insbesondere bei Kindern und der Informatik bilden hier die Arbeiten von R. M. Gagné[9], der Lernen im Wesentlichen als Informationsverarbeitung angesehen hat. Betrachtet man die verschiedenen Ansätze zu maschinellem Lernen der KI, so wird deutlich, dass sich viele Arbeiten an denen von Gagné orientiert haben.

Lernen wird von Gagné nicht als ein Phänomen aufgefasst, das durch einfache Theorien beschrieben werden kann. Als Prinzip setzt er voraus, dass Lernen als Phänomen genauer untersucht und tiefer verstanden werden kann, das von Wechselwirkungen zwischen dem Individuum und seiner Umwelt abhängt. Lernen wird als beobachtbarer *Vorgang* und nicht als Ereignis verstanden.

[9]Robert Mills Gagné (* 21. August 1916 in North Andover; † 28. April 2002) war ein US-amerikanischer experimenteller Psychologe und Pädagoge.

Für Gagné ist Lernen eine...

> *„... Änderung in menschlichen Dispositionen oder Fähigkeiten, die erhalten bleibt und nicht einfach dem Reifungsprozess zuzuschreiben ist.“* [Gag80]

Die Art des Wandels, die man Lernen nennt, zeigt sich als eine Verhaltensänderung, und man schließt auf Lernen, indem man vergleicht, welches Verhalten möglich war, bevor das Individuum in eine Lernsituation gebracht wurde, und welches Verhalten nach einer solchen Behandlung gezeigt wird. Die Änderung kann in einer verbesserten Fähigkeit zu einer bestimmten Leistung bestehen, und so ist es tatsächlich oft. Es kann auch eine veränderte Verhaltensbereitschaft von der Art sein, die man Einstellung, Interesse oder Wert nennt. Die Veränderung muss den Augenblick überdauern; sie muss über eine gewisse Zeitspanne erhalten bleiben können.

Bei jedem Lernvorgang gibt es drei wesentliche Elemente: Es gibt einen *Lernenden (1)*. Seine wichtigsten Teile sind in diesem Fall seine Sinnesorgane, sein zentrales Nervensystem (ZNS) und seine *Muskeln (2)*. Ereignisse in der Umwelt wirken auf die Sinne des Lernenden und setzen Ketten nervöser Impulse in Gang, die durch sein ZNS, besonders das Hirn, organisiert werden. Diese Nerventätigkeit läuft in bestimmten Sequenzen und Mustern ab, welche die Natur des organisierenden Prozesses selbst verändern, und diese Wirkung zeigt sich als Lernen. Schließlich wird diese Nerventätigkeit in *Handlung (3)* übersetzt, die vielleicht als Muskelbewegung in ausführenden Reaktionen verschiedener Art zu beobachten ist. Die Ereignisse, welche die Sinne des Lernenden reizen, werden zusammenfassend als die *Reizsituation* bezeichnet. Wenn ein einzelnes Ereignis zu unterscheiden ist, wird es häufig als *Reiz* benannt. Die Handlung, die aus der Reizung und der nachfolgenden Nerventätigkeit hervorgeht, wird als *Reaktion* bezeichnet.

Ein Lernvorgang findet also statt, wenn die Reizsituation auf den Lernenden in einer Weise wirkt, dass sich seine Leistung von einem Zeitpunkt vor dieser Situation zu einem Zeitpunkt nach dieser Situation ändert. Es ist die Änderung der Leistung, die zu dem Schluss führt, dass Lernen stattgefunden hat [Gag80].

Nach Gagné werden acht Lerntypen unterschieden:

1. Signallernen: einfache reflexartige Reaktion, die als Folge auf einen wiederkehrenden Reiz ausgelöst wird (Pawlowsche Hund[10]),

2. Reiz-Reaktions-Lernen: einfache Reaktion, die durch positive Verstärkung erlernt wird, z. B. Belohnung bei erfolgreichem Ergebnis, Bestrafung bei nicht erfolgreichem,

3. motorische und

4. sprachliche Kettenbildung: eine Folge von Handlungen oder anders dargestellten Abläufen wird so lange wiederholt, bis sie eingeprägt ist,

5. Lernen von Unterscheidungen: Zuordnen von Eigenschaften zu Objekten oder umgekehrt. Einordnen in Kategorien,

[10] Iwan Petrowitsch Pawlow, Nobelpreisträger; (* 14. September/26. September 1849 in Rjasan; † 27. Februar 1936 in Leningrad) war ein russischer Mediziner und Physiologe. U. a. bekannt für seine Arbeiten zur Verhaltensforschung an Tieren.

6. Begriffslernen: ein abstrakter Zustand wird aus einer Vielfalt physikalischer Konstellationen gelernt, z. B. der Begriff *mittlerer* als ein Objekt zwischen zwei anderen,

7. Regellernen: ein Objekt dehnt sich bei Erwärmung aus: das allgemeine Prinzip als Regel erkennen und übertragen können, und

8. Problemlösen: das Ausdenken einer abgewandelten oder neuen Regel für bisher unbekannte Probleme.

Diese Lernarten bauen nach Gagné hierarchisch aufeinander auf. Die vorausgehende einfachere muss jeweils beherrscht werden, bevor man eine komplexere angehen kann. Gagné hat daraus für schulisches Lernen eine Art Karte des zu lernenden Stoffes entwickelt, wobei der Lehrer die Teilaufgaben in einer hierarchischen Sequenz anordnet (Lernstruktur).

Die Informatik versucht mit Hilfe der künstlichen Intelligenz (KI), diese Lerntypen auf technische Systeme zu übertragen. Die Lerntypen *Signallernen* und *Reiz-Reaktions-Lernen* entsprechen der klassischen Konditionierung. Ihre Entsprechung findet diese Ebene des Lernens im sogenannten *Reinforcement Learning* oder *selbstverstärkendem Lernen* [KLM96], [Sto00].

Das wesentliche charakteristische Merkmal für Reinforcement Learning, das es von anderen Lernverfahren unterscheidet, ist, dass es Trainings-Informationen nutzt statt gegebener (Handlungs-) Anweisungen bei richtigen Aktionen. Dieses Vorgehen benötigt eine aktive Untersuchung des Aktionsergebnises, für eine explizite *Trial-and-Error*-Suche nach gutem und richtigem Verhalten. Eine rein bewertende Antwort *(evaluative feedback)* zeigt, wie gut das Ergebnis einer Aktion ist, jedoch nicht, ob es die best- oder schlechteste mögliche Aktion ist. Bewertende Antworten sind die Basis vieler Verfahren, wie der Funktionsoptimierung (siehe auch Abschnitt 2.2.2.1 Nichtlineare Optimierung), aber auch evolutionärer Methoden. Rein instruierende Antworten (als Handlungsanweisungen) zeigen, welche Aktion die richtige ist (oder gewesen wäre), aber berücksichtigen nicht die aktuelle Aktion. Diese Art des Lernens ist die Grundlage des *Unterweisenden Lernens (supervised learning)* und findet sich beispielsweise beim Training von künstlichen Neuronalen Netzen.

2.2.5 Selbstverstärkendes Lernen (Reinforcement Learning)

Im deutschen Sprachraum wird für das Lernverfahren für technische Systeme der Informatik im Allgemeinen die Bezeichnung Reinforcement Learning verwendet. Der Begriff selbstverstärkendes Lernen findet sich in der Psychologie und in der Pädagogik [Paw55], [Ski53] (i. A. Lernen durch positive/negative Erfahrung).

Im technischen Sinne sind es verschiedene Methoden, die durch Ausprobieren *(trial-and-error)* das Verhalten eines technischen Systems verbessern sollen. Ein guter Weg, Reinforcement Learning zu verstehen, besteht darin, verschiedene Beispiele und mögliche Anwendungen zu betrachten, die ihre Entwicklung geleitet haben.

- Ein Schachmeister macht einen Zug. Seine Wahl wird von zwei Bereichen beeinflusst: zum einen von Planung, dem Wissen über mögliche Züge und Gegenzüge und zum anderen durch plötzliche intuitive Regeln über bevorzugte Positionen und Züge.

- Ein adaptiver Regler passt die Parameter einer Raffinerie in Echtzeit an. Der Regler optimiert den Kompromiss zwischen Ausbeute/Kosten/Qualität auf der Basis von spezifizierten Grenzkosten, ohne sich dabei strikt an die ursprünglich definierten Punkte zu halten.

- Ein mobiler Roboter entscheidet, ob er einen weiteren Raum betreten soll, um dort nach Abfall zu suchen, oder ob er zurück zur Ladestation fährt. Er trifft seine Entscheidung auf der Basis der Erfahrungen, die er in der Vergangenheit gemacht hat, wie schnell er in der Lage war, die Ladestation zu finden.

Den Beispielen gemeinsam ist, dass Entscheidungen, die zukünftige Ereignisse beeinflussen (der nächste Zug beim Schachspiel, der Größe der Reserve in der Raffinerie, der nächste Ort des Roboters), auf der Basis von *Erfahrung* der Vergangenheit getroffen wurden. Die richtigen Entscheidungen implizieren indirekt die Berücksichtigung von späteren Konsequenzen. Dies erfordert Voraussicht und Planung.

In den Beispielen kann die Leistung mit der Erfahrung gesteigert werden. Der Schachmeister verfeinert seine Intuition, die er zum Überdenken der Positionen seiner Figuren benötigt, der Roboter, wie viel Zeit ihm noch bleibt, bis er zum Nachladen fahren muss. Die mitgebrachten Erfahrungen aus vorherigen Aufgaben, unabhängig davon, ob sie selbst gelernt oder vorgegeben (fest programmiert) wurden, beeinflussen das aktuelle (Lern-)Verhalten. Jedoch ist für die Verbesserung des Verhaltens immer eine Interaktion mit der Umgebung notwendig [SB98].

Nach [KLM96] ist Reinforcement Learning ein Problem, dem ein Agent (vgl. 2.5) gegenübersteht, der mit Hilfe von Trial-and-error-Interaktionen mit einer dynamischen Umgebung ein Verhalten erlernen muss. Es besteht eine enge Verbindung mit dem Begriff selbstverstärkendes Lernen aus der Psychologie, wenn auch der Begriff *selbstverstärkend* anders verstanden werden muss.

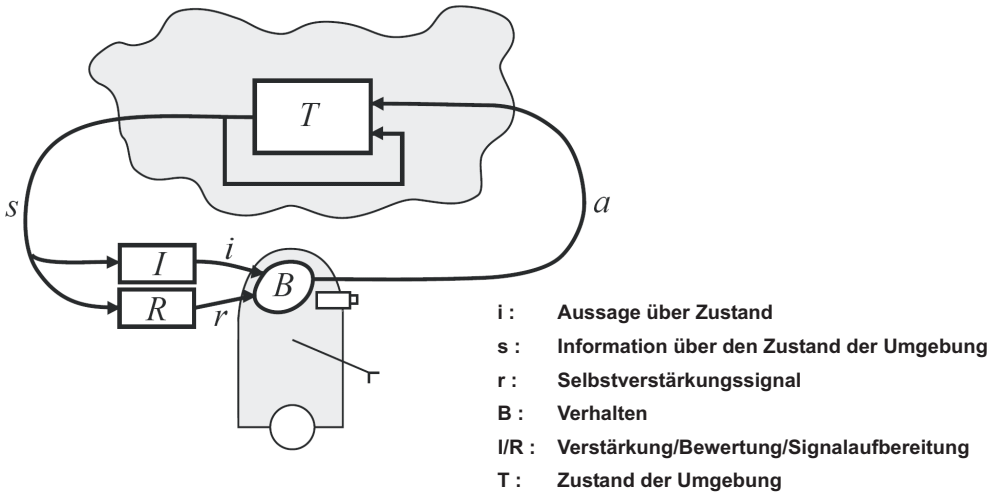

Abbildung 2.12: Das Standardmodell für Reinforcement Learning

In Abbildung 2.12 ist das sogenannte Standardmodell für Reinforcement Learning nach [KLM96] dargestellt. Es gilt unabhängig von der eingesetzten Methode und stellt den Zusammenhang bzw. den Ablauf des Lernens schematisch dar.

Die Darstellung ist folgendermaßen zu verstehen:

In jedem Schritt der Interaktion mit der Umgebung erhält der Agent den Eingang i als eine Aussage über den aktuellen Zustand s der Umgebung. Daraufhin wählt

der Agent eine Aktion a, die als Ausgang (Reaktion) umgesetzt wird. Die Aktion verändert den Zustand der Umgebung. Der Wert dieses Zustandswechsels wird an den Agenten als Selbstverstärkungssignal r (reinforcement signal) weitergeleitet. Das Verhalten B (behavior) des Agenten soll Aktionen auswählen, die auf lange Sicht zu einem Anstieg der Selbstverstärkungssignale s führen sollen. Dies kann er durch eine systematische Anwendung von Trial-and-error erreichen. Zur Steuerung dieses Verfahrens existieren zahlreiche, teils sehr unterschiedliche Algorithmen. Eine gute Übersicht der verschiedenen Methoden findet sich in [KLM96].

Insgesamt lässt sich festhalten, dass das Reinforcement Learning ein Verfahren ist, das an vielen Stellen Parallelen zu modellbasierten Verfahren zeigt. Durch die Interaktion mit der Umgebung ist eine Optimierung an die Wirklichkeit sichergestellt. Jedoch ist dieses Verfahren nicht oder nur begrenzt auf Anwendungsfälle übertragbar, bei denen ein Fehlerfall im Sinne von Versagen bei einem Schritt nicht erlaubt ist, wie beispielsweise bei sicherheitskritischen Anwendungen.

2.2.6 Neuronale Netze

Künstliche neuronale Netze gehören zu den allgemeinen Approximatoren. Im englischen Sprachraum werden diese als *artificial neural networks* (ANN) bezeichnet [GP89]. Sie eignen sich gut zur Abbildung von nichtlinearen Systemen, deren Struktur nicht sicher bekannt ist und die deshalb nicht oder nur schwer mit parametrierbaren Modellen zu modellieren sind. Durch ANN können alle nichtlinearen Funktionen, welche die Bolzano-Weiserstrass-Eigenschaft besitzen [BSMM97], beliebig genau approximiert werden. Somit ist es naheliegend, ANN auch zur Abbildung nichtlinearer Systeme und zur nichtlinearen Regelung einzusetzen [Kno01]. Jedoch hat sich gezeigt, dass sich beim Einsatz als direktes (PID-)Reglerglied neuronale Netze wie auch Fuzzy-Logik als zu komplex erweisen [Lau95].

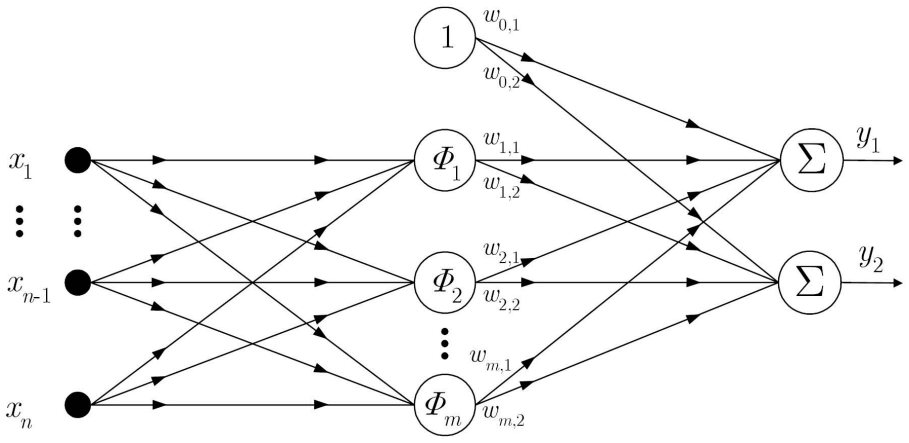

Abbildung 2.13: Radial-Basis-Funktionen (RBF) Netz

Die bedeutendste und meist angewandte Grundstruktur ist das mehrschichtige Perzeptionsnetz (engl. *multilayer perception* (MLP)). Diese Struktur wird sowohl zur Identifikation als auch zur Regelung nichtlinearer Systeme eingesetzt [HD99]. MLP sind nichtlinear in den unbekannten Parametern, was sowohl beim Training des Netzes als auch bei der Stabilität Nachteile mit sich bringt. Aus diesem Grund gewinnen

Radial-Basis-Funktionen-Netze (RBF) zunehmend an Bedeutung. Diese werden häufig zur Identifikation von nichtlinearen Systemen eingesetzt. Die Struktur ist in Abbildung 2.13 [CB92, PS92] dargestellt. Sie besteht aus einer Eingangsschicht, die den Vektor $x = [x_1, x_2, ...x_n]^T$ der Eingangssignale $x_i, i = 1, ..., n$ puffert, einer versteckten Schicht, welche die Neuronen mit dem Vektor Φ der RBF-Aktivierungsfunktion $\Phi_j, j = 1, ..., m$ enthält und einer Ausgangsschicht, in der die Aktivierungen summiert werden. Die mathematische Ein-/Ausgangsbeschreibung eines RBF-Netzes ist durch

$$y_l(k) = w_{0,l} + \sum_{j=1}^{m} w_{j,l} \Phi_j(\|x(k) - c_j\|) \tag{2.2.7}$$

gegeben. $y_l(k)$ ist das l-te Ausgangssignal, $x(k)$ der Eingangsvektor, $w_{j,l}, j = 0, ...m$ sind die Netzgewichte, und c_j sind die Zentren der RB-Funktionen. Mit $\|x(k) - c_j\| = r$ als Abstandsmaß eines Eingangsvektors von einem Zentrum ist $\Phi_j(r)$ die RB-Funktion. Für die Anwendung von neuronalen Netzen zur Regelung finden sich verschiedene Ansätze. Dies sind unter anderem:

- das Kopieren eines existierenden Reglers,
- die adaptive Regelung mit Vergleichsmodell,
- die Regelung mit internem Prozessmodell.

Erstere hat Nutzen bei der Regelung von Prozessen, bei denen die (nichtlineare) Regelung bisher durch das Bedienpersonal durchgeführt wird. Dabei wird das neuronale Netz, das die Regelungsaufgabe übernehmen soll, mittels eines Lernverfahrens trainiert. Als Eingangsgröße für das Lernverfahren kann direkt der Fehler zwischen dem Stellsignal und dem Ausgangssignal des neuronalen Netzes genutzt werden.

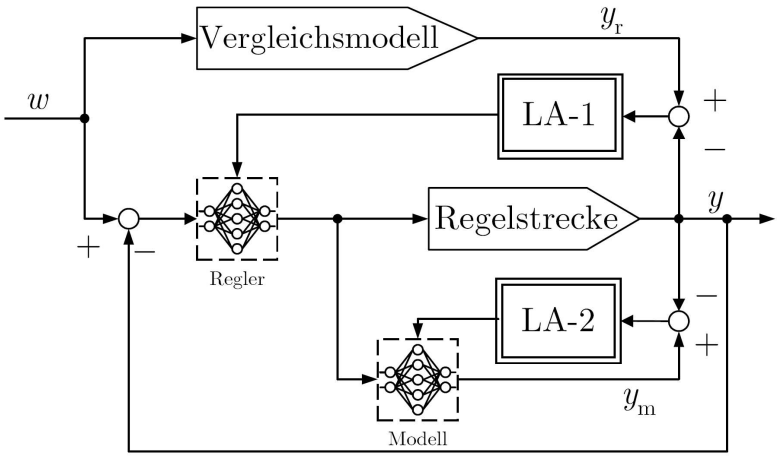

Abbildung 2.14: Indirekte adaptive Regelung mit Vergleichsmodell

Im zweiten Fall (Abbildung 2.14 [Kno01]) wird die Regelung durch neuronale Netze ersetzt. Mit Hilfe eines Lernverfahrens wird das Netz so trainiert, dass der Fehler zwischen Grundregelkreis und Vergleichsmodell klein wird. Da eine unbekannte Regelstrecke zwischen Netz und Fehlersignal liegt, kann der Fehler nicht direkt ermittelt werden. Um den Fehler zu gewinnen, muss auch das Übertragungsverhalten

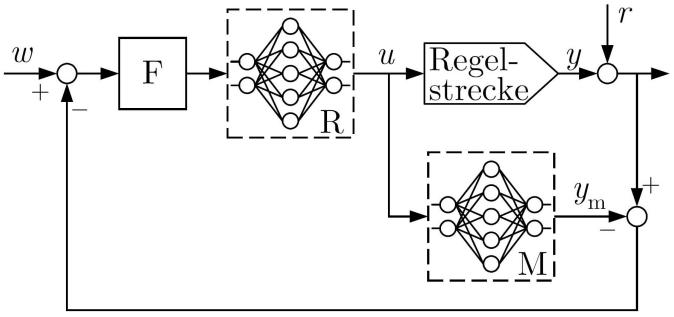

Abbildung 2.15: Regelung mit internem Prozessmodell

der Strecke durch ein zweites Netz abgebildet werden. Über dieses Netz kann dann der Fehler zum Netz im Regler geführt werden.

Der letzte Ansatz (Abbildung 2.15 [HSŻG92]) ist nur für stabile Regelstrecken mit stabiler Nulldynamik geeignet. Dabei wird durch ein parallel geschaltetes Netz, das die Streckendynamik abbildet, erreicht, dass nur die Störungen zurückgeführt werden, so dass ohne Störung eine Steuerung vorliegt. Das als Regler vorgeschaltete Netz enthält das inverse Streckenmodell. Damit kann durch den Filter F das Verhalten des geschlossenen Systems eingestellt werden.

Ein Problem bei der Anwendung von ANN ist die Abstimmung der Netzstruktur auf das jeweilige abzubildende (Teil-)System. Sie beeinflusst die Wahl der Optimierung der Netze (Verfahren), die erzielbare Reglergüte (bei Anwendung als Regler), die Rechenzeit und die Genauigkeit. Auf die Abstimmung soll hier im Detail nicht eingegangen werden. Es wird nur darauf hingewiesen, dass sich ein großer Teil der Forschung im Bereich der ANN auf diese Fragestellung konzentriert. [Kno01] weist darauf hin, dass bis heute *kein* Verfahren existiert, mit dem ein systematischer Entwurf bezüglich der Netztopologie und der Neuronenzahl in vertretbarer Zeit möglich ist. Eine weitere Anwendung ist der Einsatz von neuronalen Netzen zur Kompensation von nichtlinearem Streckenverhalten. Grundidee ist hierbei, ein lineares Übertragungsverhalten zu erzeugen, damit lineare Regler eingesetzt werden können. Dabei wird eine nichtlineare Strecke durch ein neuronales Netz so ergänzt, dass das Gesamtübertragungsverhalten wieder linear ist.

Ein Beispiel für dieses Verfahren ist die Kompensation von nichtlinearen Reibungseffekten [KHH04] in einem einfachen elektromechanischen Positioniersystem [HW98].

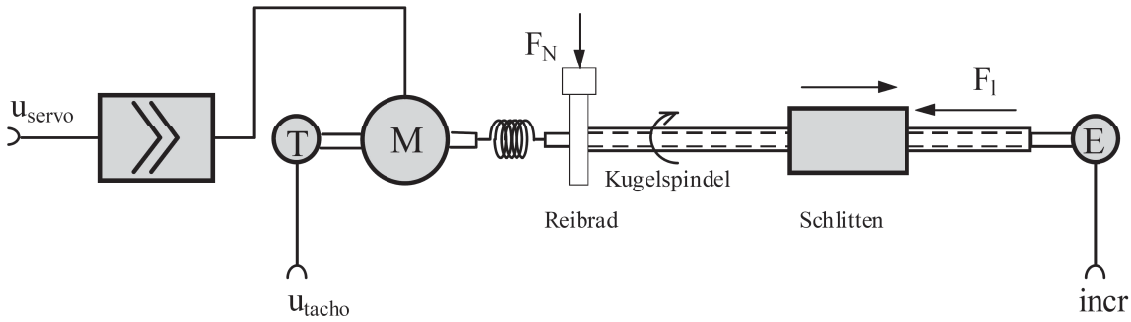

Abbildung 2.16: Elektromechanisches Positioniersystem (EMPS)

Das in Abbildung 2.16 [HW98] schematisch dargestellte elektromechanische Positioniersystem besteht aus einem stromgeregelten Gleichstrommotor und einer linearen Positioniereinheit. Die lineare Schlittenbewegung wird durch einen spielfreien Kugelgewindetrieb erzeugt. Für die Regelung werden die Ausgangsspannung u_{tacho} des motorseitigen Gleichstrom-Tachogenerators sowie der Zählerausgang *incr* des inkrementellen Drehgebers zur Messung der Schlittenposition verwendet. Stellgröße ist die Eingangsspannung u_{servo} für den Motorstromsollwert. Im System tritt trockene Reibung durch den Kugelgewindetrieb auf, der zur Vermeidung von Lose verspannt ist. Mit Hilfe eines Reibrades können unterschiedliche Reibbedingungen eingestellt und untersucht werden [HW98].

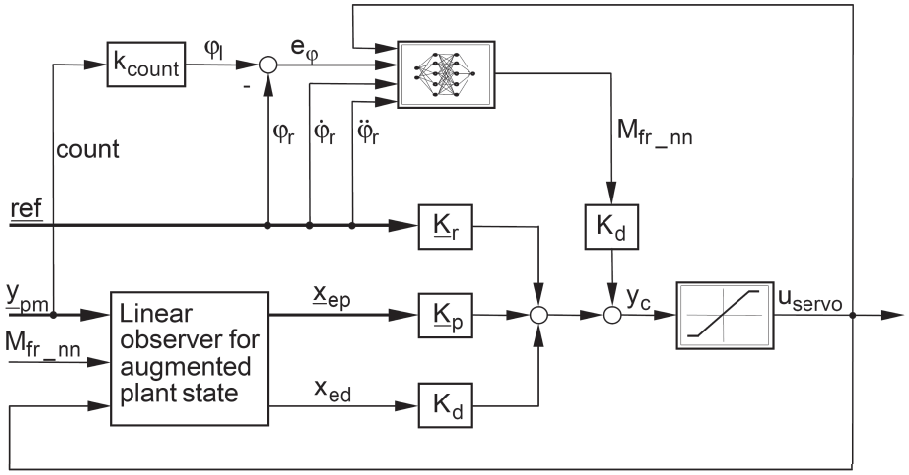

Abbildung 2.17: Struktur der Regelung mit Aufschaltung

Das Konzept der Aufschaltung ist in Abbildung 2.17 [HW98] dargestellt. Die Reglerstruktur mit Beobachter soll an dieser Stelle nicht näher betrachtet werden; sie wird in verschiedenen Arbeiten untersucht und diskutiert, unter anderem in [HW98], [Hen97], [HJW02].

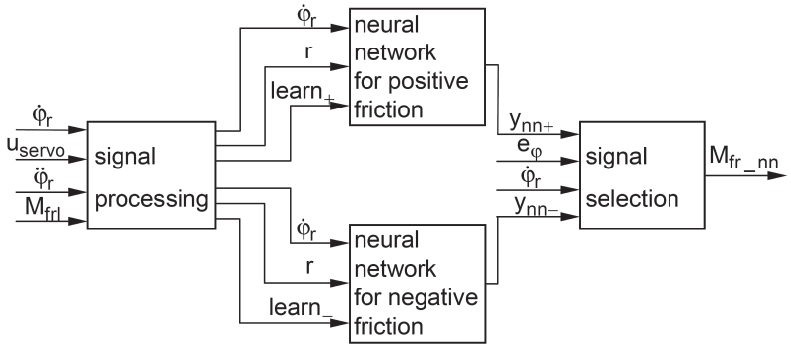

Abbildung 2.18: Neuronale Netzwerke zur Kompensation nichtlinearer Reibung

Abbildung 2.18 [KHH04] zeigt den Aufbau der Kompensation. Als Approximatoren werden zwei neuronale Netze verwendet, die jeweils einen Teil des nichtlinearen Effektes abbilden.

Insgesamt lässt sich erkennen, dass neuronale Netze als ergänzende Methode bei nichtlinearen Problemen eingesetzt werden können. Voraussetzung ist jedoch immer

die Trainierbarkeit des Netzes. Dazu muss der nichtlineare Teilaspekt isolierbar vorliegen und hinreichend reproduzierbar sein. Die direkte *Modellierung* eines gewünschten Verhaltens, wie es beispielsweise in der klassischen analytischen Regelungstechnik möglich ist – z. B. durch Vorgabe eines bestimmten Übertragungsverhaltens – ist mit neuronalen Netzen nicht möglich. Das Potential für die Selbstoptimierung liegt vielmehr in der (automatisierten) Abbildung von nichtlinearen Verhalten. Hier könnten ANN zur Abbildung eines Verhaltensmodells für eine Optimierung eingesetzt werden. Auch ist der Ansatz der nichtlinearen Kompensation viel versprechend.

2.2.7 Planungsorientierte Verfahren

Für die Durchführung von komplexeren Handlungsschritten reicht eine mehr oder weniger direkte Kopplung zwischen Sensorik und Aktorik nicht aus. Weder regelungstechnische noch verhaltensbasierte Systeme sind in der Lage, Aktionen auf abstrakter Ebene *vorauszuplanen*, d. h. strategisch zu handeln. Jedoch ist eine abstrakte und somit diskrete Planung von Handlungsschritten ohne die Berücksichtigung des technischen Systems und letzten Endes der kontinuierlichen Regelungstechnik nicht zielführend. Einige konkrete Vorschläge für die Lösung dieses Problems finden sich in der *Agententechnik* (vgl. Abschnitt 2.5) als sogenannte *hybride Schichtenansätze*.

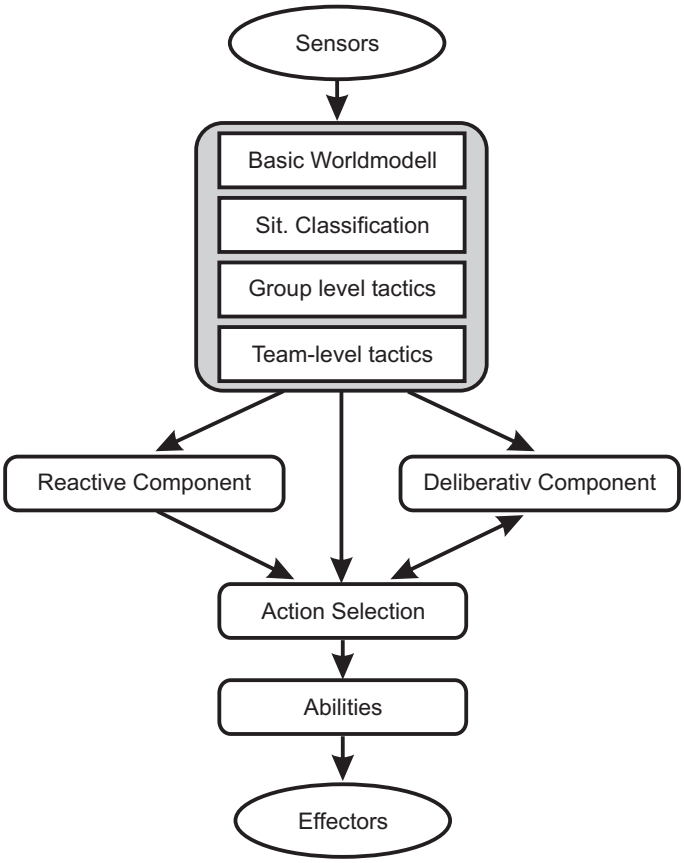

Abbildung 2.19: Hybride Architektur

Ein Beispiel für hybride Architekturen, insbesondere bei mobilen Robotern, findet sich bei [DFL02] (siehe Abbildung 2.19). Die Arbeit zeigt ein Teilergebnis

des DFG-Schwerpunktprogramms *Kooperierende Teams mobiler Roboter in dynamischen Umgebungen (SPP-1125)* [DFG07]. Weitere Beispiele für solche Architekturen sind 3T [BFG+97] oder Saphira [Kon97]. Ein Beispiel für eine agentenbasierte Steuerung des Saphira-Systems findet sich in [GCJK97].

Abbildung 2.19 [DFL02] macht auch deutlich, dass es sich bei den hybriden Schichtenmodellen nicht um einen neuen methodischen Lösungsansatz handelt, sondern um eine Kombination von reaktiven und planenden Agenten. Das Schichtenmodell zeigt einen Weg, diese Methoden zu kombinieren und umzusetzen.

Ein weiteres wichtiges Modell für die Modellierung von Planungssystemen ist das sogenannte BDI-Modell (Beliefs-Desires-Intentions) [Bra87], [RG95]. Dabei werden drei Arten von Systemzuständen unterschieden:

- Beliefs (Modell der Umwelt)
- Desires (steuert Aktionen auf der Basis der gegenwärtigen Beliefs)
- Intentions (bezeichnet (Teil-)Pläne zur Verwirklichung von Zielen)

Ein Modell der Umwelt kann beispielsweise eine bestimmte Position in einem Koordinatensystem sein, von dem der Roboter annimmt, dass er sich dort befindet. Es handelt sich also nicht um ein spezifisches mathematisches Dynamikmodell, wie es in der Mehrkörperdynamik verwendet wird, sondern allgemein um bekanntes Faktenwissen über die Umgebung. In einigen Fällen wird das technische System selbst als ein Teil des Umwelt-Modells angesehen. Die Steuerung der Aktion (Desires) gibt die aktuelle Handlung vor. Bezogen auf das Beispiel autonomer Roboter, ist dies beispielsweise eine zu erreichende Position. Die (Teil-)Pläne sind Handlungsfolgen, die als Aktionen ausgeführt können. Pläne werden aktiv, wenn das Umweltmodell neue Zustände erreicht.

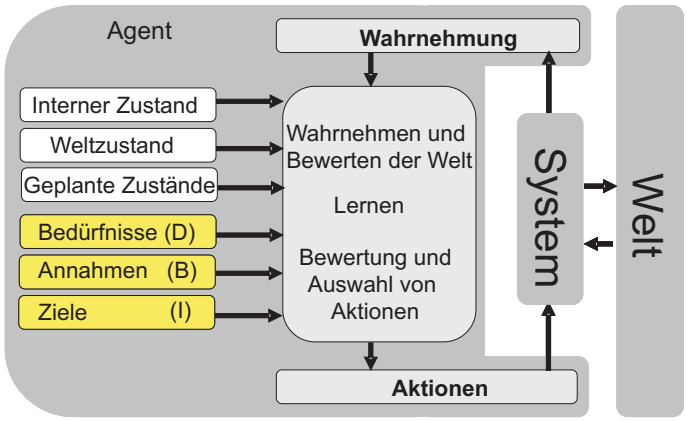

Abbildung 2.20: BDI-Architektur

Abbildung 2.20 [KK07] zeigt schematisch den Aufbau eines Agenten (vgl. Abschnitt 2.5) mit BDI-Architektur. Grundsätzlich unterscheidet dieser Ansatz nicht zwischen (eigenem) *System* und *Umwelt* (rechts). Die *Wahrnehmung* (unten) bezieht sich damit auf alle, beispielsweise durch Sensoren aufgenommene, Informationen. *Aktionen* sind Ausgangswerte oder Ausgangsinformationen aus dem Agenten, die direkt oder indirekt zu Tätigkeiten (z. B. Bewegung, Aktorik) führen. Der Ablauf in der Mitte des Agenten beschreibt eine Folge von internen Berechnungen, die sich in die Phasen *(1) Wahrnehmen und Bewerten*, *(2) Lernen* und *(3) Bewertung und*

Auswahl von Aktionen gliedert. Als Grundlage für Bewertung und Auswahl von Aktionen stehen Informationen aus der Wissensbasis zur Verfügung. Diese bestehen in diesem Fall aus *internen Zuständen*, *Weltzustand* (Wissen über die Umgebung) und dem *geplanten Zustand*. Für die Auswahl von Aktionen können allgemein *Bedürfnisse*, *Annahmen* (Erweiterung des Faktenwissens) und *Ziele* herangezogen werden. Der eigentliche Lernprozess erweitert die Wissensbasis durch die Bewertung von erfolgreichen und nicht erfolgreichen Handlungen, aber auch durch die Erweiterung von allgemeinem Faktenwissen, wie beispielsweise der Position erkannter Hindernisse. Eine erste Integration eines Planungssystems auf der Basis von Faktenwissen in die CAE-Umgebung CAMeL-View wird in [KO04] gezeigt.

2.3 Methodik zum Entwurf mechatronischer Systeme

Vorgehensmodelle für die Entwicklung technischer Systeme erlauben eine einfachere Strukturierung des Entwicklungsprozesses, bieten aber darüber hinaus die Chance, Informationen zu kanalisieren. Dies sind wesentliche Voraussetzungen für einen durchgängigen und planbaren Entwicklungsprozess. Bestimmte Vorgehensmodelle lassen sich auf die Entwicklung selbstoptimierender Systeme übertragen. Entscheidend ist hierbei die Nutzung des Informationsgewinns, der sich bei selbstoptimierenden Systemen ergibt. Im Folgenden werden zunächst, ausgehend von dem V-Modell, die einzelnen Stufen beschrieben, die in der Entwicklung auftreten, und anschließend ein mögliches Vorgehensmodell für selbstoptimierende Systeme skizziert.

2.3.1 V-Modell für die Entwicklung mechatronischer Systeme

Um die einzelnen Abschnitte bei der Entwicklung technischer, insbesondere mechatronischer Systeme formal beschreiben zu können, wurden im Laufe der Zeit verschiedene Vorgehensschemata entworfen. Einige idealisierte Vorstellungen wie etwa das *Wasserfallmodell* (vgl. [Roy87] und [Boe81]) erwiesen sich in der Praxis jedoch als nicht anwendbar [Ehr95].

Das Wasserfallmodell geht von abgestuften Entwicklungsschritten aus, bei denen die Information von einer Treppe zur nächsten „fließt“. Der Wasserfall symbolisiert dabei die Durchgängigkeit der Informationsverarbeitung. Ist ein Arbeitsschritt nicht erfolgreich, so wird zum vorherigen Schritt zurückgekehrt. Als entscheidender Erkenntnisgewinn formte sich mit der Zeit die Notwendigkeit von iterativen Zyklen bei der Entwicklung heraus. Bei vielen technischen Entwicklungen sind einzelne Schritte weder unabhängig parallel, noch sequentiell zu bearbeiten. Den entscheidenden Durchbruch brachte die Überlegung, dass auch zwischen weit auseinander liegenden Entwicklungsschritten Beziehungen existieren. Diese Beziehungen ergeben sich aus dem *Top-Down-Ansatz* vieler Entwicklungskonzepte, besonders in der Informatik, aber auch in der Konstruktionssystematik (siehe insb. [PB97]).

Beim Top-Down-Ansatz[11] wird ein komplexes Gesamtproblem durch die schrittweise Zerlegung in immer konkretere Teilprobleme aufgeteilt, die dann nach einzelner

[11] Im Gegensatz zu Bottom-Up, bei dem konkrete Teillösungen am Anfang stehen, um zum einem komplexen Ganzen zusammengefügt werden.

Implementierung wieder zu einer Gesamtlösung zusammengesetzt werden. Konkret bedeutet dies, dass beispielsweise ein Programm in Module zerlegt wird, diese Module implementiert, einzeln getestet und dann zu einem Gesamtprogramm schrittweise integriert werden.

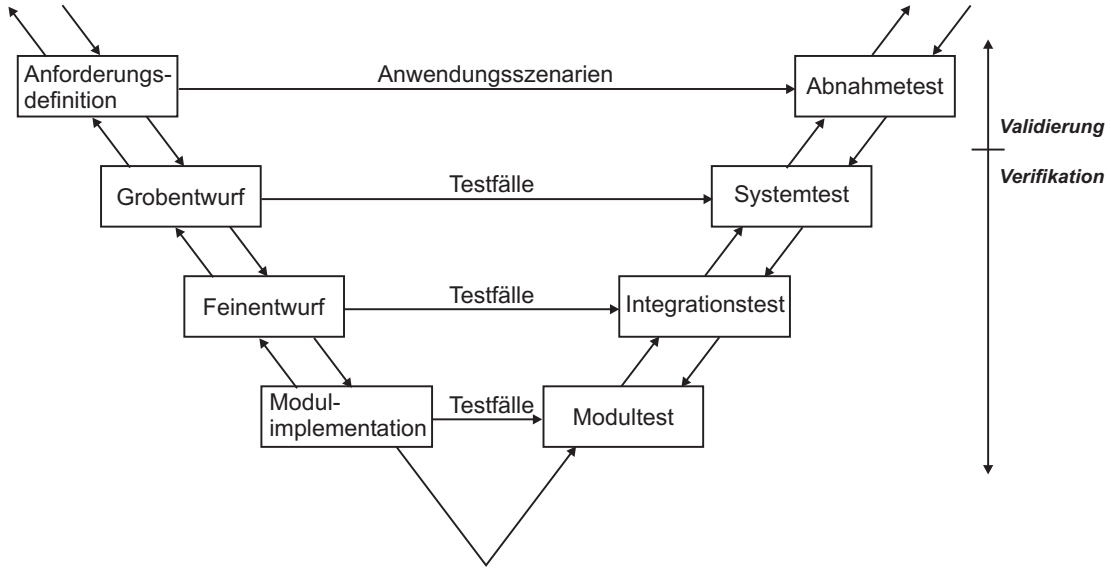

Abbildung 2.21: Das V-Modell der Informatik

Das V-Modell ist ein Ansatz zur Formalisierung des Top-Down-Ansatzes unter Berücksichtigung der Erfahrungen, die sich aus dem Wasserfallmodell ergeben haben. Wie in Abbildung 2.21 dargestellt wird, erfolgt auf der rechten Seite des V-Modells eine Zerlegung des Gesamtproblems in Teilprobleme, was einer Top-Down-Entwicklung entspricht. Ausgehend von einem globalen abstrakten Modell, verfeinert man dieses immer weiter, um letzten Endes zu einer Ausarbeitung der Teilaufgaben zu gelangen. Sind diese Teilaufgaben abgearbeitet, erfolgt eine Integration der in den Teilaufgaben entwickelten Teillösungen. Der Grad der Integration (des Systems) der linken Seite korrespondiert dabei mit dem der rechten Seite. Diese Korrespondenz erlaubt in beide Richtungen Rückschlüsse auf Verbesserungsmöglichkeiten bzw. dient der Verifikation. Ein notwendiger Iterationsschritt, der zu einem überarbeiteten Ergebnis führt, erfolgt dabei durch horizontale Bewegung von rechts nach links im V-Modell. Folgt man den Schritten bis zum Ausgangspunkt, ergibt sich ein Zyklus innerhalb des V-Modells.

Für die Abbildung komplexer Entwicklungsvorgänge in der Informationsverarbeitung ist das V-Modell gut geeignet. Jedoch beschreibt es nur unzureichend die *Reifung* eines Produktes, bei dem der beschriebene Entwicklungspfad grundsätzlich mehrfach durchlaufen wird. Beim V-Modell der Informatik wird vielmehr davon ausgegangen, dass ein Rückschritt eine Reaktion auf eine Unzulänglichkeit ist und nicht ein *regulärer* Schritt der Entwicklung. Dadurch sind teilweise Entwicklungsvorgänge, wie sie in der Praxis vorkommen, nicht oder nur sehr schwer abzubilden [Ray01]. Insbesondere der Entwurf von technischen Systemen kann nicht als linearer Prozess angesehen werden. Beispielsweise sind dynamisches Verhalten (geregeltes System) und konkrete Gestalt eines mechatronischen Systems nicht völlig unabhängig voneinander zu entwickeln, da die Gestalt wichtige Parameter wie Massen oder Dämpferkonstanten endgültig festlegt, aber die Dynamik wiederum Anforderungen

an die Gestalt, wie beispielsweise notwendige Festigkeiten, liefert. Somit sind viele technische Lösungen nur iterativ zu entwickeln. Diese aus der Praxis gewachsene Erfahrung ist in den *Entwicklungskreislauf der Mechatronik* eingeflossen (vgl. z. B. [HJW02, Toe02, Bec03, Nye06]). Der Entwicklungskreislauf der Mechatronik beschreibt die Kernphasen der Mechatronikentwicklung als einen stetig zu durchlaufenden Kreislauf, der bei jedem Durchlauf eine Verbesserung des Produktes durch Einfließen der Erfahrungen des vorherigen Schrittes vorsieht. In [Nau00] wird dieser Kreislauf bereits auf die Entwicklung *intelligenter vernetzter Systeme* vorbereitet. Jedoch standen bei diesem Ansatz vor allem die Einbindung in eine Entwicklungsumgebung und die Integration der verschiedenen Modellierungsebenen MFM, AMS und VMS (vgl. Abschnitt 2.4.3) und nicht die Nutzung von Entwicklungsergebnissen für eine spätere Optimierung im Vordergrund.

Eine wesentliche Ergänzung beschreibt die *mechatronische Komposition* als Kernkonzept des mechatronischen Entwurfs [LTB+99]. In der Phase der Konstruktion eines Produktes geht die Phase der mechatronischen Komposition dem technischen Entwurf voraus. Sie besteht selbst aus den Phasen Modellbildung, Analyse und Synthese. Sowohl die Konstruktionsphase als auch die mechatronische Komposition können dabei mehrfach durchlaufen werden, falls die Ergebnisse der Phasen etwa die Anforderungen an das Produkt nicht erfüllen oder neue Erkenntnisse durch die Untersuchungen eine Neukonzipierung sinnvoll erscheinen lassen. Gute Erläuterungen und Umsetzungen des Konzepts finden sich z. B. in [Toe02] und in [Bec03].

Weitere Ansätze zur Entwicklung eines Vorgehensmodells bezogen sich auf eine Kombination verschiedener vorhandener Ansätze. Ein Ansatz ist, den mechatronischen Entwicklungskreislauf in das V-Modell zu integrieren [GL00]. Dieses Vorgehensmodell kann als direkte Weiterentwicklung des mechatronischen Entwicklungskreislaufs angesehen werden.

Bei der Entwicklung der VDI-Richtlinie 2206 [Ver03] wurden die verschiedenen Ansätze zur strukturierten Entwicklung auf ihre Anwendbarkeit in realen Entwicklungsszenarien technischer Produkte hin diskutiert. Hierbei zeigte sich, dass das zyklische Durchlaufen verschiedener Entwicklungsphasen nicht allein auf den Mechatronikentwurf beschränkt bleiben darf. Die einzelnen Entwicklungsphasen werden durch mehrfache ineinander geschachtelte „Vs“ dargestellt. Nach jedem Durchlauf fließt der Erkenntnisgewinn in die nächste Entwicklungsphase ein – bis zum fertigen Produkt. Bei jedem Durchlauf ergeben sich neue Anforderungen an die nächste Phase. Folgt man dieser Entwicklungsspirale weiter nach außen, wächst nach jedem Zyklus der Reifegrad des Produkts.

Dieser Ansatz integriert alle Phasen der Produktentwicklung. Entscheidend ist hier die Unterscheidung in sogenannte Mikro- und Makrozyklen. Ein Mikrozyklus beschreibt die iterative Entwicklung eines technischen Aspekts bzw. das zyklische Durchlaufen einzelner Entwicklungsschritte innerhalb des V-Modells. Ein Makrozyklus wird durch das vollständige Durchlaufen eines Vs gebildet – von der Produktidee bis zum fertigen Produkt. Dabei ist das Ergebnis nicht zwangsläufig – wie beim klassischen V-Modell – ein *marktfähiges* Produkt, sondern ein zu erreichendes Zwischenergebnis, für das aber ein Entwicklungszyklus durchlaufen werden muss.

Das V-Modell nach [Ver03] beschreibt den Ablauf einer systematischen Entwicklung in mehreren Makrozyklen. Jedoch wird bei diesem Ansatz immer noch von einer Ausarbeitung der technischen Details in separaten Domänen ausgegangen. Dieser Entwicklungsschritt wird als *domänenspezifischer Entwurf* bezeichnet. Grundlage

bildet die Überlegung, möglichst genau alle Teilfunktionen schon beim mechatronischen Entwurf festzulegen, um dann in getrennten Bereichen die technischen Details auszuarbeiten. Hierbei wird jedoch nicht berücksichtigt, dass besonders bei mechatronischen Systemen eine isolierte, domänenspezifische Entwicklung von Teilfunktionen nicht zielführend ist [PB97].

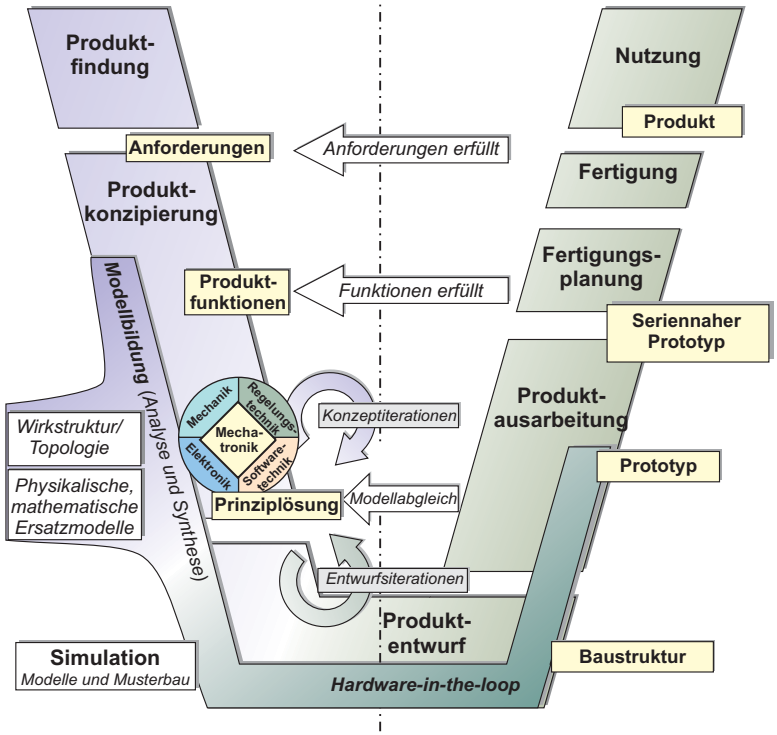

Abbildung 2.22: Das V-Modell als Referenzmodell zur Entwicklung mechatronischer Produkte

Abbildung 2.22 zeigt das V-Modell nach [GL00]. Die Entwicklung eines Produkts ist dabei in die Phasen Produktfindung, Produktkonzipierung, Produktentwurf, Produktausarbeitung, Fertigungsplanung, Fertigung und Nutzung eingeteilt. Der mechatronische Entwicklungskreislauf umschreibt die Phasen Modellbildung über Hardware-in-the-Loop-Simulation bis zum Prototyp.

2.3.2 Anwendung auf Selbstoptimierung

Die Betrachtung der verschiedenen Konzepte zur Formalisierung der Produktentwicklung zeigt eine wesentliche Gemeinsamkeit: Sie beschreiben nicht nur den formalen Ablauf, sondern auch den Informationsfluss. Jede Phase beschreibt eine Erweiterung und Verfeinerung der Erkenntnisse über ein Produkt. Während der Entwicklung werden Modelle aufgebaut, verfeinert und eventuell wieder verworfen. Die Selbstoptimierung kann diese Informationen nutzen. Insbesondere die Modellbildung kann hier eine zentrale Rolle spielen.

Zudem wird deutlich, dass die Selbstoptimierung eines fertigen Produktes als weitere Iterationsschleife des V-Modells aufgefasst werden kann. Somit ist es naheliegend, die bisherigen Ansätze durch weitere Entwicklungszyklen, die während des Betriebes folgen, zu erweitern.

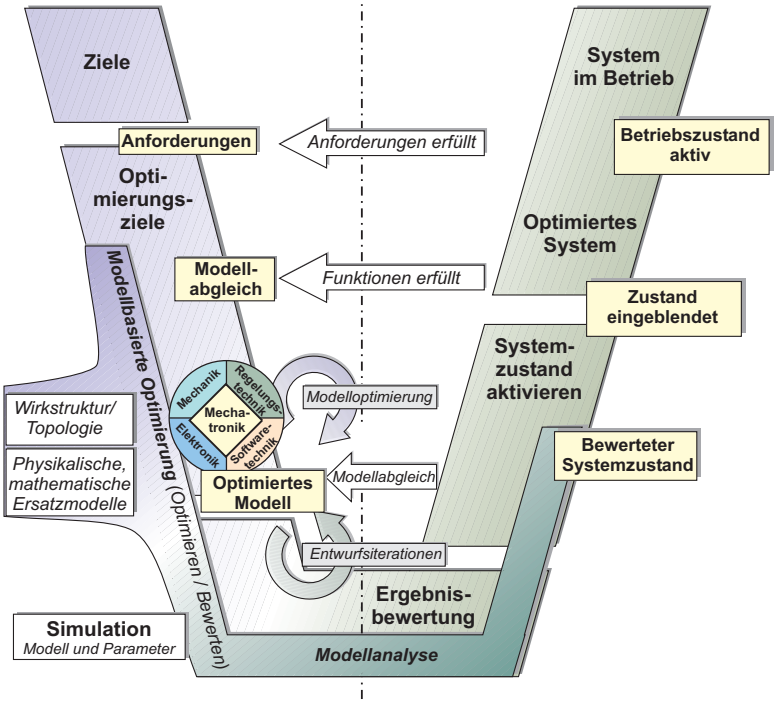

Abbildung 2.23: Vorschlag für ein V-Modell der Selbstoptimierung (modellbasiert)

Werden verschiedene Schritte der Selbstoptimierung auf das V-Modell übertragen, gelangt man zu einem Phasenmodell für Selbstoptimierung, das sich in die bisherigen Konzepte integrieren lässt. Abbildung 2.23 zeigt einen Vorschlag für den Ablauf einer modellbasierten Optimierung. Das Modell ist bewusst abstrakt gehalten. Im Gegensatz zum V-Modell mit dem integrierten mechatronischen Entwicklungskreislauf, das zur Entwicklung oder Weiterentwicklung eines Produktes dient, bezieht sich das V-Modell bei Selbstoptimierung zunächst auf die Optimierung von Systemverhalten durch Veränderung der Systemparameter. Eine Veränderung der Struktur des Systems ist nur auf Ebene der (regelnden) Informationsverarbeitung möglich, z. B. durch Austausch eines Reglers.

Die erkennbaren Parallelen deuten darauf hin, dass es das Ziel bei der Entwicklung von selbstoptimierenden Systemen sein muss, ingenieurwissenschaftliches Vorgehen im Rechner abzubilden. Dieser Weg geht damit über die bisherigen Ansätze der Informatik, wie sie im Rahmen der verhaltensbasierten Systeme und der Künstlichen Intelligenz entstanden sind, hinaus.

2.3.3 Vorgehen beim modellbasierten Systementwurf

Für die Entwicklung komplexer technischer Systeme ist die Modellbildung ein wichtiges Hilfsmittel, um Eigenschaften des späteren Produktes festzulegen und zu testen. Grob lassen sich Modelle in *reale* und *virtuelle Modelle* (z. B. mathematisch und/oder rechnergestützt) unterteilen. Reale Modelle zählen zu den ältesten Hilfsmitteln für die Entwicklung von technischen Geräten. Heute gewinnen jedoch *virtuelle* Modelle zunehmend an Bedeutung, zu denen im Maschinenbau vor allem mathematisch formulierte, physikalische Modelle gehören, die im Rechner mit Hilfe von Software abgebildet werden. Da der Bau von realen Prototypen zeit- und kostenintensiv ist, bestehen Bestrebungen, die Anzahl der physikalischen Prototypen auf ein

Minimum zu beschränken. Der Einsatz von virtuellen Prototypen kann dies, durch die Analyse im Rechner, wirkungsvoll unterstützen. Als Erweiterung des so genannten *Digital Mock-Up*, der die 3D-Gestaltmodelle und die Produktstruktur umfasst, werden beim virtuellen Prototypen zusätzliche Aspekte wie Kinematik, Dynamik, Festigkeit, etc. berücksichtigt. Das Beispiel der Boeing 777, die als erstes Flugzeug vollständig mit 3D-CAD-Systemen einschließlich des virtuellen Zusammenbaus entwickelt wurde, zeigt die Leistungsfähigkeit der virtuellen Betrachtung [GEK01].

Eine wichtige Voraussetzung dieser Vorgehensweise ist, dass die simulierten Eigenschaften mit der Realität, also mit dem zu entwickelnden oder vorliegenden System hinreichend genau übereinstimmen. Diese Überprüfung ist jedoch insbesondere bei Neukonstruktionen besonders in den frühen Phasen nicht immer realisierbar. Die Anwendung der Modellbildung und der Simulation erfordert deshalb eine ständige Überprüfung der Plausibilität und eine kritische Grundhaltung der Entwickler gegenüber den Simulationsergebnissen.

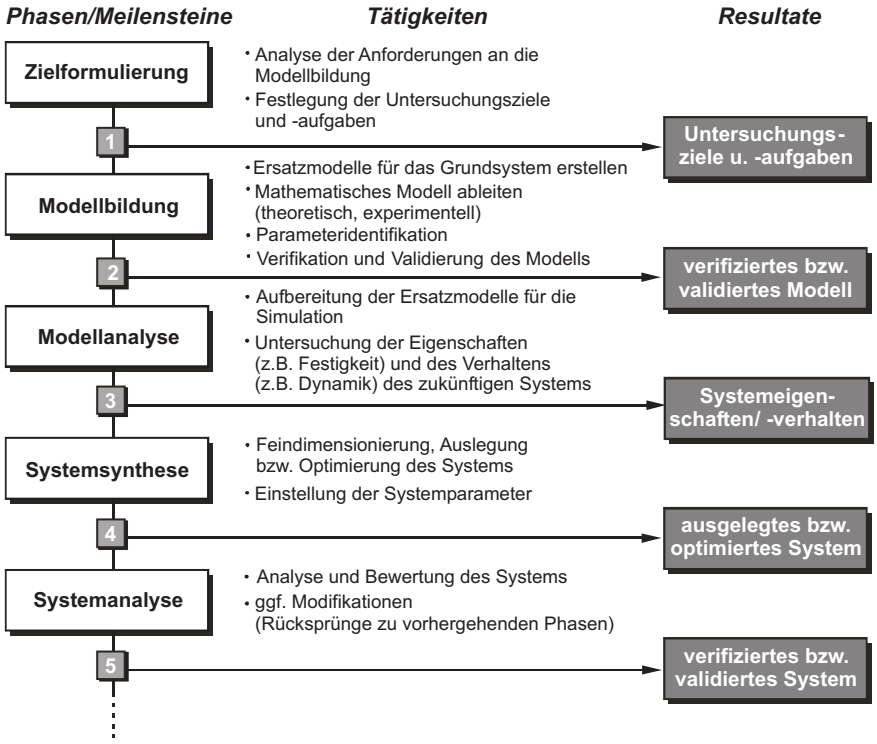

Abbildung 2.24: Vorgehen beim modellbasierten Systementwurf

Abbildung 2.24 beschreibt die Phasen Zielformulierung, Modellbildung, Modellanalyse, Systemsynthese und Systemanalyse beim modellbasierten Systementwurf [Ver03].

Zielformulierung: Der erste Schritt ist die Festlegung von Untersuchungszielen und -aufgaben. Daraus ergeben sich geeignete Methoden und Modellierungsverfahren. Folgende Gründe können eine Modellbildung motivieren:

- Prinzipuntersuchungen an neu zu entwickelnden Systemen
- Zielgerichtete Reglerauslegung
- Analyse und Optimierung bestehender Systeme
- Ergänzung oder Bestimmung von schwer messbaren Daten
- Zu hoher (Kosten-) Aufwand oder Risiko (für Mensch oder Technik) für experimentelle Untersuchungen
- Zu hoher Zeitaufwand bei experimentellen Untersuchungen oder Blick in die Zukunft (Verschleiß, Alterung, dynamische Belastung).
- Reduktion des Prototypenaufwands (schrittweise Realisierung von Teilkomponenten)
- Hardware-in-the-Loop-Simulation zur Funktionsverifikation von Teilkomponenten

Modellbildung: Es werden schrittweise physikalisch-mathematische Modelle mit Hilfe von geeigneten Entwicklungsumgebungen entworfen. In vielen Fällen gehen Skizzen oder erste mathematische Abschätzungen der Implementierung im Rechner voraus.

Modellanalyse: Auf Basis des Modells werden die Eigenschaften und das (dynamische) Verhalten des zu untersuchenden Systems analysiert. Diese Phase liefert Erkenntnisse für die nachfolgende Systemsynthese.

Systemsynthese: In der Synthese werden die Erkenntnisse und Ergebnisse der Modellanalyse auf das zu entwickelnde System übertragen. Lösungselemente werden ergänzt und mit vorhandenen Wirkprinzipien verfeinert und optimiert.

Systemanalyse: Das entwickelte System wird nun analysiert und bewertet. Dabei werden die Entwicklungsziele mit dem Ergebnis verglichen. Gegebenenfalls sind Rücksprünge in vorhergehende Entwicklungsschritte notwendig. Mit welcher Effizienz Mängel identifiziert und durch iteratives Durchlaufen der Entwicklungsschritte behoben werden, hängt wesentlich von der Wahl des Modells ab (zugrunde gelegte Mathematik/Physik, Modellierungstiefe, Vernachlässigungen/Detaillierungen etc.).

Wird dieses Vorgehensmodell auf und *in* selbstoptimierende Systeme übertragen, zeigen sich folgende Parallelen:

Zielformulierung entspricht im Kontext der selbstoptimierenden Systeme der *Bestimmung der Systemziele.*

Modellbildung und Modellanalyse entsprechen der *Analyse der Ist-Situation* z. B. durch Online-Identifikation des System- und Umgebungszustandes. Der automatisierte Aufbau von physikalisch deutbaren Modellen ist nach bisherigem Kenntnisstand zwar nicht möglich, jedoch die Bildung von Modellen mit Hilfe von universellen Approximatoren (vgl. Abschnitt 2.3.4).

Systemsynthese und Systemanalyse entsprechen der *Anpassung des Systemverhaltens.*

Damit entsprechen wesentliche Phasen des modellbasierten Entwurfs den Phasen die für selbstoptimierende Systeme definiert sind, wie in Abschnitt 2.1.2 beschrieben wurde. Somit ist eine wesentliche Eigenschaft von selbstoptimierenden Systemen, zumindest Teile des ingenieurwissenschaftlichen Vorgehens beim modellbasierten Systementwurf im Rechner abzubilden. Werden physikalische deutbare Modelle eingesetzt ist der Nutzen für eine spätere Weiterentwicklung von Modellen im modellbasierten Entwurf nicht zu unterschätzen. Selbstoptimierende Systeme haben dadurch das Potential den Aufbau von *Erfahrungswissen* zu unterstützen, der in den Phasen Produktentwurf und Produktausarbeitung gemacht werden kann.

2.3.4 Modellbildung mechatronischer Systeme

Reale Modelle bildeten schon seit jeher die Grundlage physikalischer Experimente. Der haptische Umgang mit Modellen, die an das zu lösende Problem angepasst sind, erlaubt die Untersuchung bestimmter physikalischer Wirkzusammenhänge, unabhängig von der Gesamtkomplexität eines technischen Systems.

Seit Entwicklung des Mikrorechners ist es möglich, physikalisches Verhalten vollständig im Rechner zu simulieren. Grundlage bildet hier die mathematische Abbildung physikalischer Vorgänge. Dadurch ist es dem Entwickler möglich, schon weit vor dem ersten Prototypen das Verhalten des späteren realen Systems abzuschätzen und Optimierungen durchzuführen. Viele der Mechanismen, die in der modellbasierten Optimierung eingesetzt werden, sind auf selbstoptimierende Systeme übertragbar (vgl. Abschnitt 2.2).

Damit ist die modellbasierte Optimierung zur Laufzeit eine wichtige Komponente der Selbstoptimierung von mechatronischen Systemen, die nur mit Hilfe geeigneter rechnergestützter Modelle möglich ist. Da eine Optimierung parallel zur realen Zeit erfolgen muss, gegebenenfalls sogar schneller als die reale Zeit, sind geeignete Simulationsverfahren erforderlich, die diesen Anforderungen genügen. Dies Thema wird im Kapitel 4 vertieft.

Darüber hinaus sind Modelle gegebenenfalls mit realen Systemen gekoppelt, was zu sog. Hardware-in-the-Loop (HiL) Simulationen führt. Die Kopplung digitaler Reglersysteme mit technischen Systemen stellt dabei ähnliche Anforderungen wie Hardware-in-the-Loop. Somit spielt es aus Anforderungssicht keine Rolle, ob ein (Teil-)Modell für Hil-Simulation, Echtzeit-Beobachtermodelle oder Echtzeit-Optimierung eingesetzt wird.

Bei der Simulation mechatronischer Systeme muss neben dem Verhalten der Teilsysteme auch ihr Einfluss untereinander betrachtet werden. Erschwerend kommt hinzu, dass sich die Systemelemente oft nicht einer wissenschaftlichen Disziplin zuordnen lassen; sie enthalten Funktionen mindestens zweier Disziplinen und bilden Schnittstellen zwischen den beteiligten Teilsystemen, z. B. Hydraulikzylinder oder elektrische Gleichstrommotoren. Aufgrund dieser Heterogenität sind vielfältige IT-Werkzeuge im Einsatz, um die Simulation eines mechatronischen Gesamtsystems durchzuführen: beispielsweise Werkzeuge, die möglichst viele Wissensdomänen abdecken, Import von Modellen anderer Werkzeuge in ein Simulationsprogramm oder

getrennte Simulation der Teilsysteme mit verschiedenen Werkzeugen und anschließendem Ergebnisaustausch [Dür99]. Die Herausforderung besteht darin, die relevanten IT-Werkzeuge zu einer Entwicklungsumgebung zusammenzuführen und das Zusammenspiel der Werkzeuge modell-, system-, prozess- und verfahrenstechnisch zu unterstützen.

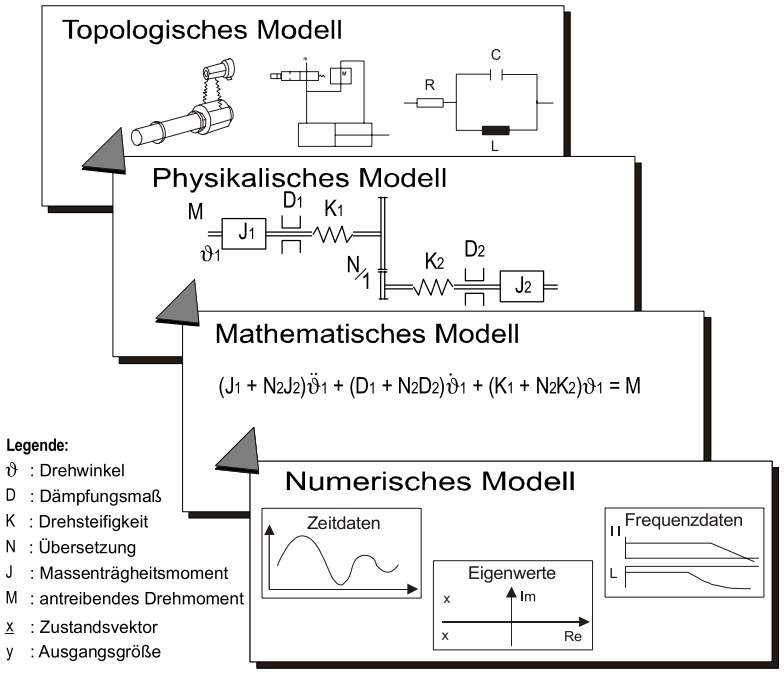

Abbildung 2.25: Abstraktionsebenen im Modellbildungsprozess

Nach [Ver03] werden vier Abstraktionsebenen des Modellbildungsprozesses unterschieden, die aufeinander aufbauen (vgl. Abbildung 2.25). Aus diesem Aufbau lässt sich ein allgemeiner Ablauf für den modellbasierten Entwurf ableiten. Dies wird im folgenden Abschnitt 2.3.5 näher erläutert. In Abbildung 2.25 ist zu erkennen, dass ein Modell aus verschiedenen Abstraktionsebenen besteht und sich schrittweise entwickeln lässt, wobei jede Ebene über eine eigene, spezifische Darstellungsform verfügt.

Eine Anwendung findet sich bereits in [Hah99] für das CAS-System CAMeL-View. Die Ebenen *topologisches Modell* und *physikalisches Modell* sind jedoch zusammengefasst. Daraus abgeleitet ist das *objektorientierte Mechatronikmodell* (vgl. Abschnitt 2.3.6).

2.3.5 Abstraktionsebenen in der Modellbildung

Grundsätzlich bestimmen das Ziel der Modellbildung und die Beschaffenheit des realen Systems die Art des Modells und die Modellierungstiefe[12]. Für Neuentwicklungen sind andere Modellierungsansätze zu wählen als für Optimierungen an bestehenden Produkten. Für eine ganzheitliche Vorgehensweise ist die Verbindung domänenspezifischer Teilmodelle erforderlich. Dabei ist jedoch zu beachten, dass die mathematische Darstellung der Teilmodelle kompatibel sein muss, sonst kann es zu schwer erkennbaren oder schwer deutbaren Effekten kommen [SS01].

[12]Modellierungstiefe bezeichnet im Allgemeinen den Detaillierungsgrad eines Modells

In den letzten Jahren wurden jedoch verschiedene Anstrengungen unternommen, die Probleme bei der Kopplung von Modellen auf Modell- und Simulationsebene zu lösen (siehe z. B. [LDHS01], [SS01]).

Einen Ausweg bieten hier eine Normierung der mathematischen Darstellung und eine Kopplung der Teilmodelle auf topologischer oder physikalischer Ebene. Insbesondere für die Darstellung dynamischer Effekte bietet sich das Vorgehen nach einem vierstufigen Modellabstraktionsprozess an.

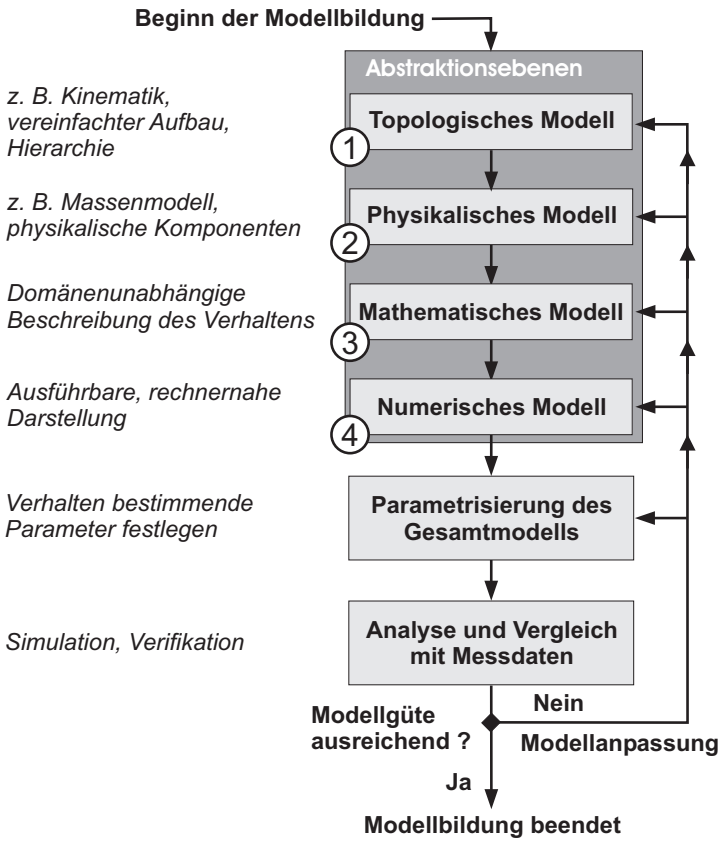

Abbildung 2.26: Modellabstraktionsebenen im Modellbildungsprozess

Abbildung 2.26 [GL00] zeigt die Ebenen, nach denen eine Modellintegration erfolgen kann. Es zeigt außerdem den Ablauf bei der Modellbildung. Die Darstellung hat vor allem bei der Modellierung dynamischer mechatronischer Systeme große Bedeutung. Die Ebenen bilden verschiedene Abstraktionslevel, auf denen das System als Modell repräsentiert wird. Jede Ebene leitet sich aus der nächst höheren ab, abstrahiert diese zum Teil und ergänzt neue Eigenschaften, die auf der höheren Ebene nicht darstellbar waren.

Das topologische Modell dient der Modellierung der Systemtopologie, also der Anordnung und Verknüpfung von Funktionselementen (Baugruppen, Module, Aggregate, Lösungselemente etc.). Dabei repräsentiert ein Element im Allgemeinen die drei Basisfunktionen *kinematische Funktion*, *dynamische Funktion* und *mechatronische Funktion*.

Die kinematische Funktion kann im Allgemeinen zunächst unabhängig vom weiteren Verhalten des Gesamtsystems entworfen werden. Dies sind beispielsweise die Stellwege einer Hydraulik, die sich aus Bauweise und Anordnung der

Zylinder ergeben, oder die Freiheitsgrade und Bewegungsmöglichkeiten eines Handhabungsgerätes. Die dynamische Funktion basiert auf dem Entwurf der Kinematik. Die Dynamik legt das Bewegungsverhalten von bewegten Massen fest. Erst in diesem Schritt können konkrete Aussagen zur notwendigen Leistung und Bandbreite von Antrieben bestimmt werden. Auch die Bewegungsgleichungen des Systems lassen sich nun bestimmen. Ist das dynamische Verhalten bestimmt, so wird das System durch die mechatronische Funktion ergänzt. Für die Kontrolle der Dynamik wird hier eine Regelung benötigt. Aber auch weitere Funktionen, die zur Kontrolle des Systems dienen, etwa Bahnsteuerungen, sind gegebenenfalls Teil der mechatronischen Funktion. Diese drei Phasen des Entwurfs können weiter untergliedert sein.

Das topologische Modell repräsentiert darüber hinaus die Beziehungen und Verkopplungen der verschiedenen Teilfunktionen und Komponenten, auch und insbesondere auf informationstechnischer Ebene.

Das physikalische Modell ergibt sich zum Teil automatisch aus dem topologischen Modell. Beispielsweise lassen sich aus der Kinematik und den bekannten Masseverhältnissen die dynamischen Bewegungsgleichungen des ungeregelten Systems ableiten.

Das mathematische Modell beschreibt im Wesentlichen das Verhalten des Systems. Dazu wird das physikalische Modell in eine abstrakte, domänenunabhängige Darstellung überführt. Die physikalischen Eigenschaften werden durch mathematische Funktionen beschrieben.

Das numerische Modell stellt zugleich ein im Rechner simulierbares Modell dar. Im einfachsten Fall ist es eine Aufbereitung des mathematischen Modells, so dass es algorithmisch behandelt werden kann. Die Simulationsergebnisse hängen dabei auch vom verwendeten Lösungsverfahren ab. Welche Verfahren zur Lösung verwendet werden können, hängt dabei vor allem von der mathematischen Darstellung ab, die letzten Endes auch über Echtzeitfähigkeit und die Möglichkeit zu Hardware-in-the-Loop-Simulationen entscheidet.

Erst alle vier Ebenen bilden gemeinsam das Gesamtmodell ab. Eine Änderung auf einer Ebene hat zwangsläufig Auswirkungen auf die darunterliegenden Ebenen.

Für die Modellbildung schließen sich weitere Schritte nach der Erstellung des Modells an. So muss das Modell parametriert werden. Dies kann durch eine Identifikation des realen Systems erfolgen. Durch Verifikation und Validierung wird abschließend sichergestellt, dass das Modell korrekt ist und die Wirklichkeit hinreichend genau abbildet. Bei einer konkreten Entwicklung würden diese Phasen nach dem V-Modell und dem Entwicklungskreislauf der Mechatronik mehrfach durchlaufen.

Durch die Darstellung in vier Modellebenen kann das Optimierungspotential für die Selbstoptimierung auch für realisierte Systeme erfasst werden. Eine Verbesserung des Systems ist demnach auf allen Ebenen denkbar. Je höher die Ebene (Topologie als höchste Ebene), umso größer sind jedoch der Änderungsaufwand und der Prüfaufwand zur Sicherstellung der Funktion *vor* der Änderung im laufenden Betrieb.

Die vier Ebenen lassen auf das folgende Optimierungspotential schließen:

Topologie Auf dieser Ebene ist eine strukturelle Optimierung denkbar. Durch Rekonfiguration und Rekombination von Elementen können die Funktionsweise

und das Verhalten verändert werden. Hierzu gehört auch das Austauschen eines Reglers.

Physik In dieser Ebene werden zur Topologie Wirkelemente beschrieben. Ein Systemelement, wie z. B. die Erzeugung einer Kraft, wird hier konkretisiert. Eine Optimierung muss an dieser Stelle an der physikalischen Wirkweise ansetzen. Dabei spielt das komplexe Zusammenspiel zwischen Komponenten eine Rolle. Ein Beispiel für eine Optimierung auf dieser Ebene ist die Veränderung der Lastverteilung zwischen verschiedenen Motoren eines Roboterarms.

Mathematik Die mathematische Darstellung eines Systems leitet sich im Allgemeinen aus der physikalisch-topologischen Sicht ab. Für rein mathematische Elemente, wie eine Regelung (analoge Regler ausgenommen), ist eine Optimierung auf Ebene der Mathematik möglich. Dazu gehört der innere Aufbau eines Regler(-systems), aber auch andere Elemente, die auf dieser Ebene modelliert werden.

Numerik Auf der untersten Ebene wird das System ausgewertet. Für ein realisiertes System ist dies der ausführbare Code der Informationsverarbeitung. Sind (Teil-)Modelle vorhanden, z. B. bei Hardware-in-the-Loop-Simulationen oder dem Einsatz von simulierten Modellen zur Optimierung, ergibt sich weiteres Optimierungspotential. Je nach numerischen Verfahren sind die Ergebnisse entweder genau oder liegen schneller vor. Für die Auswertung werden mehr oder weniger Rechenressourcen benötigt. Weiterhin bergen die Verteilung der Ausführung und das Zusammenspiel der Teilsysteme weiteres Optimierungspotential.

2.3.6 Objektorientiertes Mechatronikmodell (OMM)

Die in Abschnitt 2.3.5 dargestellten Grundlagen können auf eine objektorientierte Modellbildung übertragen werden. Unterstützend für die Wiederverwertbarkeit von Modellen oder Modellkomponenten und zur Reduktion der Gesamtkomplexität ist eine modulare, objektorientierte Sicht nötig. Für die Beschreibung von komplexen mechatronischen Systemen ist offensichtlich eine objektorientierte Beschreibung notwendig, die sich an der Physik des Systems anlehnt. Dabei ist die Modellierung auf der Ebene einer physikalisch-topologischen Darstellung sinnvoll. Jedoch müssen auf diese Weise beschriebene Modelle auch ausführbar sein und durch mathematische Funktionen ergänzbar, die keine Physik abbilden, wie die Informationsverarbeitung und die Regelungstechnik.

Für eine Transformation der physikalisch-topologisch beschriebenen physikalischen Zusammenhänge in mathematische Funktionen existieren verschiedene Ableitungsformalismen. Für eine Transformation in ausführbaren Code sind weitere Übersetzungsschritte notwendig, um letztlich zu einer numerisch auswertbaren und ausführbaren Darstellung zu gelangen.

In [Hah99] werden die allgemeinen Anforderungen an ein objektorientiertes Mechatronikmodell beschrieben (siehe Abbildung 2.27). Dabei wird von einer domänenspezifischen Beschreibung auf einer topologisch-physikalischen Ebene ausgegangen. Die fachspezifische Struktur wird in der Beschreibungsform Objective-DSS[13] aufgebaut bzw. in sie umgewandelt. Im nächsten Schritt wird diese Darstellung in eine

[13] DSS – Dynamic System Structure

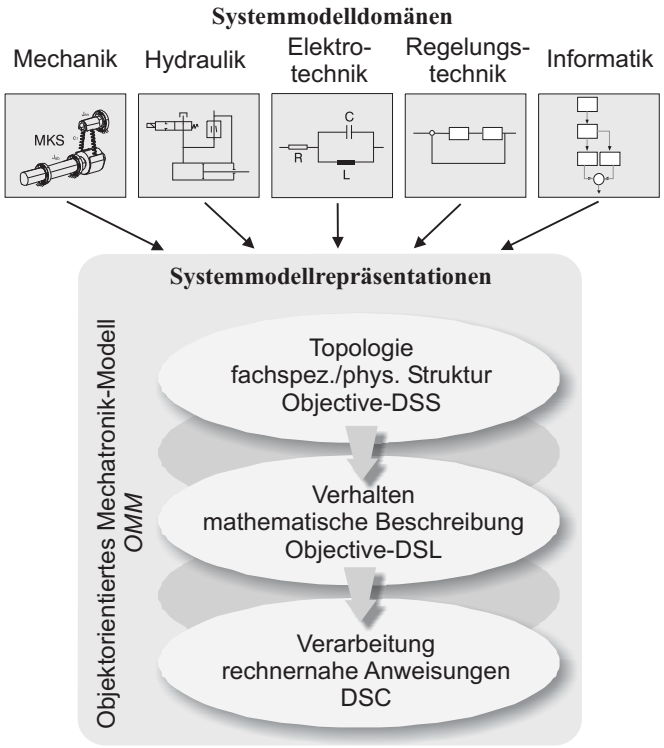

Abbildung 2.27: Objektorientiertes Mechatronikmodell (OMM)

mathematische Form transformiert (Objective-DSL[14]). Auf mathematischer Ebene können weitere Eigenschaften und Funktionen ergänzt werden, die sich nicht aus der physikalisch-technischen Darstellung ableiten lassen, wie beispielsweise die mathematische Beschreibung von Reglern. Die einheitliche, normierende Darstellung auf der mathematischen Ebene kann vollständig in eine ausführbare Form (DSC[15]) transformiert werden. Hierbei ist zu beachten, dass bei einer Simulation im Rechner weitere Effekte durch die numerische, quasi-kontinuierliche Auswertung hinzukommen, die das zu modellierende System-Verhalten verfälschen können. Daher müssen für eine Simulation die Randbedingungen, welche die beschreibende Mathematik als Resultat der zu beschreibenden Physik vorgibt, berücksichtigt werden. Dies sind beispielsweise die Steifheit des Systems, die Einfluss sowohl auf die Wahl der Schrittweite, als auch auf das Auswertungsverfahren hat [Stu96].

Deutlich wird bei dieser Systemmodellrepräsentation, dass die Entwicklung des Systemmodells in verschiedenen Phasen ablaufen muss, da die mathematische Ebene erst betreten werden kann, wenn die Topologie festgelegt wurde, und weiterhin eine Verarbeitung erst nach Festlegung der mathematischen Ebene erfolgen kann. Auf den ersten Blick erscheint dies ein Vorgehen nach dem Wasserfallmodell zu erzwingen. Jedoch ist, durch die Unterstützung der Entwicklungsumgebung CAMeL-View, welche dieses Modell verwendet, ein mehrfaches Durchlaufen der Entwicklungskette möglich. Dadurch wird das V-Modell des Mechatronikentwurfs voll unterstützt (vgl. 2.3.1).

[14]DSL – Dynamic System Language
[15]DSC – Dynamic System Code

2.3.7 Rechnergestützte Simulation

Modelle im Rechner sind zumeist mathematisch formulierte Abbildungen der Realität. Viele Eigenschaften können durch Anwendung von mathematischen Gleichungen oder anderen Formalismen auf diese Modelle ermittelt werden. Dies sind zum Beispiel die Ermittlung von Stabilitätseigenschaften in der Regelungstechnik oder die Erreichbarkeit von Zuständen bei Modellen, die endliche Automaten abbilden. In vielen Fällen ist es aber nützlich, das Verhalten eines Systems über die Zeit in einer vorbestimmten Umgebung zu untersuchen. Der Entwickler gewinnt einen „plastischen“ Eindruck vom späteren Verhalten des realen Systems, das er gerade entwickelt.

Die rechnergestützte Simulation technischer Systeme ist in der Mechatronik längst eins der wichtigsten Hilfsmittel zur Abschätzung des Verhaltens eines zu entwerfenden Systems geworden. In dieser Arbeit steht die Simulation dynamischer Vorgänge im Vordergrund, da wir zum einen selbstoptimierende Systeme zunächst auf mechatronische Systeme beschränken, und zum anderen, weil die Betrachtung aller heute praktisch relevante Simulationsbereiche den Rahmen dieser Arbeit bei Weitem sprengen würde. Jedoch lässt sich, ohne Anspruch auf Vollständigkeit, die Simulation allgemein in verschiedene Bereiche einteilen:

Nicht echtzeitfähige Simulationen [16] spielen bei der Entwicklung und der Erprobung von mechatronischen Modellen, sowie bei der Reglersynthese eine wichtige Rolle. Vor allem die nicht verteilten Simulationen sind leicht zu nutzen und sowohl im kommerziellen wie auch im nichtkommerziellen Bereich in vielfältigen Varianten vertreten. Wichtige Vertreter im Bereich Mehrkörpersimulation und rechnergestützte Modellbildung sind Matlab/Simulink [Mat06], ADAMS [MSC06] und CAMeL-View [iXt06]. Verteilte, nicht echtzeitfähige Simulationen sind am MLaP im Rahmen von IPANEMA entstanden [Hon98], [Sto04].

Verteilte Simulationen sind Anwendungen, die auf verschiedene Prozessoren oder Rechnersysteme aufgeteilt ausgeführt werden. Das Spektrum der Kopplung reicht hierbei von sehr enger Kopplung der Rechenkerne (z. B. Integratorkopplung oder Zentralintegration) bis hin zu gelegentlichem Datenaustausch über Dateien. Der Grad der Kopplung hängt dabei wesentlich vom verwendeten mathematischen Modell ab. Eine weit verbreitete Art der Kopplung ist die sog. Simulatorkopplung, bei der Zwischendaten ausgetauscht werden. Der Begriff ist jedoch nicht klar abgegrenzt.

Ein Problem bei der Kopplung verschiedener Simulationssysteme ist häufig die Nichtbeachtung der Kompatibilität des mathematischen Lösungsverfahrens oder des mathematischen Modells, so dass Ergebnisse verfälscht werden können. So ist beispielsweise eine Kopplung zwischen Mehrkörpersystemmodellen und Finite-Elemente-Modelle nur bedingt möglich, wenn Oberschwingungen des einen Modellteils im anderen nicht berücksichtigt werden können. Weitere Probleme ergeben sich durch die Kopplung unterschiedlicher Lösungsverfahren (vgl. 4.3). Am MLaP wurden seit Ende der 80er Jahre Simulationen mit massiv-parallelen Maschinen auf Transputer-Basis durchgeführt (siehe z. B. [Hon98], [Sto04]).

[16] Der Begriff der Echtzeitfähigkeit wird in Abschnitt 3.4.2.2 näher diskutiert

Echtzeitsimulationen lassen die interne Berechnungszeit synchron zur realen Zeit laufen. Im technischen Bereich ist diese Art von Simulationen vor allem für die Abschätzung von technischen Vorgängen wichtig. Große Bedeutung hat sie heute vor allem mit einer angekoppelten Visualisierung. Ein weiterer Zweig sind Echtzeitsimulationen zur Unterhaltung (Echtzeitspiele). Viele der mathematischen Modelle und Lösungsverfahren sind heute sowohl im Bereich der Spiele als auch für Anwendungen im Ingenieurbereich gebräuchlich, wenn auch mit anderem Anspruch.

Eine spezielle Form der Echtzeitsimulation ist die Hardware-in-the-Loop-Simulation.

Hardware-in-the-Loop-Simulationen koppeln ein Modell im Rechner mit einem realen technischen (Teil-)System. Das technische System wird durch eine entsprechende Aktorik mit realen physikalischen Größen beaufschlagt (z. B. Kräfte, Momente, elektrische Energie etc.). Die Rückkopplung erfolgt durch Sensoren, welche die Reaktion des technischen Systems erfassen. Weil das System in die Berechnungsschleife der Simulation eingebunden ist, wird von Hardware-in-the-Loop gesprochen.

Abbildung 2.28: Hardware-in-the-Loop: Feder-Neigetechnik-Prüfstand der *Neuen Bahntechnik Paderborn*

Hardware-in-the-Loop-Simulationen gehören zu den Echtzeitsimulationen. Sie dienen vor allem der Ermittlung von Daten über das Verhalten eines Prototyps oder einer Teilkomponente ohne eine reale Umgebung oder ohne ein vollständiges Gesamtsystem. Im letzteren Fall simuliert der Rechner einen Teil des technischen Systems, im ersten Fall nur die Umgebung [Hon98].

Abbildung 2.28 zeigt den Hardware-in-the-Loop-Prüfstand der Neuen Bahntechnik Paderborn. Links ist die Leistungselektronik mit entsprechender Echtzeithardware zu sehen (Schaltschrank). Im Hintergrund befindet sich der Rahmen des Prüfstands, der durch eine eigene Hydraulik aktuiert werden kann [LHLJ00b].

2.4 Struktur mechatronischer Systeme

Die Frage nach der Struktur eines Systems zielt in den meisten Fällen auf ein abstraktes Verständnis der Wirkzusammenhänge innerhalb des betrachteten System hin. Bei mechatronischen Systemen und den darauf aufbauenden selbstoptimierenden Systemen gibt die Struktur des Systems wichtige Hinweise auf die Struktur der zugehörigen Informationsverarbeitung. Darüber hinaus bildet die reduzierte Darstellung der Struktur die Grundlage für das Topologische Modell, mit der die objektorientierte Modellbildung beginnt. Der Kreis schließt sich bei den selbstoptimierenden Systemen bei der Frage nach der Rekonfiguration der Struktur, die zu rekonfigurierbaren Systemen führt (vgl. Abschnitt 3.2). Zunächst steht hier jedoch der Systembegriff im Vordergrund, um zu wichtigen Strukturierungsansätzen zu gelangen.

2.4.1 Systembegriff

Der Begriff System bezeichnet im Allgemeinen eine Anordnung von Gebilden. Es können reale oder imaginäre Gebilde als System aufgefasst werden. Die Definition dessen, was zu einem System gehört (Systemgrenzen), hängt wesentlich vom Betrachter und dessen Intentionen ab. Meist gibt das Untersuchungsziel die Art der Modellierung und die Systemgrenzen vor.

In der Regelungstechnik wird ein *dynamisches* System als ein Gebilde angesehen, auf das verschiedene Größen einwirken. Das dynamische System wird in diesem Zusammenhang häufig als Prozess oder Strecke bezeichnet (vgl. [Föl94]). Systeme lassen sich in Teil- und Subsysteme gliedern und können ihrerseits in übergeordnete Systeme eingebettet sein. Diese Hierarchisierung bildet die Grundlage für die abstrakte Darstellung komplexer technischer Anlagen, Prozesse, Vorgänge, aber auch Softwaresysteme. Der Systembegriff erlaubt auch die Abbildung von Beziehungen zwischen Systemen unterschiedlicher Domänen, wie beispielsweise in der Regelungstechnik, bei der mechanische Teilsysteme mit elektrischen oder informationstechnischen Systemen vernetzt sind.

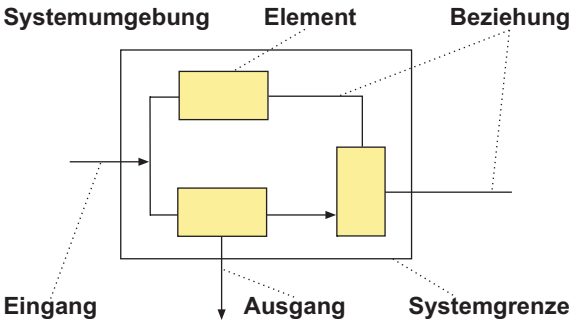

Abbildung 2.29: Definition des Begriffs System

Der Grundbaustein eines Systems ist das *Element*, das auch als Teil, Komponente oder Gebilde bezeichnet werden kann. Ein Element kann wiederum ein System sein. Die Elemente sind durch Beziehungen miteinander oder mit ihrer Umgebung verknüpft und stehen dadurch in Wechselwirkung zueinander. Beziehungen können gerichtet oder ungerichtet ausgeprägt sein. Gerichtete Beziehungen zwischen Elementen oder der Umgebung werden als *Eingänge* oder *Ausgänge* des Elements oder Systems bezeichnet. Bei Beziehungen kann es sich um Stoff-, Energie- oder Informationsflüsse handeln. Abbildung 2.29 [FGK+04] verdeutlicht diesen Zusammenhang.

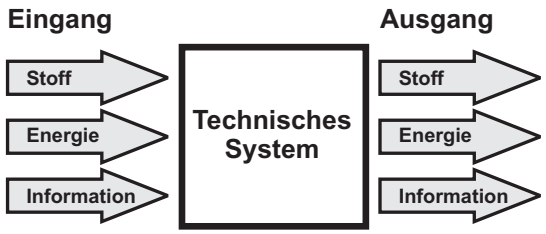

Abbildung 2.30: Allgemeine Systemdarstellung

Wie Abbildung 2.30 [HHK+03] zeigt, sind Energie-, Stoff- und Informationsfluss als gerichtete Verbindungen angelegt. Werden spezielle Wechselwirkungen betrachtet, wie beispielsweise bei mechanisch gekoppelten Systemen, müssen gerichtete Verbindungen durch ungerichtete Beziehungen oder Kopplungen ersetzt werden. Insbesondere in der Mehrkörperdarstellung sind ungerichtete Verbindungen für die Modellierung vorteilhaft, da hierbei physikalische Zusammenhänge nicht frühzeitig in mathematische Größen umgewandelt werden müssen (vgl. [Hah99]).

Die Beziehungen der Elemente und die Elemente selbst bilden insgesamt die Struktur eines Systems ab. Die Systemgrenze eines Systems trennt es von seiner Umgebung. Die Grenze muss dabei nicht physikalisch vorhanden sein, sondern kann als gedanklich gezogene Grenze definiert sein. Entscheidend für die Grenze ist letztlich der Nutzen für eine sinnvolle Strukturierung. Je nach Standpunkt des Betrachters ist alles, was außerhalb dieser Systemgrenze liegt, die Umgebung des Systems.

Da Elemente nach der vorliegenden Definition wieder Teile eines Elementes sein und Elemente wiederum als (Teil-)System aufgefasst werden können, ist es möglich, beliebige Hierarchien aufzubauen. Dabei spielt die Ebene, auf der sich das Element befindet, zunächst keine Rolle.

Nach DIN 40150 ist eine konkrete Unterteilung für technische Systeme definiert [HHK+03]:

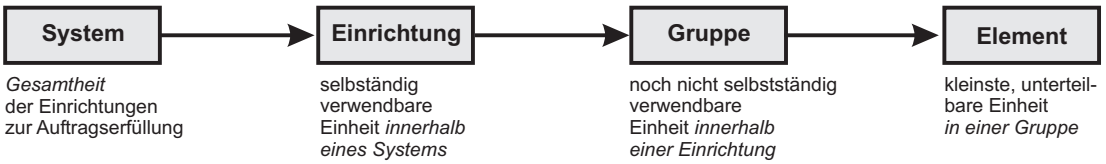

Abbildung 2.31: Unterteilung von technischen Systemen nach DIN 40150

Gut anwendbar ist diese Einteilung auf bestimmte technische Systeme wie beispielsweise Kraftfahrzeuge. Problematisch wird es, wenn informationstechnische Systeme, aber auch Regelungen, die viele Hierarchieebenen aufweisen, hinzukommen. Darüber hinaus ist an einer so konsequenten Struktur nur schwer festzuhalten, wenn technische Hilfseinrichtungen, die mehrere Hierarchieebenen berücksichtigen, hinzukommen.

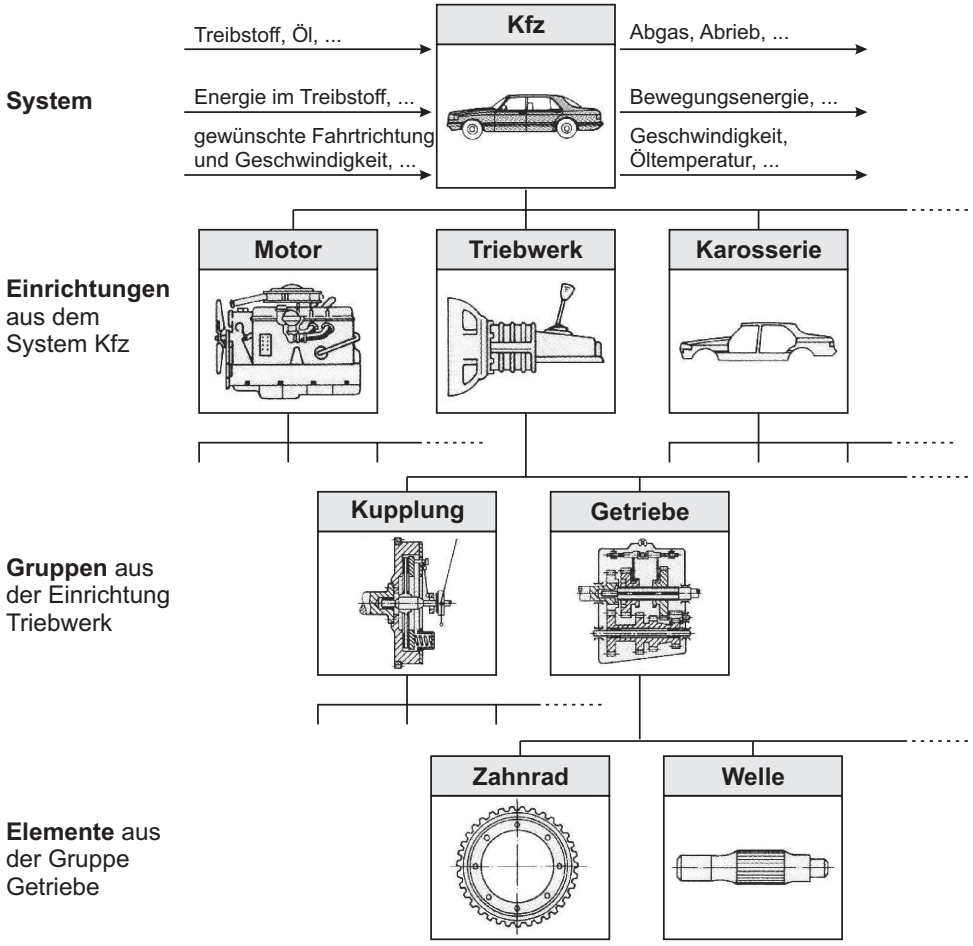

Abbildung 2.32: Unterteilung nach DIN 40150 am Beispiel eines Kfz

Wie Abbildung 2.32 [HHK+03] deutlich macht, ist die Hierarchisierung nach DIN 40150 für die Gliederung und die Modellierung technischer Systeme gut geeignet, erlaubt aber keine beliebige Erweiterung der Hierarchisierungstiefe, wie sie für die Modellierung selbstoptimierender mechatronischer Systeme nötig ist. Aus diesem Grund muss ein anderes Hierarchisierungsparadigma, in Anlehnung an den oben dargestellten Systembegriff, entwickelt werden, das diesen neuen Anforderungen entspricht.

2.4.2 Funktionsorientierte Modellierung und Strukturierung

Betrachtet man die Unterteilung in Abbildung 2.32, so ergibt sich offensichtlich aus der technischen Struktur des Systems eine „Ist-Teil-Von“-Hierarchie. Jede Komponente enthält weitere Komponenten und ist wiederum in andere eingebettet. Auf der obersten Ebene befindet sich das Produkt.

Wird das konkrete technische System in den Hintergrund gestellt und seine abstrakte Funktion in den Vordergrund, so ergibt sich eine Funktionsstruktur, die der technischen Struktur des Systems entspricht.

In dem gezeigten Beispiel wurde das technische System abstrahiert – dadurch wurde seine Funktionsstruktur, also das Zusammenwirken von verschiedenen Einzelfunktionen, sichtbar. Wird der Prozess umgekehrt, kann, ausgehend von der Funktion und der Funktionsstruktur, die Struktur des technischen Systems hergeleitet

werden. Durch eine Konkretisierung der Teilfunktionen durch reale technische Geräte, Aggregate und Komponenten lassen sich auf diese Weise technische Systeme aus ihrer Funktion heraus entwerfen. Diese Vorgehensweise wird auch als *Funktionsorientierter Entwurf* verstanden [LK03], [Wal03] bzw. [PB97].

Für selbstoptimierende Systeme hat diese Art der Modellierung einen entscheidenden Vorteil: Hier wird von einer abstrakten Eigenschaft auf eine konkrete technische Lösung geschlossen und nicht von einer konkreten Teillösung auf ein Gesamtsystem. Weiterhin sind Systemstruktur und Funktionsstruktur weitgehend identisch. Somit sind eine Modellierung der Systemeigenschaften und die Zuordnung von Wirkursachen leichter. Verbindet man dies mit dem zuvor dargestellten Systembegriff, ergibt sich ein erster Ansatz für die Darstellung selbstoptimierender Systeme.

2.4.3 Modular-hierarchische Bauteilstruktur

Modulbildung und Hierarchien unterstützen den Entwurf von komplexen technischen Systemen. Sie reduzieren die Komplexität des Gesamtsystems auf viele einfachere Teilsysteme. Im Maschinenbau ist die konkrete Ausprägung dieses Ansatzes die Bildung von technischen Aggregaten im Sinne von Funktionseinheiten. Grundidee ist dabei, Baugruppen losgelöst vom Gesamtsystem entwerfen, entwickeln, testen und fertigen zu können. Durch diesen Ansatz werden Entwicklung und Fertigung von Produkten, angefangen von Haushaltsgeräten über das Auto bis zu modernen Flugzeugen, durch mehrere Arbeitsgruppen und durch verschiedene Zulieferer erst möglich.

Auch ein Aggregat kann ein mechatronisches System mit eigener Aktorik, Sensorik und Regelungstechnik sein. Wird dieses Teilsystem in ein Gesamtsystem mit einer Regelung eingebaut, ergibt sich eine kaskadierte Regelung. Kaskadenregler haben ihre Vorteile gegenüber komplexen Mehrgrößenreglern insbesondere in Entwurf, Auslegung und Inbetriebnahme. Hier führt es zu einer Vereinfachung der Verfahren. Ist ein mechatronisches System aus mechatronischen Aggregaten aufgebaut, so ist die regelungstechnische Kaskade eine inhärente Eigenschaft dieses Systems.

Im Sinne der Objektorientierung handelt es sich bei der Bildung von Aggregaten um eine Objekthierarchie, also um eine Gliederung im Sinne von *ist Teil von*. Ein PKW besteht beispielsweise aus Karosserie, Antriebsstrang, Bremssystem, Fahrwerk etc. Der Antriebsstrang besteht wiederum aus Motor, Getriebe, Kardan, Differential etc. Diese Gliederung lässt sich beliebig bis zur kleinsten Schraube fortsetzen – es ist eine Zerlegung in Bauteile (vgl. Abbildung 2.32).

Eine Objekthierarchie, die sich aus der Funktionsorientierung ableitet, bildet Funktionsgruppen, von denen sich Baugruppen (Aggregate) ableiten lassen. Aggregate können primär Aktorik und Sensorik anbinden, sie können jedoch auch unterlagerte Aggregate als Aktorik oder Aktor-Sensor-Gruppen enthalten. Solche primären Aggregate sind mechatronische Funktionseinheiten oder Funktionsmodule. Sie bestehen aus Aktorik, Sensorik und Mikroelektronik, welche die mechatronische Funktion, etwa die Regelung, realisiert. Ein solches *mechatronisches Funktionsmodul (MFM)* kann auch hierarchisiert werden; dann besteht die Aktorik aus einem oder mehreren unterlagerten MFM.

Ein mechatronisches Gesamtsystem unterscheidet sich von seinen Baugruppen dadurch, dass es autonom betrieben werden kann. Die Tragstruktur verbindet alle Aggregate zu einem Gesamtsystem. Durch die Verbindung zum Gesamtsystem werden neue überlagerte Funktionen möglich, die nicht einem speziellen Aggregat bzw.

MFM zugeordnet werden können. Es sind Funktionen der Objektebene des autonomen Systems. Ein Beispiel für eine Funktion auf dieser Ebene ist das elektronische Stabilitätsprogramm ESP in Kfz, das die Fahrzeugstabilität durch Eingriff in das unterlagerte Bremssystem realisiert. Das Gesamtsystem bildet somit ein *autonomes mechatronisches System (AMS)*.

Wenn mechatronische Systeme autonom sind, heißt dies nicht zwangsläufig, dass sie auch autonom im Sinne der *Hauptgebrauchsfunktion* der *Funktionsorientierung* handeln können. Für das Beispiel *Fahrzeug* ist die Hauptgebrauchsfunktion der Transport von Personen oder Gütern zwischen zwei Punkten. Ein konventionelles Kfz benötigt immer noch einen Fahrer, der Fahrziel, Fahrweg, Geschwindigkeit etc. festlegt bzw. steuert. Auch wenn es alltäglich erscheint, ist dies ein höchst komplexer Prozess! Ein Überholvorgang beispielsweise erfordert die Berücksichtigung des übrigen Verkehrs. Der Fahrer muss hierfür (stellvertretend) für das autonome mechatronische System in Interaktion mit anderen Verkehrsteilnehmern und der Umwelt treten.

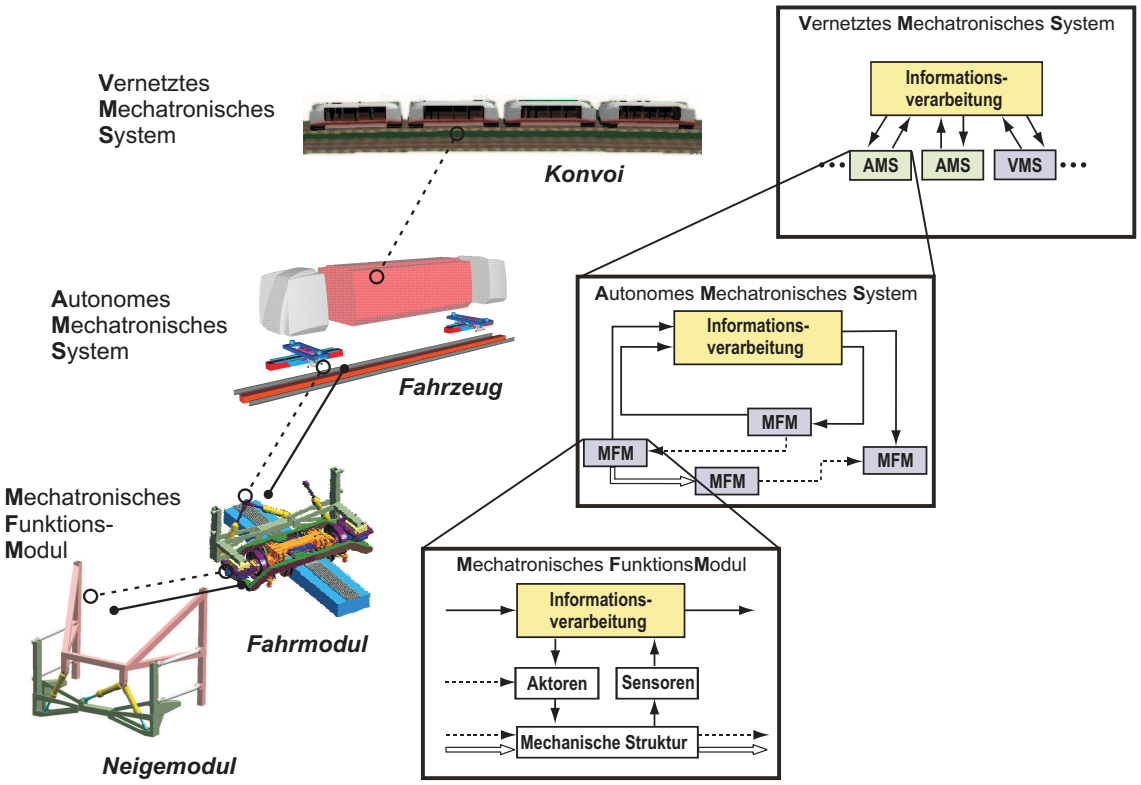

Abbildung 2.33: Modular-hierarchische Bauteilstruktur

Es gibt aber auch autonom handelnde Systeme, die selbstständig Ziele verfolgen. Moderne Flurförderfahrzeuge transportieren eine Last ohne Fahrer von einem Punkt zu einem anderen. Die Steuerung kann zentral durch eine Leitwarte erfolgen oder dezentral, d. h. jedes Fahrzeug ermittelt seine Fahrstrecke selbstständig und interagiert mit der Umwelt und anderen Fahrzeugen.

Wenn mechatronische Systeme autonom handeln und in Interaktion mit anderen mechatronischen Systemen treten müssen, werden Funktionen wie Kommunikation oder Koordination (auch für gemeinsame Planung) benötigt. Regelungstechnische Funktionen, wie die Abstandsregelung in einer Kolonne, können auf dieser Funktionsebene ebenfalls realisiert werden. Da sich diese Funktionen nicht einem einzelnen

autonomen System zuordnen lassen, wird eine Ebene in der Objekthierarchie benötigt, die eine Vernetzung autonomer mechatronischer Systeme ermöglicht. Auf dieser Ebene der *vernetzten mechatronischen Systeme (VMS)* existieren nur noch informationstechnische Kopplungen zwischen den einzelnen Teilsystemen, die aus AMS oder anderen VMS bestehen können.

Die Abbildung 2.33 illustriert die Beziehungen zwischen MFM, AMS und VMS [LHLH01], [GO03]. Wie das Beispiel zeigt, können auf jeder Ebene verschiedene Module existieren, die untereinander verkoppelt sind. Auch eine weitere Hierarchisierung ist möglich. Aufgrund der guten Skalierbarkeit ist die modular-hierarchische Bauteilstruktur ohne weiteres auf selbstoptimierende mechatronische Systeme übertragbar. Die Informationsverarbeitung einer Komponente (MFM, AMS, VMS) wird jedoch durch das Operator-Controller-Modul (OCM) konkretisiert (vgl. Abschnitt 3.4).

2.5 Agententechnik als Entwurfsparadigma für proaktive Informationsverarbeitung

Die Objektorientierung als ein Paradigma für die Entwicklung von informationstechnischen Systemen brachte nach der prozedualen Programmierung einen neuen Ansatz, der es erlaubt, mit der zunehmenden Komplexität der Programmentwicklung fertig zu werden. Nachdem Softwarekomponenten als Objekte aufgefasst wurden, fiel es leichter, auch große Projekte mit vielfältigen Funktionen umzusetzen. Im Maschinenbau ist der Weg zu Modulen und Komponenten schon vor mehr als einem Jahrhundert begonnen worden. Heute werden mechatronische Produkte ganz selbstverständlich aus aktiven Komponenten zusammengesetzt. Der Weg von Komponenten zu Aggregaten kann verglichen werden mit dem Weg vom Objekt zum Agent. Wird ein Objekt aktiv, so dass es aus sich selbst heraus handelt und Ziele verfolgt, so wird von einem (Software-)Agenten gesprochen. Dies vereinfacht den Entwurf von Systemen mit vielen parallel ablaufenden und einander beeinflussenden Teilkomponenten. Die Agententechnik ist somit ein Paradigma, das es erleichtert komplexe, informationstechnische Systeme zu entwerfen.

2.5.1 Einführung

Der Begriff der Agententechnik ist nur schwer in eine allgemeingültige Definition zu fassen – jedoch sind eine Anzahl von Arbeiten in diesem Bereich hervorzuheben. Insbesondere die Arbeiten von Ferber [Fer99], Wooldridge und Jennings [WJ95], [JW98] sind weithin anerkannt. Allgemein ist die Agententechnik ein abstraktes Paradigma zum Entwurf von komplexen Software-Systemen. Grundlage bildet die Objektorientierung. Während ein Objekt jedoch passiv ist und nur durch einen Impuls von außen aktiv wird, ist ein Agent ein aktives Objekt, das *selbsttätig* Handlungen aufgrund seines inneren Zustands ausführt.

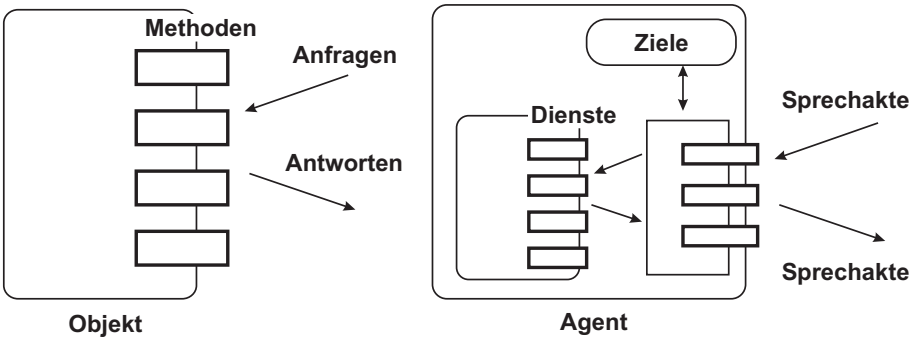

Abbildung 2.34: Unterschied zwischen Objekt und Agent

Ferber [Fer99] unterscheidet Objekt und Agent nach der inneren Struktur (Abbildung 2.34). Ein Objekt antwortet auf eine Anfrage mittels Methoden direkt (Abbildung 2.34, links), während ein Agent in Dialog mit seiner Umwelt bzw. mit anderen Agenten tritt. Die Kommunikation wird quasi durch Methoden gefiltert. Dadurch sind die inneren Dienste von der Außenwelt gekapselt. Die Aktionen, die Agenten ausführen, sind dabei nicht nur Antworten auf Anfragen von außen, sondern werden außerdem von eigenen Zielen stimuliert (Abbildung 2.34, rechts).

Für die Strukturierung der Informationsverarbeitung in selbstoptimierenden Systemen bietet sich die Agententechnik zur Bildung von selbsttätig optimierenden Funktionselementen an, die auch als selbstoptimierende Agenten bezeichnet werden können. Hierbei ist der Agent zunächst nicht selbstoptimierend, sondern führt selbsttätig Handlungen aus, die zur Optimierung eines technischen (Regler-)Systems führen. Insbesondere die strikte Kapselung der Dienste erlaubt die Anwendung vielfältiger Methoden aus der KI, die insbesondere die Planung und die Bewertung von Zielen für eine modellbasierte Optimierung ermöglichen. Dementsprechend wird in [GLR01] ein Agent wie folgt definiert:

> *„Ein Agent ist ein autonomes, proaktives, kooperatives und hochgradig adaptives Funktionsmodul. Autonom impliziert eine eigenständige Kontrolle, die von sich aus Aktionen initiiert (proaktiv). Agenten werden als Funktionsmodule angesehen, die in Kooperation oder Konkurrenz zueinander handeln. Adaptiv bezeichnet ein zur Laufzeit generisches Verhalten, das beispielsweise auch Lernfähigkeit beinhalten kann. Ein Funktionsmodul wird als heterogenes Teilsystem mit elektronischen, mechanischen und informationstechnischen Komponenten verstanden.“* [GLR01]

Daraus lassen sich folgende Grundeigenschaften ableiten:
- Unabhängiges Handeln (Autonomie)
- Selbsttätige Verfolgung von Zielen (Proaktivität)
- Kommunikation mit anderen Agenten (Kooperativität)
- Anpassung an Veränderungen (Anpassungsfähigkeit)

Führt man sich vor Augen, dass es sich um Softwarekomponenten handelt, kann man feststellen, dass diese Eigenschaften von einer ganzen Reihe klassischer Systeme erfüllt werden. Beispielsweise sind adaptive Reglersysteme in gewisser Weise autonom, da sie ohne Zutun von außen Messwerte vom technischen Prozess aufnehmen, verarbeiten und entsprechende Ausgangswerte aufschalten (Autonomie). Anpassungsfähig sind solche Systeme durch entsprechende Adaptionsalgorithmen (Anpassungsfähigkeit). Bei einer Reglerkaskade werden kontinuierlich Werte zwischen

verschiedenen Reglerebenen ausgetauscht, was als eine einfache Form der Kooperation gedeutet werden kann. Auch wenn eine Regelung die eindeutige Aufgabe der Beeinflussung der Dynamik des technischen Systems oder Prozess hat, fällt es schwer, Proaktivität in eine adaptive Regelung hinein zu interpretieren. Dazu fehlt die Wahlfreiheit bei den auszuführenden Aktionen. Des Weiteren darf nicht vergessen werden, dass ein Agent auch alle Forderungen der Objektorientierung zu erfüllen hat, etwa die nach Abgeschlossenheit.

Ein anschauliches Beispiel für die Anwendung der Agententechnik im Gegensatz zur Objektorientierung ist die *Digitale Fabrik* nach [Rit07]. Die Teilsysteme, die am Produktionsprozess beteiligt sind, werden als Agenten aufgefasst. Deutlich wird das am Beispiel eines Transportauftrages für einen Flurförderer: während in der Objektorientierung ein freier Flurförderer vom System gesucht werden muss und daraufhin der Transportauftrag zugeteilt, bietet im Ansatz der Agententechnik ein frei werdender Flurförderer seine Transporttätigkeit an. Auf diese Weise wird keine zentrale Stelle mehr benötigt, die Transportaufträge zuteilt.

Das Beispiel zeigt, wie auch in technischen Problemstellungen die Agententechnik als ein Entwurfsparadigma wirken kann, das zu dezentralen Architekturen führen kann. Selbsttätigkeit und Autonomie der Agenten führen zu *selbstorganisierenden Systemen*. Die Selbstorganisation ist vor allem auf den höheren Ebenen selbstoptimierender Systeme anwendbar, nämlich dann, wenn verschiedene autonome mechatronische Systeme (AMS) auf Ebene der vernetzten mechatronischen Systeme (VMS) zusammenwirken. Bei der Regelung von Prozessen auf der Ebene der Mechatronischen Funktionsmodule (MFM), bei der die Dynamik des Systems eine größere Rolle spielt, müssen strengere Maßgaben, z. B. bei der Einhaltung von Zeitschranken, angewendet werden, als es die Selbstorganisation vorgibt.

Ein weiteres populäres Beispiel für die Anwendung der Agententechnik beim Entwurf und der Realisierung von mechatronischen Systemen ist die Realisierung der Raumsonde *Deep Space 1*[17] der NASA, die am 24.10.1998 zum Kometen *Borrelly* gestartet wurde. Wesentliche Komponenten waren dabei die *Autonome Navigation* und ein *Autonomous Remote Agent*, als intelligenter Assistent des Bodenpersonals an Bord der Sonde.

2.5.2 Klassifizierung von Agenten

Agenten können in verschiedene Klassen eingeteilt werden. Häufig genannt werden dabei *Reaktiver Agent* (reactive agent), *Verarbeitender Agent* (processing agent) und *Kognitiver Agent* (cognitive agent) [Att00].

Reaktiver Agent (reactive agent) Dies ist ein Agent mit der Fähigkeit, auf externe Ereignisse zu reagieren, um das Verhalten eines technischen Systems und die Kontrolle darüber aufrechtzuerhalten oder als kontrollierender Agent andere Agenten zu überwachen und die Aktivitäten zwischen einem Satz von Agenten zu koordinieren (Kontrollinstanz in einem Multi-Agentensystem).

Verarbeitender Agent (processing agent) Die Rolle dieses Agenten ist es, Umwandlungen auszuführen. Dabei werden Prozesse oder Berechnungen auf einen eingehenden Datenfluss hin ausgeführt, um einen ausgehenden Datenfluss zu

[17] http://nmp.jpl.nasa.gov/ds1/

produzieren. Der Agent sichert die Ausführung eines (komplexen) Algorithmus (operativer Teil eines Multi-Agenten-Systems).

Kognitiver Agent (cognitive agent) Dies ist ein Agent, der nicht nur Daten und prozedurale Methoden enthält, sondern auch eine Wissensbasis. Er ist außerdem in der Lage, Schlussfolgerungen aus seinem Wissen oder dem Wissen anderer zu initiieren.

Die Eigenschaft eines Agenten, mit anderen Agenten in Dialog treten zu können, ist insbesondere bei hierarchischer Optimierung nützlich, wenn verschiedene selbstoptimierende Agenten Informationen austauschen können. Dies können etwa Daten über das aktuelle Optimierungsziel, einfache Parameter oder komplexere Informationen sein. Für das Zusammenspiel von Agenten existiert eine weitere, auf der Agententechnik aufbauende Theorie der *Multiagentensysteme*. Multiagentensysteme ergeben sich dann, wenn einzelne Agenten miteinander in Aktion treten.

2.5.3 Multiagentensysteme

Nach der Agententheorie können einzelne Agenten in Interaktion mit ihrer Umgebung (auch Softwareumgebung) treten. Ein Sonderfall ist dabei die Interaktion zwischen Agenten. Nach [WJ95] ist ein Multiagentensystem ein Informationssystem der verteilten Wissensverarbeitung, das durch die Eigenschaften Autonomie, Kollaborativität, Reaktivität und Proaktivität gekennzeichnet ist. Der Begriff Autonomie bedeutet hier vor allem selbstständiges Handeln und Entscheiden. Mit Kollaborativität ist die Fähigkeit gemeint, mit anderen Agenten und ggf. mit der Umgebung und den Menschen in Beziehung zu treten und sie in die eigene Handlung einzubeziehen. Die Reaktivität bezeichnet die Fähigkeit, auf Ereignisse (in der Umgebung) zu reagieren. Der Begriff der Proaktivität ist eine wesentliche Forderung an Agenten. Er meint in diesem Zusammenhang vor allem

> *„...ein frühzeitiges initiatives (handeln) im Gegensatz zu einem abwartenden reaktiven Handeln...“* [Wik07]

Somit tritt ein Agent bzw. ein Multiagentensystem selbstständig in Aktion und reagiert *nicht nur* auf Ereignisse. Eine andere Betrachtung geht davon aus, dass Multiagentensysteme eine konsequente Erweiterung objektorientierter Entwurfstechniken darstellen [WC01].

In Multiagentensystemen werden zwei wesentliche Ebenen unterschieden: Die untere Schicht beschreibt im Wesentlichen das lokale Verhalten eines Agenten und wird als *Mikrosicht* bezeichnet. Die überlagerte Ebene dient der Kollaboration der Agenten, wobei hier der Schwerpunkt auf Management und Kommunikation liegt. Diese Schicht wird als *Makrosicht* eines Agenten bezeichnet. Interessanterweise erfordert die Realisierung dieser Ebenen unterschiedliche Ansätze und Werkzeuge. So wird die Mikrosicht aufgrund ihres maschinennahem Abstraktionsniveaus häufig in Sprachen wie C, C++, Prolog oder Lisp kodiert, während auf der Makroebene häufig abstraktere Sprachen Anwendung finden. Definition und Bearbeitung von Zielen und Strategien erfolgten oft auf Basis von kognitiven, deskriptiven Konzepten der BDI-Architektur (Belief-Desire-Intention) [RG95] (vgl. 2.2.7 und insbesondere Abbildung 2.20).

Kapitel 3

Konzept für Entwurf und Struktur selbstoptimierender Systeme

> Es kommt nicht darauf an, mit dem Kopf durch die Wand zu rennen, sondern mit den Augen die Tür zu finden (Werner von Siemens)

In diesem Kapitel werden Ansätze zum Entwurf und zur Strukturierung selbstoptimierender Systeme vorgestellt. Dabei bilden bekannte Methoden zur systematischen Entwicklung und zur Strukturierung mechatronischer Systeme die Grundlage für die Entwicklung und die Strukturierung selbstoptimierender Systeme.

3.1 Einführung

Die Entwicklung komplexer mechatronischer Systeme, wie etwa moderner Kraftfahrzeuge, ist ohne genaue Planung und systematisches Vorgehen nicht durchführbar. Vorhandene Entwicklungsressourcen gezielt und effizient einzusetzen, ist hierbei der entscheidende Faktor, um Produkte in kurzer Zeit mit der vom Kunden erwarteten Qualität auf den Markt zu bringen. Entwicklungsystematiken kanalisieren hierbei die zu leistenden Entwicklungstätigkeiten und tragen zur Modularisierung des Entwicklungsprozesses bei. Für die Entwicklung selbstoptimierender mechatronischer Systeme gilt dies in besonderem Maße, da hier der Komplexitätsgrad weiter steigt.

Jedoch gibt es keine allgemeingültige, auf alle Entwicklungen mechatronischer Produkte zu übertragende kanonisierbare Vorgehensweise [Ver03]. Erfolgversprechender sind flexible Vorgehensmodelle, die der innovativen Entwicklung technischer Produkte dort genug Freiraum lassen, wo er nötig ist.

Die klassische Methode zur Unterstützung und vor allem zur Formalisierung der Konstruktion im Maschinenbau ist vor allem die Konstruktionssystematik [PB97]. Neben Methoden zur Entwicklung von konstruktiven Lösungsansätzen werden in der Konstruktionssystematik auch Vorgehensmodelle betrachtet. Aufgrund der Entwicklung der Mechatronik, die unterschiedliche Domänen verbindet, genügten die Ansätze der Konstruktionssystematik nicht mehr, um moderne mechatronische Systeme ganzheitlich zu entwerfen.

3.2 Rekonfigurierbare Systeme als Grundlage für Selbstoptimierung in mechatronischen Systemen

Der Begriff der *Rekonfiguration* ist bisher in der Mechatronik noch nicht etabliert. Er stammt vor allem aus dem Bereich der Rechentechnik (siehe z. B. [TGR+03]). Hierbei geht es vor allem um rekonfigurierbare Schaltungen und die damit zusammenhängende Informationsverarbeitung.

Für technische Systeme muss der Begriff aber erst noch genauer spezifiziert werden. Rekonfiguration eines Systems allgemein bedeutet: Vorhandene Komponenten eines Ganzen werden neu zueinander angeordnet. Dies schließt alle Domänen der Mechatronik ein. Somit kann Rekonfiguration in den Domänen der Mechanik, der Elektrotechnik und der Informationsverarbeitung auftreten. Für die weitere Diskussion wird sie wie folgt definiert:

> Rekonfigurierbare Systeme sind mechatronische Systeme, die ihre innere Struktur oder Wirkstruktur verändern können, ohne dass dabei Komponenten hinzugefügt oder ergänzt werden.

Ein einfaches Beispiel für ein rekonfigurierbares System ist die klassische Zweikreisbremse (siehe Abbildungen 3.1 und 3.2, [FGH+04]). Bei Druckverlust[18] in einem Kreis der Bremsanlage wird der beschädigte hydraulische Bremskreis automatisch verschlossen. Die Konfiguration der Komponenten verändert sich so, dass nur noch ein Bremskreis im Eingriff bleibt. Dabei ändern sich auch die technischen Eigenschaften des Systems, wie Bremskraft und -wirkung und die Dynamik des Systems insgesamt.

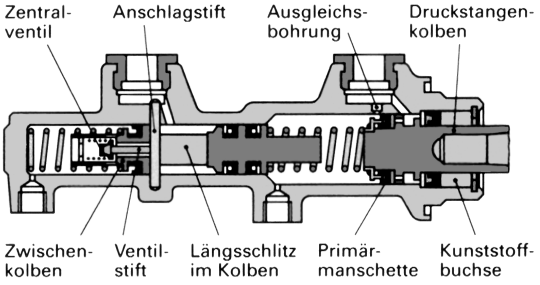

Abbildung 3.1: Gestufter Tandemhauptzylinder mit Zentralventil

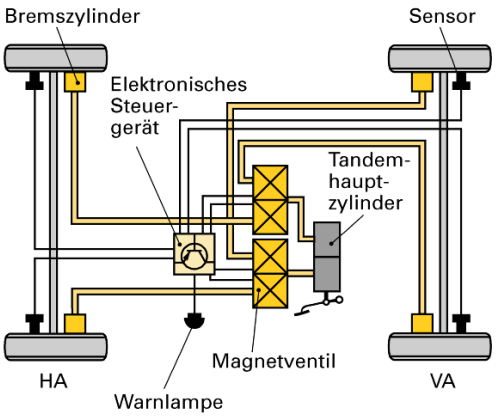

Abbildung 3.2: Schematischer Aufbau einer Zweikreis-Bremsanlage

Als rekonfigurierbares System beschrieben, besteht das Zweikreis-Bremssystem aus einer Konfiguration von mechanisch-hydraulischen Komponenten, die drei mögliche Konfigurationen annehmen können:

a) Zweikreisbetrieb – beide Bremskreise aktiv

[18] Der Verlust von Bremsflüssigkeit als Masse kann vernachlässigt werden. Die Systemkomponenten ändern sich nicht in ihrer Anzahl, sondern wirken anders zusammen.

b) Bremskreis 2 ausgefallen – Bremskreis 1 aktiv

c) Bremskreis 1 ausgefallen – Bremskreis 2 aktiv

Streng genommen müsste auch der Ausfall beider Bremskreise als mögliche Konfiguration betrachtet werden. Ob weitere *gültige* Konfigurationen mit in das Modell eines rekonfigurierbaren Systems aufgenommen werden, ist jedoch eine Frage der Modellierungstiefe. Für die Betrachtung des Problems sollen drei Konfigurationen genügen.

Schematisch kann das Problem der Zweikreisbremse, stark vereinfacht, als ebenes Blockschaltbild dargestellt werden:

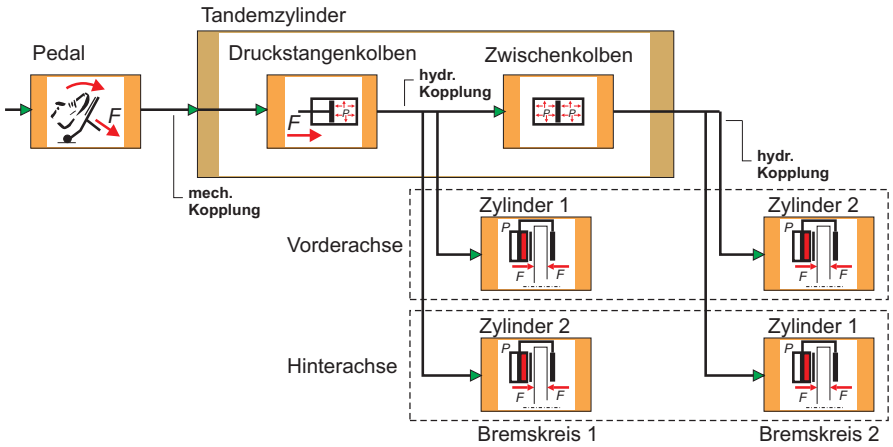

Abbildung 3.3: Grundkonfiguration

1. In der Grundkonfiguration (Abbildung 3.3) der Zweikreisbremse wirkt die Pedalkraft über eine mechanische Kopplung auf den Druckstangenkolben. Dieser baut Druck auf. Über eine hydraulische Kopplung wird dieser Druck an die Bremszylinder des 1. Bremskreises und den Zwischenkolben weitergeleitet. Der Zwischenkolben baut wiederum Druck auf und leitet diesen über eine hydraulische Kopplung an die Bremszylinder des 2. Bremskreises weiter.

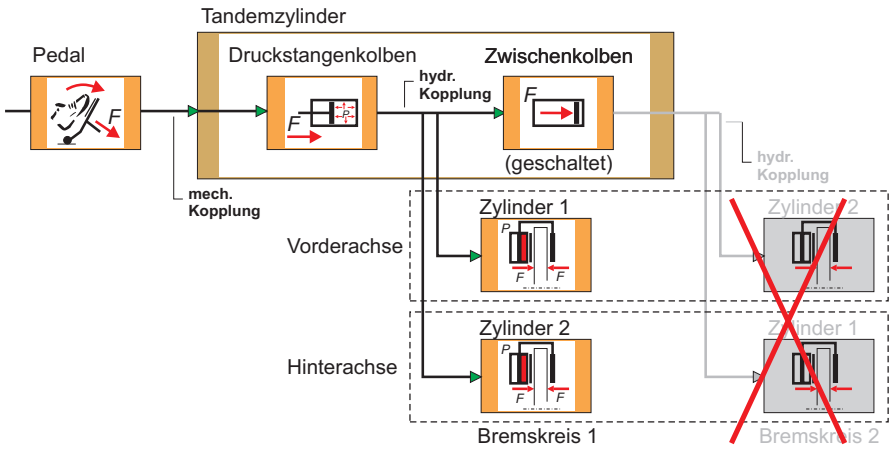

Abbildung 3.4: Bremskreis 2 ausgefallen

2. Fällt der 2. Bremskreis aus, bewirkt der Zwischenkolben, dass der zweite Kreis hydraulisch entkoppelt ist. Der Zwischenkolben fährt bis in die Endposition, der 1. Kreis bleibt funktionsfähig.

   Da der Zwischenkolben in dieser Konfiguration nun nicht mehr bewegt werden kann, wird diese Komponente statisch, d. h. für die Beschreibung der Eigenschaften des Rest-Bremssystems spielen die ursprünglichen Eigenschaften dieser Komponente (Dynamik, innere Reibung etc.) keine Rolle mehr.

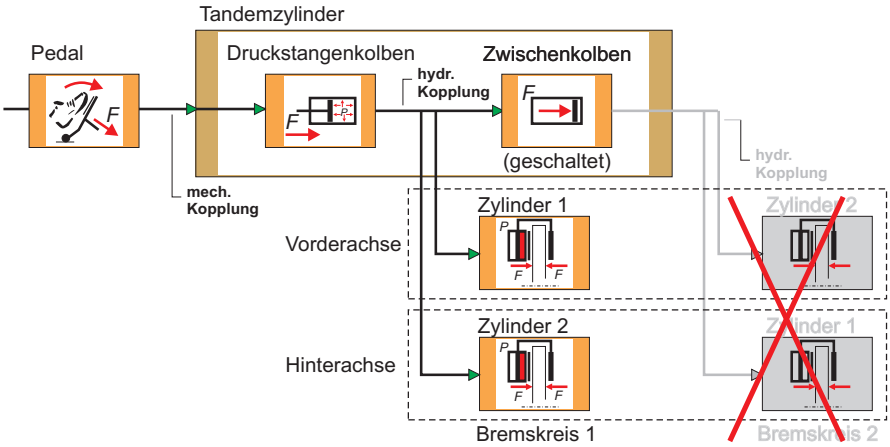

Abbildung 3.5: Bremskreis 1 ausgefallen

3. In der dritten Konfiguration ist der 1. Bremskreis ausgefallen und der 2. intakt. Der Druckstangenkolben drückt nun direkt auf den Zwischenkolben. Aus der hydraulischen Kopplung zwischen Druckstangenkolben und Zwischenkolben wird eine mechanische Kopplung.

   Die Eigenschaften des Druckstangenkolbens und des Zwischenkolbens ändern sich durch den Verlust des 1. Bremskreises deutlich. Die Struktur dieser Konfiguration unterscheidet sich durch den Austausch einer Kopplungsart (von hydraulisch zu mechanisch) noch deutlicher von der Ausgangskonfiguration.

Obwohl dieses Beispiel schon stark vereinfacht ist, bringt es doch die klassische statische Beschreibung von Modellen an neue Grenzen. Bisherige Umgebungen zur Modellbildung, die in der Lage sind Modelle auf topologischer Ebene zu beschreiben, wie beispielsweise CAMeL-View [iXt06], ermöglichen es nicht, diesen Wechsel der Konfigurationen durch einfache Mittel zu beschreiben. Ein Wechsel der Kopplungen und der Eigenschaften der Komponenten ist bestenfalls auf mathematischer Ebene möglich.

Ein allgemeiner Ansatz leitet sich aus dem Objektorientierten Mechatronikmodell (OMM) ab (vgl. Abschnitt 2.3.6). Dabei wird das OMM auf allen Ebenen des Entwurfs um diskrete Komponenten erweitert:

Wie in Abbildung 3.6 zu sehen ist, wird jede Ebene des OMM um weitere Aufgaben erweitert, die unabhängig von der zeitkontinuierlichen Auswertung zeitdiskret ablaufen. Besonders hervorzuheben ist hier die Ebene der Topologiesicht, auf der ein mechatronisches System in seinen verschiedenen Domänen entworfen wird. Eine Rekonfiguration führt zu einer Kontrolle genau dieser Ebene durch zugeordnete Kontrollmechanismen.

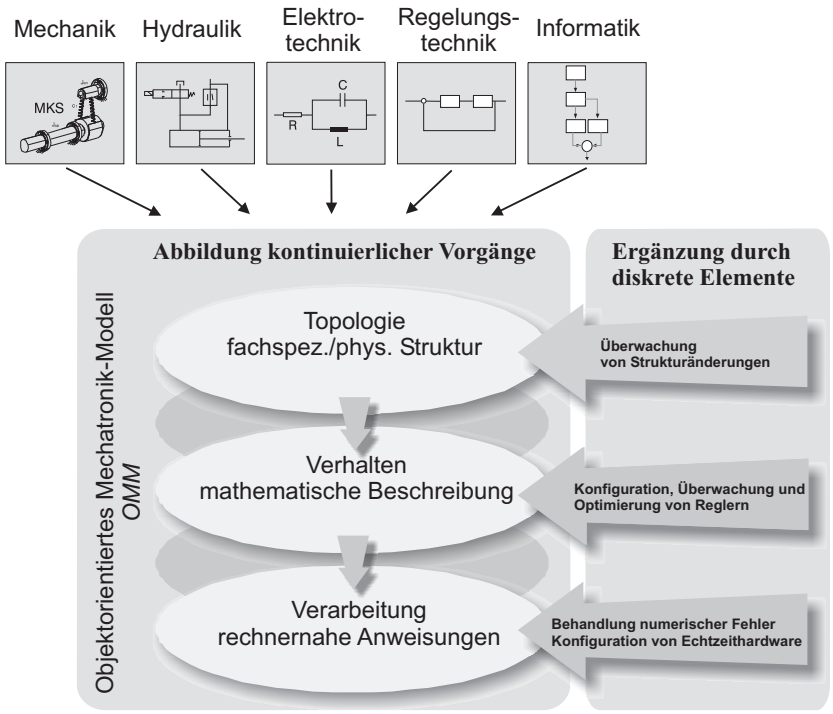

Abbildung 3.6: Erweitertes objektorientiertes Mechatronikmodell

Aus weiteren Ebenen schließen sich weitere Mechanismen an, die für die Selbstoptimierung wichtig sind. Auf der Ebene des *Verhaltens* finden sich Verfahren zur Parameteroptimierung und zur Überwachung von mathematisch zu beschreibenden Komponenten wie Reglern (siehe auch [DO00, DOM01]). Auf der Ebene der Verarbeitung wird die Auswertung selbst überwacht. Im Sinne der Selbstoptimierung spielt hier vor allem die Wahl der richtigen Verfahren und Konfigurationen für die numerische Auswertung eine wichtige Rolle (vgl. Kapitel 4).

Wie das Beispiel der Zweikreisbremse illustriert, ist es für die Modellierung von rekonfigurierbaren Systemen hilfreich, die möglichen Konfigurationen separat, also diskret voneinander zu betrachten. Der Vorteil dieser Darstellung liegt auf der Hand: Jede Konfiguration kann nun unabhängig von den anderen modelliert und beschrieben werden. Besonders umfassend wird der Begriff, wenn er auf nichtlineare Systeme angewendet wird, die stückweise linear sind oder sich durch stückweise lineares Verhalten beschreiben lassen. Dies trifft auf viele nichtlineare Regelungsprobleme zu, die stückweise linearisiert werden können. Ein Wechsel zwischen linearisierten Zuständen des Systems kann als Rekonfiguration dargestellt werden – ein weitreichender Vorteil für die Entwicklung.

Rekonfiguration kann somit als Hilfsmittel für die Modellierung nichtlinearer Probleme (mechatronische Systeme) dienen. Sie bietet aber auch die Möglichkeit, Änderungen in der Struktur eines Systems abzubilden, die zu einer Verhaltensänderung führen. Die Änderung des Verhaltens ist dabei nicht auf mechanische Komponenten wie im oben beschriebenen Beispiel beschränkt. Wird der Wechsel eines Reglers oder des Teils eines Reglers als eine Rekonfiguration des Gesamtsystems beschrieben, lässt sich dies für eine Optimierung zur Laufzeit ausnutzen, womit rekonfigurierbare Systeme eine wichtige Grundlage für die Modellierung selbstoptimierender mechatronischer Systeme bilden können.

Für die Modellierung rekonfigurierbarer Systeme ist es allerdings nötig, ein geeignetes abstraktes Modell zu entwickeln, das die Grundlage für die Benutzersicht späterer Werkzeuge darstellt. Dieses Modell liefern die so genannten *Hybriden Komponenten*, die als Grundlage für die Modellierung rekonfigurierbarer Systeme dienen können.

3.3 Modellierung von hybriden Systemen

Die Modellierung von Systemen, die zeitdiskrete und (quasi-)zeitkontinuierliche Eigenschaften vereinen, ist ein beständig wiederkehrendes Problem in der Informatik. Dabei werden verschiedene Zielsetzungen verfolgt. Auch ist der Begriff selbst nicht eindeutig. Im Allgemeinen sind aber Systeme gemeint, die zum Teil auf analogen technischen Systemen und zum Teil auf zeitdiskreten logischen Abfolgen basieren.

> *„Hybride eingebettete Systeme sind durch die Mischung von diskreter und kontinuierlicher Dynamik gekennzeichnet, wie sie etwa bei digitaler Software in Wechselwirkung mit einer analogen Umgebung vorkommt. Bei dem Entwurf solcher Systeme werden unterschiedliche Zeitmodelle und Techniken aus unterschiedlichen Disziplinen, vor allem aus der Informatik und der Regelungstechnik, eingesetzt. Wenn die diskreten und kontinuierlichen Anteile eines hybriden Systems bereits zu Beginn des Entwicklungsprozesses isoliert betrachtet werden, und wenn zudem die Wechselwirkung dieser Teile nicht präzise definiert ist, so können ungeeignete oder fehlerhafte Entwürfe entstehen.“* [Sta01]

Der Begriff *hybrides System* wird hier gemeinsam mit dem Begriff *Gemischt diskret-kontinuierliches System* verwendet.

In der Realität hybride Systeme sehr komplex werden. Vor allem die Vermischung von kontinuierlichen und diskreten Komponenten führt dazu, dass viele Methoden der jeweiligen Domäne nicht mehr ohne weiteres anwendbar sind, z. B. analytische Verfahren zum Reglerentwurf, Stabilitätsnachweis etc. [Sta01].

Das Problem der *formalen Beschreibung* der Informatik soll an dieser Stelle nicht näher beleuchtet werden, da es das Problem der systematischen Entwicklung solcher Systeme nicht deutlicher macht. Zu diesem Thema sei insbesondere auf die Arbeiten von Burmester [Bur06] und Stauner [Sta01] verwiesen, die sich vor allem mit der Beschreibung hybrider Systeme aus informationstechnischer Sicht befassen. Im Folgenden sollen wichtige Strömungen zu diesem Thema beleuchtet werden.

3.3.1 Beschreibung hybrider Systeme

Zur Unterstützung der gemischt diskret-kontinuierlichen Modellierung von hybriden Systemen wurden in den letzten Jahren eine Reihe von Beschreibungstechniken vorgeschlagen (z. B. [ACH+95], [Lam93], [Wie96]). Tatsächlich steht bei vielen dieser Formalismen aber nicht die adäquate Modellierung realer Systeme, sondern eher die Eignung zum effizienten Einsatz mathematischer Analysemethoden im Vordergrund. Bestes Beispiel sind die (linearen) hybriden Automaten [ACH+95], die den Einsatz von Model-Checking-Algorithmen erlauben, aber aufgrund fehlender Modularität und nur geringer Strukturierungsmöglichkeiten zum praktischen Systementwurf wenig geeignet sind [MS00].

Ausnahmen sind hier die hybride Statechart-Erweiterung von Kesten und Pnueli [KP92] und die Arbeiten im Rahmen des KONDISK[19]-Teilprojekts HYFOS von Jähnichen und Weber [Deu07]. In diesem Projekt wurde untersucht, wie aufbauend auf einer hybriden objektorientierten Erweiterung der Spezifikationssprache Z [Fri97], [FNW99] praxistaugliche hybride Beschreibungstechniken im Stil der UML-Klassendiagramme [IBM07] entwickelt werden können. In [FNW99] werden Verwendungsrichtlinien für diese Klassendiagramme als ein Teil einer Entwicklungsmethodik vorgestellt. Dabei handelt es sich im Wesentlichen um eine Übertragung von Konzepten aus der Informatik; eine Integration mit ingenieurtechnischen Vorgehensweisen findet hier nicht statt.

[CWM98] skizziert einige methodische Paradigmen zur Verwendung von Statecharts beim Entwurf hybrider Systeme. Der dort verfolgte Ansatz ist stark durch die heutige Praxis im Automobilbau geprägt und baut daher allein auf existierenden Techniken, nämlich auf Statecharts und Blockdiagrammen, und kommerziellen Werkzeugen auf. Hybride Systeme können damit erst behandelt werden, nachdem sie in den frühen Entwurfsphasen in rein diskrete und rein kontinuierliche Teile zerlegt wurden. Die Durchgängigkeit der Methode ist daher nicht gegeben.

Ebenfalls interessant ist in diesem Zusammenhang das KONDISK-Teilprojekt *Funktionsbausteinsysteme als diskret-kontinuierliche Steuerungsnetzwerke – formale Struktur und Klassifikation* von Epple (RWTH Aachen). In diesem Projekt wurde an einer Entwicklungsmethodik von Steuerungssoftware für die Prozessleittechnik gearbeitet, die auf Funktionsbaustein-Bibliotheken basiert [Deu07].

In [Sta01] werden formale, integrierte Notationen und Entwicklungstechniken für hybride Systeme eingeführt, die Entwicklungsprozesse unterstützen, in denen eine Partitionierung hybrider Systeme erst in späteren Entwicklungsphasen stattfindet. Dies ist ein wesentlicher Schritt, um dem ingenieurwissenschaftlichen Vorgehen entgegenzukommen. [Bur06] führt diesen Gedanken fort und erweitert ihn für die Rekonfiguration. Die dort vorgeschlagene Notation kann als Grundlage für einen Modellierungsansatz aus technischer bzw. mechatronischer Sicht dienen.

3.3.2 Hybride Hierarchieelemente

Dieser Abschnitt stellt die Anwendung der Blockschaltbilddarstellung für die Definition von rekonfigurierbaren Systemen vor, welche die Grundlage für strukturvariante selbstoptimierende Systeme bilden. Blockschaltbilder dienen allgemein der abstrakten Modellierung technischer Systeme. In zahlreichen CAE-Werkzeugen sind sie die Grundlage der abstrakten Modellierung. Ursprünglich stammen Blockschaltbilder aus der Regelungstechnik, wo sie zur graphischen Darstellung von mathematischen Übertragungsfunktionen dienen.

Blockschaltbilder setzen sich aus Komponenten (Blöcken) zusammen, die durch Verknüpfungen zueinander in Beziehung stehen. Ein Block kapselt eine Funktion oder ein Verhalten. Typischerweise wird dieses Verhalten mathematisch beschrieben, etwa als Differentialgleichungen in Zustandsraumdarstellung für die Beschreibung der Dynamik. Darüber hinaus ist auch eine Beschreibung mit physikalischen Beschreibungselementen, wie etwa in Mehrkörpersystemmodellen, möglich (vgl. Abschnitt 2.3.6). Die Mathematik wird dabei durch Transformationsverfahren wie Newton oder Lagrange automatisch abgeleitet. Zwischen den einzelnen Blöcken bestehen

[19] KONDISK: Schwerpunktprogramm der Deutschen Forschungsgemeinschaft „Analyse und Synthese kontinuierlich-diskreter Systeme“ (1996–2001).

Verknüpfungen oder Verkopplungen, die als gerichtete oder ungerichtete Verbindungen ausgeprägt sein können. Im Falle der gerichteten Verkopplungen werden Daten ausgetauscht, während ungerichtete Verkopplungen häufig funktionelle Beziehungen oder physikalische Bindungen beschreiben, wie etwa eine Verkopplung zwischen Masse und Feder in der Mehrkörpersystemdarstellung.

Dieser zunächst einfache Ansatz lässt sich durch die Einführung von Hierarchien deutlich erweitern. Hierarchische Blockschaltbilder erlauben eine komplexere Modellierung, indem mehrere Blöcke zu Hierarchien gruppiert werden, die wiederum andere Hierarchien enthalten können. Dies ermöglicht einen strukturierten Entwurf und reduziert die Komplexität eines Blockschaltbilds auf der jeweiligen Ebene.

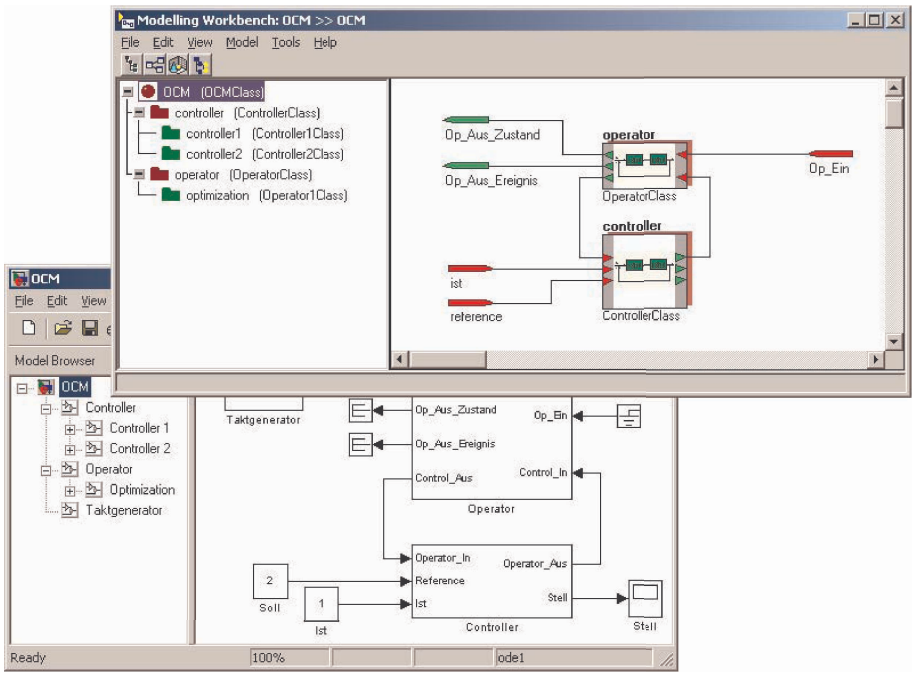

Abbildung 3.7: Blockschaltbilder in den CAE-Werkzeugen CAMeL-View und MATLAB

Abbildung 3.7 zeigt Beispiele für typische Blockschaltbilder in den Werkzeugen CAMeL-View und MATLAB. Im jeweils linken Teil der Fensters sind die Baumstrukturen dargestellt, während auf der rechten Seite die Topologie der jeweils aktuellen Ebene erscheint.

Die Topologie hierarchischer Blockschaltbilder kann als Baum dargestellt werden. Die Blätter dieses Baumes bilden dabei das Verhalten bzw. die Funktion ab (als Block), während die Knoten die Verkopplungen und die Struktur des Systems beschreiben (siehe Abbildung 3.8, [OHG04], [HOG04]).

Bei dieser allgemeinen Betrachtung fällt auf, dass in Blockschaltbildern die Struktur der Verknüpfungen und der Verhaltensbeschreibungen selbst getrennt dargestellt sind. Dies wird auch in typischen Entwurfswerkzeugen umgesetzt, wie Abbildung 3.7 zeigt. Es bietet sich daher an, die Trennung zwischen Struktur (Hierarchie) und Funktion (Block) für rekonfigurierbare Systeme zu nutzen. Wie in Abschnitt 3.2 beschrieben, ist eine Rekonfiguration die Änderung der Struktur bzw. Teilstruktur eines Systems. Bei einer Rekonfiguration ändert sich die Topologie des Systems, also die Beziehungen zwischen den Blöcken, die das mathematische Verhalten oder die Physik abbilden. Werden alle in einer bestimmten Konfiguration verwendbaren

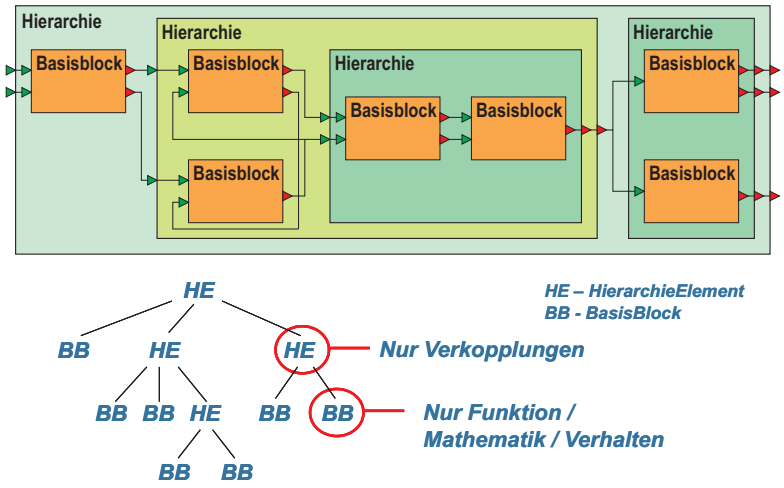

Abbildung 3.8: Hierarchische Blockschaltbilder

Funktionsblöcke als gegeben und definiert vorausgesetzt, so ändern sich bei einer Rekonfiguration nur hierarchische Elemente. Somit bedeutet Rekonfiguration letzten Endes die Veränderung des Topologiebaumes.

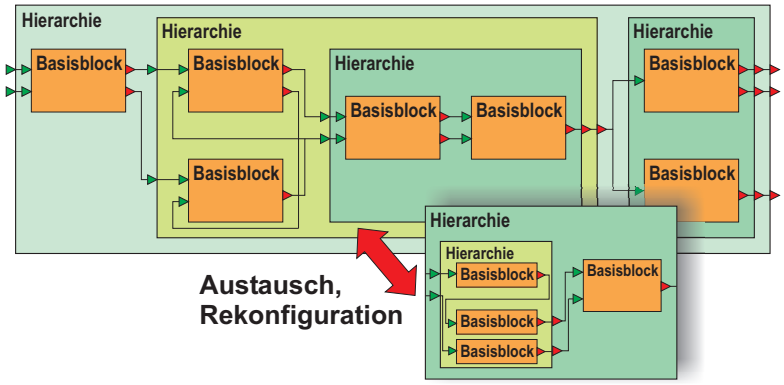

Abbildung 3.9: Rekonfiguration auf Topologieebene

Da in der hierarchischen Blockschaltbilddarstellung eine Hierarchieebene wiederum als Block dargestellt wird, kann eine Rekonfiguration auch durch Austausch von Hierarchieelementen in der Blocksicht erfolgen. Wird nur ein Teil des Systems verändert, entspricht dies dem Austausch nur eines Teils des Topologiebaums. Eine wichtige Randbedingung ist hierbei, dass die Schnittstellen der auszutauschenden Hierarchieelemente kompatibel sind. Abbildung 3.9 [OHG04] zeigt das beschriebene Prinzip des Austauschs eines Hierarchieelements. Das Element und alle unterlagerten Komponenten werden hierbei ausgetauscht. Die Änderung der Topologie bleibt lokal beschränkt.

Die Randbedingung für den Austausch von Hierarchieelementen, die besagt, dass alle umzuschaltenden Komponenten existieren und alle Schnittstellen kompatibel sein müssen, kann nun zur Definition eines neuen Hierarchieelements genutzt werden, das *verschiedene* Topologien beschreibt. Hierbei kann ein Hierarchieelement *verschiedene Konfigurationen in Form von Topologien* und inneren Verkopplungen enthalten, die zu unterschiedlichen Zeiten gültig sind. Die überlagerte Hierarchie

sieht die topologische Änderung der unterlagerten Hierarchie nicht, da diese nach außen nur durch ihre Schnittstellen beschrieben wird (Abbildung 3.10, [OHG04]).

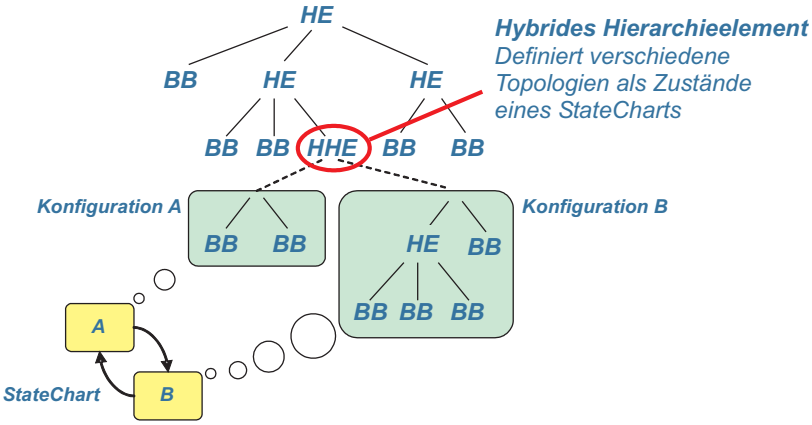

Abbildung 3.10: Hybride Hierarchieelemente

Die unterschiedlichen Topologien eines rekonfigurierbaren Hierarchieelements können als verschiedene Zustände dieses Elements interpretiert werden. Es bietet sich an, den Wechsel zwischen den verschiedenen Topologien durch eine Zustandsmaschine bzw. einen endlichen Automaten, zu kontrollieren. Die Beschreibung eines solchen Automaten kann als Zustandsübergangsdiagramm (StateChart [HP87]) erfolgen. Mit Hilfe von StateCharts ist es möglich, alle Konfigurationswechsel genau zu spezifizieren und quasi als Schnittstelle für die Rekonfiguration zu verwenden. Wird die Beschreibungsform StateChart mit den Hierarchieelementen von Blockschaltbildern kombiniert, führt dies zu speziellen hybriden Systemen, wobei die Einheit aus rekonfigurierbarem Hierarchieelement und StateChart als *hybrides Hierarchieelement* bezeichnet wird. Ein großer Vorteil dieses Ansatzes ist, dass klassische, nicht rekonfigurierbare Blockschaltbilder als Sonderfall enthalten sind, nämlich als hybrides Hierarchieelement mit nur *einem* Zustand. Somit können auch rekonfigurierbare Systeme nach bekannten Methoden entworfen werden.

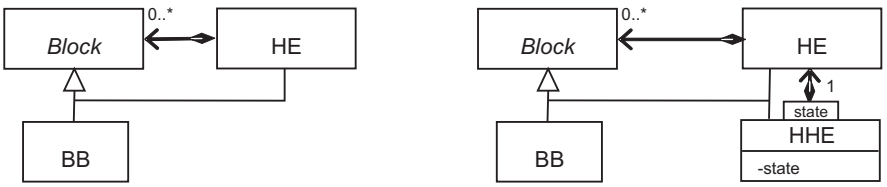

Abbildung 3.11: Struktur der Blockverschaltung ohne und mit Rekonfiguration

In Abbildung 3.11 [OHG04] sind die benötigten Elemente des Integrationsmodells mit und ohne Rekonfiguration und ihre Hierarchie mit Hilfe eines UML-Klassendiagramms [The07] beschrieben. Zu den Basisblöcken (BB) und den Hierarchieelementen (HE) im Modell ohne Rekonfiguration (Abbildung 3.11, links) kommen im integrierten Modell (Abbildung 3.11, rechts) noch hybride Hierarchieelemente (HHE) hinzu.

Basisblöcke können, wie oben bereits erwähnt, Verhalten als mathematische Funktionen oder in domänenspezifischer, physikalischer Beschreibung abbilden. Darüber hinaus können sie aber auch Schnittstellen zu Aktoren und Sensoren bilden, wie

beispielsweise beim Einsatz der Laufzeitumgebung IPANEMA [Hon98] (vgl. auch Abschnitt 5.3). Ein Hierarchieelement verbindet eine beliebige Anzahl von Blöcken der jeweiligen Typen und bildet somit die klassischen hierarchischen Blockschaltbilder ab. Die zusätzlich vorhandenen hybriden Hierarchieelemente erlauben es darüber hinaus, pro innerem, lokalem Zustand ein Hierarchieelement zu verwenden. Diese Zustände bilden so die Rekonfiguration der Blockschaltbilder entsprechend ab. Hat ein einfaches HHE nur untergeordnete Elemente der Typen HE und BE, entspricht dies den üblichen Ansätzen für die Modellierung hybrider Systeme mit Differentialgleichungen und Automaten (vgl. [HHWT95]). Durch die Möglichkeit, solche HHE-Blöcke mehrfach zu verwenden, geht der Ansatz der hybriden Hierarchieelemente jedoch wesentlich über die bekannten Ansätze hinaus [OHG04], [HOG04].

3.4 Operator-Controller-Modul (OCM)

Neben einem konkreten Ansatz für die Modellierung rekonfigurierbarer Systeme durch die im vorherigen Abschnitt beschriebenen hybriden Blockschaltbilder und den daraus resultierenden hybride Komponenten (vgl. Abschnitt 5.2.2) wird weiterhin ein Konzept benötigt, um die Informationsverarbeitung zu strukturieren.

3.4.1 Grundlagen zum OCM

Ausgehen sollen die Überlegungen zunächst vom Begriff des mechatronischen Funktionsmoduls (MFM) (vgl. Abschnitt 2.4.3), das als Funktionseinheit aus regelnder Informationsverarbeitung und technischer Komponente mit Aktorik und Sensorik die vitalen Teile eines mechatronischen Gesamtsystems enthält. Das mechatronische Funktionsmodul kombiniert im Ansatz schon für den Entwurf die verschiedenen Domänen der Mechatronik in einem Aggregat. Wird diese Komponente im Kontext von adaptiven Systemen mit diskreten Entscheidungsprozessen gesehen, gelangt man zunächst zu der Darstellung nach [Nau00].

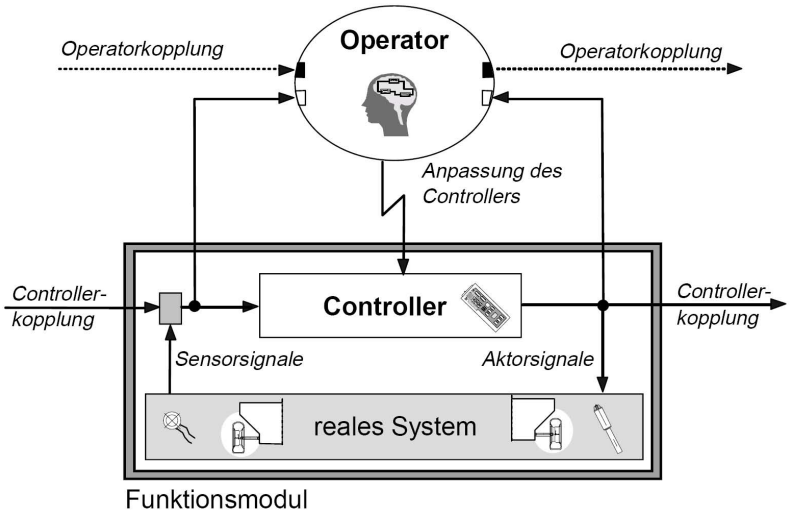

Abbildung 3.12: OCM nach Naumann

Das OCM nach Naumann [Nau00] (Abbildung 3.12) besteht aus einem Funktionsmodul, das einem MFM entspricht, und einem sogenannten Operator, der in

der Lage ist, den Regler (Controller) anzupassen. Diese Trennung hat verschiedene Gründe:

- Der Controller arbeitet quasi-kontinuierlich, der Operator diskret (Zustandsmaschine).
- Der Controller unterliegt durch die Kopplung mit dem realen System harten Echtzeitbedingungen, der Operator ggf. weichen Echtzeitbedingungen.
- Der Entwurf des Controllers basiert auf physikalisch-mathematisch hergeleiteten Differentialgleichungen oder Differenzengleichungen, der Entwurf des Operators i. A. auf üblichen Verfahren für Zustandsmaschinen wie beispielsweise StateCharts.

Der Vorteil für Entwurf und Betrieb liegt auf der Hand: Durch die Trennung können beide Hauptelemente des OCM mit den üblichen Verfahren der Regelungstechnik und der Informatik entworfen werden. Im Betrieb sind für beide Module separate Zeitschranken möglich. Für die Auswertung können jeweils Standardverfahren (vgl. Kapitel 4) benutzt werden. Die Kopplung kann nach bekannten Verfahren, etwa wie in der Laufzeitplattform IPANEMA (vgl. 5.3, [Hon98]) vorgesehen, erfolgen.

Das Konzept des OCM nach Naumann erlaubt eine kaskadierte Verkopplung auf Ebene der Operatoren und der Controller. Ein Einsatz zur Online-Optimierung ist möglich, sowohl als direkte als auch als indirekte Online-Optimierung [DO00], [DOM01].

Jedoch bringt das OCM nach Naumann auch Einschränkungen mit sich, die beim Entwurf von selbstoptimierenden Systemen schnell an die Grenzen des Konzeptes führen. So bietet das OCM prinzipiell keinen Raum für eine vorausschauende Optimierung eines im Operator eingebetteten kontinuierlichen Modells, da eine quasi-kontinuierliche Informationsverarbeitung hier nicht vorgesehen ist. Auch erlaubt das OCM nur eine Optimierung der Parameter des Controllers; eine strukturelle Veränderung, wie sie rekonfigurierbare Systeme benötigen, ist nicht vorgesehen. Für die Selbstoptimierung muss der Ansatz des OCM überdacht und erweitert werden.

3.4.2 Aufbau des erweiterten OCM

Im folgenden Abschnitt wird eine neue, erweiterte Struktur für ein OCM vorgestellt, die den Anforderungen an selbstoptimierende mechatronische Systeme mehr gerecht wird als der ursprüngliche Ansatz.

3.4.2.1 Funktionelle Anforderungen

Die Informationsverarbeitung eines selbstoptimierenden mechatronischen Systems muss eine Vielzahl von Funktionen erfüllen. Unter anderem sind dies:

- Regeln der Bewegung bzw. Dynamik der Strecke des technischen Systems
- Überwachung der Strecke auf auftretende Fehler
- Verkopplung des regelnden Systems mit anderen Reglern
- Anpassung der Regelung an veränderte Umgebungszustände durch Adaption der Reglerparameter

- Anpassung der Regelung an veränderte Umgebungszustände durch Wechsel der Reglerstruktur (Rekonfiguration)
- Optimierung der Reglerparameter durch vorausschauende Optimierung am Referenzmodell der geregelten Strecke, durch klassische Optimierungsverfahren und verhaltensbasierte Ansätze
- Wissensbasierte Optimierung durch Austausch von Optimierungsergebnissen anderer OCM mit gleichen Streckenverhältnissen
- Vernetzung der Überwachungs-, Kontroll- und Optimierungsfunktionen (Kaskadenregelung)

Die Komplexität des Gesamtsystems wird durch die Kombination der einzelnen Teilfunktionen, die unterschiedliche Anforderungen an die Informationsverarbeitung stellen, noch erhöht.

3.4.2.2 Echtzeitanforderungen

Eine generelle Anforderung an die Informationsverarbeitung von mechatronischen Systemen ist die Einhaltung von *Echtzeitbedingungen*. Dabei wird zwischen harter und weicher Echtzeit unterschieden. *Harte Echtzeitbedingungen* liegen vor, wenn Ergebnisse einer Berechnung zu einem bestimmten Zeitpunkt vorliegen *müssen*, um die Funktion sicherzustellen. Dies gilt für digitale Regler, die in festgelegten Zeitabständen die Stellwerte der Aktorik aktualisieren müssen. Werden diese Grenzen verletzt, kann es zu einer Destabilisierung des Systems kommen, was in einem realen technischen Prozess auf keinen Fall passieren darf (Beispiel: Regelung des Leitwerks bei Flugzeugen). *Weiche Echtzeitanforderungen* liegen dann vor, wenn Daten zu einem bestimmten Zeitpunkt vorliegen *sollten*, jedoch die Verzögerung nicht zu dramatischen Auswirkungen führt. Ein Beispiel ist hier die Übertragung von Überwachungsdaten an eine Leitstelle: Werden Messwerte nicht rechtzeitig übertragen, kann es zu einer Verzögerung der Anzeige kommen, oder im schlimmsten Fall fehlen Messwerte; eine Verletzung der Echtzeitschranken führt jedoch nicht zu einer unmittelbaren Gefährdung des technischen Prozesses (vgl. auch 5.3).

Allgemein lassen sich die Anforderungen für selbstoptimierende OCM wie folgt zusammenfassen:

- Die Regelung und alle an ihr direkt beteiligten Prozesse (Überwachungs- und Notfallroutinen; Überblend- und Umschaltmechanismen) unterliegen *harten Echtzeitbedingungen* [DOM01].
- Die Optimierung und alle an ihr beteiligten Prozesse unterliegen *weichen Echtzeitbedingungen*.

Das OCM nach Naumann führt diese Anforderungen bereits an die Grenzen des Konzepts, da hier beispielsweise Notfallroutinen, die, wenn sie auch sehr einfache, aber bereits diskrete Entscheidungsmechanismen erfordern, weder Controller noch Operator zuzuordnen sind. Ebenso steht es mit Steuerungen der Konfiguration (Rekonfiguration) oder Ablaufsteuerungen, z. B. bei Handhabungsgeräten. Auch diese unterliegen harten Echtzeitbedingungen, basieren aber nicht zwingend auf kontinuierlicher Informationsverarbeitung.

3.4.2.3 Erweiterte Struktur des OCM für selbstoptimierende mechatronische Systeme

Eine erweiterte Struktur muss das strenge Paradigma, dass diskrete Prozesse dem Operator und kontinuierliche dem Controller vorbehalten bleiben, verfeinern. Nach den in Abschnitt 3.4.2.2 gemachten Annahmen können die aus der Entwicklung der Laufzeitumgebung IPANEMA gewonnenen Erfahrungen in die neue Struktur einfließen (vgl. Abschnitt 5.3). Das dort entwickelte 3-Ebenen-Modell aus Calculator/Adaptor (harte Echtzeit), Assistant (harte/weiche Echtzeit) und Moderator (weiche Echtzeit) lässt sich im Ansatz für die Neudefinition des OCM nutzen. Während beim OCM nach Naumann die Trennung zwischen weicher und harter Echtzeit zwischen Operator und Calculator erfolgt, erfolgt bei IPANEMA diese Trennung innerhalb des Assistant-Prozesses (Überwachung). Dies hat den Vorteil, dass sich die Pufferung von Daten und die Weitergabe von Befehlen auf einer Zwischenebene außerhalb des kritischen, regelnden Prozesses kontrollieren lassen.

Der erweiterte Ansatz des OCM (Abbildung 3.13) greift die Anforderungen an selbstoptimierende Systeme auf. Sie nimmt die in [NR98] und [Nau00] vorgestellten Ansätze auf und verbindet diese mit Ergebnissen aus [HBN01], [OHKK02] und [OHG04]. Im erweiterten Ansatz besteht das OCM aus drei Hauptelementen:

Der Kognitive Operator umfasst alle Funktionen, die der Optimierung des Systems dienen und allgemein *nicht* harten Echtzeitbedingungen unterliegen.

Der Reflektorische Operator umfasst vor allem diskrete Funktionen, die

- der Umsetzung von Optimierungsergebnissen dienen (Parameterwechsel, Reglerumschaltung etc.),
- Notfallfunktionen ausführen,
- weitere diskrete Abläufe steuern,
- Datenkopplung zwischen Kognitivem Operator und Controller vornehmen.

Der Controller enthält die regelnde Informationsverarbeitung und alle Funktionen, die einen *direkten Zugriff* auf Aktorik und Sensorik benötigen. Alle Zugriffe auf den technischen Prozess bzw. das technische System erfolgen *nur* auf dieser Ebene.

Die strikte Trennung in diskrete und kontinuierliche Verarbeitung wird aufgehoben. Jedoch gilt weiterhin, dass der Controller *keine weitreichende* diskrete Verarbeitung enthält, wie beispielsweise Zustandsmaschinen, sondern nur für Rekonfiguration bzw. Notfallfunktionen notwendige diskrete Teilfunktionen. Die Aufteilung erfolgt nach den Anforderungen der Echtzeitverarbeitung, die sich wiederum aus der Art des Durchgriffs auf das technische System ergeben.

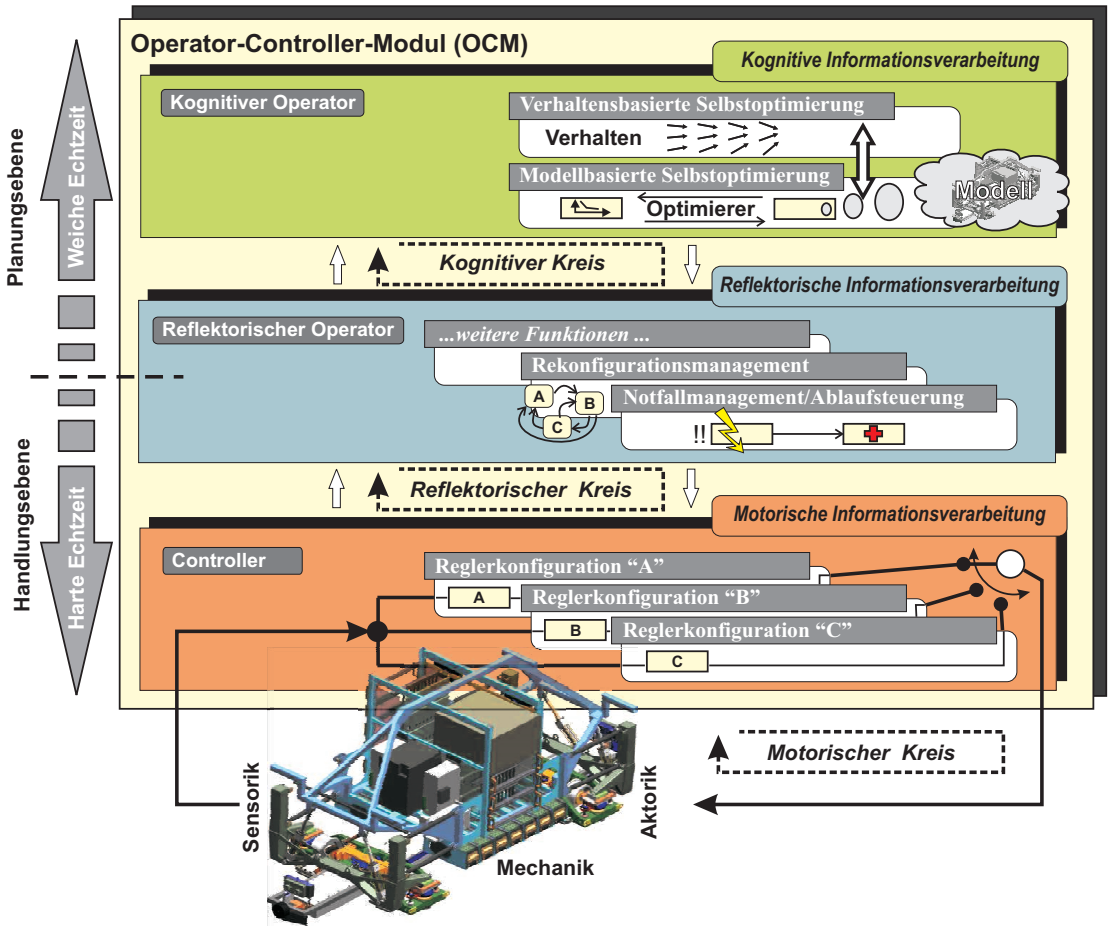

Abbildung 3.13: Schematischer Aufbau des erweiterten Operator-Controller-Moduls

Der Controller liegt auf der untersten Ebene des OCM. Der Informationsfluss bildet zusammen mit dem technischen System einen *inneren Kreis*. In direktem Durchgriff werden Messsignale der Sensoren ausgewertet, verarbeitet und neue Stellsignale an die Aktorik ausgegeben. Dieser innere Kreis wird deshalb auch als *motorischer Kreis* bezeichnet.

Die Informationsverarbeitung erfolgt quasi-kontinuierlich, bis auf die oben genannten Beschränkungen für die Verarbeitung von speziellen, notwendigen, diskreten Ereignissen. Für eine Reglerumschaltung kann der Controller bereits vorkonfigurierte Regler enthalten, zwischen denen umgeschaltet werden kann. Die Umschaltung erfolgt zeitdiskret zwischen zwei kontinuierlichen Zeitschritten. Überblendungsvorgänge sind eigene, besondere Reglerprozesse.

Der Reflektorische Operator ist ein in diesen Ansatz neu eingeführte Komponente. In ihm sind Überwachungs- und Steuerungskomponenten zusammengefasst. Der Kontrollfluss bildet einen weiteren Kreis zusammen mit dem Controller, der Messwerte, aber auch allgemein Informationen über den Zustand des regelnden Systems an den Operator weitergibt und wiederum Steuersignale, neue Parametersätze u. ä. erhält.

Der Name des Reflektorischen Operators leitet sich vom *reflexartigen* Handeln ab. Dabei soll zum Ausdruck kommen, dass auf dieser Ebene ohne lange Verarbeitungsketten Informationen direkt, quasi reflexartig verarbeitet werden.

Dazu gehören z. B. so genannte *Watchdogs*[20], die eine einfache Prozessüberwachung erlauben, aber auch kontinuierliche Adaptionsalgorithmen zur Regleranpassung. Die Komponenten, die direkt mit dem Controller verbunden sind, erfordern harte Echtzeitbedingungen für die Informationsverarbeitung.

Der Reflektorische Operator dient darüber hinaus als ein Verbindungselement zur kognitiven Informationsverarbeitung. Dieses Verbindungselement stellt auch die Grenze zwischen harter und weicher Echtzeitverarbeitung dar. Das Interface dient als Puffer und stellt den echtzeitfähigen Betrieb der unterlagerten Ebenen sicher. Weiterhin ist es denkbar, hier eine Überprüfung neuer Vorgabewerte in Abhängigkeit vom Systemzustand durchzuführen. Beispielsweise dürfen im Notfallbetrieb keine neuen, optimierten Reglerparameter gesetzt werden.

Der Kognitive Operator bildet die oberste Ebene der Informationsverarbeitung innerhalb des OCM. Auf dieser Ebene befinden sich alle höherwertigen Funktionen der Selbstoptimierung. Die Methoden können dabei vielfältig sein. Wesentliche Ansätze wurden in Abschnitt 2.2 behandelt. Dies sind vor allem Methoden, die ohne genau vorhersehbare Zeitschranke arbeiten, wie z. B. iterative Optimierungsverfahren. Diese Verfahren basieren häufig auf der Verwendung von Modellen des unterlagerten Systems zur Optimierung, wie bei der modellbasierten Optimierung und bei der modellgestützten verhaltensbasierten Optimierung. Darüber hinaus können auch wissensbasierte Systeme verwendet werden, insbesondere in Kombination mit Pareto-Optimierung zur Selektion eines konkreten Punktes auf der Paretofront. Weitere Verfahren sind Lernverfahren. Ein Beispiel hierfür ist das sogenannte selbstverstärkende Lernen, das vom Formalismus her mit der nichtlinearen Optimierung verwandt ist (vgl. Abschnitt 2.2).

Die optimierende Informationsverarbeitung des Kognitiven Operators kann grob in modellbasierte und verhaltensbasierte Selbstoptimierung eingeteilt werden. Die *modellbasierte Optimierung* ermöglicht eine vorausschauende Optimierung unabhängig vom Zustand des technischen Prozesses bzw. Systems. Die *verhaltensbasierte Optimierung* enthält Funktionen zur Planung und zur Bewertung der aktuellen Zielvorgaben (siehe [OHKK02], [HO03], sowie [OHG04] und [HOG04]).

Der Kognitive Operator benötigt keine engen Zeitschranken wie der Reflektorische Operator und der Controller. Durch die Kopplung an den Reflektorischen Operator, der in der Kopplung unter weichen Echtzeitbedingungen arbeitet, unterliegen auch Teile des Kognitiven Operators weichen Echtzeitbedingungen. Hierbei ist die Grenze durch die Verwertbarkeit der Optimierungsergebnisse festgelegt – hat sich der Zustand des unterlagerten technischen Systems und der Umgebung so verändert, dass die Optimierungsergebnisse nicht mehr aktuell sind, ist die Zeitschranke überschritten. Eine genaue Zeitschranke, als Verfallsdatum für den Optimierungsprozess, kann daher nicht immer vorherbestimmt werden.

[20] Der Begriff Watchdog (englisch: Wachhund) wird für eine Komponente eines Rechnersystems verwendet, welche die Funktion anderer Komponenten überwacht. Wird dabei eine Fehlfunktion erkannt, so wird diese entweder signalisiert, oder es werden geeignete Anweisungen zur Problembehandlung ausgeführt [OHG04].

Eine Kopplung mit anderen Prozessen ist auf der Ebene des Kognitiven Operators möglich. Der Import von Optimierungsergebnissen anderer OCM ist hier nur ein Beispiel. Ein wesentlicher Punkt ist das Empfangen und Setzen neuer Optimierungsziele und die Kommunikation von Optimierungsergebnissen und Systemzuständen an andere OCM.

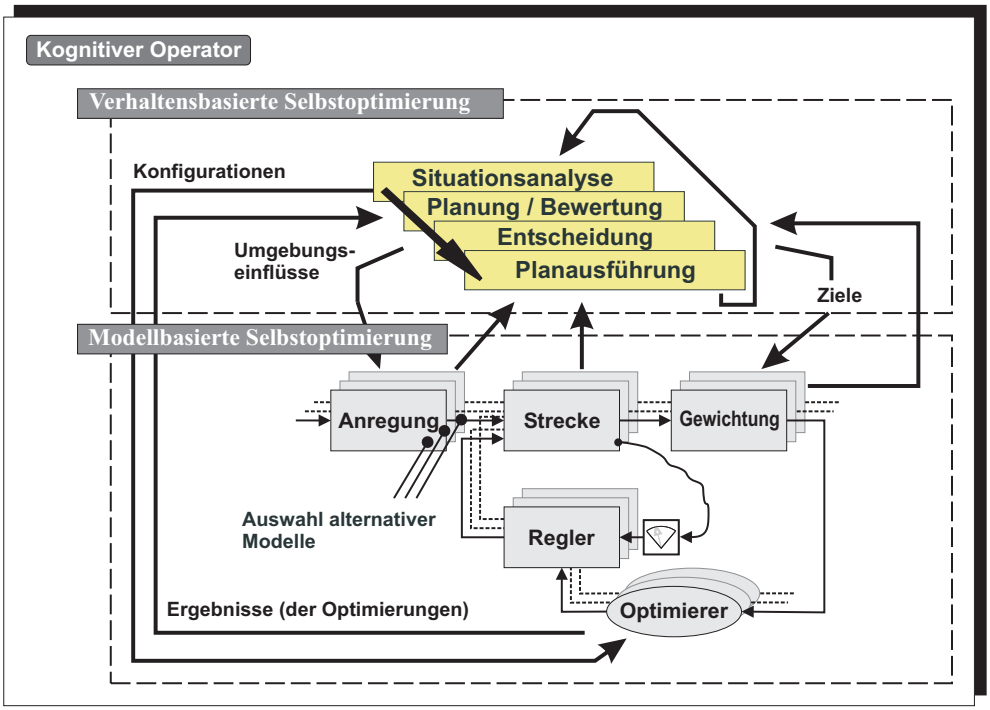

Abbildung 3.14: Beispiel für den inneren Aufbau eines Kognitiven Operators

Ein Beispiel für den Aufbau eines Kognitiven Operators ist in Abbildung 3.14 zu sehen [OHKK02]. Der Operator ist in zwei Bereiche aufgeteilt: die *Verhaltensbasierte Optimierung* (oben) und die *Modellbasierte Optimierung* (unten). Der Aufbau der verhaltensbasierten Optimierung entspricht in wesentlichen Zügen dem eines *Deliberativen Agenten* [JW98]. Dies deutet zunächst nicht auf einen verhaltensbasierten Ansatz hin. Da jedoch das *Verhalten* des simulierten Systems und der Modellbasierten Optimierung von der Planungsebene direkt beeinflusst werden, kann im Allgemeinen von Verhaltensbasierung gesprochen werden.

Planung und Ausführung der Optimierung werden durch die Module/Phasen (1) Situationsanalyse, (2) Planung/Bewertung, (3) Entscheidung und (4) Planausführung durchgeführt. In der Situationsanalyse (1) wird der Zustand des Systems und ggf. der Umgebung erfasst. Nach der Planung/Bewertung (2), bei welcher der Zustand bewertet wurde, wird aufgrund der Faktenlage die Entscheidung (3) für *eine* konkrete Handlung getroffen. In der Phase der Planausführung (4) wird die Handlung ausgeführt; die getroffenen Entscheidungen werden z. B. als neue Ziele und Zielgrößen der Modellbasierten Optimierung mitgeteilt. Auch die Auswahl einer geeigneten Anregung kann darunter fallen.

Die Modellbasierte Optimierung führt daraufhin eine Simulation und eine Optimierung durch. Ist eine verbesserte Konfiguration des geregelten Systems

gefunden, werden die Ergebnisse der Modellbasierten Optimierung an den Reflektorischen Operator weitergeleitet, der für die Umsetzung der neuen Konfiguration zuständig ist. Der Prozess der Optimierung kann indessen durch den Kognitiven Operator fortgesetzt werden.

Die gesamte Struktur orientiert sich nunmehr nicht allein an der Trennung von diskreter und (quasi-)kontinuierlicher Informationsverarbeitung, sondern an derjenigen zwischen weicher und harter Echtzeit und dem Durchgriff auf das technische System.

Die drei Ebenen kommunizieren hierarchisch in drei *Kreisen*, dadurch werden die Ebenen voneinander entkoppelt. Der Kognitive Operator hat keinen direkten Durchgriff auf den Controller, der Reflektorische Operator keinen auf den technischen Prozess. Auf jeder Ebene ist jedoch eine Kopplung mit anderen OCM möglich: Für den Controller sind dies kaskadierte Regelungen, für den Reflektorischen Operator ist dies z. B. Notfallroutinen[21], für den Kognitiven Operator sind dies der Austausch von Systemzuständen sowie Optimierungsergebnissen und -zielen mit anderen OCM.

Die genaue Aufteilung der Funktionen auf die drei Elemente Controller, Reflektorischer und Kognitiver Operator hängt stark von der Aufgabenstellung ab. Die genannten Beispielfunktionen dienen nur als Anhaltspunkt.

3.4.3 Rekonfiguration mit Hilfe des OCM

Wie in Abschnitt 3.3 beschrieben wurde, lassen sich klassische Blockschaltbilder systematisch zu rekonfigurierbaren Blockschaltbildern erweitern. Für eine Realisierung auf dieser Basis muss das Konzept auf OCM-Architekturen übertragen werden.

Jede Konfiguration eines Blockschaltbildes besteht aus einer eigenen Hierarchie und einem definierten Wirk- und Informationsfluss, der aus quasi-kontinuierlichen und diskreten Datenkanälen bestehen kann. Die ausgetauschten Daten beinhalten Informationen wie Stellgrößen auf Ebene des Controllers oder Ereignisse und andere diskrete Signale auf Ebene des Reflektorischen und des Kognitiven Operators. Neben den gerichteten Verbindungen existieren auch bidirektionale Verbindungen zwischen den Ein- und Ausgängen der Blöcke.

Wird ein komplexes mechatronisches System betrachtet, so besteht dieses aus vielen (mechatronisch) aktiven Komponenten (z. B. MFM), die das OCM als Strukturansatz für die Informationsverarbeitung nutzen. Wird dieses Bild auf rekonfigurierbare Systeme erweitert, so besteht die Informationsverarbeitung eines komplexen Systems aus ggf. hierarchisch verkoppelten OCM, die eine rekonfigurierbare Baumstruktur abbilden.

Nach [OHG04] ist die Zuordnung in die verschiedenen Komponenten des OCM nur dann eine *wohlstrukturierte* Architektur, wenn gilt:

- Alle Basisblöcke und hierarchischen Blöcke, die direkt über Signale im Wirkfluss auf einen Aktor des OCM oder untergeordneter OCM Einfluss haben, und nur diese, müssen dem Controller des OCM zugeordnet sein.

- Für die Trennung in Reflektorischen Operator und Kognitiven Operator muss des Weiteren gelten, dass der Kognitive Operator nur *ereignisgesteuert* auf den Reflektorischen Operator und *gar nicht* auf den Controller einwirken kann. Der

[21] Man denke an kontrollierte Notabschaltung.

Reflektorische Operator muss jederzeit auch ohne agierenden Kognitiven Operator handlungsfähig bleiben. Dies bedeutet, dass der Reflektorische Operator soweit autonom handlungsfähig sein muss, dass sein Verhalten ausreichend für den sicheren Betrieb des OCM ist (siehe [GTB+03]).

- Für die Verbindungen zwischen den Komponenten der einzelnen OCM muss darüber hinaus gelten, dass Controller nur mit Controllern des *übergeordneten* und der *untergeordneten* OCM verbunden sind und Reflektorische Operatoren nur mit den Reflektorische Operatoren des *übergeordneten* und der *untergeordneten* reaktiven Operatoren. Für die Kognitiven Operatoren ist eine ähnliche Struktur sinnvoll, aber nicht zwingend erforderlich [OHG04], [HOG04].

Mit der beschriebenen Beschränkung auf sogenannte *wohlstrukturierte* Architekturen wird sichergestellt, dass die in Abschnitt 3.4.2 beschriebene Trennung zwischen den einzelnen Ebenen der OCM (Controller, reflektorischer und kognitiver Operator) auch im Falle hierarchischer Verkopplung erhalten bleibt und auch im Falle einer Rekonfiguration kein unkontrollierter Zugriff auf den Controller durch den kognitiven Operator möglich ist.

Darüber hinaus gilt, bedingt durch den sicherheitskritischen Charakter mechatronischer Systeme, dass nur Blockschaltbilder erlaubt sind, die ein vorhersehbar sicheres Verhalten zeigen. Folgende Eigenschaften gelten zusätzlich zur Forderung der Wohlstrukturiertheit:

1. Um ein undefiniertes Verhalten zu vermeiden, gilt eine Konfiguration nur dann als *strukturell korrekt*, wenn keine unverbundenen Ein- oder Ausgänge vorhanden sind.

   Wenn alle möglichen Konfigurationen der hybriden Hierarchieelemente strukturell korrekt sind, so gelten die Konfigurationen als *statisch korrekte Rekonfiguration.*

   Sind jedoch nicht alle (möglichen) Konfigurationen korrekt, sondern nur die, welche durch die Interaktion des Systems erreichbar sind, wird von *dynamisch korrekter Rekonfiguration* gesprochen.

   Um einen sicheren Betrieb zu gewährleisten, muss ein System mindestens einer dynamisch korrekten Rekonfiguration entsprechen. Im Falle einer statisch korrekten Rekonfiguration sind alle Konfigurationen sicher.

2. Speziell für den Controller gilt ergänzend, dass die Konfiguration nicht nur mindestens einer dynamisch korrekten Rekonfiguration entsprechen muss, sondern dass darüber hinaus auch alle erreichbaren Zustände (z. B. verschiedene Reglertypen und Verkopplungen, auch mit Sensorik/Aktorik) die notwendige *regelungstechnische Stabilität* aufweisen müssen.

3. Sicherheitskritische Elemente, wie bestimmte Notfallverhalten, müssen in jeder Konfiguration erreichbar sein. Dies gilt vor allem für den Controller und den reflektorischen Operator.

Kapitel 4

Numerische Simulation und Ausführung von modularen Systemen

Wir verstehen die Zahl, aber nie das Gezählte (Blaise Pascal)

In diesem Kapitel werden Verfahren vorgestellt, die für die korrekte Simulation und Ausführung von verkoppelten Teilsystemen wesentlich sind. Solche verkoppelten Teilsysteme kommen insbesondere in selbstoptimierenden mechatronischen Regelungssystemen vor. Im Folgenden werden Probleme und Lösungsansätze für die numerische Simulation und Ausführung untersucht und beschrieben, die in s.o. Systemen genutzt werden können.

4.1 Mathematische Modelle

Mathematische Modelle dynamischer Systeme und kontinuierliche Regelungssysteme werden üblicherweise durch zeitkontinuierliche Gleichungen dargestellt, wie DAE (Differential-Algebraic Equations) und ODE (Ordinary Differential Equations). Da für das numerische Lösen von DAEs im Allgemeinen iterative Verfahren benötigt werden, haben sich ODEs als Beschreibungsform für echtzeitfähige Systeme durchgesetzt. Bei der Betrachtung von s. o. Systemen, die auf rekonfigurierbaren Systemen und hybriden Komponenten basieren, ergeben sich besondere Anforderungen an die Ausführung im Rechner. Dabei spielt es zunächst keine Rolle, ob (Teil-)systeme simuliert werden oder nur die Informationsverarbeitung des realisierten Systems betrachtet wird – die mathematischen Grundlagen sind die gleichen. In jedem Fall geht es um die Lösung von gewöhnlichen Differentialgleichungen, die für die Echtzeitverarbeitung die größte Bedeutung haben und auf denen sowohl die Systemmodelle als auch die Regelungstechnik basieren (vgl. 2.3.5).

4.2 Numerische Simulation mechatronischer Systeme

Die Simulation technischer bzw. mechatronischer Systeme im Digitalrechner besteht im Wesentlichen aus der numerischen Integration der systembeschreibenden

Differentialgleichungen. Dazu dienen leistungsfähige Verfahren, wie z. B. Runge-Kutta-Methoden (Einschrittverfahren) oder die Methoden nach Adams-Bashforth (Mehrschrittverfahren) [Stu96]. Dabei werden die Systemgleichungen zu diskreten Zeitpunkten ausgewertet. Die Schrittweite zwischen zwei solchen Zeitpunkten wird hauptsächlich von der Geschwindigkeit, also den Zeitkonstanten der Teilsysteme, bestimmt. Allgemein kann man sagen, dass die Schrittweite umso kleiner gewählt werden muss, je schneller das System ist, damit der numerische Fehler, der bei der Berechnung gemacht wird, nicht zu groß wird.

Wie bereits im Kapitel 3 diskutiert wurde, setzen sich s. o. Systeme aus mechatronischen Teilsystemen zusammen, die ggf. in ihrem Wirkzusammenhang variieren können (Rekonfiguration). Rekonfiguration ist besonders leicht in der Informationsverarbeitung zu realisieren. Dabei werden neue Regler umgeschaltet oder Teilmodelle zur Optimierung ausgetauscht oder variiert.

Solche Systeme setzen sich häufig aus Teilsystemen mit stark unterschiedlichen Zeitkonstanten zusammen. Dies ist besonders dann der Fall, wenn die Teilsysteme unterschiedlichen Disziplinen (Domänen) entstammen, wie beispielsweise schnelle elektronische und damit verglichen langsame mechanische Teilsysteme. Dies gilt auch für die hierarchische Verkopplung von Reglern (vgl. verallgemeinerte Kaskadenregelung [LHLH01, Hes06]) auf unterschiedlichen Ebenen, bei denen verschiedene Zeitkonstanten gelten.

Wird ein einziges Integrationsverfahren zur Simulation bzw. Berechnung des Gesamt(regelungs)systems verwendet, muss die Schrittweite klein genug für eine hinreichende Genauigkeit bei der Berechnung des schnellsten Teilsystems gewählt werden. Entsprechend werden aber auch alle anderen Differentialgleichungen im System mit dieser Schrittweite integriert, auch wenn diese keine so kleine Schrittweite erfordern. Dieses Vorgehen ist nicht effizient [KR99].

Um die Effizienz der Auswertung zu verbessern, kann die modular-hierarchische Struktur eines Gesamtsystems ausgenutzt werden. Die Gruppierungen und Kopplungen, die sich aus einer an der Realität orientierenden Modellierung des technischen Systems ergeben, führen automatisch zu Teilsystemen mit unterschiedlichen Zeitkonstanten. Durch eine getrennte Berechnung der Teilmodelle mit Schrittweiten, die an die jeweiligen Zeitkonstanten angepasst sind, kann die Effizienz gesteigert werden. So wird verhindert, dass langsame Teilsysteme, also Systeme mit großen Zeitkonstanten, mit viel kleinerer Schrittweite und dadurch größerem Rechenaufwand integriert werden, als für die geforderte Genauigkeit notwendig ist.

Für die Praxis ist also der Einsatz von Integrationsverfahren geboten, welche die Anforderungen an die Zeitkonstanten der Teilsysteme berücksichtigen. Solche Verfahren werden auch als *Multirate-Verfahren* bezeichnet [Vöc03], [OV04].

Die Regelung mechatronischer Systeme erfolgt heute praktisch ausschließlich durch digitale Regler. In komplexen Systemen wie Kraftfahrzeugen werden die Regler der einzelnen Teilsysteme auf Steuergeräten ausgeführt. Dabei sind die Steuergeräte oft hierarchisch verkoppelt, d. h. ein Steuergerät bekommt Vorgabedaten von einem anderen Steuergerät höherer Ordnung. Dies ist beispielsweise beim ESP (Electronic Stability Program) der Fall, bei dem das Steuergerät für das ESP mit dem Steuergerät für ABS (Antiblockiersystem) verkoppelt ist. In neueren ESP-Varianten werden weitere Funktionen wie ASR (Antischlupfregelung) integriert und innerhalb eines Steuergerätes realisiert [Rob07].

Die lokalen digitalen Regler berechnen die neuen Ausgabewerte in eigenen zeitlichen Abständen. Diese Schrittweiten können sich denjenigen der anderen Regler

unterscheiden. Bei unterschiedlichen Schrittweiten kann es zu besonderen Effekten kommen, die großen Einfluss auf das Ergebnis der Regelung haben [Vöc03], [OV04].

S. o. Systeme unterliegen durch ihre Komplexität und den Einsatz von simulierten Teilsystemen ebenfalls diesen Effekten. Daher ist es sinnvoll, diese Effekte zu modellieren und näher zu untersuchen. Für die Untersuchung bietet sich eine Modellsimulation an. Die auftretenden Effekte lassen sich stellvertretend an Multirate-Systemen untersuchen, um Rückschlüsse auf das Verhalten von komplexen Reglersystemen zu gewinnen. So lassen sich sowohl die unterschiedlichen Geschwindigkeiten der digitalen Komponenten als auch die verteilte modulare und hierarchische Struktur realer Systeme abbilden. Der Prinzipaufbau eines einfachen Multirate-Systems ist in Abbildung 4.1 dargestellt.

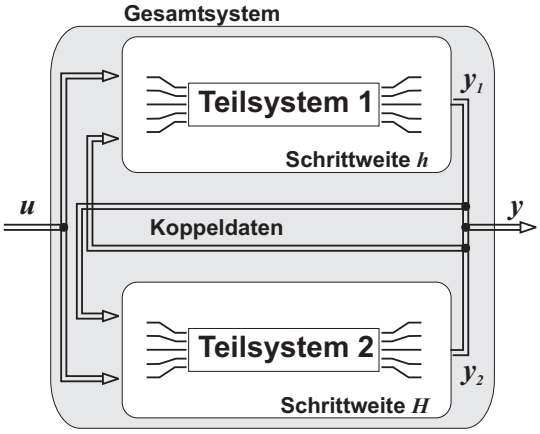

Abbildung 4.1: Multirate-System mit zwei Teilsystemen

Das Gesamtsystem besteht aus zwei unterschiedlich schnellen Teilsystemen. Jedes Teilsystem wird einzeln mit einer Schrittweite ausgewertet, die der geforderten Genauigkeit angepasst ist. Für sich betrachtet, ergibt sich für die Berechnung der einzelnen Teilsysteme dadurch kein Problem, wenn die jeweiligen Eingangsgrößen für jeden Berechnungsschritt zur Verfügung stehen. Jedoch werden häufig die Ausgangsgrößen des einen Systems zur Berechnung des anderen Systems benötigt (rekursive Verkopplung). Bei Systemen mit unterschiedlicher Schrittweite müssen diese Kopplungen besonders behandelt werden.

In den folgenden Abschnitten sollen die Effekte, die zu Problemen führen können, näher untersucht werden. Darüber hinaus werden auch Ansätze vorgestellt, die eine Kompensation ermöglichen. Für s.o. Systeme sind dies Grundlagen, die bei jeder Realisierung in Betracht gezogen werden müssen. Im Falle der Rekonfiguration ist darüber hinaus die veränderte Verkopplung von Teilsystemen mit unterschiedlichen Zeitkonstanten zu beachten.

4.2.1 Genauigkeit und Stabilität numerischer Berechnungsverfahren

Die Auswertung der mathematischen Gleichungen, die das System beschreiben, erfolgt zeitdiskret. Dadurch entsteht ein numerischer Fehler, dessen Größe, Qualität und Verwertbarkeit der Simulation bestimmt. Mit der Genauigkeit der Berechnung steigt allerdings auch der Aufwand, im Allgemeinen der Rechenaufwand. Einen weiteren Einfluss hat das gewählte Integrationsverfahren. Ziel ist es, den Fehler bei

akzeptablem Aufwand hinreichend klein zu halten. Eine Möglichkeit Aussagen über die Qualität eines numerischen Verfahrens zu gewinnen, ist des Anwendung des sogenannten Standard- oder Dahlquist-Test-Problems [But87], [HW91].

$$\begin{aligned} y' &= \lambda y \,;\ \lambda \text{ komplex} \\ y(0) &= 1 \end{aligned} \tag{4.2.1}$$

Die analytische Lösung ist bekannt:

$$y(x) = e^{\lambda x} \tag{4.2.2}$$

Diese Lösung konvergiert genau dann gegen 0, wenn $Re(\lambda) < 0$ gilt. Die numerischen Verfahren müssen mindestens diese Eigenschaft wiedergeben. Die häufig verwendeten expliziten Runge-Kutta-Verfahren der Stufe s mit Schrittweite h, Zwischenstufen g_i und den Koeffizienten a_{ij} und b_j liefern bei Anwendung auf das Test-Problem 4.2.1:

$$\begin{aligned} g_i &= y_m + h\lambda \sum_{j=1}^{i-1} a_{ij} g_j \\ y_{m+1} &= y_m + h\lambda \sum_{j=1}^{s} b_j g_j \end{aligned} \tag{4.2.3}$$

mit $z = h\lambda$ und nach mehrfachem Einsetzen von g_i ergibt sich:

$$y_{m+1} = R(z) y_m \tag{4.2.4}$$

wobei

$$R(z) = 1 + z \sum_j b_j + z^2 \sum_{j,k} b_j a_{jk} + z^3 \sum_{j,k,l} b_j a_{jk} a_{kl} + ... \tag{4.2.5}$$

Aufgrund von Gleichung 4.2.4 muss $|R(z)| < 0$ gelten, damit die Eigenschaft des Testproblems, für ein λ mit negativem Realteil gegen 0 zu konvergieren, erhalten bleibt. Der Bereich

$$S = \{z \in C; |R(z)| < 0\} \tag{4.2.6}$$

wird als *Stabilitätsbereich des Verfahrens* bezeichnet. Das numerische Integrationsverfahren konvergiert gegen 0, wenn h so gewählt wird, dass $z = h\lambda$ innerhalb dieses Bereiches liegt. Beispielsweise ergibt die Anwendung eines Runge-Kutta-Verfahrens auf ein System linearer Differentialgleichungen:

$$y' = Ay \tag{4.2.7}$$

mit der Stabilitätsfunktion $R(z)$ aus Gleichung 4.2.5:

$$y_{m+1} = R(hA) y_m \tag{4.2.8}$$

Für den Erhalt der numerischen Stabilität muss h bei $Re(\lambda_i) < 0$ somit so gewählt werden, dass $h\lambda_i \in S$ ist. Die λ_i stellen die Eigenwerte der Systemmatrix A

dar. Die Eigenwerte des Systems bestimmen die zu wählende Integrationsschrittweite. Für explizite Runge-Kutta-Verfahren mit der Ordnung p, die ihrer Stufenzahl s entspricht, lautet die Stabilitätsfunktion nach [HW91]:

$$R(z) = 1 + z + \frac{z^2}{2} + ... + \frac{z^s}{s!} \tag{4.2.9}$$

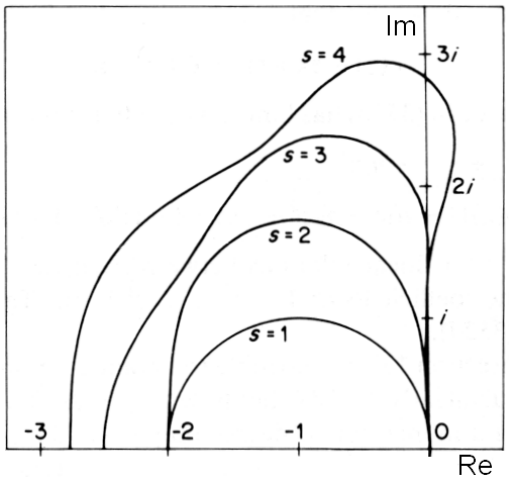

Abbildung 4.2: Stabilitätsbereiche expliziter Runge-Kutta-Verfahren

In Abbildung 4.2 [But87] sind die Stabilitätsbereiche S für explizite Runge-Kutta-Verfahren bis zur Stufe $s = 4$ zu sehen, die sich durch diese Stabilitätsfunktion ergeben.

4.2.2 Voraussetzungen für unterschiedliche Schrittweiten

Insgesamt lässt sich anhand der oben genannten Stabilitätsbedingung $h\lambda_i \in S$ aussagen, dass bei betragsmäßig großen Eigenwerten (schnelle Systeme) die Schrittweite klein gewählt werden muss, um numerische Stabilität zu gewährleisten. Für kleine Eigenwerte (langsame Systeme) darf die Schrittweite größer sein.

Wenn Systeme schnelle und langsame Anteile besitzen, also sowohl kleine als auch große Eigenwerte aufweisen, ist der größte Eigenwert für die Wahl der Schrittweite h bestimmend. Sie muss so klein gewählt werden, dass für alle Eigenwerte die Stabilitätsbedingung erfüllt wird. In der Praxis wird die Schrittweite bei nichtsteifen Systemen oft nicht durch die Stabilitätsbedingung beschränkt, sondern durch Genauigkeitsforderungen, die eine Schrittweite vorgeben, mit der die Stabilitätsbedingung ohnehin erfüllt ist [RSW96]. Grundidee der Multirate-Verfahren ist nun, ein Gesamtsystem in Teilsysteme aufzuteilen, um diese mit unterschiedlichen, ihren Eigenwerten entsprechenden Schrittweiten berechnen zu können. Dadurch kann Genauigkeit oder auch Rechenleistung gewonnen werden.

4.3 Verkopplung von Teilsystemen

Besondere Aufmerksamkeit bei der Anwendung von Multirate-Verfahren ist bei der Verkopplung von Teilsystemen gegeben. Bei einer solchen Verkopplung, wie in Abbildung 4.1 dargestellt, können Teilsysteme mit unterschiedlichen Geschwindigkeiten

verkoppelt werden. Für die Berechnung eines schnellen Teilsystems sind beispielsweise berechnete Werte eines langsameren Teilsystems erforderlich. Für einen Zeitschritt des schnellen Systems stehen die für den aktuellen Zeitschritt benötigten Daten des langsamen Systems möglicherweise aktuell nicht zur Verfügung, da zu diesem Zeitpunkt das entsprechende Teilsystem nicht ausgewertet wurde. Zur Verfügung stehen somit nur Daten von einem früheren Zeitpunkt, wodurch ein numerischer Fehler entsteht.

Im umgekehrten Fall, wenn ein langsames System Daten eines schnelleren benötigt, treten ebenfalls numerische Effekte auf. Zwischen dem Schritt des langsamen Systems berechnet das schnelle mehrere Schritte. Während eines Zwischenschrittes ignoriert und nur der jeweils letzte Wert vom langsamen System verwendet, entsteht ein weiterer numerischer Fehler. Die Untersuchung dieser Effekte und die Entwicklung von Maßnahmen zur Kompensation stehen im Mittelpunkt der folgenden Abschnitte.

4.3.1 Numerische Fehler durch Aliasing

In diesem Abschnitt wird der Effekt beschrieben, der auftreten kann, wenn ein langsames System Koppeldaten von einem schnellen System erhält. Nach dem Abtasttheorem von Shannon können die Eigenschaften eines Systems verloren gehen, wenn mit zu geringer Frequenz abgetastet (digitalisiert) wird. Dies spielt insbesondere bei der Analog-/Digitalwandlung eine große Rolle, wobei die Theorie besagt, dass mindestens mit der doppelten Frequenz des Systems abgetastet werden muss. Liegt die Abtastrate zu niedrig, kann beispielsweise ein Signal mit viel niedrigerer Frequenz entstehen. Dieser Effekt wird als *Aliasing* [LJ96] bezeichnet und ist kein multiratespezifisches Problem, sondern kann überall dort vorkommen, wo Signale abgetastet oder nur zu diskreten Zeitpunkten ausgewertet werden (siehe auch: Abtasttheorem von Shannon[22] [Ste88]).

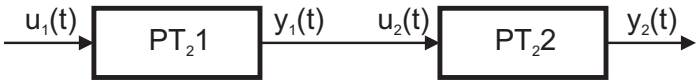

Abbildung 4.3: Testsystem aus zwei PT_2-Gliedern

Deutlich wird das am Beispiel zweier in Serie geschalteter PT_2-Systeme, wie in Abbildung 4.3 dargestellt. Das erste hat eine hohe und das zweite eine niedrige Eigenfrequenz. Damit die Pole des langsamen zweiten Systems im Stabilitätsbereich des Integrationsverfahrens liegen, reicht für dieses System eine größere Schrittweite H als für das schnelle erste System aus. Das hochfrequente Ausgangssignal y_1 dieses Systems wird bei der Simulation und der daraus resultierenden Diskretisierung vom zweiten System zu den Zeiten $t_0 + nH$ ausgewertet. Diese Abtastfrequenz reicht jedoch nicht aus, um die Eigenschaften von y_1 hinreichend genau wiederzugeben. Es entsteht ein völlig verändertes Eingangssignal für das zweite System.

[22] Claude Elwood Shannon (* 30. April 1916 in Petoskey, Michigan; † 24. Februar 2001)

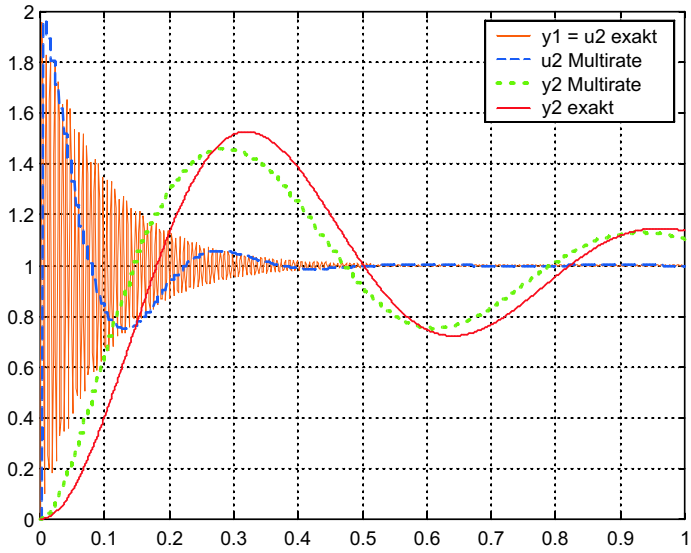

Abbildung 4.4: Aliasing-Effekt bei Multirate-Integration

Abbildung 4.4 [OV04] zeigt das Simulationsergebnis y_2 exakt (gepunktete Linie) bei Sprunganregung ohne Multirate-Verfahren, also mit gleicher hinreichend kleiner Integrationsschrittweite für beide Systeme, und mit unterschiedlichen Schrittweiten für das langsame und das schnelle System (durchgezogene Linie). Deutlich zu erkennen sind die unterschiedlichen Verläufe des Eingangssignals in das zweite PT_2-System sowie die daraus resultierenden Unterschiede im Ausgang aus diesem System.

Für die Modellierung der Effekte muss die Diskretisierung modelliert werden. Dazu dient der sogenannte δ-Abtaster. Er erzeugt aus einem kontinuierlichen Signal eine Impulsfolge. Um den abgetasteten Wert über die Schrittweite konstant zu halten, wird ein Halteglied benötigt. Die Impulsfolge des δ-Abtasters wird dadurch zu einer Treppenfunktion. Der δ-Abtaster bildet zusammen mit dem Halteglied das sogenannte Abtast-Halteglied.

Im oben beschriebenen Beispiel hat das erste PT_2-System eine wesentlich kleinere Zeitkonstante als das zweite Teilsystem. Zur Untersuchung der daraus resultierenden Koppeleffekte kann es als quasi-kontinuierlich angenommen werden.

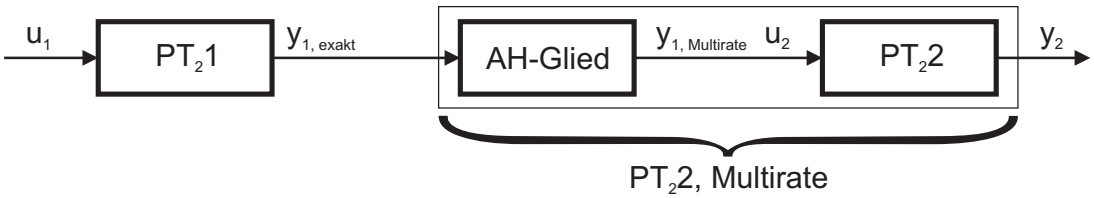

Abbildung 4.5: Ersatzmodell der Serienschaltung zweier PT_2-Glieder bei Kopplung vom langsamen zum schnellen System

Der Ausgang des ersten Systems wird bei der Berechnung des zweiten Systems aufgrund der Multirate-Integration aber nur zu diskreten Zeitpunkten berücksichtigt. Dies wird durch ein zusätzlich eingefügtes Abtast-Halteglied berücksichtigt, wie in Abbildung 4.5 [OV04] zu sehen ist. Die größere Schrittweite, mit der das zweite System ausgewertet wird, geht dabei als Totzeit T in das Halteglied ein.

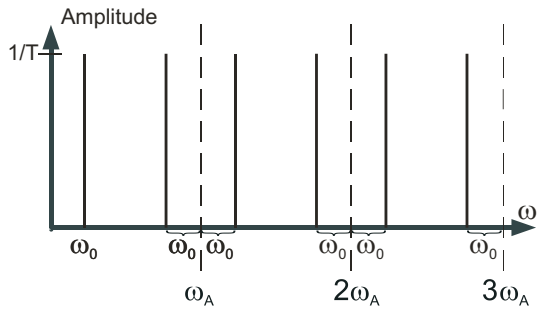

Abbildung 4.6: Spektrum des δ-Abtasters

Abbildung 4.6 [LJ96] zeigt das Spektrum eines δ-Abtasters bei einer Sinusanregung mit der Frequenz ω_0. Es weist unendlich viele Peaks der Höhe $1/T$ bei den Frequenzen $n\omega_A \pm \omega_0$ auf.

Der Amplitudengang des Haltegliedes ergibt, multipliziert mit dem Spektrum des Abtasters, das Gesamtspektrum des Abtast-Halteglieds, wie in Abbildung 4.7 zu sehen ist.

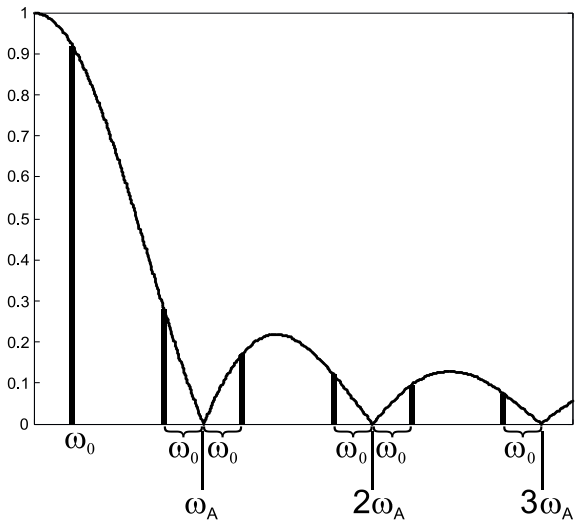

Abbildung 4.7: Spektrum des Abtast-Halteglieds

Das Spektrum zeigt deutlich die Oberschwingungen bei $n\omega_A \pm \omega_0$. Mit kleiner werdendem ω_A und konstantem ω_0 wird die Amplitude der ersten Oberschwingung im Verhältnis zu derjenigen der Grundschwingung immer größer. Bei $\omega_A = 2\omega_0$ haben die Grundschwingung und die erste Oberschwingung dieselbe Frequenz. Wenn die Abtastfrequenz ω_A noch weiter reduziert wird, tritt die erste Oberschwingung bei einer niedrigeren Frequenz auf als die Grundschwingung. Außerdem hat sie dann eine größere Amplitude als die Grundschwingung. Das Eingangssignal wird also völlig verändert wiedergegeben. Nach dem Abtasttheorem von Shannon [Ste88] sollte diese Grenze nicht überschritten werden:

> *„Um fähig zu sein, $f(t)$ exakt wiederzugewinnen, muss man $f(t)$ mit einer Rate abtasten, die größer als das Doppelte seiner höchsten Frequenzkomponente ist.“* [Ste88]

In der Praxis wird eine doppelte Abtastrate nicht ausreichen, um das Signal hinreichend genau wiederzugeben. Lediglich die Frequenz kann ab dieser Schwelle korrekt wiedergegeben werden (siehe z. B. [LJ96]).

Der Effekt kann ebenfalls mit Hilfe der z-Transformation gezeigt werden. Ein schwingungsfähiges PT_2-Glied hat im Laplace-Bereich einen Nenner der Form:

$$\begin{aligned} N(s) &= s^2 + 2\zeta\omega_0 s + \omega_0^2 \\ &= s^2 + 2\zeta\omega_0 s + \zeta^2\omega_0^2 - \zeta^2\omega_0^2 + \omega_0^2 \\ &= \left(s + \underbrace{\zeta\omega_0}_{a} \right)^2 + \underbrace{\omega_0^2 \left(1 - \zeta^2\right)}_{b^2} \end{aligned} \tag{4.3.1}$$

Anhand der Polstellen (für $\zeta < 1$) $s_{1,2} = -\zeta\omega_0 \pm j\omega_0\sqrt{1-\zeta^2}$ lassen sich charakteristische Werte des Systems ablesen. Wesentlich sind im Allgemeinen die Eigenfrequenzen des gedämpften und des ungedämpften Systems sowie die Dämpfung [Föl94]. Die Polstellen der z-transformierten Übertragungsfunktion geben, ähnlich der laplace-transformierten Übertragungsfunktion, über wichtige Eigenschaften Auskunft. Die Rücktransformation in den Zeitbereich erfolgt durch Partialbruchzerlegung und Anwendung von Korrespondenztabellen. Ein komplexes Polpaar einer z-Übertragungsfunktion ergibt einen Partialbruch mit einem Nenner der Form:

$$(z - \alpha + j\beta)(z - \alpha - j\beta) = z^2 - 2\alpha z + \alpha^2 + \beta^2 \tag{4.3.2}$$

Aus der Korrespondenztabelle [Fei90] ergibt sich:

$$N(z) = z^2 - 2ze^{-aT}cos(bT) + e^{-2aT} \tag{4.3.3}$$

Durch Koeffizientenvergleich ergeben sich aus Gleichung 4.3.2 und 4.3.3:

$$e^{-2aT} = \alpha^2 + \beta^2 \Rightarrow e^{-aT} = \sqrt{\alpha^2\beta^2} \tag{4.3.4}$$

und

$$\alpha = e^{-aT}cos(bT) \tag{4.3.5}$$

Damit folgt nach kurzer Rechnung:

$$\beta = e^{-aT}sin(bT) \tag{4.3.6}$$

Mit Gleichung 4.3.4 in 4.3.5 ergibt sich:

$$cos(bt) = \frac{\alpha}{\sqrt{\alpha^2 + \beta^2}} \tag{4.3.7}$$

$$b = \frac{1}{T} arccos\left(\frac{\alpha}{\sqrt{\alpha^2 + \beta^2}} \right) \tag{4.3.8}$$

Die z-Übertragungsfunktion des Nenners (Gleichung 4.3.3) korrespondiert aber mit der Laplace-Übertragungsfunktion des Nenners (Gleichung 4.3.1), der einem PT_2-System gehört wie im oben dargestellten Beispiel. Die zugehörige Zeitfunktion lässt sich ermitteln mit:

$$e^{-at}cos\,(bt)\,\text{bzw.}e^{-at}sin\,(bt) \tag{4.3.9}$$

Die in den Korrespondenztabellen enthaltenen kontinuierlichen Zeitbereichsfunktionen stimmen zu den Abtastzeitpunkten nT mit den Werten der Punktfolge überein. Jedoch wird über den Verlauf zwischen zwei Abtastzeitpunkten bei der Rücktransformation keine Aussage getroffen. Somit ist die Rücktransformation in den Zeitbereich nicht eindeutig. Wird jedoch die Abtastzeit hinreichend klein gewählt, gibt das rücktransformierte Signal im Zeitbereich die wichtigsten Eigenschaften wieder.

Aus den Pollagen der z-transformierten Übertragungsfunktion lässt sich die Kreisfrequenz b des Antwortsignals bestimmen. Der Aliasing-Effekt lässt sich dadurch erklären, dass *cos* eine periodische Funktion ist. Deshalb ist sie nicht über den ganzen Bereich der reellen Zahlen injektiv, sondern nur intervallweise [BSMM97], [Vöc03].

Für den Übergang von 4.3.7 nach 4.3.8 muss folglich gelten: $bT < \pi$. Wird die Abtastzeit T durch die Abtastkreisfrequenz ω_A beschrieben als $T = 2\pi/\omega_A$, ergibt sich die Bedingung zu:

$$b\frac{2\pi}{\omega_A} < \pi \Leftrightarrow 2b < \omega_A \tag{4.3.10}$$

Die Kreisfrequenz b stellt nach Gleichung 4.3.8 die Eigenfrequenz der Schwingung des Antwortsignals dar, was das Shannon-Theorem wiedererkennen lässt. Wird die Bedingung 4.3.10 nicht eingehalten, entsteht durch die Abtastung eine zu einer niedrigeren Frequenz passende Punktfolge. Durch Einsetzen von 4.3.4 und 4.3.5 in Gleichung 4.3.8 ergibt sich:

$$b = \frac{1}{T}arccos(cos(bT)) \tag{4.3.11}$$

Obwohl diese Gleichung auch nach 4.3.10 richtig ist, ergibt sich wegen der Mehrdeutigkeit von *cos* eine niedrigere tatsächliche Kreisfrequenz ω_T für das rücktransformierte Signal:

$$\omega_T = \frac{1}{T}arccos(cos(bT)) \tag{4.3.12}$$

Die Abbildung 4.8 zeigt den Verlauf dieser tatsächlichen Kreisfrequenz für eine zunehmende Abtastschrittweite T bei konstanter Kreisfrequenz $b = 100rad/s$ des abgetasteten Signals. Zunächst stimmen ω_T und b überein. Wird jedoch Bedingung 4.3.10 verletzt, nimmt mit zunehmendem T die Frequenz ω_T ab. Im weiteren Verlauf steigt und fällt die Funktion wieder, was an der Periodizität des *cos* liegt. Die Obergrenze ergibt sich durch π/T, was durch die gestrichelte Linie dargestellt wird. Für $bT > \pi$ ist diese stets kleiner als b.

Die Darstellung in Abbildung 4.8 macht deutlich, dass sich bei zu groß gewählter Abtastzeit T bzw. zu großer Schrittweite ein verändertes Antwortsignal ergibt. Dies hat stets eine kleinere Frequenz als die Ausgangsfrequenz.

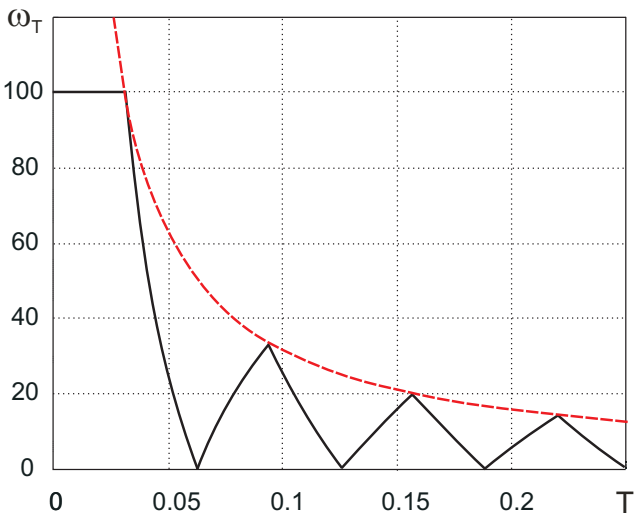

Abbildung 4.8: Kreisfrequenzen der rücktransformierten Funktion

Aus den dargestellten Überlegungen ergibt sich nun die Frage, wie bei einem Multirate-System, bei dem Bedingung 4.3.10 verletzt wird, trotzdem eine numerisch stabile und hinreichend genaue Berechnung möglich gemacht werden kann. Offensichtlich ist, dass Mechanismen benötigt werden, um die Effekte zu kompensieren, die solche – streng genommen mit zu niedriger Frequenz abgetasteten und ausgewerteten – Signale mit sich bringen.

4.3.2 Erzeugung von Störfrequenzen durch Kopplung von Teilsystemen

Nachdem im vorherigen Abschnitt 4.3.1 die Effekte untersucht wurden, welche die Kopplung eines schnellen Systems mit einem langsamen mit sich bringen, soll im Folgenden die Kopplung eines langsamen mit einem schnellen Teilsystem untersucht werden. Als Beispiel dient wieder die Reihenschaltung zweier PT_2-Systeme wie im Abschnitt 4.3.1, jedoch besitzt diesmal das erste System die größere Zeitkonstante und das nachfolgende die kleinere. Entsprechend ist die Integrationsschrittweite für das erste größer als für das zweite gewählt worden. Für sich betrachtet sind beide Teilsysteme numerisch stabil.

Der zu untersuchende Effekt ergibt sich bei diesem Multirate-System durch die Verkopplung des Ausgangs des ersten, langsamen Systems mit dem Eingang des schnellen: Während der Auswertung des schnellen Systems kann das Eingangssignal nicht zu jedem Zeitschritt aktualisiert werden. Als Eingang wird der letzte bekannte Wert benutzt. Das Ausgangssignal $y_1 = u_2$ wird für die Dauer eines Integrationsschrittes des langsamen Systems konstant gehalten. Nach diesem großen Schritt (*Makroschritt*), der mehrere Auswertungen des zweiten Systems (*Mikroschritte*) umfasst, wird ein neuer Wert für y_1 berechnet. Dieser Wert wird dann für die Dauer des nächsten Makroschrittes zur Berechnung des zweiten Systems genutzt. Aus Sicht des schnellen Systems kommt es bei einem Makroschritt zu einer plötzlichen sprunghaften Aktualisierung des Eingangssignals, was zu einer zusätzlichen Anregung des schnellen Systems führen kann. Dabei kann die Frequenz der Anregung viel höher sein als die Frequenz des Ausgangssignals des langsameren Systems.

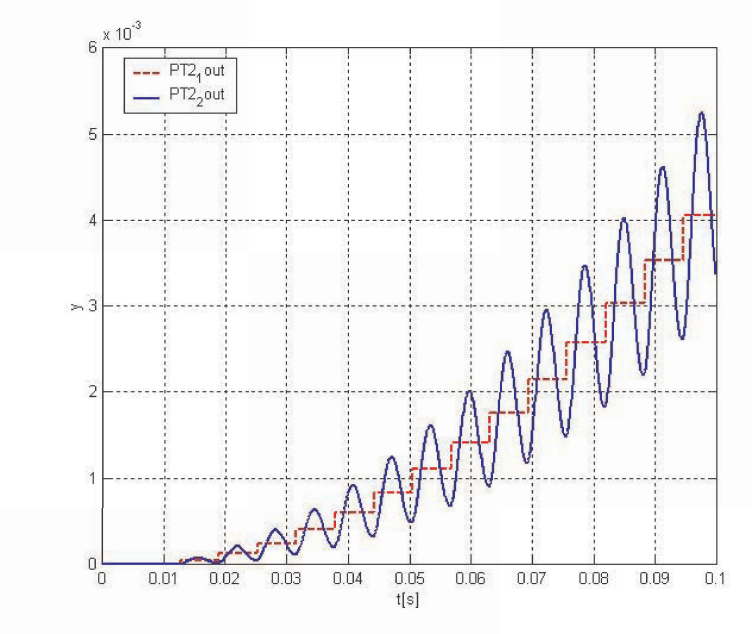

Abbildung 4.9: Zusätzliche hohe Frequenzanteile durch Multirate-Integration

Abbildung 4.9 [Vöc03] zeigt das Simulationsergebnis für einen Eingangssprung. Integrationsschrittweite und Zeitkonstanten wurden dabei *ungünstig* gewählt, so dass der Effekt besonders zu Tage tritt. Die Anregung erfolgt im Bereich der Eigenfrequenz des schnellen, schlecht gedämpften Systems, da das erste System mit eben dieser Frequenz ausgewertet wird. Dadurch entstehen Sprünge im Ausgangssignal (gestrichelte Linie).

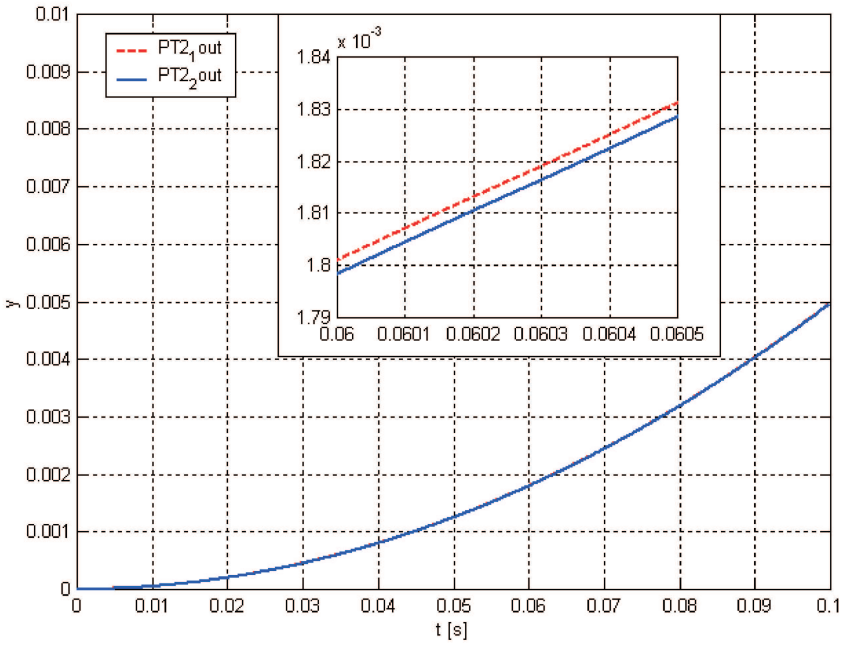

Abbildung 4.10: Integration ohne Multirate

Erfolgt eine Auswertung ohne Multirate mit der Schrittweite des schnellen Teilsystems für das Gesamtsystem, liegen Eingangs- (gestrichelt) und Ausgangssignal (durchgezogene Linie) nahezu direkt übereinander. Wie im vergrößerten Bereich in

Abbildung 4.10 [Vöc03] deutlich zu erkennen ist, werden keine zusätzlichen Schwingungen aufgebaut. Der Effekt kann dadurch erklärt werden, dass, wie schon im Abschnitt 4.3.1, ein zwischengeschaltetes Abtast-Halte-Glied angenommen wird, das den Eingang des zweiten PT_2-Systems für die Dauer eines Integrationsschrittes des ersten konstant hält. Da die Integrationsschrittweite des zweiten Systems als viel kleiner gegenüber dem ersten angenommen wird, kann dieses Teilsystem zur Vereinfachung als kontinuierliches System modelliert werden. Die Diskretisierung dieses Systems wird also vernachlässigt. Das Modell ist als Blockschaltbild in Abbildung 4.11 dargestellt.

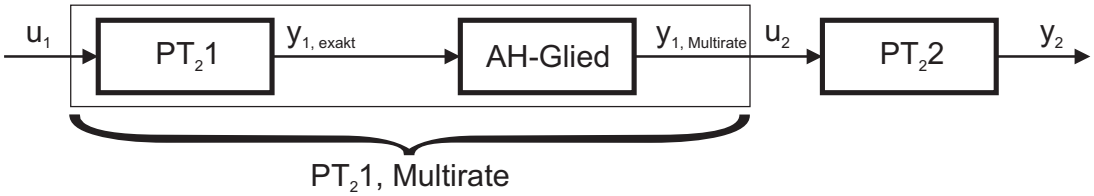

Abbildung 4.11: Ersatzmodell für Multirate-Integration bei Kopplung vom schnellen zum langsamen System

Wie in Abbildung 4.7 aus Abschnitt 4.3.1 zu sehen ist, weist es Oberschwingungen bei $n\omega_A \pm \omega_0$ auf, wobei ω_A die Abtastfrequenz darstellt und ω_0 die Anregungsfrequenz. Somit gibt ω_A für das hier beschriebene Modell die Frequenz der Berechnung des ersten Teilsystems an. Ist die Auswertefrequenz deutlich höher als die Frequenz des Systems, kann ω_0 für die Untersuchung der Oberschwingungen vernachlässigt werden, deren Frequenzen im Bereich von ganzzahligen Vielfachen von ω_A liegen. Wenn die Eigenfrequenz des folgenden Systems in diesem Bereich liegt, wirkt sich der Effekt besonders stark aus. Außerhalb können jedoch immer noch unerwünschte Oberschwingungen auftreten.

Da das schnellere Teilsystem als zeitkontinuierlich angenommen wurde, lässt sich die modifizierte z-Transformation [Fei90] auf dieses Problem anwenden. Sie erlaubt unter den hier vorliegenden Bedingungen die Berechnung der Signalverläufe zwischen den Abtastzeitpunkten.

Durch das Konstanthalten des Ausgangssignals des ersten Systems entsteht eine Treppenfunktion, die für die zusätzliche Anregung verantwortlich ist. Der Verlauf des ursprünglichen Signals spielt dabei keine Rolle. Deshalb genügt es, das Verhalten des zweiten PT_2-Systems zu untersuchen. Das Eingangssignal wird auf ein einfaches Rampensignal $u(t)$ reduziert. Durch ein nachgeschaltetes Abtast-Halteglied erhält das Signal einen treppenförmigen Verlauf $u_{Treppe}(t)$, der bei der Integration eines Systems mit geringerer Schrittweite entstehen würde. Die Konfiguration ist in Abbildung 4.12 zu sehen.

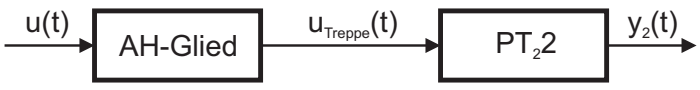

Abbildung 4.12: Modell zur Anwendung der modifizierten z-Transformation

Mit Hilfe der modifizierten z-Transformation soll nun das Ausgangssignal des angeregten zweiten PT_2-Systems nach [Fei90] untersucht werden. Im ersten Schritt

wird die z-Transformierte des Eingangssignals $u(t) = t$ mit Hilfe einer Korrespondenztabelle ermittelt:

$$U(z) = \frac{Tz}{(z-1)^2} \tag{4.3.13}$$

Die laplace-transformierte Übertragungsfunktion des PT_2-Gliedes lautet:

$$G_{PT_2}(s) = \frac{K}{T_0^2 s^2 + 2\zeta T_0 s + 1} = \frac{K\omega_0^2}{s^2 + 2\zeta\omega_0 s + \omega_0^2} \tag{4.3.14}$$

mit $\omega_0 = 1/T_0$ als Eigenfrequenz des ungedämpften Systems und ζ als Lehrsches Dämpfungsmaß. Nach Umformung in eine entsprechende Form ergibt sich mit $a = \zeta\omega_0$ und $\omega = \omega_0\sqrt{1-\zeta^2}$:

$$G_{PT_2}(s) = \frac{K\omega_0}{(s+a)^2 + \omega^2} \tag{4.3.15}$$

Die Übertragungsfunktion des Haltegliedes:

$$F_H(s) = \frac{1 - e^{-Ts}}{s} \tag{4.3.16}$$

lässt sich in die Faktoren $1-e^{-Ts}$ und $1/s$ aufteilen. Mit $G_{PT_2}(s)$ wird der Faktor $1/s$ zusammengefasst zu:

$$G(s) = \frac{K\omega_0}{s((s+a^2) + \omega^2)} \tag{4.3.17}$$

Die sich ergebende Übertragungsfunktion wird in Partialbrüche aufgeteilt:

$$G(s) = A\frac{1}{s} + B\frac{(s+a)}{(s+a)^2 + \omega^2} + C\frac{\omega}{(s+a)^2 + \omega^2} \tag{4.3.18}$$

Aus den Summanden lässt sich mit Hilfe einer Korrespondenztabelle die z-Übertragungsfunktion bestimmen:

$$\begin{aligned} G(z,\gamma) = {} & A_1\frac{z}{z-1} \\ & + B_1\frac{(zcos(\omega\gamma T) - e^{-aT}cos((1-\gamma)\omega T))ze^{-a\gamma T}}{z^2 - 2ze^{-at}cos(\omega T) + e^{-2aT}} \\ & + C_1\frac{(zsin(\omega\gamma T) + e^{-aT}sin((1-\gamma)\omega T))ze^{-a\gamma T}}{z^2 - 2ze^{-at}cos(\omega T) + e^{-2aT}} \end{aligned} \tag{4.3.19}$$

T ist die Abtastperiode. $\gamma \in [0,1]$ wird zur Berechnung der Werte zwischen den Abtastzeitpunkten nT verwendet. Die z-Transformierte von $1 - e^{-Ts}$ aus dem Halteglied

$$1 - z^{-1} = \frac{z-1}{z} \tag{4.3.20}$$

wird zusammen mit der Übertragungsfunktion des Eingangssignals $U(z)$ zur Berechnung des Ausgangssignals genutzt:

$$Y_2(z,\gamma) = U(z)\frac{z-1}{z}G(z,\gamma) \tag{4.3.21}$$

Nach Einsetzen und Rücktransformation mit Hilfe der Korrespondenztabelle ergibt sich der zeitliche Verlauf des Ausgangssignals zu:

$$\begin{aligned} y_2(t) = & A_2 \\ & + B_2 kT \\ & + C_2 e^{-a(kT+\gamma T)} cos(\omega(kT+\gamma T)) \\ & + D_2 e^{-a(kT+\gamma T)} sin(\omega(kT+\gamma T)) \end{aligned} \tag{4.3.22}$$

mit den Koeffizienten A_2, B_2, C_2 und D_2, die von ω_0, ζ, γ und T abhängen[23]. Das PT_2-System wird mit der Abtastperiode $T = 2\pi/\omega_0$ bei niedriger Dämpfung ($\zeta = 0.05$) genau mit seiner Eigenfrequenz zusätzlich angeregt. Das Ergebnis der Berechnung ist in Abbildung 4.13 zu sehen. Die Auswirkungen des treppenförmigen Eingangssignals auf das nachfolgende System stimmen mit der Simulation überein.

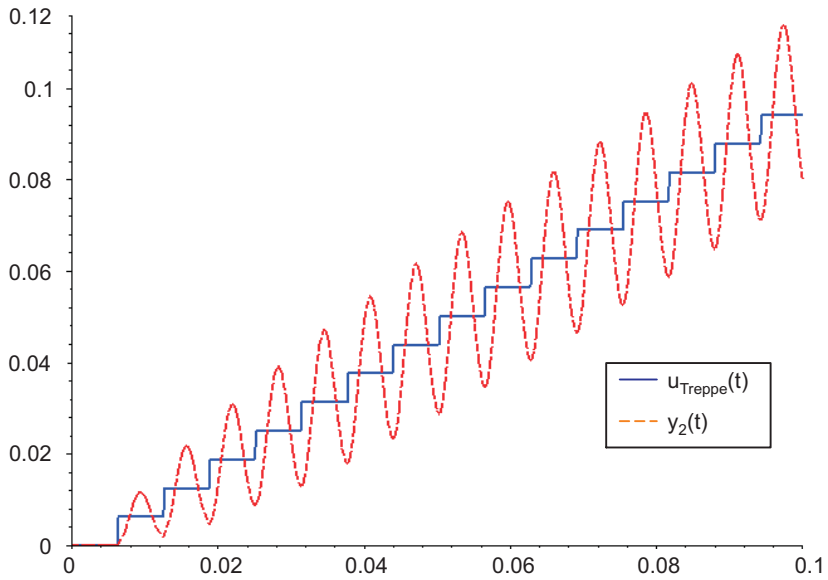

Abbildung 4.13: Antwortsignal mittels modifizierter z-Transformation

Wird die Abtastzeit anders gewählt, z. B. $T = (2/3) \cdot 2\pi/\omega_0$, was nicht zu einer Anregung im Bereich der Eigenfrequenz führt, fallen die Auswirkungen geringer aus. Zusammenfassend lässt sich sagen, dass die angestellten Überlegungen mit der Herleitung der Koppeleffekte aus dem Spektrum des Abtast-Haltegliedes übereinstimmen. Besonders kritisch sind die Auswirkungen, wenn die gewählte Abtastperiode zu Anregungen im Bereich der Eigenfrequenz oder ganzzahligen Vielfachen des angeregten Systems liegt. Die Tastperiode entspricht im Falle der Multirate-Integration der Schrittweite, mit der das erste, langsame System ausgewertet wird. Eine Veränderung der Abtastperiode kann den Effekt zwar verringern, jedoch kann nicht ohne Betrachtung der einzelnen Teilsysteme garantiert werden, dass diese Reduzierung hinreichend ist. Für die Kompensation der störenden Einflüsse dieser Oberschwingungen sind weitere Maßnahmen nötig. Für automatisch verkoppelte Systeme im

[23] Auf eine ausführliche Darstellung wurde aus Gründen der Übersichtlichkeit verzichtet, da es sich um längere Terme handelt, die für das hier angestrebte Verständnis nicht von Bedeutung

Falle von Rekonfiguration werden Methoden benötigt, die eine numerische Stabilität sicherstellen oder zumindest eine definierte Aussage über ein späteres Verhalten erlauben.

4.3.3 Multirate-Systeme und Multirate-Integration

Nach der Untersuchung und der Darstellung von Effekten, die bei der Kopplung von Systemen mit unterschiedlichen Schrittweiten auftreten können, wurden zunächst einzelne Effekte mit Hilfe von eingefügten Abtast-Halteglieder untersucht. Um ein Gesamtsystem zu untersuchen, müssen an allen Stellen, an denen Signale zwischen unterschiedlichen Schrittweiten übergeben werden, Abtast-Halteglieder eingefügt werden. Unter Vernachlässigung des numerischen Fehlers des Integrationsverfahrens kann der Einfluss der Auswertung unterschiedlicher Schrittweiten untersucht werden. Dazu werden die Teilsysteme als kontinuierliche Systeme betrachtet[24]. Mit Hilfe zweier Abtast-Halteglieder kann dann die Diskretisierung jedes Teilsystems modelliert werden.

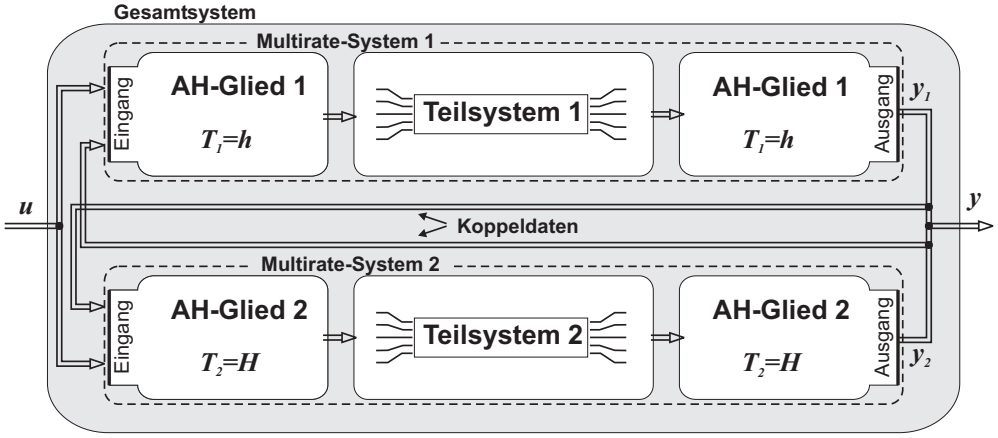

Abbildung 4.14: Modell der Diskretisierung

In der Abbildung 4.14 ist diese Konfiguration zu sehen. Sie kann als Erweiterung der Konfiguration aus Abbildung 4.1 betrachtet werden. In der Erweiterung besitzt jedes Teilsystem sowohl am Eingang als auch am Ausgang ein Abtast-Halteglied. Die zeitdiskrete Auswertung des Teilsystems wird durch das AH-Glied am Eingang dargestellt (vgl. 4.3.1). Die Effekte, die sich aus der Diskretisierung des Ausgangs ergeben, werden durch das nachgeschaltete AH-Glied modelliert (vgl. 4.3.2).

4.3.4 Grenzen des Modellierungsansatzes

Die bisherigen Betrachtungen konzentrierten sich auf die Untersuchung von Koppeleffekten zwischen Teilsystemen mit unterschiedlicher Schrittweite. Die Erweiterung durch Abtast-Halteglieder erlaubt mit Hilfe der z-Transformation Vorhersagen über die Veränderung des Systemverhaltens bei der Simulation.

[24]Die betrachteten Teilsysteme sind Modelle kontinuierlicher Systeme, die zur Abbildung im Rechner diskretisiert werden müssen (quasi-kontinuierlich). Wird diese Diskretisierung vernachlässigt, gelangt man wieder zu kontinuierlichen Systemen.

Jedoch stößt dieser Ansatz schnell an Grenzen, da zum einen selbst einfache Systeme schnell so komplex werden, dass sie mit Hilfe von analytischen Beschreibungsformen, wie der z-Transformation nach [Fei90], zu keiner handhabbaren Darstellung führen; zum anderen ist der Ansatz durch die z-Transformation selbst begrenzt. Diese ist für unterschiedliche Schrittweiten nicht vorgesehen. Alle Signale müssen zu festen Zeitpunkten $t = n \cdot T$ vorliegen, damit die z-Transformation überhaupt angewendet werden darf [BSMM97]. Somit ist es nicht ohne weiteres möglich, die z-Transformation zur vollständigen Beschreibung von Multirate-Systemen nach dem bisher dargestellten Ansatz anzuwenden.

Für eine vollständige Beschreibung von Multirate-Systemen ist eine Erweiterung oder Modifikation der z-Transformation notwendig, was eine Herausforderung für zukünftige Arbeiten darstellt. Im Sinne von selbstoptimierenden mechatronischen Systemen kann eine solche Beschreibungsform einen entscheidenden Fortschritt bringen, da damit Vorhersagen über das Systemverhalten *einer* bestimmten Konfiguration schon beim Entwurf getroffen werden können. In Ergänzung der Rekonfiguration ist es dann möglich, das Systemverhalten selbst bei unterschiedlichen Systemkonfigurationen *und* dem Einsatz von Multirate-Verfahren vorherzusagen. So weit gehen die bisherigen Untersuchungen in dieser Arbeit nicht. Sie zeigen aber die Notwendigkeit, diesen Weg zu gehen.

4.4 Ansätze zur Vermeidung von Störeffekten in Multirate-Systemen

Zur numerischen Lösung von Differentialgleichungssystemen mit unterschiedlichen Schrittweiten werden Integrationsverfahren verwendet, die aufgrund ihrer unterschiedlichen Zeitkonstanten als *Multirate-Verfahren* bezeichnet werden. Es handelt sich dabei um gewöhnliche Integrationsverfahren, die für diesen Zweck erweitert wurden. Auch Ansätze zur Kompensation der bereits dargestellten Koppeleffekte sind in diesen Verfahren enthalten. Eine Zusammenfassung der Verfahren und Ansätze nach [Blu78] findet sich z. B. bei [Obe98a].

In [Blu78] wird ein System von Differentialgleichungen, bestehend aus zwei Subsystemen, untersucht:

$$\begin{aligned} \dot{x} &= f(x, y) \\ \dot{y} &= g(x, y) \end{aligned} \tag{4.4.1}$$

x stellt hier den Zustand des langsamen und y den des schnellen Teilsystems dar. Das schnelle Teilsystem wird mit der Schrittweite h, das langsame mit dem ganzzahligen Vielfachen $H = Kh$ berechnet. Nach insgesamt n Schritten sind sowohl das schnelle als auch das langsame System bis zum Zeitpunkt $t = t_0 + nH = t_0 + nKh$ ausgewertet. Ab diesem Zeitpunkt kann ein neuer Schritt $H = Kh$ beginnen. Dieser Schritt, zu dem beide Systeme ausgewertet wurden, wird im Folgenden als *Gesamtschritt* bezeichnet. Die Betrachtung eines Gesamtschrittes genügt für die Untersuchung der Multirate-Verfahren. Die Synchronisation nach K Teilschritten erspart außerdem die Interpolation von Zwischenschritten ([GW84]). Ausgehend von dem Gesamtsystem nach 4.4.1, ergibt sich das generelle Modell für ein Multirate-Verfahren, das in Abbildung 4.15 zu sehen ist.

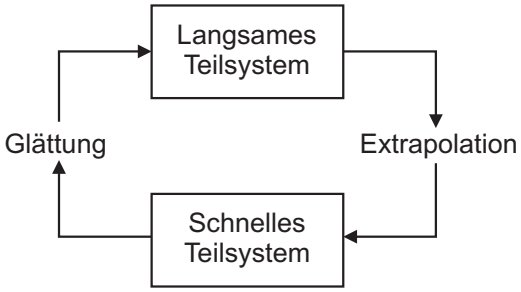

Abbildung 4.15: Schema eines Multirate-Verfahrens

Zur Unterdrückung der Koppeleffekte werden die Daten des schnellen Systems y zum langsamen System x geglättet und die Daten des langsamen Systems x zum schnellen System y extrapoliert. Zur besseren Darstellung des Ansatzes wird ein lineares Gesamtsystem (4.4.2) betrachtet:

$$\begin{aligned} \dot{x} &= A_{11}x + A_{12} \\ \dot{y} &= A_{21}x + A_{22} \end{aligned} \tag{4.4.2}$$

Mit Hilfe dieses Systems werden die Erweiterungen des Euler-Verfahrens beschrieben. Die einfachste Form eines auf dem Euler-Verfahren beruhenden Multirate-Verfahrens für das System aus 4.4.2 ist in 4.4.3 angegeben:

$$\begin{aligned} x_{i+1,0} &= x_{i,0} + Kh(A_{11}x_{i,0} + A_{12}y_{i,0}) \\ y_{i,j+1} &= y_{i,j} + h(A_{21}x_{i,0} + A_{22}y_{i,j});\ j = 1, ..., K-1 \\ y_{i+1,0} &= y_{i,K} \end{aligned} \tag{4.4.3}$$

Die Indizes i und j bestimmen dabei den betrachteten Zeitpunkt:

$$\begin{aligned} t_{i,j} &= t_0 + (iK + j)h;\ i \geq 0;\ 0 \leq j \leq K \\ x_{i,j} &= x(t_{i,j}) \\ y_{i,j} &= y(t_{i,j}) \end{aligned} \tag{4.4.4}$$

Das Verfahren beschreibt so einen Gesamtschritt von $t_{i,0}$ nach $t_{i+1,0}$. Das langsame System x wird über den Gesamtschritt berechnet. Die Ableitung $\dot{x}$ wird zu Beginn dieses Gesamtschrittes $t_{i,0}$ benutzt.

Innerhalb eines Gesamtschrittes bleiben Zustandsänderungen des schnellen Systems y unberücksichtigt. Das schnelle System y wird in K Teilschritten zum Gesamtschritt integriert. In jedem Teilschritt von $t_{i,j}$ nach $t_{i,j+1}$ wird zur Berechnung der Ableitung $\dot{y}$ die Änderung des schnellen Systems berücksichtigt. Da das langsame System nicht aktualisiert wird, wird sein Zustand als konstant angenommen. Alle im Folgenden beschriebenen Erweiterungen basieren auf diesem grundlegenden Verfahren.

4.4.1 Kompensation der Aliasing-Effekte durch Glättung der Koppeldaten

Das aus Abschnitt 4.3.1 bekannte Aliasing-Problem kann durch die Glättung der Koppeldaten vom langsamen zum schnellen System kompensiert werden. Da nach

Gleichung 4.4.3 zur Berechnung des Gesamtschrittes des langsamen Teilsystems x der Zustand $y_{i,0}$ des schnellen Systems benutzt wird und die anderen Zwischenschritte unberücksichtigt bleiben, kann es zu den bereits beschriebenen Störeffekten kommen. Das Ergebnis des letzten Schrittes des schnellen Systems ($y_{i,0}$) allein spiegelt nicht das gesamte Verhalten des Teilsystems während des Intervalls $[t_{i,0}, t_{i+1,0}]$ wieder. In [Blu78] wird deshalb das Glätten der Koppeldaten des schnellen Teilsystems vorgeschlagen. So ist es möglich, das Verhalten des schnellen Systems für das langsame System besser abzubilden. Der veränderte Gesamtschritt hat dann folgende Form:

$$\begin{aligned} x_{i,j+1} &= x_{i,0} \\ y_{i,j+1} &= y_{i,j} + h(A_{21}x_{i,0} + A_{22}y_{i,j}) \end{aligned} \tag{4.4.5}$$

$$\begin{aligned} \bar{y}_{i,0} &= 0 \\ \bar{y}_{i,j+1} &= \bar{y}_{i,j};\ j = 0, 1, ..., K-1 \end{aligned} \tag{4.4.6}$$

$$\begin{aligned} x_{i+1,0} &= x_{i,0} + KhA_{11}x_{i,0} + hA_{12}\bar{y}_{i,K} \\ y_{i+1,0} &= y_{i,K} \end{aligned} \tag{4.4.7}$$

Wie in Gleichung 4.4.6 zu sehen ist, werden die einzelnen Zwischenschritte des schnellen Systems in $y_{i,j}$ festgehalten und aufsummiert. Im letzten Teilschritt wird der arithmetische Mittelwert gebildet, der als Eingang in das langsame System verwendet wird.

Die Anwendung dieses Verfahrens auf das Beispiel zweier in Reihe geschalteter PT_2-Glieder aus Kapitel 4.3.1 ergibt gute Ergebnisse. In Abbildung 4.16 werden die unterschiedlichen Verfahren in der Simulation zweier in Reihe geschalteter PT_2-Glieder aus Kapitel 4.3.1 verglichen. Wie zu sehen ist, sind die Unterschiede zwischen dem System mit Multirate (durchgezogene Linie) und Multirate mit Glättung (gestrichelte Linie) deutlich. Das Multirate-System mit Glättung kommt der analytisch berechneten, genauen Lösung schon sehr nahe, wie ein Vergleich der beiden Verläufe (gepunktete und gestrichelte Linie) zeigt.

4.4.2 Extrapolation der Koppeldaten

Für die Kompensation der Oberschwingungen, die durch die Verkopplung vom langsamen zum schnellen System entstehen, wird in [Blu78] eine lineare Extrapolation der Koppeldaten vorgeschlagen:

$$\begin{aligned} \hat{x}_{i,j+1} &= \hat{x}_{i,j} + h(A_{11}x_{i,0} + A_{12}y_{i,0});\ j = 1, ..., K-1 \\ x_{i+1,0} &= \hat{x}_{i,K} \end{aligned} \tag{4.4.8}$$

$$\begin{aligned} \hat{y}_{i,j+1} &= \hat{y}_{i,j} + h(A_{21}\hat{x}_{i,j} + A_{22}\hat{y}_{i,j});\ j = 1, ..., K-1 \\ y_{i+1,0} &= \hat{y}_{i,K} \end{aligned} \tag{4.4.9}$$

Wie Gleichung 4.4.8 zeigt, wird bei der Auswertung zunächst ein Gesamtschritt für das langsame System x ausgeführt. Dazu werden die Daten des schnellen Systems

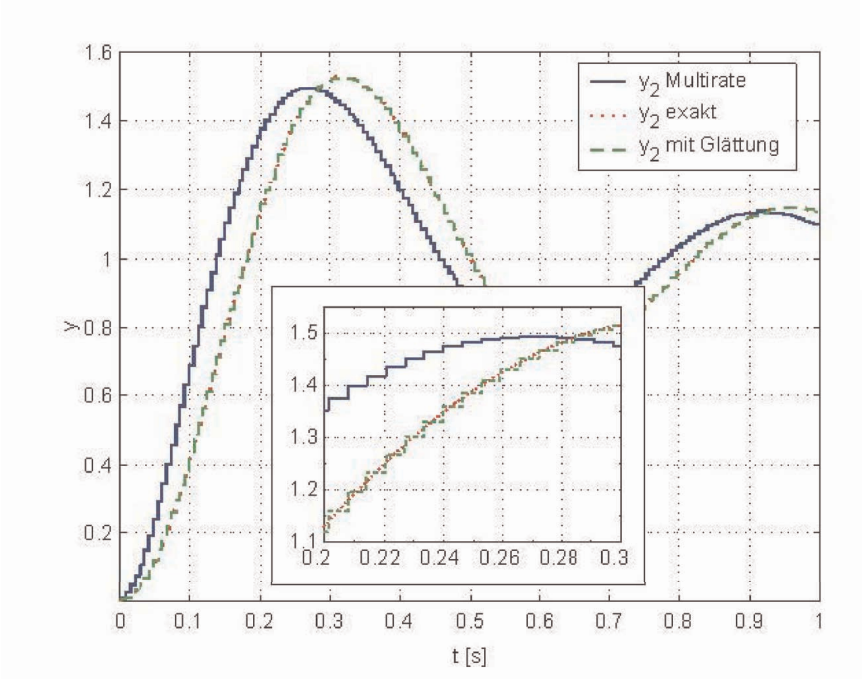

Abbildung 4.16: Vergleich der Systemantworten mit und ohne Glättung zum exakten Verlauf

y aus dem Anfangszustand extrapoliert (Gleichung 4.4.8). Der Gesamtschritt ist in K Zwischenschritte $\hat{x}_{i,j};\ j = 1, ..., K$ aufgeteilt, für die jedoch die Multiplikation $h(A_{11}x_{i,0} + A_{12}y_{i,0})$ nur einmal ausgeführt werden muss. Die interpolierten Werte $\hat{x}_{i,j}$ werden für die Berechnung der Zwischenschritte des schnellen Systems y benutzt. Dadurch soll eine treppenförmige Aktualisierung der Koppeldaten (vgl. Abschnitt 4.3.2) vermieden werden.

Das Ergebnis des Verfahrens, auf das Beispiel aus Abschnitt 4.3.2 angewendet, ist in Abbildung 4.17 zu sehen. Ohne die Extrapolation ergeben sich die bereits in Abschnitt 4.3.2 diskutierten Oberschwingungen im Ausgang des schnellen Systems (durchgezogene Linie). Der treppenförmige Verlauf des langsamen Systems (gestrichelte Linie) wird durch die Extrapolation mit zusätzlichen Zwischenwerten versehen. Daraus ergibt sich ein gleichmäßiger Verlauf (gepunktete Linie), wodurch das Entstehen von Oberschwingungen in diesem Beispiel verhindert wird. Das schnelle System folgt dadurch dem Verlauf besser (Strichpunktlinie).

4.4.3 Gleichzeitiger Einsatz von Glättung und Extrapolation

Nach der separaten Betrachtung der Kompensationsverfahren Glättung und Extrapolation bleibt zu klären, wie sich die Einführung der Methoden auf das Gesamtverhalten eines rückgekoppelten Systems auswirkt. Rückkopplungen treten vor allem bei geregelten mechatronischen Systemen auf, wie vereinfacht in Abbildung 4.1 dargestellt.

Nach [Blu78] werden zur Berechnung von $\hat{y}_{i,j+1}$ aus Gleichung 4.4.7 die durch lineare Extrapolation bestimmten Werte $\hat{y}_{i,j}$ aus Gleichung 4.4.9 benutzt. Daraus ergibt sich folgendes Verfahren:

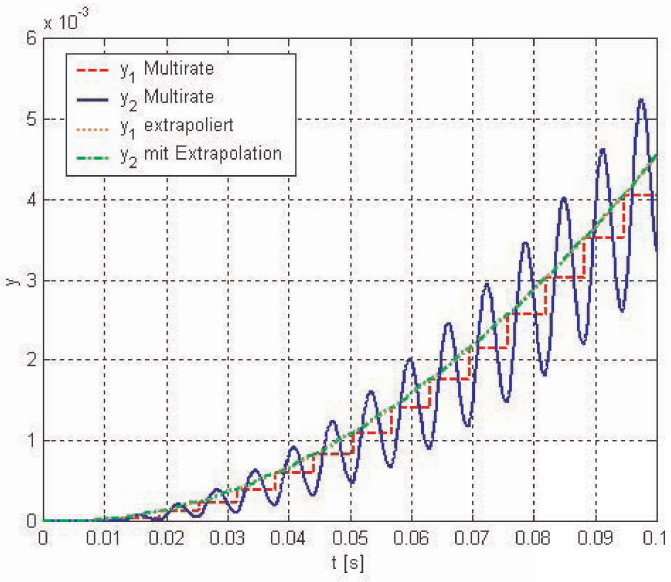

Abbildung 4.17: Multirate-Integration mit und ohne linearer Extrapolation

$$\begin{aligned} \hat{x}_{i,j+1} &= \hat{x}_{i,j} + h(A_{11}x_{i,0} + A_{12}y_{i,0});\ j = 1, ..., K-1 \\ \hat{y}_{i,j+1} &= \hat{y}_{i,j} + h(A_{21}\hat{x}_{i,j} + A_{22}\hat{y}_{i,j});\ j = 1, ..., K-1 \end{aligned} \tag{4.4.10}$$

$$\begin{aligned} \bar{y}_{i,0} &= 0 \\ \bar{y}_{i,j+1} &= \bar{y}_{i,j} + \hat{y}_{i,j};\ j = 0, 1, ..., K-1 \end{aligned} \tag{4.4.11}$$

$$\begin{aligned} x_{i+1,0} &= x_{i,0} + KhA_{11}x_{i,0} + hA_{12}\bar{y}_{i,K} \\ y_{i+1,0} &= \hat{y}_{i,K} \end{aligned} \tag{4.4.12}$$

Die extrapolierten Werte aus Gleichung 4.4.10 des langsamen Teilsystems werden zur Berechnung der Teilschritte des schnellen Systems benutzt. In Gleichung 4.4.11 werden diese aufsummiert ($\bar{y}_{i,j}$) und zur Bildung des Durchschnitts genutzt. Schließlich kann das langsame System mit geglätten Koppeldaten ausgewertet werden, wie Gleichung 4.4.12 zeigt.

4.4.4 Aufwand und Fehler

Multirate-Verfahren sind aus der Motivation heraus entstanden, Rechenzeit einzusparen. Auch wenn bei zunehmender Rechenleistung das Problem in den Hintergrund zu rücken scheint, darf nicht vergessen werden, dass der Aufwand nur deshalb eine geringe Rolle spielt, weil das Verhältnis von Rechenleistung zu Aufwand – also die Komplexität der zu berechnenden Systeme – sich günstig entwickelt hat. Sollten jedoch verstärkt selbstoptimierende mechatronische Systeme zum Einsatz kommen, die zur Optimierung rekonfigurierbare Regelungssysteme einsetzen, kann sich das Verhältnis zwischen Aufwand und vorhandener Rechenleistung wieder anders darstellen. Daher soll an dieser Stelle kurz der Nutzen in Bezug auf eine Ersparnis an Berechnungen nach [Blu78] dargestellt werden.

In [Blu78] wird der Aufwand anhand des Vergleichs der benötigten Multiplikatoren abgeschätzt. Dazu werden Verfahren auf Basis des einfachen Eulerverfahrens verglichen. Das Beispiel, das aus zehn langsamen Gleichungen für x und zwei schnellen Gleichungen für y besteht, wird mit den Integrationsschrittweiten $H = Kh$ mit $K = 100$ ausgewertet. Das Multirate-System wird zusätzlich mit den Verfahren Glättung und Extrapolation ergänzt. Das Referenzsystem wird mit der Schrittweite des schnellen Teilsystems ausgewertet. Hierbei ergibt sich ein Verhältnis des Aufwands von Multirate zu Nicht-Multirate von 0.29.

Darüber hinaus besteht ein weiterer, nicht unerheblicher Vorteil in der Reduzierung des Kommunikationsaufwands, da die Koppeldaten des langsamen Teilsystems nicht bei jedem Schritt gesendet und empfangen werden müssen.

Leider wird in [Blu78] nicht näher auf Fehlerbetrachtungen oder Stabilitätsuntersuchungen eingegangen. Jedoch lässt sich der Fehler eines Integrationsverfahrens abschätzen. Dazu wird zur Berechnung einer Näherung der y_k der exakten Lösung $y(t_k)$ nach [Sch86] eine Fehlerordnung definiert:

> Ein Einschrittverfahren besitzt die *Fehlerordnung* p, falls für seinen lokalen Diskretisierungsfehler d_k die Abschätzung
>
> $$max_{(1 \leq k \leq n)} |d_k| \leq D = const \cdot h^{p+1} = O(h^{p+1}) \tag{4.4.13}$$
>
> gilt, so dass der globale Diskretisierungsfehler g_n... beschränkt ist durch
>
> $$|g_n| \leq \frac{const}{L} e^{nhL} \cdot h^p = O(h^p) \tag{4.4.14}$$

Der lokale Diskretisierungsfehler d_k beschreibt die Abweichung der exakten Lösung von dem berechneten Wert nach einem Schritt. Der globale Diskretisierungsfehler $g_n = y(t_n) - y_n$ beschreibt die Abweichung der Auswertung des numerischen Verfahrens von der exakten Lösung an der Stelle t_n. Besonders hervorzuheben ist, dass nach [Hau83] die Extrapolation mit einem Polynom vom Grad $p - 1$ zu einem zusätzlichen Fehler der Ordnung $O(Kh)^p$ führt. Die Konvergenzordnung wird somit nur eingehalten, wenn die Verfahren zu Extrapolation und Interpolation die gleiche Fehlerordnung aufweisen wie das verwendete Integrationsverfahren.

Dieser Zusammenhang hat erhebliche Konsequenzen: Mit der Ordnung des Integrationsverfahrens steigt der Aufwand für Extrapolation und Interpolation erheblich, so dass der Nutzen des Einsatzes von Multirate dahin sein kann [Hau83]. Aus diesem Grund sollte die Ordnung der Integrationsverfahren beim Einsatz von Multirate nicht zu hoch sein.

4.4.5 Reihenfolge der Auswertung der Teilsysteme

Für die Realisierung von Multirate-Systemen im Sinne der Informationsverarbeitung stellt sich die Frage, in welcher Reihenfolge die einzelnen Teilsysteme des Gesamtsystems ausgewertet werden. Zwei wesentliche Strategien werden in [GW84] und [RG94] unterschieden:

- *Fastest first*-Strategie: Das schnelle Teilsystem wird zuerst in K Schritten mit der Schrittweite h über einen Gesamtschritt berechnet. Die Werte für die

Zwischenschritte des langsamen Teilsystems werden aus dem vorangegangenen Zeitschritt extrapoliert. Anschließend wird das langsame Teilsystem über einen Gesamtschritt mit der Schrittweite H integriert. Die zur Auswertung des schnellen Systems nötigen Werte werden durch Interpolation ermittelt und gegebenenfalls geglättet.

- *Slowest first*-Strategie: Hierbei wird das langsame Teilsystem zunächst in einem Gesamtschritt mit der Schrittweite H integriert. Die für die Berechnung nötigen Werte aus dem schnellen Teilsystem werden mittels Extrapolation aus dem vorherigen Zeitschritt ermittelt. Zur Berechnung des schnellen Teilsystems werden aus dem zuvor durchgeführten Gesamtschritt des langsamen Systems die benötigten Werte interpoliert. Das schnelle System wird in K Teilschritten berechnet.

Die Fastest first-Strategie ist Voraussetzung für die in [Blu78] vorgestellte Glättung der Koppeldaten des schnellen Teilsystems und somit auch für die Kombination von Glättung und Extrapolation des langsamen Systems. Jedoch hat dieser Ansatz einen entscheidenden Nachteil bei Integrationsverfahren, die mit einer Schrittweitensteuerung arbeiten: Bei einer Schrittweitensteuerung wird jeder Schritt auf seine Genauigkeit überprüft. Ist die geforderte Genauigkeit nicht erreicht, wird die Schrittweite verkleinert und der Schritt wiederholt.

Bei einem Multirate-Integrationsverfahren, das nach der Fastest first-Strategie arbeitet, ergibt sich folgende Konsequenz: Stellt sich nach der abschließenden Auswertung des Gesamtschrittes für das langsame Teilsystem heraus, dass die Schrittweite zu groß gewählt wurde, beruhen alle zuvor berechneten Zwischenschritte des schnellen Teilsystems auf der zuvor durchgeführten fehlerhaften Extrapolation des langsamen Systems und müssen ebenfalls wiederholt werden [GW84], [KR99], [Hau83].

Der Einsatz der Slowest first-Strategie resultiert aus einer Überlegung: Multirate-Verfahren bieten sich vor allem bei Teilsystemen an, die untereinander nur schwach verkoppelt sind [RG94]. Wird schwache Kopplung vorausgesetzt, wirken sich die größeren Fehler durch die Extrapolation der Koppeldaten zur Berechnung des langsamen Systems nicht so gravierend aus. Bei Einsatz einer Schrittweitensteuerung unter den getroffenen Annahmen ist dieses Verfahren somit günstig. Allerdings ist der Einsatz einer Schrittweitensteuerung bei Echtzeitsystemen ohnehin nicht geeignet, da dort solche Verfahren eingesetzt werden müssen, bei denen die maximale Berechnungsdauer für einen Zeitschritt im Voraus bestimmt werden kann. Bei Verfahren mit Schrittweitensteuerung ist das nicht möglich, da nicht von vornherein feststeht, mit welcher Schrittweite die geforderte Genauigkeit erreicht wird [Obe98a]. Im Zusammenhang mit selbstoptimierenden Systemen eignet sich die Slowest first-Strategie für nicht echtzeitabhängige Simulationen, etwa für vorausschauende Optimierungen.

4.4.6 Erweiterung von Runge-Kutta-Verfahren zu Multirate-Verfahren

Abschließend sollen nun Multirate-Runge-Kutta-Verfahren betrachtet werden, die insbesondere bei nicht steifen Systemen, wie sie häufig in mechatronischen Systemen auftreten, von Bedeutung sind. Am Beispiel zweier expliziter Multirate-Runge-Kutta-Verfahren nach [KR99] soll kurz skizziert werden, dass sich solche Verfahren, basierend auf *normalen* Runge-Kutta-Verfahren, gut zu Multirate-Verfahren erweitern lassen.

Die in [KR99] vorgestellten Verfahren arbeiten nach der *Slowest first*-Strategie. Um zu verhindern, dass für Interpolation und Extrapolation der Koppeldaten die Daten aus dem vorhergehenden Schritt gespeichert werden müssen, werden die Koppeldaten aus dem aktuellen Zeitschritt heraus bestimmt. Zunächst wird das schnelle Teilsystem über einen Gesamtschritt extrapoliert, damit die Zwischenschritte zur Berechnung des langsamen Teilsystems vorliegen. Zur Verbesserung der Genauigkeit und der Stabilität des Verfahrens wird in einem zweiten Verfahren vorgeschlagen, die Zwischenwerte mit Hilfe eines Verfahrens niedriger Ordnung zu bestimmen; in diesem Fall das Euler-Verfahren. Damit die Ordnung des Gesamtverfahrens nicht reduziert [Hau83] wird, schlägt [KR99] eine Modifikation dieser Vorgehensweise vor, mit der die Fehlerordnung des Multirate-Verfahrens voll erhalten bleibt. Die beschriebenen Multirate-Runge-Kutta-Verfahren (MRK-Verfahren) beinhalten eine Schrittweitensteuerung, die, wie bereits erwähnt, für eine Echtzeitverarbeitung nicht verwendet werden kann. Jedoch ist es durchaus möglich auf den Einsatz einer Schrittweitensteuerung zu verzichten. Somit sind die Verfahren auch für eine Berücksichtigung der *Koppeleffekte* gut geeignet.

Schon [Rük96] nennt Schwierigkeiten beim Nachweis von numerischer Stabilität bei Multirate-Verfahren. Eine interessante Untersuchung der Stabilität dieser MRK-Verfahren findet sich bei [Kvæ00].

$$\begin{bmatrix} \dot{y}_A \\ \dot{y}_L \end{bmatrix} = A \begin{bmatrix} y_A \\ y_L \end{bmatrix} ; \; A \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \tag{4.4.15}$$

y_A stellt ein schnelles (*active*) und y_L ein langsames (*latent*) Teilsystem dar. Dieses Testproblem ist für die Darstellung verschiedener Eigenschaften eines Differentialgleichungssystems geeignet. Es wird durch folgende Parameter beschrieben:

$$\begin{aligned} \gamma &= \frac{a_{12}a_{21}}{a_{21}a_{22}} \\ \kappa &= \frac{a_{22}}{a_{11}} \end{aligned} \tag{4.4.16}$$

Für die Parameter werden a_{11}, $a_{22} < 0$ und $\gamma < 1$ vorausgesetzt, damit A Eigenwerte mit negativem Realteil besitzt. g spiegelt die Stärke der Kopplung wieder. Der Parameter $\kappa > 1$ kann als Maß für die Steifigkeit des Systems angesehen werden. Die oben beschriebenen Verfahren werden auf dieses Testproblem angewendet. Ein Gesamtschritt mit K Schritten der Schrittweite h zur Berechnung des schnellen Teilsystems y_A und einem Schritt der Schrittweite H für y_L lässt sich dann allgemein darstellen als:

$$\begin{bmatrix} y_{A,K} \\ y_{L,1} \end{bmatrix} = \Phi \begin{bmatrix} y_{A,0} \\ y_{L,0} \end{bmatrix} \tag{4.4.17}$$

Für das gewählte Verfahren F gilt: Es ist genau dann asymptotisch stabil, wenn die Eigenwerte von Φ innerhalb des Einheitskreises liegen. Weiterhin müssen Methoden und Schrittweiten so gewählt werden, dass das MRK-Verfahren für das unverkoppelte System ($a_{12} = a_{21} = 0$) stabil ist. Das erfordert, dass die Stabilitätsfunktionen $R_A(ha_{11})$ und $R_L(Ha_{22})$ (siehe Gleichung 4.2.5) vom Betrag her kleiner als 1 sind. In [Kvæ00] werden Bedingungen für die Stärke der Kopplung γ und die Stabilitätsfunktion des schnellen Teils R_A hergeleitet, welche die Forderung an die Eigenwerte

von Φ in jedem Fall erfüllen. In Abhängigkeit von der Steifigkeit κ und dem Faktor zwischen den Schrittweiten $K = H/h$ ergeben sich daraus Stabilitätsbereiche ähnlich der Abbildung 4.2.

In Abbildung 4.18 [Kvæ00] sind im Vergleich dazu diese Stabilitätsbereiche für ein nichtsteifes System ($\kappa = 1$) zu sehen.

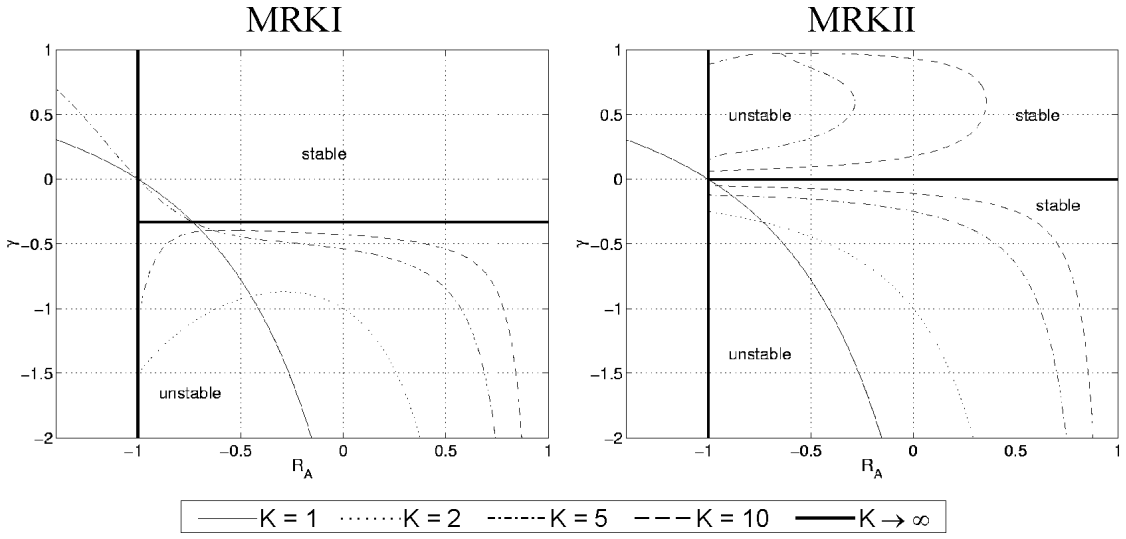

Abbildung 4.18: Stabilitätsbereiche von Multirate-Runge-Kutta-Verfahren

Die Abbildung für MRKI zeigt dabei den Stabilitätsbereich der Methode, bei der die Zwischenwerte des schnellen Teilsystems ausschließlich aus einer Extrapolation des ersten Teilschritts gewonnen werden. Die Abbildung für MRKII zeigt den Stabilitätsbereich der Methode, bei der ein Integrationsverfahren niedriger Ordnung zur Berechnung der Zwischenwerte verwendet wurde. Erstaunlicherweise hat MRKI einen größeren Stabilitätsbereich als die aufwendigere MRKII. Jedoch wird in [Kvæ00] darauf hingewiesen, dass dieses Ergebnis nicht verallgemeinert werden darf, sondern nur für die in [KR99] gewählten Parameter gilt. Das Ergebnis zeigt jedoch insgesamt, dass sich Runge-Kutta-Verfahren gut auf Multirate-Systeme erweitern lassen.

4.4.7 Fazit

Zusammenfassend lässt sich sagen, dass der Einsatz von Multirate-Verfahren für die Simulation und die Realisierung komplexer mechatronischer Systeme insbesondere bei stark unterschiedlich schnellen Teilsystemen nicht nur sinnvoll, sondern notwendig ist. Dies gilt in besonderem Maße für selbstoptimierende mechatronische Systeme.

Weiterhin ist festzustellen, dass es hinreichende Verfahren gibt, um numerische Stabilität abzuschätzen; für eine genaue Modellierung *aller* Effekte in komplexen Systemen fehlen bisher jedoch geeignete, vor allem leicht handhabbare Methoden. Das vorherige Kapitel zeigt jedoch Möglichkeiten und erste Ansätze für weitergehende Arbeiten. Sollten Multirate-Verfahren in selbstoptimierenden Systemen mit Rekonfiguration zum Einsatz kommen, ist eine Abschätzung jedoch unumgänglich.

Kapitel 5

Informationsverarbeitung – Entwurf und Implementierung

> Gute Informationen sind schwer zu bekommen. Noch schwerer ist es, mit ihnen etwas anzufangen (Sir Arthur Conan Doyle)

Für die Realisierung selbstoptimierender mechatronischer Systeme werden neben Entwurfs- und Strukturierungsmethoden für das technische Teilsystem auch Ansätze für eine geeignete Informationsverarbeitung benötigt. Dabei spielt der Ansatz der *rekonfigurierbaren Systeme* und der *Hybriden Komponenten* [Bur06] (vgl. Kapitel 3) eine wichtige Rolle. Der Entwurf solcher Systeme wurde bereits aus technisch-mechatronischer Sicht dargestellt. Der Entwurf der Informationsverarbeitung benötigt darüber hinaus formale Darstellungen, um zu einer Abbildung im Rechner zu gelangen.

Für Selbstoptimierung auf der Ebene der Informationsverarbeitung ist es notwendig, Teilkomponenten zur Laufzeit auszutauschen. Da es sich hierbei um die mathematische Abbildung mechatronischer Systeme und speziell um Reglersysteme handelt, ist neben der formalen Beschreibung auch eine modulare Codegenerierung erforderlich.

5.1 Modularisierung von Modellen

In diesem Abschnitt werden grundsätzliche Fragen bei der Aufteilung von Teilsystemen auf der Ebene der Informationsverarbeitung beleuchtet. Dabei steht der Aufbau von Verständnis für Probleme und Lösungsansätze im Vordergrund. Es werden verschiedene Strategien zur Modularisierung des Systemcodes vorgestellt, die bei der Rekonfiguration selbstoptimierender Systeme eine Rolle spielen. Darüber hinaus sind die Ansätze aber auch für eine effiziente Generierung nicht rekonfigurierbarer Systeme interessant, insbesondere, wenn Systeme aus vorgegebenen Teilkomponenten erstellt werden sollen.

5.1.1 Zerlegung in Teilkomponenten

Die bisherigen Überlegungen zu einer Modularisierung von Systemen und Systemmodellen gingen von der modular-hierarchischen Bauteilstruktur mechatronischer Systeme bzw. von hierarchischen Reglersystemen aus. Jedes Teilsystem wird jedoch durch mathematische Gleichungen beschrieben, die in Bezug auf das Gesamtsystem

nicht unabhängig voneinander sind. Bei der mathematischen Darstellung dynamischer Systeme in Form von gewöhnlichen Differenzialgleichungen und anschließender Transformation in numerisch berechenbare Differenzengleichungen ergibt sich eine bestimmte Reihenfolge für die Berechnung der Systemgleichungen, die *Auswertereihenfolge*. Fatalerweise ist diese Auswertereihenfolge abhängig von der Verkopplung der Teilsysteme. Für rekonfigurierbare Systeme bedeutet dies, dass mit der Rekonfiguration auch die Auswertereihenfolge neu bestimmt werden muss. Werden Teilsysteme ohne Betrachtung der inneren Abhängigkeiten der Gleichungen verkoppelt, kann es zu zyklischen Abhängigkeiten bei der Berechnung kommen. Diese führen zu Verklemmungen bei der Berechnung, sogenannten *Deadlocks*.

Ein Beispiel für die zyklische Verkopplung zweier einfacher Teilsysteme ist in Abbildung 5.1 zu sehen. Die dargestellten Teilsysteme stehen symbolisch für komplexe Teilsysteme und sind hier zur Verdeutlichung auf ein Minimum reduziert.

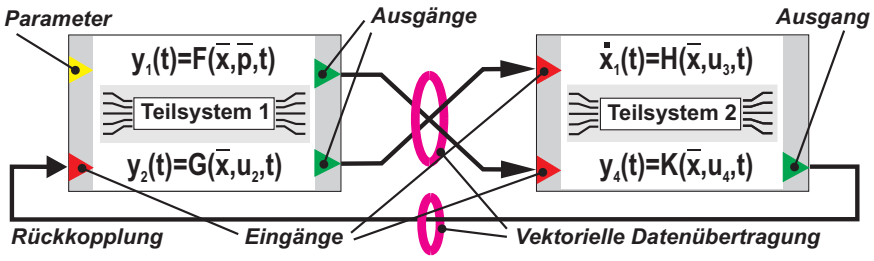

Abbildung 5.1: Gekoppelte Teilsysteme mit vektorieller Datenkopplung

Das System besteht aus zwei Teilsystemen, die durch Ein- und Ausgänge miteinander verkoppelt sind. Teilsystem 1 berechnet zwei Ausgangswerte y_1 und y_2, Teilsystem 2 den Ausgang y_4 und die Ableitung des Zustands $\dot{x}_1$[25]. Soll nun ein Teilsystem nach dem anderen berechnet werden, so ist dies aufgrund der Verkopplung nicht möglich: $\dot{x}_1$ ist abhängig von einem Ausgang des Teilsystems 1 und somit von y_2. Der Ausgang y_2 hängt wiederum von y_4 ab und dies von y_1. Eine schrittweise oder parallele Berechnung der Auswertung der Teilsysteme ist in dieser Konfiguration nicht möglich.

Eine Variante, dieses Problem zu lösen, ist die Betrachtung auf Gleichungsebene. Wird die Auswertereihenfolge unabhängig von den Teilsystemen global festgelegt, ist eine Auswertung problemlos möglich, sofern keine zyklische Abhängigkeit auf Gleichungsebene existiert. Dieser Ansatz führt jedoch nicht zu einer echten Modularisierung, da alle Teilsysteme in Kenntnis des Gesamtsystems erzeugt werden müssen. Die Teilsysteme sind dann zwar verteilt ausführbar, jedoch *nicht wieder verwendbar* und auch nicht für eine Rekonfiguration geeignet, in der sich die Auswertereihenfolge ändert (siehe z. B. [Gam02, GOD03, GO03, Sto04]). Um Auswertereihenfolge und Codegenerierung voneinander zu trennen, müssen die auszuwertenden Systeme näher betrachtet werden.

5.1.2 Modularisierung der Systemgleichungen

Mit einer einfachen Überlegung kann die Abhängigkeit der Teilsysteme voneinander sehr weitgehend gelöst werden. Die Systemgleichungen in Zustandsraumdarstellung,

[25] Das Szenario entspricht typischerweise einem System in Zustandsraumdarstellung.

die zur Berechnung eines einzigen Zeitschrittes nötig sind, lassen sich in drei Gruppen einteilen [Hon98], [GO03]:

Nichtdurchgriffscode (ND-Code) : Dazu gehören alle Gleichungen, die nur von inneren Zuständen des Systems oder deren Parametern abhängen:

$$\underline{y}_N(t) = \underline{F}(\underline{x}, \underline{p}, t) \tag{5.1.1}$$

Durchgriffscode (D-Code) : Zu dieser Kategorie gehören alle Gleichungen, die von einem Eingang abhängen und einen Ausgang (eines Teilsystems) berechnen:

$$\underline{y}_D(t) = \underline{F}(\underline{x}, \underline{\mathbf{u}}, \underline{p}, t) \tag{5.1.2}$$

Zustandscode (S-Code) : Diese Gleichungen berechnen die Zustandsgleichungen des Systems:

$$\dot{\underline{x}}(t) = \underline{F}(\underline{x}, \underline{\mathbf{u}}, \underline{p}, t) \tag{5.1.3}$$

Werden die Gleichungen der Teilsysteme entsprechend diesen Kategorien erzeugt, lassen sich eine große Zahl von Teilsystemen ohne Verklemmung miteinander verkoppeln. Wie die Berechnung abläuft, ist in Abbildung 5.2 illustriert.

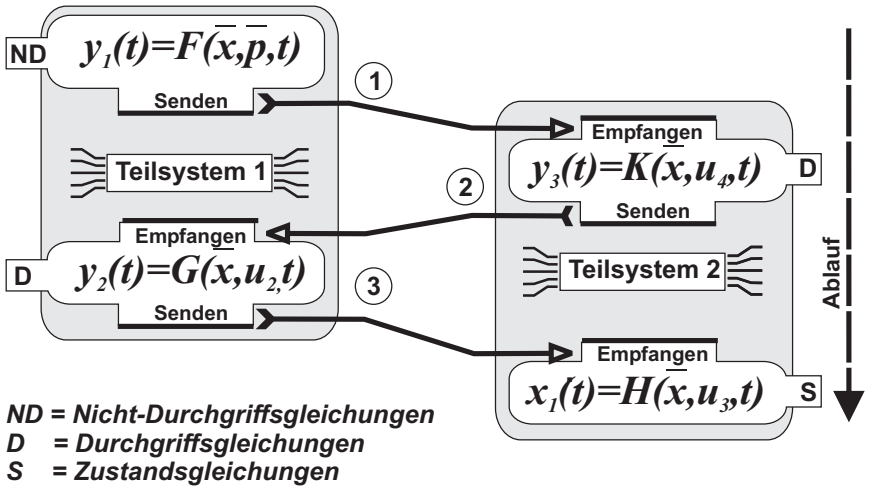

Abbildung 5.2: Kommunikation zweier Teilsysteme mit Aufteilung in ND-, D- und S-Code

Zunächst werden der ND-Code berechnet und die Ausgänge (linke Seite der Gleichungen) an verkoppelte D-Codes anderer Teilsysteme geschickt (1). Im nächsten Schritt werden der Durchgriffscode berechnet und das Ergebnis an andere Teilsysteme geschickt (2). Danach wird weiterer D-Code berechnet, sobald die Eingänge vorliegen. Die Ergebnisse werden an den S-Code weitergeleitet (3). Im letzten Schritt wird der S-Code ausgewertet. Die Auswertungen des ND-Codes und des S-Codes lassen sich ggf. parallelisieren, nur verkoppelter D-Code muss entsprechend der Auswertereihenfolge sequentiell berechnet werden.

Diese Art der Aufteilung erhöht die Anzahl der Möglichkeiten, Verkopplungen durchzuführen, ohne eine Neugenerierung der Teilsysteme vornehmen zu müssen,

schon erheblich. Allerdings lassen sich nicht alle Kopplungsfälle lösen. Insbesondere bei größeren Teilsystemen mit vielen Verkopplungsmöglichkeiten steigt die Wahrscheinlichkeit einer Verklemmung, wie Abbildung 5.3 beispielhaft zeigt.

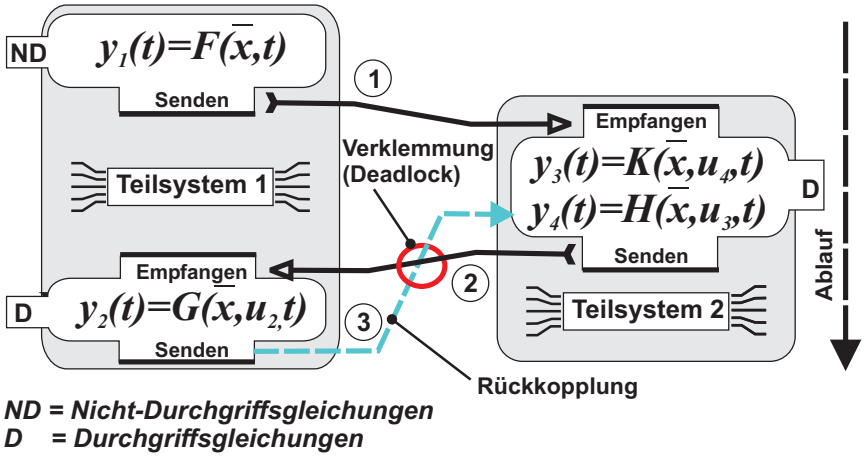

Abbildung 5.3: Verklemmung durch eine Rückkopplung im D-Code

Mehrfachverkopplungen zwischen D-Code führen unter Umständen zu einer Verklemmung. In Abbildung 5.3 ist zu sehen, dass die Kommunikation (1) kein Problem bringt, jedoch Schritt (2) nicht vor (3) berechnet werden kann und umgekehrt.

Für die Lösung dieses Problems existieren verschiedene Ansätze. Ein Vorschlag ist die Verwendung von Iterationsschleifen, um so die fehlenden Eingangsgrößen zu bestimmen [Rük96]. Allerdings ist dieser Ansatz nur bei Simulationen möglich und nicht für Echtzeitverarbeitung geeignet. Ein weiterer Weg, das Problem der Rückkopplung zu lösen, ist der Einsatz von Filtern [Azi90]. Im einfachsten Fall wird bei diskreter Berechnung der Wert des letzten Zeitschritts für fehlende Eingänge verwendet. Der Einsatz von Filtern führt jedoch in jedem Fall zu einer Veränderung des Systemverhaltens, was in einigen Fällen zu Fehlern bei der Auswertung führen kann, die nicht zu tolerieren sind [Vöc03].

5.1.3 Modularisierung nach Ausgangsblöcken

Für eine *echte* Modularisierung von Teilsystemen ist ein noch weiterreichender Ansatz nötig. Ziel muss sein, Teilsysteme so zu erzeugen, dass, abgesehen von algebraischen Schleifen, die sich grundsätzlich nur durch die Modellierung verhindern lassen, beliebige Kopplungszustände zwischen Ein- und Ausgängen möglich sind.

Die bisher diskutierten Ansätze verfolgen das Ziel, über einfache, leicht zu bestimmende Auswertereihenfolgen das Gesamtsystem auszuwerten. Die Auswertereihenfolge wird dabei zum Zeitpunkt der Codegenerierung bestimmt. Im Fall von verkoppelten Teilsystemen werden die Teilsysteme nacheinander berechnet. Im Fall der Modularisierung der Systemgleichungen wird die Reihenfolge ND – D – S festgelegt, wobei bei mehreren verkoppelten D-Codes wiederum die Reihenfolge der modularen Teilsysteme verwendet wird. In einem rekonfigurierbaren System *kann* aber die Reihenfolge der Auswertung nicht in der Codegenerierung festgelegt werden, sondern erst im Betrieb und das bedeutet: zur Auswertezeit oder zumindest zwischen (Neu-)Verkopplung und Auswertung. Deshalb wird ein neuer Ansatz benötigt, der die Rekonfiguration berücksichtigt.

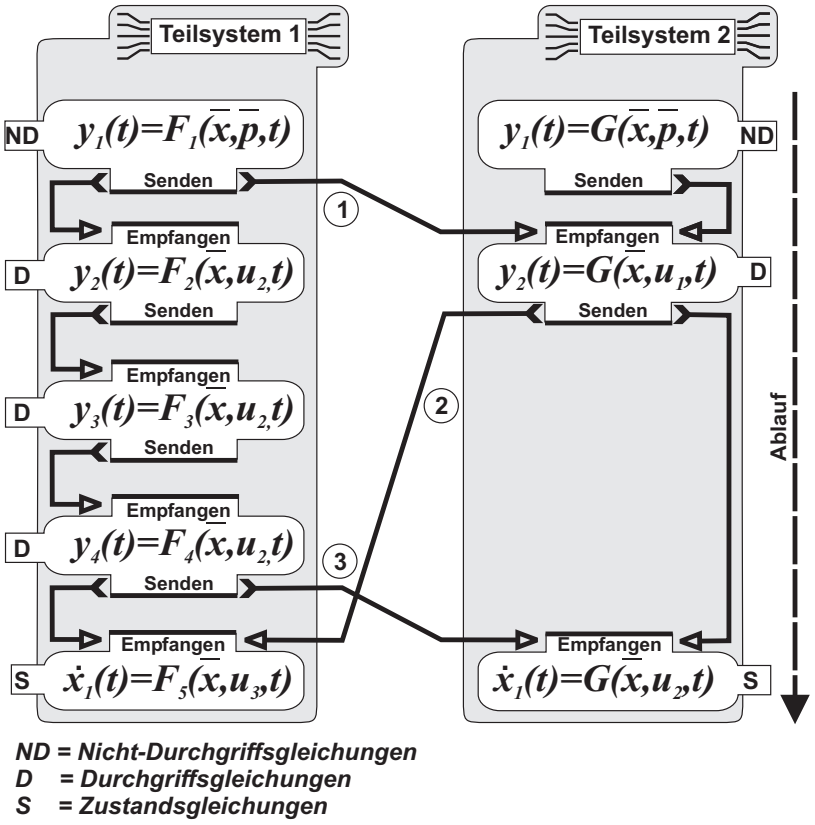

Abbildung 5.4: Berechnung der Auswertereihenfolge nach Ausgangsblöcken

In der Abbildung 5.4 [OHD+01] sind vereinfacht die Auswertung und die Kommunikation zwischen zwei Teilsystemen nach Bildung von Ausgangsblöcken dargestellt. Dabei muss folgendes beachtet werden: Im ND-Code sind *alle* Gleichungen zusammengefasst, die nur Ausgänge produzieren, nicht von Eingängen abhängen und deshalb zu Beginn eines Zeitschrittes sofort berechnet werden können. Im S-Code sind *alle* Gleichungen zusammengefasst, die *keine* Ausgänge produzieren und Zustände für den nächsten Zeitschritt berechnen. Dies entspricht dem Ansatz der *Modularisierung der Systemgleichungen.*

Wesentliche Änderungen finden sich in zwei Bereichen:

1. Alle Gleichungen die, *einen* Ausgang produzieren und von einen oder mehreren Eingängen abhängig sind, werden in einzelnen D-Code-Blöcken zusammengefasst.

2. Neben der Kommunikation der Ausgänge über Teilsystemgrenzen hinaus müssen auch interne Kopplungen berücksichtigt werden, da die Gesamtauswertereihenfolge über alle Kopplungen zwischen Gleichungen bestimmt werden muss.

Das jeweilige Teilsystem besteht somit aus Blöcken von Gleichungen die jeweils einen oder mehrere Ausgänge produzieren bzw. Zustände berechnen. Die Reihenfolge der Berechnung ist nach der Erzeugung der Teilsysteme *nicht* festgelegt. Um eine Berechnung durchführen zu können, müssen die Teilsysteme erst gebunden werden, das heißt, es müssen alle Abhängigkeiten zwischen Gleichungsblöcken gesetzt und auf der Basis des entstehenden *Abhängigkeitsgraphen* die Auswertereihenfolge berechnet werden. Die geschieht unmittelbar vor der Berechnung.

5.2 Modulare Codegenerierung und Steuerung

Nachdem verschiedene Möglichkeiten der Modularisierung beschrieben wurden, soll im Folgenden ein Vorgehen zur Erzeugung von modularem Beschreibungscode gezeigt werden. Modulare Codegenerierung muss sich im Kontext rekonfigurierbarer Systeme zwei wesentlichen Fragen stellen:

1. Wie kann der Code geschickt partitioniert werden?

   Ziel muss es sein, möglichst große zusammenhängende Gleichungen zu finden, die jedoch ein gerichtetes Ein-Ausgangsverhalten zeigen. Mit anderen Worten: Finde Gleichungen, die stückweise immer in der gleichen Reihenfolge zu berechnen sind. Dort, wo Veränderungen auftreten können, müssen die Gleichungen zerlegt (partitioniert) werden. Dabei ist die Zuordnung zum Teilmodell zu erhalten.

2. Wie kann modularer Code verkoppelt und effizient ausgewertet werden?

   Bei Echtzeitsystemen ist eine vorhersagbare Berechnungszeit immer Grundvoraussetzung für die Einhaltung von harten Echtzeitbedingungen. Bei einer Rekonfiguration muss der Code in kürzester Zeit wieder auswertbar sein. Dies erfordert entsprechende Auswertemöglichkeiten und Verfahren, welche die Auswertereihenfolge schnell neu berechnen können.

Die Berechnung der Auswertereihenfolge erfolgt üblicherweise mit Hilfe eines Auswertegraphen, der die Abhängigkeiten der einzelnen Gleichungen von einander beschreibt. Auf Basis dieses Auswertegraphen, der sich bei jeder Rekonfiguration mit dem System ändert, kann eine Steuerung der Auswertung erfolgen.

5.2.1 Partitionierung

Zur Erläuterung der Partitionierung soll auf das Beispiel aus [OGBG04] zurückgegriffen werden. Für eine weitergehende Betrachtung des Problems sei auf verschiedene Arbeiten verwiesen, vor allem [Bur06] und [BGGO04] sowie [GO03]. Ein Algorithmus für die Partitionierung findet sich in [OGBG04].

Ein Teilmodell kann im einfachsten Fall durch die Gleichungen repräsentiert werden. In Anlehnung an die Beschreibungsformen der Entwicklungsumgebung CAMeLView [iXt06] in Zustandsraumdarstellung ist die kleinste Einheit ein Basisblock (z. B. ein PIDT1-Regler). Dieser Basisblock enthält Gleichungen, die sein gerichtetes Ein-/Ausgangsverhalten beschreiben. Die Variablen lassen sich in drei Klassen einteilen:

1. Eingänge (Inputs) sind Variablen, die zu jedem Zeitschritt von außen neu besetzt werden, z. B. Verkopplungen mit Ausgängen anderer Systeme.

2. Ausgänge (Outputs) sind Variablen, die zu jedem Zeitschritt vom Teilsystem durch die Gleichungen neu berechnet werden.

3. Zustände (States) sind die inneren Zustände des Systems.

4. Parameter (Auxiliars) sind Variablen, die als Hilfsgrößen zur Berechnung verwendet werden, aber nicht mit dem Zeitschritt aktualisiert werden.

Zu jedem Zeitschritt (Hauptschritt, vgl. 4) werden alle Eingänge ausgewertet und neue Ausgänge sowie Zustände berechnet. Für die Verkopplung zu anderen Teilsystemen sind die Abhängigkeiten zwischen Ein- und Ausgängen sowie den Zuständen interessant. Die gerichteten Abhängigkeiten können als mehrere unabhängige Teilgraphen (Wald, englisch: *forest*) dargestellt werden.

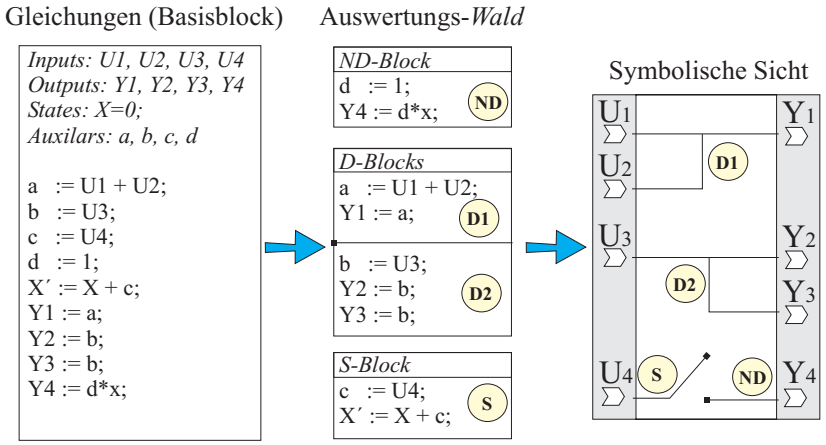

Abbildung 5.5: Auswertegraph eines Basisblocks

Die Abbildung 5.5 [OGBG04] zeigt drei Schritte zur Berechnung des Auswertegraphen. Im ersten Schritt liegen die Systemgleichungen vor (links). Im Folgenden (Mitte) sind die Gleichungen nach den in Abschnitt 5.1 Kategorien sortiert. Die Abhängigkeiten bilden einen Wald von Teilgraphen. Im letzten Schritt liegt der Block als Auswertegraph symbolisch vor; genau genommen als eine *Vielzahl von Teilgraphen*, die einen Wald bilden. Das Beispiel ist so gewählt, dass alle vier Grundfälle einfach dargestellt werden. Komplexere Systeme unterscheiden sich nur noch quantitativ von der dargestellten Struktur.

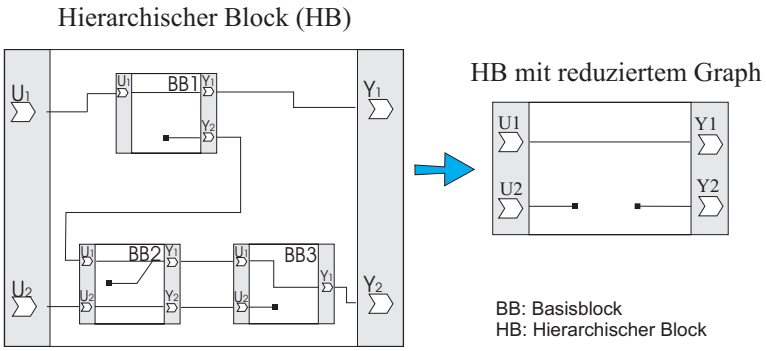

Abbildung 5.6: Auswertegraph einer Hierarchie

Wie die symbolische Sicht in Abbildung 5.5 zeigt, genügt für die Berechnung der Auswertereihenfolge die Beziehung zwischen Eingang, Ausgang und Zustand. Alle Gleichungen, die *innerhalb* eines Blockes liegen, sind für die Auswertereihenfolge der verkoppelten Systeme uninteressant. Daraus ergibt sich eine sehr praktische Konsequenz für hierarchisch verkoppelte Systeme: Der Algorithmus für die Berechnung der Auswertereihenfolge kann verallgemeinert auf *hierarchische* Systeme übertragen werden. Die Berechnung erfolgt dabei von innen nach außen, d. h. ausgehend von

Basisblöcken zu immer höher gelegenen hierarchischen Teilsystemen, bis schließlich das Gesamtsystem ausgewertet werden kann. Dabei entstehen für jede Ebene der Hierarchie reduzierte Auswertegraphen.

Die Abbildung 5.6 zeigt beispielhaft, wie aus einer Hierarchie von verkoppelten Teilsystemen ein reduzierter Graph abgeleitet werden kann. Die innere Struktur ist für die Bestimmung der Auswertereihenfolge der nächst höheren Ebene nicht von Bedeutung, lediglich der abgeleitete reduzierte Graph wird als quasi als *Extrakt* weiterhin benötigt.

5.2.2 Steuerung der Auswertung

Die Ergebnisse aus Abschnitt 5.2.1 können, in Verbindung mit den Annahmen aus Abschnitt 3.2 für eine Steuerung der Auswertung in hierarchischen Systemen genutzt werden. Dies führt wiederum zu den in Abschnitt 3.3 beschriebenen Hybriden Komponenten, die als Ergebnis den Auswertegraphen zur Kontrolle der Auswertereihenfolge nutzen. Eine Beschreibung des Mechanismus zur Kontrolle findet sich in [BGO04a, Boi05] und [Bur06].

5.3 Laufzeitumgebung IPANEMA

Nachdem ein Modell für die Erzeugung von modularen, rekonfigurierbaren Teilsystemen dargestellt worden ist, stellt sich nun die Frage, welchen Rahmen solche Systeme benötigen, um ausgeführt werden zu können. Für die Ausführung wird eine Plattform benötigt, die Systeme unabhängig von der informationstechnischen Hardware und dem Betriebssystem testen wie auch in Betrieb nehmen kann. Darüber hinaus sind Fähigkeiten wie *Echtzeitfähigkeit*, *verteilte (parallele) Verarbeitung* und die Bereitstellung *verschiedener numerischer Verfahren zum Lösen der ODEs* unbedingt notwendig. Eine Umgebung, die diese Anforderungen erfüllt, ist die Laufzeitplattform IPANEMA.

IPANEMA wurde schon in zahlreichen Veröffentlichungen und verschiedenen Dissertationen beschrieben und dokumentiert. Wichtige Meilensteine bilden dabei [Scz95, Hon98, Gam02] und [Sto04].

Aufgrund der Dynamik des Projektes soll an dieser Stelle nur ein kurzer Überblick über Grundkonzepte und Struktur von IPANEMA und die Erweiterungen zur Unterstützung von modularen Teilsystemen gegeben werden, soweit es für das Verständnis der Problematik nötig ist.

5.3.1 Grundkonzept

IPANEMA (Integration Platform for Networked Mechatronic Applications) wurde primär für Ausführung, Überwachung und Simulation von kontinuierlichen Systemen entwickelt. Ziel war die Entwicklung einer systemunabhängigen Umgebung, die in der Lage ist, Modelle bzw. Teilmodelle mechatronischer Systeme gegebenenfalls verteilt zu simulieren und zu überwachen. IPANEMA sollte insbesondere bei der verteilten Simulation die Komplexität des Netzwerks, auf dem es ausgeführt wird, verbergen; das heißt: Das Netzwerk soll für den Anwender *transparent* sein. Darüber hinaus

sollte IPANEMA in der Lage sein, harte Echtzeitbedingungen zu erfüllen, auch externe Hardware ansprechen können und somit für Hardware-in-the-Loop-Simulation geeignet sein.

Diese genannten Anforderungen lassen sich nicht allein durch die Laufzeitumgebung erfüllen. Darüber hinaus werden auch sowohl eine geeignete Mathematik als auch Werkzeuge für Modellbildung und Codegenerierung benötigt. Die Topologie von IPANEMA basiert auf einem objektorientierten Ansatz, der das Gesamtsystem in kleinere spezialisierte Objekte aufteilt. Ausgehend von einer Zerlegung des Modells in Teilkomponenten (vgl. Abschnitt 5.1.1), werden die einzelnen Komponenten des Modells auf die sogenannten *Calculator*-Objekte verteilt, welche die Ausführungsebene von IPANEMA repräsentieren. Calculatoren werten die eingebetteten Systemgleichungen auf Basis gewöhnlicher Differentialgleichungen mit Hilfe von numerischen Solvern aus. Die Solver arbeiten nach Runge-Kutta- und Adams-Verfahren[26].

Die Calculatoren können Daten entsprechend der Kopplungen der Teilsysteme untereinander austauschen. Wird IPANEMA in der Echtzeit-Verarbeitung eingesetzt, so unterliegen die Calculatoren harten Echtzeitbedingungen.

Ein zentraler Gedanke bei der Konzeption von IPANEMA war der Einsatz als Umgebung für die Hardware-in-the-Loop-Simulation. Ausgehend von einer vollständigen Simulation im Rechner, sollte es möglich sein, Teilsystemmodelle schrittweise durch reale Systeme zu ersetzen. Die Kopplung zu einem realen Teilsystem wird durch ein entsprechendes *Adaptor*-Objekt realisiert, das ein Calculator-Objekt ersetzen kann. Adaptor-Objekte binden das reale Teilsystem an das (Rest)Modell und sorgen somit für eine Kopplung zwischen Echtzeitsystem und Aktorik/Sensorik (vgl. [Obe98b], [Sto04]).

Die Überwachung der Calculator- und Adaptor-Objekte erfolgt durch die *Assistant*-Objekte. Pro Calculator bzw. Adaptor existiert genau ein Assistant-Objekt. Die Aufzeichnung von Daten und die Weitergabe von Befehlen erfolgen nur über diese Objekte, was einen ungestörten Echtzeitbetrieb der Calculatoren sicherstellen soll.

Die Kommunikation zwischen Leitwarte und Assistant-Objekten erfolgt über eine zentrale Komponente namens *Moderator*. Dieser sorgt für eine Netzwerk-Transparenz, d. h. aus Anwendersicht ist es unerheblich, wo genau ein Calculator tatsächlich ausgeführt wird. Auch spielt es keine Rolle, ob ein Teilsystem simuliert wird oder real vorhanden ist. Die zentrale Datenerfassung durch den Moderator sorgt auch hier für Transparenz.

In Abbildung 5.7 [Hon98] ist die beschriebene Struktur von IPANEMA vereinfacht dargestellt. Die Verteilung eines Gesamtsystems auf verschiedene Calculator-Objekte ist von IPANEMA nicht streng vorgegeben und kann nach unterschiedlichen Kriterien erfolgen. Die Aufteilung kann nach Gesichtspunkten der Modularisierung der Teilsysteme, aber auch beispielsweise nach Optimierung der Ausführung (minimale Kommunikation, maximale Auswertegeschwindigkeit etc.) erfolgen.

Die modulare Realisierung ist aber ohne Zweifel der wichtigste Aspekt von IPANEMA. Nur so ist eine schrittweise Realisierung von den durch die modular-hierarchische Struktur vorgegebenen Teilsystemen überhaupt möglich.

Wichtige Erweiterungen und Ergänzungen fanden in [Gam02] statt. Dabei wurde die Modellierungsumgebung CAMeL für IPANEMA angepasst, um schon auf der

[26] Als echtzeitfähige Verfahren werden Solver nach Euler und Heun angeboten.

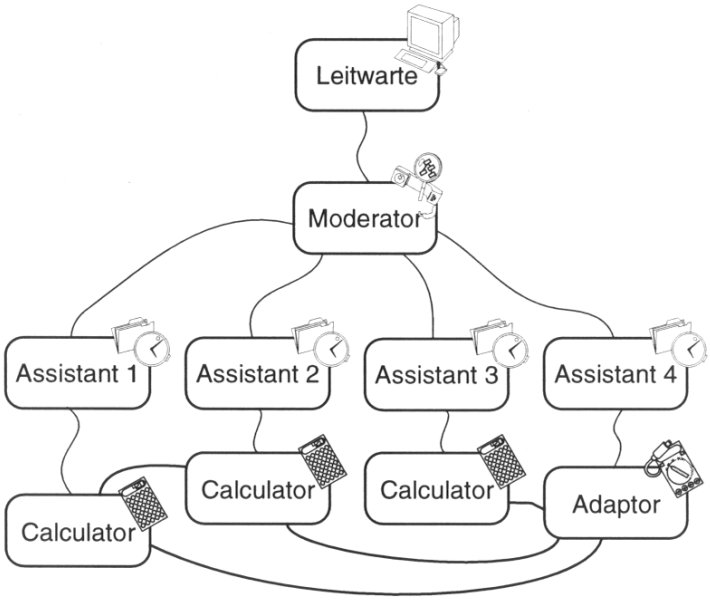

Abbildung 5.7: Typische Objekttopologie einer IPANEMA-Anwendung

Ebene der Modellbildung die Zerlegung in Teilsysteme vorzubereiten. Weitere Arbeiten untersuchten die Möglichkeit von unterschiedlichen Zeitmodellen und Verfahren zur Multirate-Simulation [Obe98a], [Vöc03], die als wichtiger Schritt zu rekonfigurierbaren Systemen und somit zur Selbstoptimierung angesehen werden können.

Des weiteren wurde parallel zu der vorliegenden Arbeit IPANEMA für selbstoptimierende Systeme vorbereitet und erweitert. Insbesondere die modulare Codegenerierung [GBSO04], [Boi05] wie auch die Anwendung hybrider Komponenten standen dabei im Vordergrund [BGO04a, Bur06, BGO06].

5.4 Informationstechnische Realisierung hybrider Komponenten

Aus mechatronischer Sicht bedeutet Rekonfiguration die Veränderung von *Wirkzusammenhängen* (vgl. 3.2). Dies gilt für alle Teildomänen der Mechatronik. Zentraler Ansatzpunkt für Selbstoptimierung ist jedoch die Informationsverarbeitung eines mechatronischen Systems, da hier am ehesten ein Eingriff zur Laufzeit möglich ist. Für den elektrischen oder mechanischen Teil des Systems ist der Aufwand für eine solche Rekonfiguration erheblich größer, was schon das Beispiel des Tandemhauptzylinders (Abbildung 3.1) aus Abschnitt 3.2 deutlich macht.

Für eine Umsetzung dieses Ansatzes sind auf verschiedenen Ebenen Entwicklungen zu leisten. Wird von dem Ansatz des Objektorientierten Mechatronikmodells ausgegangen (vgl. Abschnitt 2.3.6), können drei zur Umsetzung notwendige Schritte identifiziert werden:

1. Entwicklung einer geeigneten Modellierungsumgebung auf mechatronischer Ebene (Entwicklersicht).

2. Entwicklung einer mathematischen Darstellungsform (Modellierung).

3. Entwicklung eines geeigneten Laufzeitmodells (Ausführung).

Im Rahmen des Sonderforschungsbereichs 614 [GLR01] entstand im Rahmen einer Kooperation zwischen den Arbeitsgruppen von Professor Lückel und Professor Schäfer ein neuer Ansatz für die Modellierung und die Implementierung von Werkzeugen zur Unterstützung von rekonfigurierbarer Informationsverarbeitung für mechatronische Systeme. In der ersten Phase wurden neben allgemeinen Grundlagen, auf die sich diese Arbeit konzentriert, erste Schritte für eine Realisierung geleistet (z. B. [GBSO04]). Als ein wesentlicher Schritt ist dabei die Arbeit von Burmester [Bur06] anzusehen. Sie fasst die Aspekte der Modellierung auf Laufzeitebene mit Hilfe eines UML-Modells zusammen. Aus diesem Grund werden an dieser Stelle nur die wesentlichen Ergebnisse zusammengefasst und aus Sicht der Mechatronik beleuchtet. Als weiterführende Arbeit ist noch [BGM+08] zu erwähnen, hier wird auch ein weiteres Anwendungsbeispiel gezeigt.

5.4.1 Diskussionsgrundlagen

Um den Gesamtprozess deutlicher darzustellen, wird ein einfaches Anwendungsbeispiel verwendet. Es handelt sich dabei um einen Prüfstand für Versuche im Rahmen der Lehre. Das Versuchsmodell dient der Erläuterung von Grundlagen für das Verständnis der Aufbaudynamik einer Magnetbahn und der zugehörigen Regelung. Das Experiment ist vergleichbar mit dem Simulationsmodell in Abschnitt 6.2.

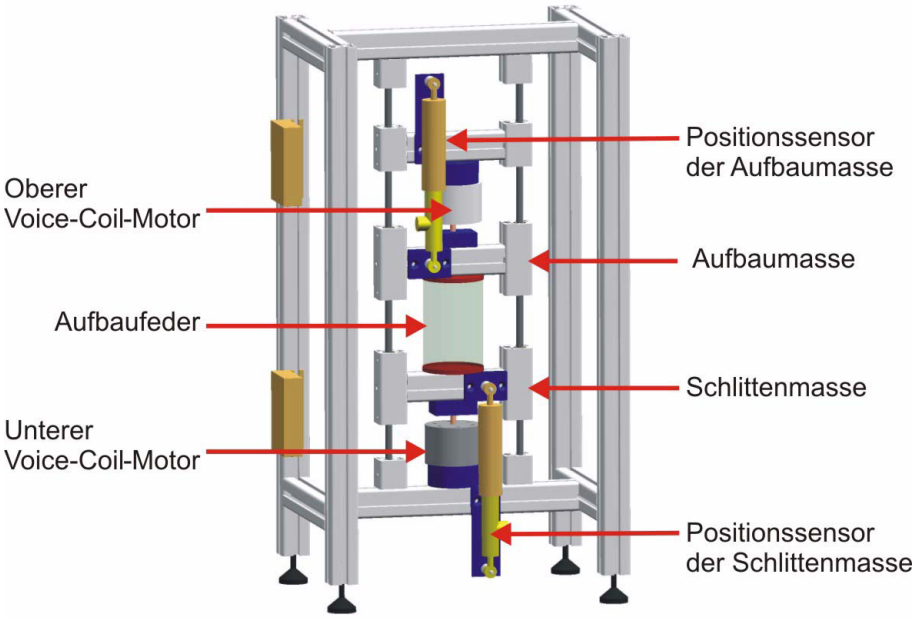

Abbildung 5.8: Versuchsmodell einer Magnetbahn

Abbildung 5.8 zeigt den Versuchsstand. Er besteht im Wesentlichen aus einem Ständer, der die beweglichen Komponenten führt, einer Aufbaumasse, einer Fahrwerksmasse und zwei Voice-Coil-Aktuatoren, die aktives Fahrwerk und Störanregung simulieren. Die Aufbaumasse stützt sich über eine Feder auf der Fahrwerksmasse ab. Der untere Aktuator dient zur Einleitung von Kräften, die der Fahrwerksaktorik und der Schienenanregung entsprechen. Der obere Aktuator leitet weitere Störanregungen in die Aufbaumasse.

Der Versuchsaufbau erlaubt die Darstellung und den Vergleich verschiedener Regler. In dem hier betrachteten Beispiel soll zwischen drei verschiedenen Reglertypen umgeschaltet werden.

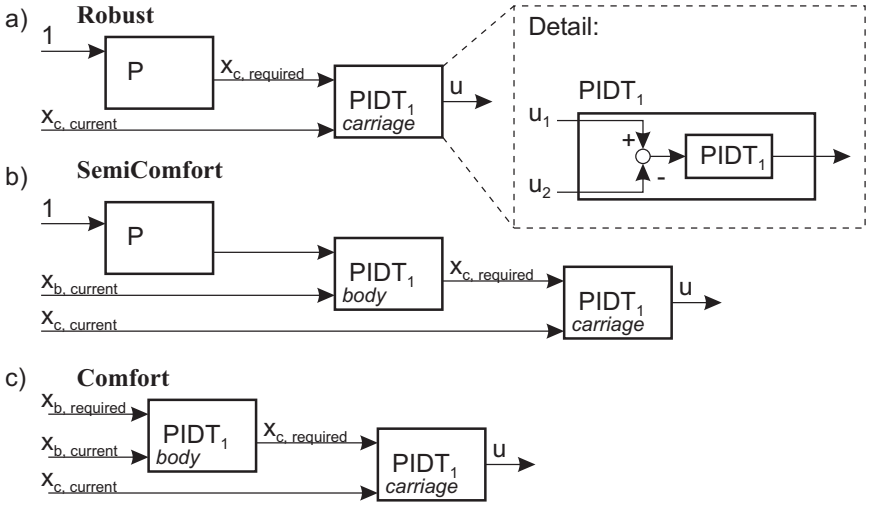

Abbildung 5.9: Verschiedene Reglerkonfigurationen

Die verschiedenen Reglerkonfigurationen sind in Abbildung 5.9 zu sehen.

Die Reglerkonfiguration Typ (a) wird als *Comfort*-Regler bezeichnet. Sie besteht aus einem Regler für die Fußpunktverstellung des Fahrwerks ($\mathrm{PIDT}_{1,\ carriage}$) und einem Regler für die Aufbauposition ($\mathrm{PIDT}_{1,\ body}$). Die Regler sind zu einem Kaskadenregler verschaltet. Eingänge in diese Konfiguration sind die Soll-Position des Aufbaus $x_{body,\ required}$ und die gemessenen Ist-Positionen der Aufbaulage und des Fahrwerks in x-Richtung. Die Soll-Position wird durch den Anwender[27] vorgegeben, die Ist-Positionen werden durch Sensoren gemessen.

Reglerkonfiguration Typ (b) geht davon aus, dass eine Benutzervorgabe nicht vorhanden ist. In diesem Fall wird der Vorgabewert auf einen konstanten Wert gesetzt (P-Glied).

In der Reglerkonfiguration Typ (c) ist der Messwert der Aufbauposition nicht verfügbar. Daher wird für die Position des Fahrwerks hier ein konstanter Wert angenommen (P-Glied). Diese Konfiguration kann auch als Rückfallebene betrachtet werden, falls bei einer Störung die Sensorwerte der Aufbauposition nicht verfügbar sind.

Die verschiedenen alternativen Strukturen a, b und c sind in Abbildung 5.9 zusammengefasst dargestellt. Wesentlich beim Wechsel der Konfiguration ist die Änderung der Anzahl der Eingänge in das jeweilige Reglersystem. Im Falle eines Wechsels muss Sorge getragen werden, dass die neue Konfiguration entsprechend initialisiert wird. Die richtige Initialisierungsstrategie kann ggf. beim Entwurf bereits festgelegt werden. In der Informationsverarbeitung muss die Möglichkeit einer Initialisierung

[27] Die Reglerkaskade kann im Sinne der verallgemeinerten Kaskade durch weitere Ebenen erweitert werden.

und einer Überleitung zwischen verschiedenen Reglern vorgesehen werden. Hierzu ist ein geeigneter Modellierungsansatz für den Entwurf der Informationsverarbeitung nötig.

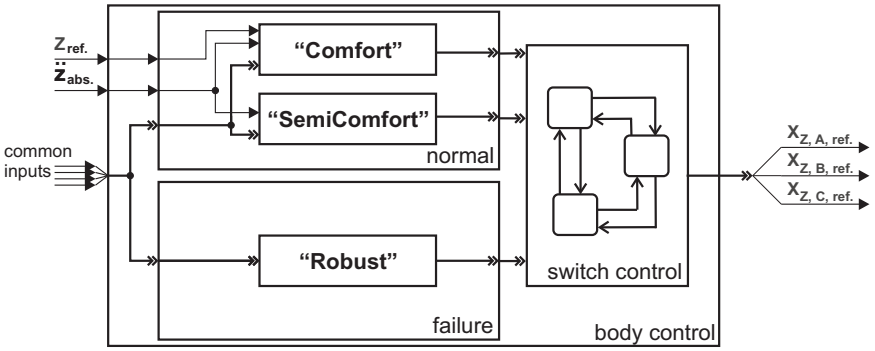

Abbildung 5.10: Kontrolle der Überblendung mit Hilfe eines Statecharts

Die übergeordnete Struktur der Regler ist in Abbildung 5.10 schematisch dargestellt. In dieser Variante wird die Überblendung der Ausgänge der Regler durch ein Statechart kontrolliert. Die drei Regler bleiben zunächst alle im Eingriff; lediglich die Ausgänge werden manipuliert. Das angedeutete Statechart dient dabei lediglich als ein Element, das den Ablauf steuert; der Informationsfluss aus den Reglern bleibt weiterhin kontinuierlich.

Dieser Aufbau hat jedoch wesentliche Nachteile: Zum einen besteht die Gefahr, dass ein Regler, der keinen Durchgriff auf die Ausgänge hat, „übersteuert“, d. h. er wird numerisch instabil; zum anderen wird bei vielen parallelen Reglervarianten unnötig Rechenzeit verbraucht. Eine Kontrolle der Reglerumschaltung darf sich somit nicht auf die Manipulation der Ausgänge beschränken; sie muss vielmehr die gesamte Reglerstruktur kontrollieren und diejenigen Regler initialisieren, starten und wieder stoppen, die aktuell benötigt werden. Dies muss weiterhin im oben angesprochenen Modellierungsansatz ermöglicht werden.

5.4.2 Hybride Modellierung

Die Umschaltvorgänge zwischen den einzelnen Reglerkonfigurationen können durch einen *Hybriden Automaten* beschrieben werden. In jedem Zustand ist eine bestimmte Reglerkonfiguration gültig. Da bei bestimmten Wechseln Überblendungen zwischen *zwei* Reglern verwendet werden, müssen auch diese Konfigurationen als separate Zustände beschrieben werden.

Eine mögliche Konfiguration skizziert die Abbildung 5.11. Die Darstellung zeigt, dass schon dieses einfache Beispiel zu einer komplexen Darstellung führt. Die jeweils gültigen Reglerkonfigurationen sind in den einzelnen Zuständen angedeutet. Die Übergänge zwischen den Zuständen sind Umschaltungen der Konfigurationen, somit Rekonfigurationen des Reglersystems.

Ein Überblendvorgang ist hier nur *ein Zustand*, in dem zwei Reglerkonfigurationen für eine vorgegebene Zeit gleichzeitig existieren und durch eine Überlagerung der Ausgangswerte miteinander verknüpft sind. Ist ein Regler aus dem Eingriff genommen, wird er beim Übergang zum Folgezustand mit nur einer Reglerkonfiguration abgebaut.

Besonders hervorzuheben ist bei der Betrachtung der Abbildung 5.11, dass die Übergänge zwischen zwei Zuständen prinzipiell zwischen zwei Zeitschritten der diskreten Auswertung des Gesamtsystems erfolgen. Daher „sieht“ das kontinuierliche Reglersystem die Rekonfiguration nicht.

Dieser Ansatz erlaubt nun eine vollständige Beschreibung der Abläufe möglicher Rekonfigurationen, ist jedoch in seiner Darstellung etwas unhandlich. Für eine Modellierung wird im Folgenden eine vereinfachte Darstellung vorgeschlagen [GBSO04], [BGO04a], [BGO04b].

Abbildung 5.11 zeigt die Reglerumschaltung als Statechartmodell. Die abgerundeten Kästen stellen Zustände dar, in denen eine bestimmte Reglerkonfiguration gültig (geschaltet) ist. Die Pfeile (Kanten) sind zeitlich diskrete Übergänge zwischen zwei Zuständen. An den Kanten stehen die Bedingungen für den Wechsel in einen anderen Zustand. Der Zustandswechsel wird durch eine weitere Überwachungsebene, den s. g. *Monitor* (siehe 5.12), generiert, die den Zustand der Regelung bewerten kann. In diesem einfachen Beispiel werden nur Eingangssignale bewertet. Ist ein Eingang nicht mehr verfügbar, z. B. durch den Ausfall eines Sensors, erzeugt der Monitor ein Ereignis, das an den hybriden Statechart des Reglersystems weitergeleitet wird. Ein Beispiel ist das Signal *zAbsFail*, das in Abbildung 5.11 oben links zu sehen ist. Wenn der Statechart gerade im Zustand der Überblendung zwischen *Robust* $\rightarrow$ *SemiComfort* ist, muss beim Ausfall des Aufbausensors sofort auf *Robust* umgeschaltet werden. Das Signal *zAbsOk* führt wiederum zu einem Wechsel auf die Überblendung, die nach einer bestimmten Zeit $d^1_{low} \leq t_0 \leq d^1_{up}$ über den Zustandswechsel rechts zum Zustand *SemiComfort* führt.

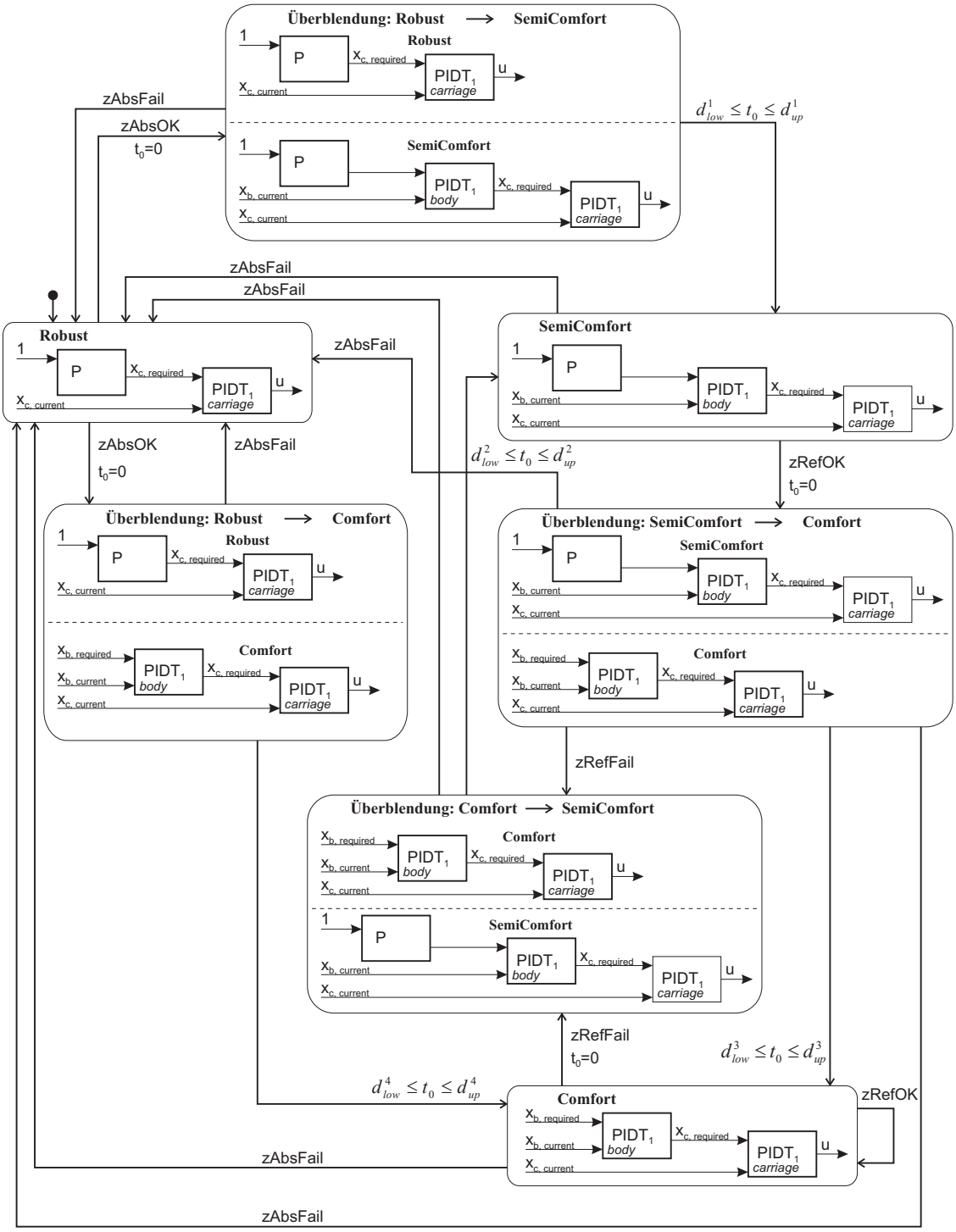

Abbildung 5.11: Hybride Sicht des Aufbaureglers mit Überblendung in vollständiger Darstellung

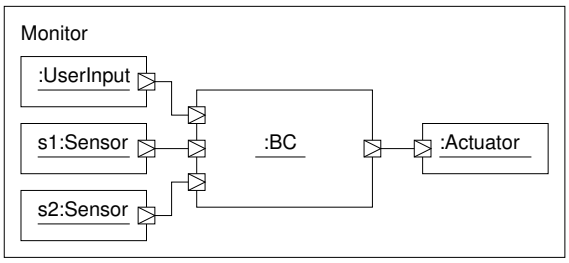

Abbildung 5.12: Struktur des Reglersystems als Monitor

In Abbildung 5.12 ist die übergeordnete Struktur des Reglersystems dargestellt. Aus Sicht der Reglerumschaltung ist dies die Ebene des *Monitors*, der den Gesamtzustand der Regelung überwacht und entsprechende Signale generiert. In Bezug auf das angesprochene Beispiel wird ein möglicher Sensorausfall auf dieser Ebene erkannt und in das entsprechende Signal (Ereignis) umgesetzt. Im oben betrachteten Fall ist dies das Signal *zAbsFail*.

Mögliche Verfahren zur Online-Überwachung der Reglerzustände und zur Reglerumschaltung im laufenden Betrieb finden sich in [DO00] und [DOM01]. Auch werden dort Probleme der Echtzeitverarbeitung näher diskutiert.

5.4.3 Hybride Statecharts

Das in Abbildung 5.11 dargestellte Statechart kann mit Hilfe einer abgeänderten Darstellung übersichtlicher dargestellt werden. Hierzu ist eine Erweiterung der bisherigen Darstellungsregeln für Statecharts notwendig. Diese Darstellung soll den Entwurf der Informationsverarbeitung für rekonfigurierbare Systeme unterstützen und dabei weiterhin die Möglichkeit bieten, entsprechende Prüfverfahren für die formale Richtigkeit des Statecharts anzuwenden. Diese Darstellungsform wird als *hybride Statecharts* bezeichnet [GBSO04, BGO04a, BGO04b] und insbesondere [Bur06].

Eine vollständige Beschreibung von hybriden Statecharts findet sich bei [Bur06]. Da diese Arbeit die hybriden Statecharts ausführlich aus informationstechnischer Sicht darstellt, soll an dieser Stelle das Thema nicht tiefer untersucht werden. Die Bedeutung für die Modellierung von rekonfigurierbaren Systemen und selbstoptimierenden Systemen wird aber als hoch eingeschätzt.

Abbildung 5.13 zeigt das in Abbildung 5.11 dargestellte Statechart als hybrides Statechart. Der Unterschied liegt in der Darstellung der Überblendvorgänge.

Da diese Überblendvorgänge grundsätzlich aus der Kombination zweier verbundener *statischer*, also zeitlich unbefristeter Zustände gebildet werden, können sie vereinfacht als *besondere* Kante zwischen zwei Zuständen beschrieben werden. Dies wird durch die fett ausgeprägten Pfeile ausgedrückt. Es handelt sich um Zustandswechsel, die eine bestimmte Zeit benötigen *(fading transitions)*. In dieser Zeit wird der Zustand a auf den Zustand b überblendet. Die übrigen Zustandswechsel sind wie bisher dünn gezeichnet. Sie können auch als Sonderfall der *fading transitions* betrachtet werden, die für einen Wechsel keine Zeit benötigen[28]. Diese Übergänge werden als *atomic transitions* bezeichnet.

Da für den primären Entwurf der Umschalt- und Überblendregeln als Statechart die statischen Zustände im Vordergrund stehen, ist diese Darstellung für den Entwurf

[28]Real wird Zeit benötigt. Da ein Wechsel idealerweise zwischen zwei kontinuierlichen Zeitschritten erfolgt, vergeht aus Sicht der kontinuierlichen Auswertung keine Zeit.

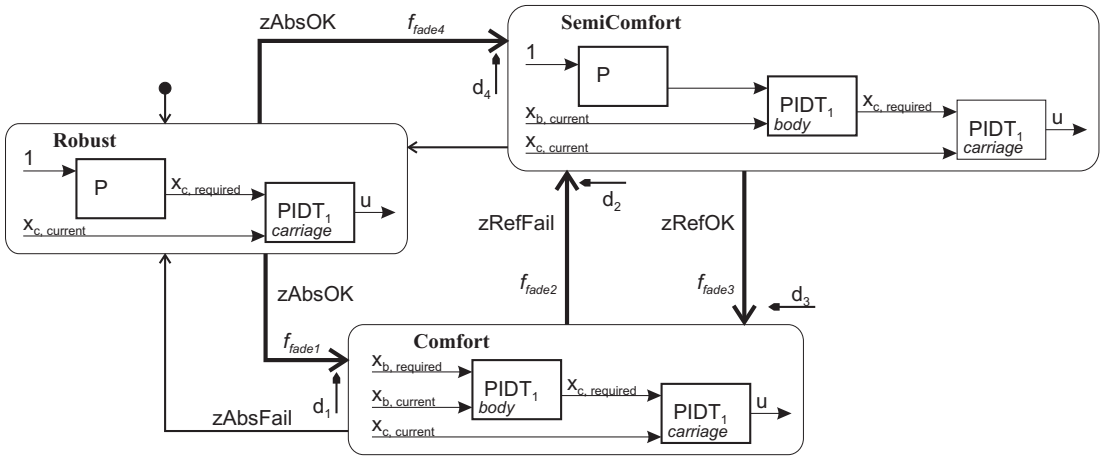

Abbildung 5.13: Hybride Sicht des Aufbaureglers mit Überblendung (Verhalten der Komponente)

von rekonfigurierbaren Systemen, insbesondere rekonfigurierbaren Reglersystemen in selbstoptimierenden Systemen, besonders zweckmäßig und überschaubarer.

5.4.4 Schnittstellen-Statecharts

Ein weiterer Schritt, bestehende Beschränkungen hybrider Systeme zu überwinden, ist das Konzept der hybriden Statecharts auf hierarchische Systemmodelle zu erweitern. Bei rekonfigurierbaren Systemen, wie beim Beispiel in Abschnitt 5.4.1, verändern Teile des kontinuierlichen Teilsystems die Anzahl der Ein- und Ausgänge. Um die Veränderung zu beschreiben, wird eine weitere Notation benötigt. Diese beschreibt die veränderlichen Schnittstellen eines rekonfigurierbaren Teilsystems und die dazugehörigen Zustände. Der Vorteil liegt vor allem darin, dass mit dem Zustandswechsel die Verkopplung der kontinuierlichen Ein- und Ausgänge geprüft werden kann. Dadurch kann die Korrektheit bezüglich offener Schnittstellen schon bei der Modellbildung geprüft werden.

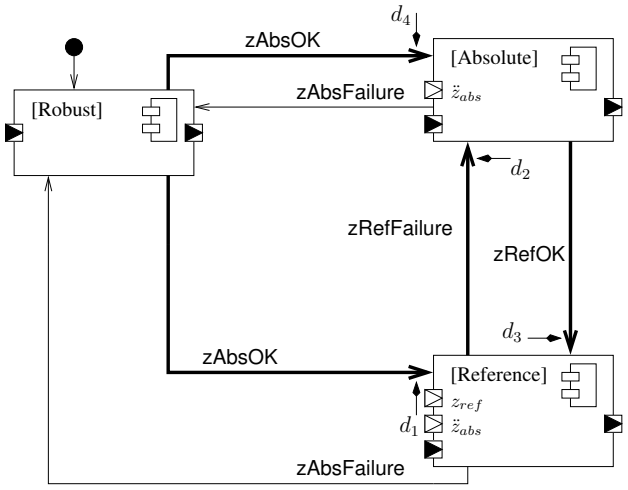

Abbildung 5.14: Interface-Statechart der Komponente Aufbauregler

Die Schnittstellen, die inneren Zustände und die Übergänge (Transitionen) zwischen den Zuständen des Systems können durch einen Schnittstellen-Statechart *(interface statechart)* beschrieben werden. Abbildung 5.14 zeigt einen solchen Automaten. Schnittstellen *(ports)*, die in allen Konfigurationen benötigt werden, sind hier schwarz dargestellt. Alle anderen, die nur in bestimmten Konfigurationen benötigt werden, sind weiß abgebildet. Für alle möglichen Zustandswechsel werden nur die außerhalb benötigten Informationen dargestellt.

Schnittstellen-Statecharts abstrahieren die Details eines Teilsystems. Dadurch ist es möglich, ein unterlagertes Teilsystem als *hybride Komponente* (vgl. Abschnitt 3.3) einzubauen, ohne Details des inneren Aufbaus zu benötigen[29]. Dadurch wird die Komplexität des Entwurfs auf die aktuelle Ebene, in der sich der Entwickler befindet, beschränkt [BGO04a]. Darüber hinaus kann das Modell auf Korrektheit aller Konfigurationsmöglichkeiten geprüft werden [BGO04a, Bur06]. Eine Prüfung des kontinuierlichen Anteils etwa auf Stabilität muss nach bisherigem Kenntnisstand für jede mögliche Konfiguration separat erfolgen.

[29] In Bezug auf das kontinuierliche Verhalten ist eine Betrachtung der inneren Dynamik eines Teilsystems für den Entwurf in vielen Fällen zwingend. Für die Modellierung der Zustandsmaschine gilt dies jedoch nicht.

Kapitel 6

Anwendungsbeispiele für Selbstoptimierung

> Wenn Du willst, dass die Leute ein Schiff bauen, dann zeige ihnen nicht, wie man Bäume fällt, sondern erzähle ihnen von der wunderschönen Welt auf der anderen Seite des Meeres (Antoine de Saint-Exupéry)

Die unterschiedlichen Aspekte der Selbstoptimierung aus den vorangegangenen Kapiteln sollen im Folgenden anhand verschiedener Beispiele skizziert werden. In Abschnitt 6.1 (Magnetbahn) werden Hierarchisierungsansätze, die für die Modellierung von selbstoptimierenden mechatronischen Systemen wichtig sind, gezeigt. Ein wichtiges Seitenthema ist dabei der Einsatz von Multirate-Verfahren für die Simulation und die Ausführung. Abschnitt 6.2 (Aktives Fahrwerk) geht auf die Verwendung von Verhaltensbasierung bei der Vorgabe von Steuerdaten ein. Das Zusammenspiel verschiedener Komponenten in einer verteilten Optimierung wird in Abschnitt 6.3 (Trajektorienoptimierung bei schienengebundenen Fahrzeugen) dargestellt.

6.1 Ein Beispiel für Hierarchisierung und Multirate: Magnetbahn

Die Kapitel 4, Numerische Simulation und Ausführung von modularen Systemen, und 5, Informationsverarbeitung – Entwurf und Implementierung, beschäftigten sich mit Verfahren zur Modularisierung und verteilten Multirate-Simulation und -Ausführung von selbstoptimierenden mechatronischen Systemen. In diesem Abschnitt sollen anhand des Beispielmodells einer Magnetbahn diese Themenbereiche näher untersucht werden. Das ausgewählte Beispiel ist im Wesentlichen an einem Beispiel aus der Vorlesung *Grundlagen der Regelungstechnik* von Prof. Dr.-Ing. Lückel [Lüc02] angelehnt. Aufgrund seiner klaren Struktur und guten Verständlichkeit wurde es als Beispiel für die Darstellung von angewandten Multirate-Verfahren und des Ansatzes zur Hierarchisierung gewählt. In [Vöc03] wurden darüber hinaus an diesem Anwendungsbeispiel noch *Umschaltstrategien* untersucht, die für den Einsatz in rekonfigurierbaren Systemen verwendet werden können. Weitere Arbeiten beschäftigten sich mit grundsätzlichen Fragen zur Theorie und zur Anwendung von Multirate [VO04], [OV04].

6.1.1 Modellierung

Das Gesamtsystem *Magnetbahn* besteht im Wesentlichen aus zwei hierarchisch verkoppelten Teilen: dem Aufbau und dem Schlitten. Der Schlitten wird über einen Elektromagneten geführt, der unter der Schiene angebracht ist. Die Kraft des Magneten, der den Schlitten und den Aufbau trägt, hängt zum einen vom Strom ab, der durch die am Magneten angelegte Spannung induziert wird, zum anderen aber auch von der Größe des Luftspaltes zwischen Schiene und Schlitten. Um den Fahrkomfort zu erhöhen, ist der Aufbau über ein Feder-/Dämpfersystem an den Schlitten gekoppelt. Die Gesamtregelung besteht aus einer zweistufigen Kaskade. Auf unterer Ebene wird zunächst die Größe des Luftspaltes geregelt, die möglichst konstant gehalten werden soll. Außerdem dient eine übergeordnete Regelung dazu, über die Größe des Luftspaltes Einfluss auf die Aufbaubewegungen und -beschleunigungen zu nehmen. Als Anregung dient im Modell eine Störkraft, die von außen auf den Aufbau wirkt.

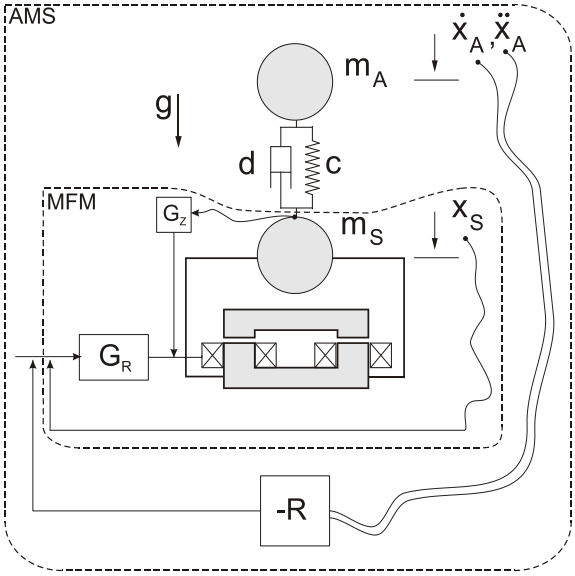

Abbildung 6.1: Modular-hierarchische Struktur des Gesamtmodells mit Regelung

In Abbildung 6.1 ist das modular-hierarchische Modell des Gesamtsystems mit der zugehörigen Regelung dargestellt. Daran lässt sich eine Einteilung in das unterlagerte MFM Schlitten und das umfassende AMS Aufbau, das im Gegensatz zum MFM keine eigene Aktorik besitzt, erkennen. Beide Teilsysteme verfügen jedoch über eine eigene mechanische Tragstruktur und Komponenten zur Sensorik und zur Informationsverarbeitung.

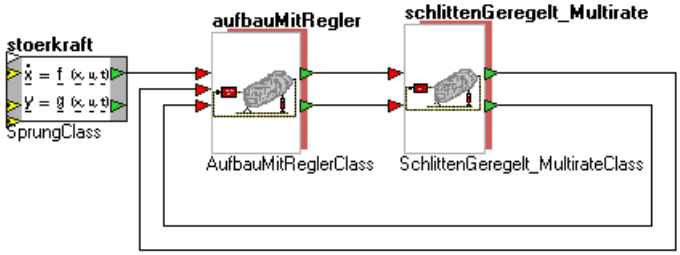

Abbildung 6.2: Gesamtmodell der Magnetbahn in CAMeL

Abbildung 6.2 zeigt das zugehörige, mit dem Programm CAMeL-View [iXt06] erstellte Modell. Die untergeordneten Teilmodelle werden im Folgenden näher erläutert.

6.1.1.1 MFM *Schlitten*

Das MFM *Schlitten* besteht aus drei Hauptkomponenten: der Schlittenmasse m_s, dem Elektromagneten und der Spaltregelung G_R zur Vorgabe eines gewünschten Spaltabstands $x_{s,soll}$. Zur Verbesserung des Einschwingverhaltens wird die Kraft F_{cd}, die über das Feder-Dämpfer-Element auf den Schlitten wirkt, über eine Störgrößenrückführung G_Z hinter dem Regler aufgeschaltet. Das zugehörige Modell in CAMeL-View ist in Abbildung 6.3 zu sehen.

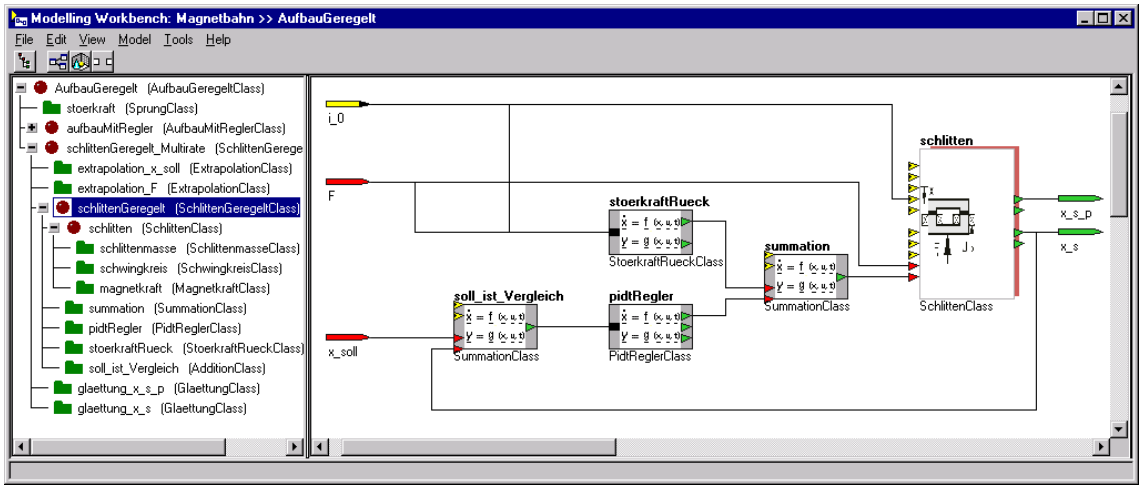

Abbildung 6.3: CAMeL-Modell des MFM Schlitten

Die mathematischen Beschreibungen der Schlittenmasse und des Elektromagneten wurden aus Gründen der Übersichtlichkeit in einem eigenen Block zusammengefasst. Der Block besteht aus den Bewegungsgleichungen der Masse, der Berechnung der Magnetkraft und den Gleichungen zur Abbildung des elektrischen Schwingkreises. Das zugehörige Modell dieses Subsystems wird in Abbildung 6.4 gezeigt.

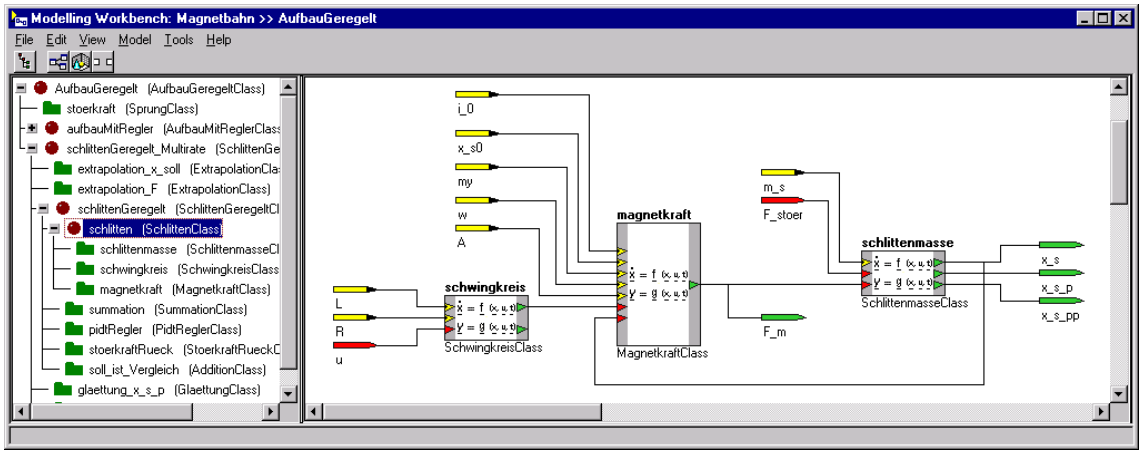

Abbildung 6.4: Modell der mechanischen und der elektromagnetischen Elemente

Die Kraft, die im Magnetspalt wirkt, lässt sich aus der im Magnetfeld gespeicherten Energie ermitteln:

Zeichen	Bezeichnung	Einheit	Zahlenwert
x_S	Schlittenposition	m	
F_m	Magnetkraft	N	
i	Stromstärke	A	
u	Spannung	V	
m_S	Schlittenmasse	kg	500.0
μ	Permeabilitätskonstante	Vs/Am	0.001256
A	Spulenquerschnitt	m^2	0.25
w	Windungszahl		46
L	Induktivität	Vs/Am	10.0
R	Ohmscher Widerstand	Ω	1000.0
g	Erdbeschleunigung	m/s^2	9.81

Tabelle 6.1: Modellgrößen und Parameter im MFM Schlitten

$$F_m\left(x,t\right) = k_f \cdot \frac{i^2\left(t\right)}{x^2\left(t\right)} \text{ mit } k_f = \frac{\mu w^2 A}{2} \tag{6.1.1}$$

Die Eigenschaften des elektrischen Schwingkreises lassen sich mit Hilfe der Kirchhoffschen Maschengleichung, angewandt auf die Induktivität L und den Widerstand R des Elektromagneten, abbilden:

$$L \cdot \dot{i}\left(t\right) + R \cdot i\left(t\right) = u(t) \tag{6.1.2}$$

Nach den Sätzen von Newton-Euler lässt sich der mechanische Teil darstellen. Dabei wird die Erdanziehung berücksichtigt. Eine aus dem Aufbau stammende mögliche Störkraft $F_{c,d}$ wirkt ebenfalls ein:

$$m_s \cdot \ddot{x}_s = F_{c,d} + m_s \cdot g - F_m \tag{6.1.3}$$

Aus der statischen Ruhelage des Systems Elektromagnet mit $\ddot{x}_s = \dot{x}_s = x_s = \dot{x}_A = 0$ und der Voraussetzung $F_{c,d0} = 0$ ergibt sich die erforderliche Spannung U_0. Die Magnetkraft ergibt sich durch:

$$F_{m0} = m_s \cdot g \tag{6.1.4}$$

Die elektrische Spannung U_0 kann damit berechnet werden:

$$F_{m0} = k_f \cdot \frac{i^2(t)}{x^2(t)} \text{ mit } k_f = \frac{\mu w^2 A}{2} \text{ und } x = x_{S0} - x_s = x_{S0} \tag{6.1.5}$$

x_{S0} stellt hier die Größe des Luftspalts im Ruhezustand des Schlittens dar. Weiter gilt:

$$i = x \cdot \sqrt{\frac{F_{m0}}{k_f}} \tag{6.1.6}$$

$$L \cdot \dot{i}\left(t\right) + R * i\left(t\right) = U(t) \text{ mit } \dot{i}\left(t\right) = 0 \tag{6.1.7}$$

was zur Spannung U_0 führt mit:

$$U_0 = R \cdot i = x \cdot R \cdot \sqrt{\frac{F_{m0}}{k_f}} \tag{6.1.8}$$

Durch Linearisierung ergeben sich folgenden linearen Gleichungen des Systems:

$$m_S \cdot \triangle \ddot{x_S} = -\triangle F_m + \triangle F_{c,d} \tag{6.1.9}$$

$$\triangle F_m = 2k_f \cdot \frac{i_0}{x_{S0}^2} \cdot \triangle i - 2k_f \cdot \frac{i_0^2}{x_{S0}^3} \cdot \triangle x_S = k_i \cdot \triangle - k_S \cdot \triangle x_S \tag{6.1.10}$$

$$L \cdot \triangle i = -R \cdot \triangle i + \triangle u \tag{6.1.11}$$

Nach der Transformation in den Laplace-Bereich ergeben sich die Gleichungen mit:

$$\triangle X_S(s) = -\frac{\triangle F_m(s)}{m_S \cdot s^2} + \frac{\triangle F_{c,d}(s)}{m_S \cdot s^2} \tag{6.1.12}$$

$$\triangle F_m(s) = k_i \cdot \triangle I(s) - k_S \cdot \triangle X_S(s) \tag{6.1.13}$$

$$\triangle I(s) = \frac{1}{L \cdot s + R} \cdot \triangle U(s) \tag{6.1.14}$$

Zur Regelung des Spaltabstandes dient ein $PIDT_1$-Regler. Das Verhalten des geregelten Systems gegenüber einer Störung lässt sich durch die Störgrößenrückführung G_Z noch verbessern. Aus den Laplace-transformierten der Gleichungen 6.1.12 bis 6.1.14 ergibt sich:

$$\triangle X_S(s) = \underbrace{-\frac{k_i}{(m_S - k_S \cdot s^2)(Ls + R)}}_{G_1(s)} \triangle U(s) + \underbrace{\frac{1}{(m_S - k_S \cdot s^2)}}_{G_2(s)} \triangle F_{c,d}(s) \tag{6.1.15}$$

Um den Einfluss der Störgröße $\triangle F_{c,d}$ auf $\triangle X_S$ zu kompensieren, wird sie über G_Z zurückgeführt und hinter dem Regler G_R aufgeschaltet, wie in Abbildung 6.1 dargestellt. Für eine ideale Störgrößenaufschaltung gilt dann:

$$\begin{aligned} \tilde{G}_Z(s) &= G_2(s) G_1^{-1}(s) \\ &= \frac{1}{(m_S - k_S) \cdot s^2} \frac{(m_S - k_S \cdot s^2)(Ls + R)}{k_i} \\ &= \frac{Ls + R}{k_i} \end{aligned} \tag{6.1.16}$$

Diese Übertragungsfunktion ist nicht realisierbar, da der Zählergrad höher als der Nennergrad ist. Dieses Problem wird durch die Einführung einer zusätzlichen Nennerzeitkonstante T_1 behoben. Daraus ergibt sich:

$$G_Z(s) = \frac{Ls + R}{(T_1 s + 1) k_i} \tag{6.1.17}$$

6.1.1.2 AMS *Aufbau*

Das dem Modell des Schlitten überlagerte Modell des Aufbaus besteht aus drei Komponenten: der Aufbaumasse, dem Feder-Dämpfer-System und dem unterlagerten MFM Schlitten selbst. Das Verhalten des Schlittens wird vereinfacht als ideal angenommen, um Modellierung und Auslegung zu erleichtern. Zwischen $xS, soll$ und x_S wird deshalb ein proportionaler Zusammenhang vorausgesetzt. Die Regelung erhält durch die Verstellung des Fußpunktes Einfluss auf das Aufbauverhalten. In Abbildung 6.5 ist das mechanische Teilsystem zu sehen.

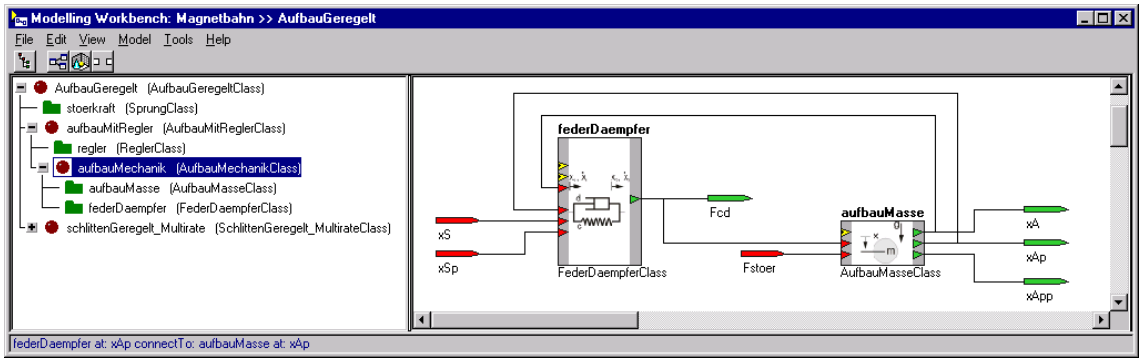

Abbildung 6.5: Mechanisches Teilsystem des AMS Aufbau

Durch die Anwendung der Sätze von Newton-Euler lassen sich auch hier wieder die Gleichungen zur Beschreibung des mechanischen Systems ableiten. Die Tabelle 6.2 zeigt die zugehörigen Größen und Parameter des AMS Aufbau.

x_A	Position des Aufbaus	[m]	
$F_{c,d}$	Feder-Dämpfer-Kraft	[N]	
F_Stoer	Störkraft von außen	[N]	
m_A	Aufbaumasse	[kg]	5000.0
c	Federkonstante	[N/m]	100000.0
d	Dämpfungskonstante	[Ns/m]	12000.0
g	Erdbeschleunigung	[m/s]	9.81

Tabelle 6.2: Modellgrößen und Parameter im AMS Aufbau

$$m_A \cdot \ddot{x}_A = -F_{c,d} + m_A \cdot g + F_{stoer} \tag{6.1.18}$$

mit der Feder-Dämpfer-Kraft $F_{c,d}$:

$$F_{c,d} = c \cdot (x_A - x_S) + d \cdot (\dot{x}_A - \dot{x}_S) + F_{c,d0} \tag{6.1.19}$$

Die Federvorspannung $F_{c,d0}$ ergibt sich aus der statischen Ruhelage des Systems. Die Gleichungen 6.1.18 und 6.1.19 ergeben:

$$m_A \cdot \ddot{x}_A = -c \cdot (x_A - x_S) - d \cdot (\dot{x}_A - \dot{x}_S) - F_{c,d0} - m_A \cdot g + F_{Stoer} \tag{6.1.20}$$

Aus der Voraussetzung:

$$\ddot{x}_A = \dot{x}_A = x_A = \dot{x}_s = x_s = F_{Stoer} = 0 \tag{6.1.21}$$

folgt:

$$F_{c,d0} = m_A \cdot g \tag{6.1.22}$$

Die statische Kraft $F_{c,d0}$ wird benötigt, um die Gewichtskraft des Aufbaus im unterlagerten MFM Schlitten zu berücksichtigen. Die notwendige Magnetkraft im Ruhezustand F_{m0} aus Gleichung 6.1.4 ergibt sich als:

$$F_{m0} = m_S \cdot g + m_A \cdot g \tag{6.1.23}$$

Die benötigte elektrische Spannung U_0 lässt sich nach Gleichung 6.1.8 bestimmen.

Analog zu den linearisierten Gleichungen 6.1.9 bis 6.1.11 des MFM *Schlitten* werden die Gleichungen zur Beschreibung des Aufbauverhaltens abgeleitet. Die Abweichungen der Ruhelage lassen sich beschreiben als:

$$m_A \cdot \triangle\ddot{x}_A = -\triangle F_{c,d} + \triangle F_{Stoer} \tag{6.1.24}$$

$$\triangle F_{c,d} = c \cdot (\triangle x_A - \triangle x_S) + d \cdot (\triangle\dot{x}_A - \triangle\dot{x}_S) \tag{6.1.25}$$

Aufbaubeschleunigung $\ddot{x}_A$ und -geschwindigkeit $\dot{x}_A$ werden über die Matrix R zurückgeführt, wie auch Abbildung 6.1 zeigt. Die Aufschaltung des Wertes für $x_{S,soll}$ dient zur Beeinflussung der Aufbaudynamik. Dies geschieht indirekt mit Hilfe des unterlagerten MFM, da das AMS *Aufbau* keine eigene Aktorik besitzt. Die Regelung des Aufbaus kann beispielsweise zur Verbesserung des Komforts genutzt werden.

6.1.2 Multirate-Integration

Nach der Beschreibung des Modells soll nun exemplarisch die Multirate-Integration dargestellt werden. Besonderes Augenmerk liegt dabei auf den Eigenwerten der unverkoppelten Teilsysteme, die sich in dem gezeigten Modell stark unterscheiden, wie die Tabellen 6.3 und 6.4 zeigen.

Re	Im
-5.56906	3.43737
-24.0792	190.646
-200.000	0.00000
-1040.70	0.00000

Tabelle 6.3: Eigenwerte des MFM *Schlitten*

Re	Im
-1.20000	4.30813

Tabelle 6.4: Eigenwerte des AMS *Aufbau*

Diese Unterschiede zeigen viele (große) dynamische Systeme, worauf bereits in [Rük96] hingewiesen wurde. Verantwortlich für die Unterschiede sind beispielsweise große Massendifferenzen, steife Federn oder hochfrequente Regelungen.

Werden die beiden Teilsysteme Aufbau und Schlitten verkoppelt, verschieben sich die Pollagen des Gesamtsystems. In dem betrachteten Beispiel sind die Verschiebungen allerdings klein genug, um noch die Zugehörigkeit der Eigenwerte zu

den Teilsystemen erkennen zu lassen. Die Eigenwerte des Gesamtsystems und ihre Zuordnung finden sich in Tabelle 6.5.

Re	Im	Teilsystem
-1.20242	4.30785	Aufbau
-5.10195	4.10389	Schlitten
-24.4388	226.545	Schlitten
-205.551	0.00000	Schlitten
-1059.36	0.00000	Schlitten

Tabelle 6.5: Eigenwerte des Gesamtsystems und ihre Zuordnung

Wie die Tabelle 6.5 zeigt, besitzt der Aufbau das Eigenwertepaar mit der geringsten Eigenfrequenz. Da die Aufbaumasse die größte Trägheit besitzt, war dies zu erwarten. Die Eigenwerte der Schlittenmassen und der elektrischen Komponenten besitzen höhere Eigenfrequenzen. Die Unterschiede zeigen, dass hier der Einsatz von Multirate sinnvoll ist. Mit Hilfe der Erweiterungen von CAMeL-View aus [Gam02] wird eine IPANEMA-Applikation mit zwei Calculator-Objekten erstellt. Die Calculatoren arbeiten mit unterschiedlichen Schrittweiten entsprechend dem Multirate-Ansatz.

6.1.3 Modellerweiterungen für Multirate

Für die Realisierung von Multirate ist nach Abschnitt 4.4 der Einsatz von Filtern für Glättung und Extrapolation notwendig. Sind diese Filter nicht in der Laufzeitumgebung (z. B. IPANEMA) vorhanden, müssen sie mit in das Modell implementiert werden. Nach Abschnitt 4.4 werden diese Filter im schnellen Teilsystem ergänzt, um Ein- und Ausgangsdaten in jedem Zwischenschritt zu glätten bzw. zu extrapolieren. Wie Abbildung 6.2 zeigt, besitzt das MFM *Schlitten* sowohl zwei zu extrapolierende Eingangs- als auch zwei zu glättende Ausgangsgrößen. Das Modell wurde mit entsprechenden Filtern erweitert. Die erweiterte Struktur ist in Abbildung 6.6 zu sehen.

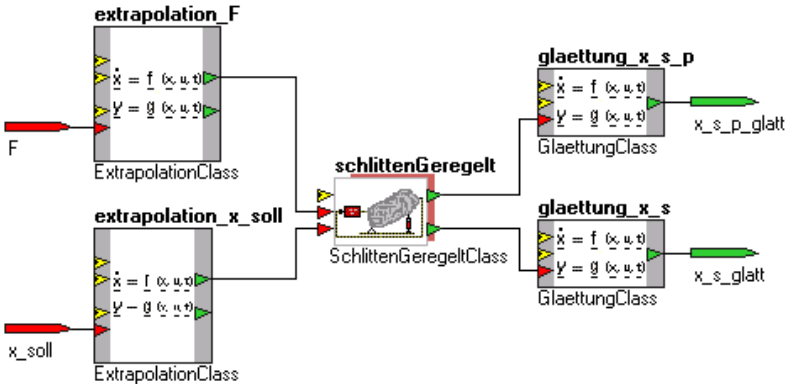

Abbildung 6.6: Erweiterung des MFM *Schlitten* durch Filter

Für die Simulation wurde in diesem Fall zur Integration das explizite Euler-Verfahren gewählt, auch, um das Verhalten im Echtzeit-Fall abschätzen zu können. Da das Euler-Verfahren die geringste Genauigkeit pro Schritt aufweist, eignet es sich

gut als *Worst-Case-Fall*. Für die modulare Multirate-Simulation müssen die höherwertigen Verfahren in IPANEMA angepasst werden. Hierbei wäre eine Integration der entsprechenden Glättungs- und Extrapolationsverfahren wünschenswert. Die für die dargestellte Simulation eingesetzten Filter entsprechen den in Abschnitt 4.4.1 und 4.4.2 erläuterten Methoden nach [Blu78].

6.1.4 Simulationsergebnisse

In einem ersten Schritt wurden beide Teilsysteme separat simuliert, um real geeignete Schrittweiten der Einzelsysteme abzuschätzen. Für das langsamere Teilsystem *Aufbau* erwies sich eine Schrittweite von $h = 0.001s$ als günstig, für das schnellere Teilsystem *Schlitten* war eine kleinere Schrittweite von $h = 0.0001s$ notwendig. Werden beide Teilsysteme mit der größeren Schrittweite $h = 0.001s$ simuliert, wird das Gesamtsystem instabil. Mit der Schrittweite $h = 0.0001s$ ist eine stabile Simulation möglich. Die Simulationsergebnisse beider Fälle für den Verlauf von x_S sind in Abbildung 6.7 dargestellt. Als Anregungsfunktion diente eine auf den Aufbau wirkende sprungförmige Störkraft F_{Stoer} bei $t = 0.1s$.

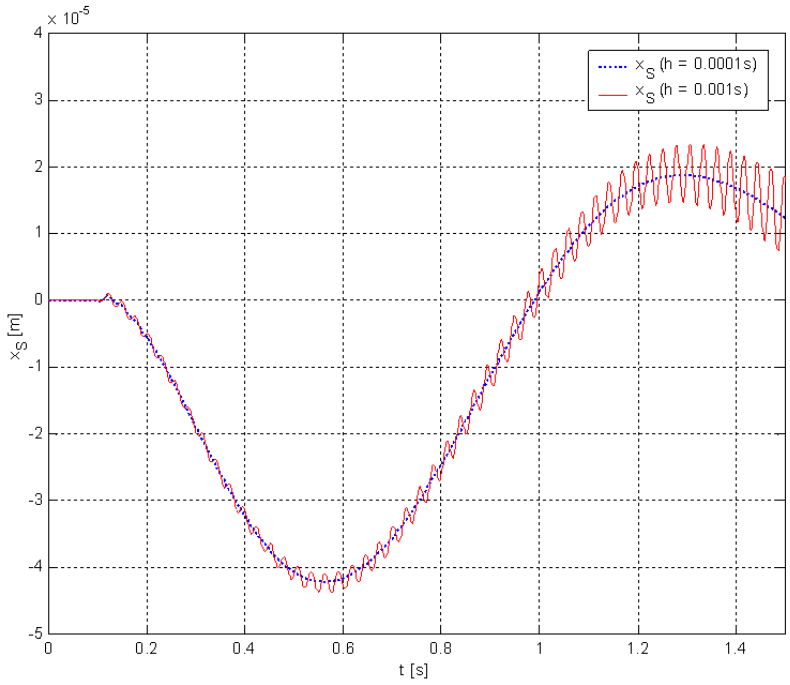

Abbildung 6.7: Simulation mit gemeinsamer Schrittweite für beide Teilsysteme

Der nächste Schritt ist der Einsatz unterschiedlicher Berechnungsschrittweiten für beide Teilsysteme. Die gewählten Schrittweiten entsprechen denen der separat simulierten Teilsysteme, und zwar $h = 0.0001s$ für das schnelle MFM *Schlitten* und $H = 0.001s$ für das langsamere AMS *Aufbau*. Das Ergebnis der Simulation für die Spaltbreite x_S ist in Abbildung 6.8 zu sehen. Dabei zeigt die durchgezogene Linie den Verlauf für die Simulation mit Multirate (unterschiedliche Schrittweiten) und die gepunktete Linie den Verlauf ohne Multirate (nur die kleinere Schrittweite $h = 0.0001s$ für beide Teilsysteme). Obwohl das langsame System im Multiratefall nur bei jedem zehnten Teilschritt des schnellen Systems ausgewertet wird, ist der Unterschied zwischen beiden Simulationsverläufen kaum erkennbar.

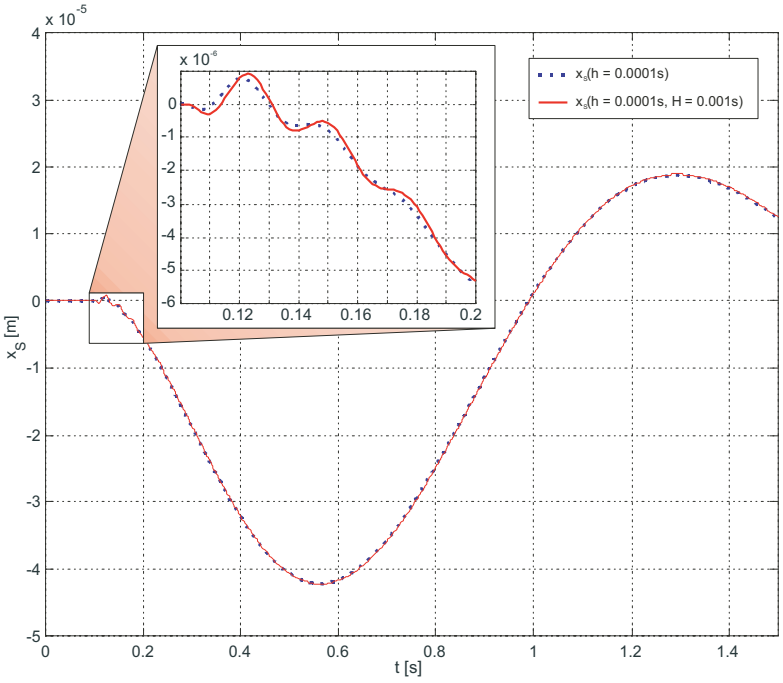

Abbildung 6.8: Verlauf der Spaltbreitenänderung bei Multirate-Integration

Die Unterschiede zeigen sich erst in der vergrößerten Darstellung im Ausschnitt der Abbildung 6.8. Der Einfluss des zusätzlichen numerischen Fehlers wird jetzt sichtbar. Er ist jedoch so gering, dass auch das Multirate-System stabil arbeitet.

Der simulierte Verlauf der Aufbaubeschleunigung in Abbildung 6.9 zeigt ein ähnliches Bild. Auch die Größe x_A unterscheidet sich im Ergebnis bei Auswertung der beiden Teilsysteme mit unterschiedlicher Schrittweite h (durchgezogene Linie) nicht wesentlich von dem mit gemeinsamer Schrittweite berechneten Verlauf (gepunktete Linie).

Die Ergebnisse zeigen, dass sich die in Kapitel 4 beschriebenen Multirate-Verfahren auf das verwendete Beispielsystem anwenden lassen. Die Verfahren zur Glättung und zur Extrapolation der Ein- und Ausgangsdaten zeigen ebenfalls die zu erwartenden Ergebnisse. Die Anwendung des einfachen Eulerverfahrens unterstreicht noch einmal die Robustheit und die Übertragbarkeit auf die Echtzeit-Simulation bzw. Hardware-in-the-Loop. Für Simulationsaufgaben sollten höherwertige Verfahren wie Runge-Kutta 4. Ordnung noch bessere Ergebnisse liefern.

Das gewählte Beispiel hat eine klassische hierarchische (Regler-)Struktur, so dass die Ergebnisse auf andere komplexere Systeme übertragbar sind, die sich im Wesentlichen nur quantitativ durch die Anzahl der Ein- und Ausgänge und der Hierarchiestufen unterscheiden. Besonders attraktiv ist im Gesamtbild, dass der Schnitt der Teilsysteme, also die aus dem technischen Kontext heraus entstandene Modularisierung, ohne weiteres auf die modulare Multirate-Simulation zu übertragen ist. Im Sinne der Selbstoptimierung bedeutet dies, dass eine modulare Rekonfiguration auch im Multirate-Fall möglich und vor allem notwendig ist. Weiterführende Untersuchungen sind für eine automatisierte Bestimmung der Fehlerabschätzung sinnvoll, um im Falle der Rekonfiguration die Qualität des Ergebnisses kontrollierbar zu machen. Darüber hinaus bleibt offen, wie Schrittweiten automatisch bestimmt werden können. Für die Überwachung solcher Systeme bieten sich Verfahren nach [DO00] und [DOM01] an.

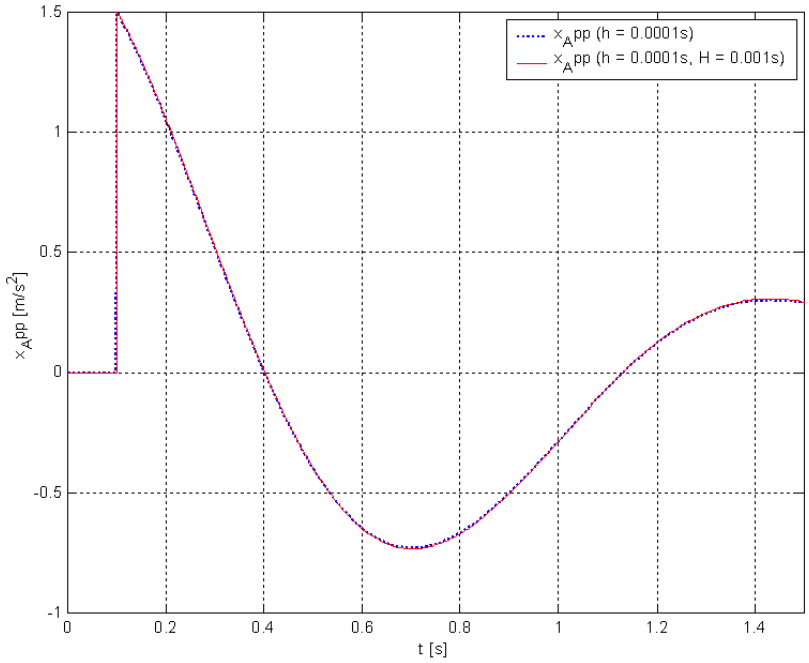

Abbildung 6.9: Verlauf der Aufbaubeschleunigung bei Multirate-Integration

6.2 Ein Beispiel für Verhaltensbasierung: Aktives Fahrwerk

Verhaltensbasierte Verfahren eignen sich vor allem für die Steuerung der Sollgrößen, der eine Regelung folgen soll. Das folgende Beispiel skizziert die Ergebnisse aus [OHKK02], [HO03], [MOH+04].

Aktive Fahrwerke spielen in der Fahrzeugtechnik einen zunehmende Rolle. Vor allem in den letzten Jahren wurden aktive Fahrwerke vor allem in Sonderfahrzeugen eingesetzt, wie Land- und Forstmaschinen, Gabelstapler, Geländefahrzeuge [SJW06] etc. Aber auch in PKWs der Oberklasse finden sich heute aktive Fahrwerke, z. B. *Dynamic Drive* der Firma BMW oder *Active Body Control* der Firma Daimler. Neue Entwicklungen bringen aktive Fahrwerke in schienengebundene Fahrzeuge, wie beim schwedischen Neigetechnikzug X2000, dem italienischen Pendolino ETR 460/ETR 470, dem Neigetechnikzug ICE-T (z. B. [Wer07]) oder dem RailCab. Beim RailCab ist das aktive Fahrwerk zu einem integrierten Konzept zur Kontrolle der Aufbaudynamik geworden [LHLJ00a, Neu06], das, neben dem sekundären Federungssystem des *Mechatronic Bogie* [Bom07] der Firma Bombardier, eine der wenigen Lösungen mit mehr als nur Kompensation der Kurvenneigung darstellt.

6.2.1 Aufgabenstellung

Die Hauptaufgabe von aktiven Fahrwerken mit vertikaler Aktorik ist die Entkopplung der Aufbaubewegung von der Straße oder dem Fahrweg. Idealerweise bleibt der Aufbau in vertikaler Richtung völlig ruhig; das Fahrwerk kompensiert jedwede störende Anregung durch die Strecke[LCJR92]. Da jedoch der Fahrweg – sei es Straße oder Schiene – nicht horizontal verläuft und die Höhenunterschiede weit größer sind

als die Verstellwege der Fahrwerke, kommt es zwangsläufig zu einem Zielkonflikt zwischen *Folgen des Fahrweges* und *Reduzierung der Aufbaubeschleunigung* [Hes00].

Die meisten Ansätze gehen zunächst von einer analytischen Betrachtung der vorliegenden Anregungsfälle aus. Betrachtet man das Problem jedoch aus einer abstrakten, vereinfachten Sichtweise heraus, so kann das Fahrzeug auch als ein *autonomes Objekt* modelliert werden, das durch eigenes Verhalten auf seine Umgebung reagiert. Ähnlich wie ein Skifahrer, der durch die Bewegung der Beine die Unebenheiten der Piste ausgleicht und so den Oberkörper ruhig hält, aber bei zu großer Auslenkung zunehmend dem Streckenverlauf folgt, lässt sich dieser Ansatz auf die Modellierung eines Fahrwerkes übertragen.

Fahrzeugmodell Eine Möglichkeit für die Modellierung eines einfachen Fahrzeugmodells ist das *Viertelfahrzeug*. Dabei wird vereinfacht angenommen, dass das Fahrzeug in zwei Achsen symmetrisch ist. Bei einem vierrädrigen Fahrzeug ergibt sich dabei jeweils ein Viertel des Fahrzeugs, bestehend aus der anteiligen Aufbaumasse, dem Feder-Dämpfer-System zwischen Aufbau und Lenker, der Radmasse (inkl. Radaufhängung) und dem Reifen als Feder gegenüber der Fahrbahn.

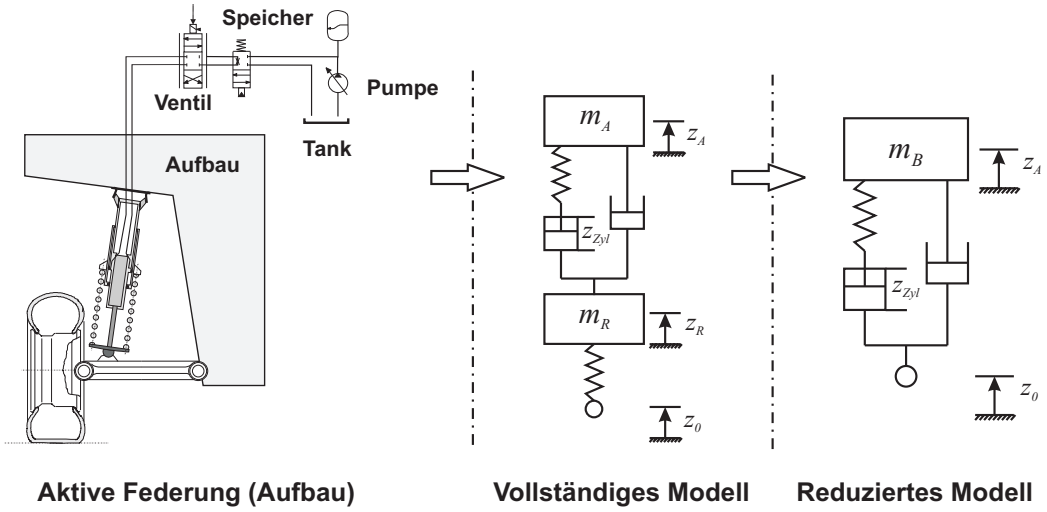

Abbildung 6.10: Entwicklung eines Viertelfahrzeugmodells

Abbildung 6.10 [OHKK02], [HO03] zeigt links den beschriebenen schematischen Aufbau eines Viertelfahrzeugs. Bei einem aktiven Fahrwerk besteht das Federbein aus einem Öldämpfer und einer Kombination aus Feder und Zylinder. Über ein Ventil kann mit Hilfe einer Pumpe kontrolliert Öl in den Zylinder gefördert werden. Damit kann die Feder vorgespannt werden. Die erzeugte, zusätzliche Kraft wirkt zwischen Aufbau und Querlenker zusätzlich zu den passiven Kräften, die im Feder-Dämpfer-System wirken. In erster Näherung wird davon ausgegangen, dass die Verschiebung des Zylinders (z_{Zyl}) direkt vorgegeben werden kann. Daraus ergibt sich der Ansatz, das Fahrzeugmodell als ein passives Fahrzeug mit einer zusätzlichen parallelen aktiven Kraft zu modellieren.

Eine weitere Vereinfachung führt zu dem Ersatzmodell, das in Abbildung 6.10 Mitte dargestellt ist. Dabei wurden die Lenkerkinematik vereinfacht und die Aufbaumasse über der Radmasse angeordnet. Um den Einfluss der Lenker zu berücksichtigen, wird ein Übersetzungsfaktor bei Federsteifigkeit und Dämpfung eingeführt. Die

Anregung durch die Straße (z_0) wird in diesem Modell über Radfeder und Radmasse auf das Feder-Dämpfer-System übertragen, das mit der Aufbaumasse verbunden ist. Dies erzeugt eine Rad- und Aufbaubewegung in vertikaler Richtung (z_R, z_A).

Wird weiterhin ein lineares Übertragungsverhalten aller Komponenten angenommen, so kann der Zusammenhang zwischen Straßenanregung und Aufbaubewegung mit Hilfe einer Übertragungsfunktion von z_0 nach z_A dargestellt werden. Das resultierende Modell 4. Ordnung besteht aus einer Radmasse und einer Aufbaumasse, die ein Verhältnis von ca. 1:10 aufweisen[30]. Für die Regelung der Aufbaudynamik kann deshalb die Radmasse vernachlässigt werden. Dies führt zu einem noch einfacheren Modell, das in Abbildung 6.10 rechts dargestellt ist. In diesem Modell regt die Straße direkt den Fußpunkt des Feder-Dämpfer-Systems an.

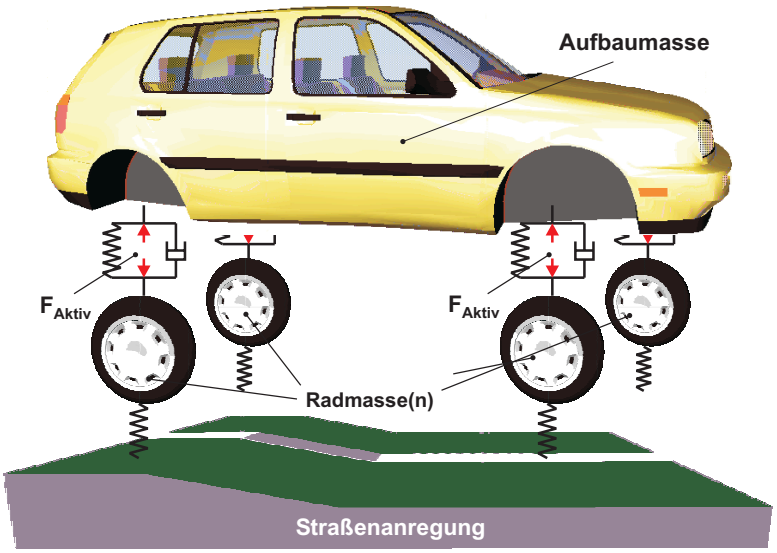

Abbildung 6.11: Starrkörpermodell mit Rädern

Die Kriterien, die für die Auslegung von passiven Fahrwerken gelten, sind auf die Auslegung von aktiven Fahrwerken ebenfalls anwendbar. Es gilt, Fahrkomfort, i. A. durch eine bewertete Insassenbeschleunigung [Ver87], und die Fahrsicherheit durch die dynamische Radlast zu optimieren. Jedoch lässt sich kein allgemeines Optimum finden, da die Kriterien einander widersprechen. Durch die Reduktion des Fahrzeugsmodells auf die Aufbaumasse nach Abbildung 6.10 rechts, entfallen die Effekte der Radmasse, und die Kriterien widersprechen einander in diesem einfachen Fall nicht mehr. Jedoch bleibt ein Zielkonflikt erhalten: Der Aufbau muss aufgrund des begrenzten Stellweges des Fahrwerks der Straße folgen, soll aber gleichzeitig aus Gründen des Komforts möglichst ruhig gehalten werden. Das Problem tritt insbesondere dann besonders anschaulich auf, wenn das Fahrzeug eine Rampe hinauf oder hinab fahren soll: Im ersten Fall neigt das Fahrzeug dazu, aufzusetzen, da die Regelung versucht, den Aufbau auf dem bisherigen Niveau zu halten, im anderen Fall fährt das Fahrwerk bis zum Ende des Stellwegs hin aus. Für passive Fahrwerke ist dies im Allgemeinen nach dem *Durchfedern* kein Problem. Der Fehler entsteht durch den Versuch der Ausregelung der absoluten Aufbaubewegung. Dieser Regelungsansatz wird auch häufig als *Skyhook* bezeichnet (z. B. [Mit97]).

Für die Modellierung eines Gesamtfahrzeugs lassen sich die oben gemachten Vereinfachungen auf ein Gesamtfahrzeugmodell übertragen. In Abbildung 6.11 [HO03]

[30] z. B. 30 kg Radmasse und 300 kg anteilige Aufbaumasse.

ist ein solches vereinfachtes Gesamtfahrzeugmodell schematisch dargestellt. Der Aufbau ist als ein Starrkörper modelliert. Die Radmassen können wie in Abbildung 6.10 Mitte mitmodelliert oder nach Abbildung 6.10 rechts weggelassen werden.

6.2.2 Verhaltensbasierter Ansatz

Ausgangspunkt der Überlegung ist zunächst, dass zwischen zwei Extremen bei der Auslegung der Regelung unterschieden werden muss:

1. Die Dämpfung des Aufbaus zur Umgebung ist optimal. Das Fahrwerk gleicht alle Unebenheiten aus, und der Aufbau erfährt keine vertikale Beschleunigung. Dieser Ansatz entspricht einem klassischen Skyhook-Ansatz [Mit97].

2. Das aktive Fahrwerk ist ähnlich einem passiven Fahrwerk ausgelegt. Aufbaubeschleunigung und Fahrverhalten entsprechen im Wesentlichen einem klassischen Fahrwerk aus passiven Komponenten (Feder-Dämpfer-System). Der Kontakt der Räder zur Straße wird verbessert. Der Aufbau folgt dem Verlauf des Fahrwegs (gedämpft).

Ausgehend von diesen beiden Einstellmöglichkeiten, kann nun eine Veränderung der Reglerparameter durch ein überlagertes Verhalten modelliert werden. Die durch das Verhalten wählbaren Reglerparameter entsprechen paretooptimalen Einstellungen (vgl. Abschnitt 2.2.2). Dieses *Verhalten* reagiert auf die Veränderung der Federbeinauslenkungen. Dazu wird der Federbeinweg in Zonen eingeteilt. In der Komfortzone liegt das Verhalten *Komfort* vor. In diesem Bereich wird der Aufbau nach der Methode 1 optimal ruhig gehalten. Verlässt die Federbeinauslenkung diesen Komfortbereich nach oben oder unten, so wird durch eine einfache lineare Überlagerung je nach Auslenkungsgrad zwischen Komfortregelung und passiver Auslegung überblendet. Erreicht die Auslenkung einen kritischen Bereich, so wird ganz auf die passive Auslegung (*Sicherheit*) des Reglers umgeschaltet, und das Fahrzeug reagiert wie ein passiv gedämpftes Fahrzeug.

Regelungstechnisch ist dieser Ansatz mit einem *Gain-Scheduling* vergleichbar (z. B. [AW94], [Hel95]). Bei dem vorgestellten Ansatz steht jedoch die Art der Modellierung im Vordergrund. Durch eine verhaltensbasierte Modellierung kann diese Form der Adaption leicht erweitert werden.

6.2.3 Aufbau des Operator-Controller-Moduls (OCM)

Die Struktur der Optimierung orientiert sich in wesentlichen Zügen an dem Ansatz gemäß Abschnitt 3.4.2. Dabei wird die Konfiguration aus [HO03] betrachtet, die den Ansatz nach [OHKK02] konkretisiert.

Aufgrund des Widerspruchs zwischen der Forderung, der Straße zu folgen, und dem Wunsch nach Fahrkomfort kann eine Optimierung nur eine Folge von optimalen Kompromissen als Paretomenge ergeben (vgl. Abschnitt 2.2.2.3). Aufgabe des verhaltensbasierten Teils des kognitiven Operators ist es, aus dieser Paretomenge den für die augenblickliche Situation optimalen Parametersatz auszuwählen. In [HO03] wird davon ausgegangen, dass keine Online-Identifikation des Systems erfolgt. Somit ist es sinnvoll, die Paretomenge auf Basis eines simulierten Modells der geregelten Strecke vorab, also offline, zu bestimmen und nur die berechnete Paretomenge zu nutzen.

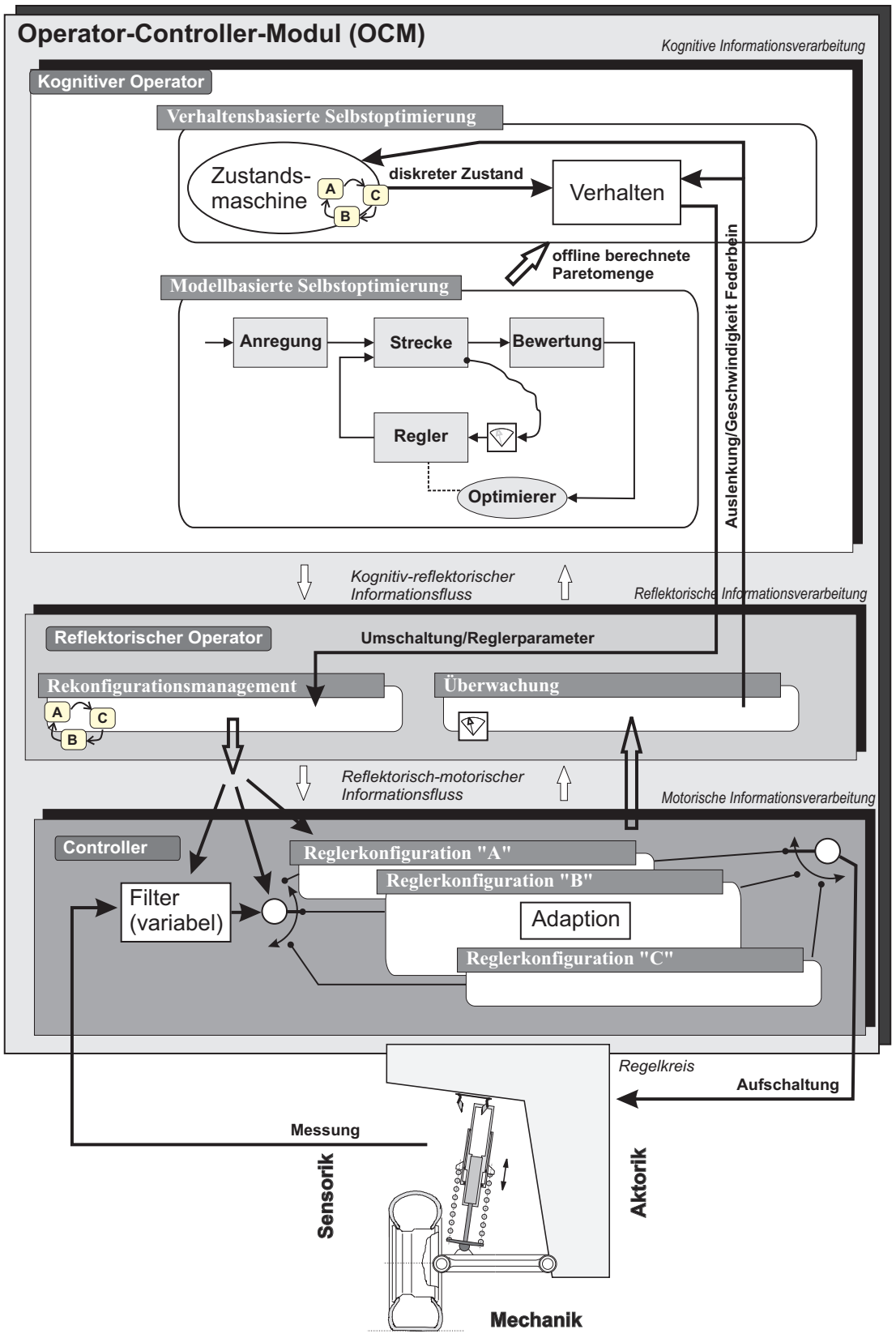

Abbildung 6.12: OCM-Struktur der selbstoptimierenden Fahrwerksregelung

Die gewählte Struktur behält aber auch bei einer möglichen Online-Identifikation und entsprechenden Online-Bestimmung der Paretomenge ihre Gültigkeit.

In Abbildung 6.12 ist die Struktur des OCMs dargestellt. Es besteht aus dem kognitiven Operator, der mit Hilfe einer einfachen Zustandsmaschine das entsprechende Verhalten aus der vorab bestimmten Paretomenge bestimmt. Dies geschieht auf Basis der aktuellen Auslenkung und Geschwindigkeit des Federbeins. Der durch das *Verhalten* gewählte Parametersatz der Reglung wird durch die Umschaltung (Rekonfigurationsmanagement) des reflektorischen Operators in die Reglerstruktur des Controllers übertragen.

Das Verhalten kann aus drei verschiedenen Reglertypen wählen. Regler A ist auf die Einhaltung der Relativkoordinaten optimiert, bei Regler C handelt es sich um einen komfortoptimierten Regler und bei Regler B um einen adaptiven Regler, der zwischen Komfort und Relativbewegung überblendet. Die Steuerung der Adaption erfolgt über das Verhalten, das die augenblicklichen Reglerparameter des adaptiven Reglers B verändert.

Nach [HO03] wird neben der Veränderung der Reglerparameter noch eine weitere Technik verwendet, die komfortables Fahren auch bei wechselndem Fahrbahnniveau ermöglicht. Hierbei wird eine Vorsteuerung verwendet, welche die Straßenanregung in tief- und höherfrequente Anregungen aufteilt. Die tieffrequenten Anregungen werden als Sollbahn vorgegeben, die höherfrequenten als Störung angesehen. Bei klassischen Strukturen ist eine solche Trennung nicht ohne weiteres möglich, da die tieffrequenten Anregungen meist stärkere Störungen hervorrufen als die hochfrequenten. Durch den im Beispiel aus [HO03] verwendeten Adaptionsalgorithmus wird die Eckfrequenz jedoch verändert. Dies geschieht in Abhängigkeit zur Fähigkeit, Störungen zu kompensieren. Ausschlaggebend ist hier die Federbeinauslenkung. Dabei wird ein möglichst großer Frequenzbereich als Störung angesehen.

6.2.4 Kognitiver Operator: Zustandsmaschine und Verhalten

Der kognitive Operator besteht im Wesentlichen aus zwei Elementen: einem Zustandsautomaten, der zwischen den verschiedenen Zuständen selektiert, und einem einfachen Verhalten, das die eigentliche Adaption ermöglicht.

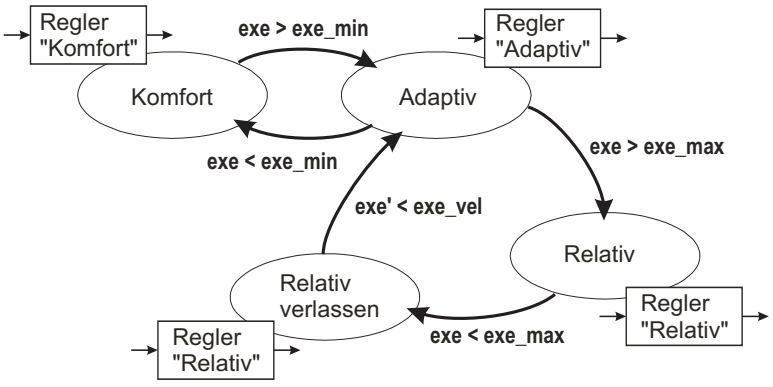

Abbildung 6.13: Zustandsautomat der selbstoptimierenden Fahrwerksregelung

Der Zustandsautomat ist in Abbildung 6.13 [HO03] schematisch dargestellt. Er besteht aus den vier Zuständen *Komfort*, *Relativ*, *Relativ verlassen* und *Adaptiv*. Die

Übergangsbedingungen werden bei Über- oder Unterschreitung bestimmter Grenzwerte aktiv.

Der Wert exe_{min} ist hier die Obergrenze der Auslenkung, bei der vom *Komfort* in den Zustand *Adaptiv* gewechselt wird. Bei Unterschreitung dieses Wertes wird zurück in den Zustand *Komfort* gewechselt. Bei Überschreitung einer gewählten maximalen Auslenkung exe_{max} wechselt der Zustandsautomat in den Zustand *Relativ*, der auch nur über den Zustand *Relativ verlassen* verlassen werden kann. Durch diesen zusätzlichen Zustand wird sichergestellt, dass nicht sofort bei Unterschreitung der Auslenkung in den adaptiven Regler geschaltet wird, sondern erst bei Erfüllung der weiteren Bedingung, welche die Obergrenze für die Federbeingeschwindigkeit exe_{vel} festlegt.

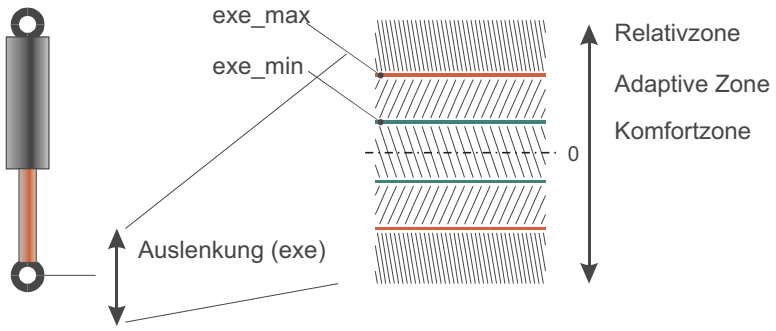

Abbildung 6.14: Verhalten der selbstoptimierenden Fahrwerksregelung (schematisch)

Der Block *Verhalten* modelliert durch Gleichungen die Anpassung der Regler- bzw. Filterparameter. Das Gesamtverhalten wird zusammen mit dem Zustandsautomaten abgebildet. Verdeutlicht wird dieses Verhalten in Abbildung 6.14. Die Federbeinauslenkung ist in die drei Zonen *Komfort*, *Adaptiv* und *Relativ* eingeteilt. In der Komfortzone verhält sich das Fahrzeug komfortoptimiert, d. h. die Vertikalbeschleunigung des Aufbaus wird soweit wie möglich unterdrückt[31]. In der Zone *Relativ* folgt das Fahrzeug der Straße, d. h. es verhält sich ähnlich einem Fahrzeug mit passivem Fahrwerk. In der Zone *Adaptiv* werden die Parameter der Regelung über das Verhalten, in Abhängigkeit der Federbeinauslenkung, angepasst. Dabei sind obere und untere Grenze der Parameterveränderungen so gewählt, dass sie je nach Grenzfall genau dem Relativregler oder dem Komfortregler entsprechen. Beide Grenzen stellen somit nur die Ränder der oben beschriebenen paretooptimalen Lösung dar, die mit Hilfe der modellbasierten Optimierung vorab bestimmt wurden.

Prinzipiell können die unterschiedlichen Zonen auch als eine Überlagerung zweier *Basisverhalten* interpretiert werden. Ein Verhalten versucht, dem Straßenverlauf zu folgen und die Federbeinauslenkung zu minimieren, das andere versucht, den Aufbau möglichst ruhig zu halten. In der Zone *Relativ* ist das erste Verhalten dominant, in der Zone *Komfort* das zweite. In der adaptiven Zone werden beide Verhalten überlagert.

[31] In realen Systemen setzt die Bandbreite der Aktorik meist die Grenze der kompensierbaren Störanregungen.

6.2.5 Simulationsergebnisse

Die Simulation eines Fahrversuchs zeigt nach [HO03] die in Abbildung 6.16 und 6.17 dargestellten Ergebnisse. Abbildung 6.15 zeigt die verwendete Anregungsfunktion bzw. den modellierten Fahrweg. Er besteht aus einem Bereich sinusförmiger Anregungen (links) und einer Rampe (rechts). Die Anregungsfunktion wurde bewusst extrem gewählt, um die Effekte deutlich zu visualisieren.

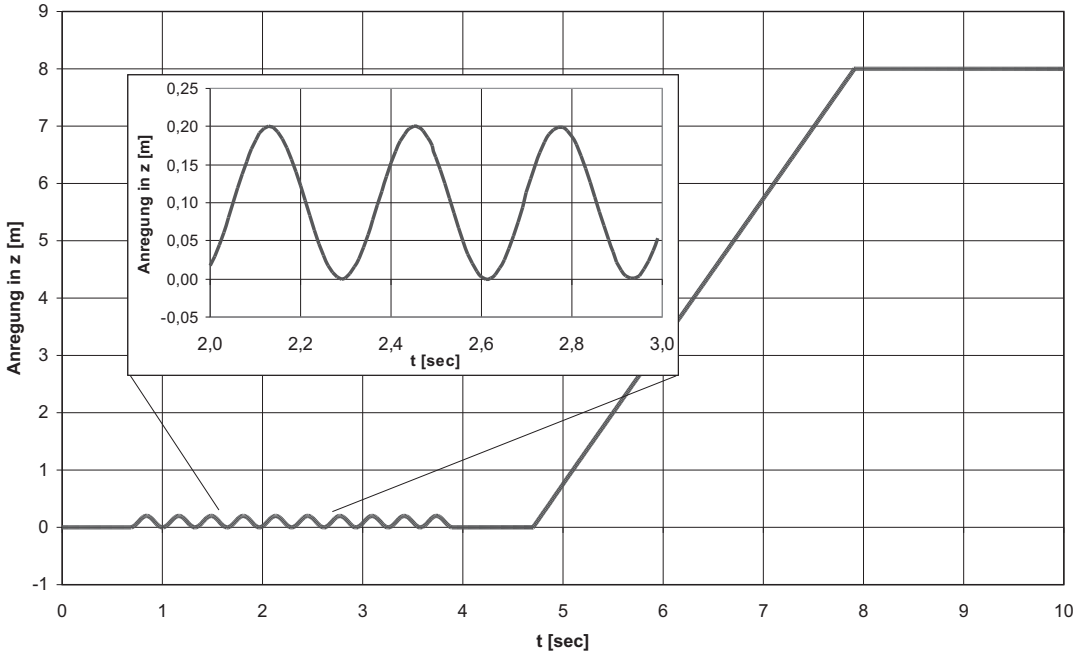

Abbildung 6.15: Anregungsfunktion der Fahrsimulation

In Abbildung 6.16 ist der Verlauf der Aufbaubeschleunigung für drei verschiedene Regler abgebildet. Der adaptive Regler erkennt den Umgebungszustand und kann somit die optimalen Fahreigenschaften der anderen Regler (*Komfort* und *Relativ*), die für eine bestimmte Streckenanregung ausgelegt sind, erreichen.

Hervorzuheben ist, dass der adaptive Regler bzgl. des benötigten Federweges günstiger als alle anderen Regler ist, wie in Abbildung 6.17 zu sehen ist. Dafür liegen die Komfortwerte[32] (Vertikalbeschleunigung, Abbildung 6.16) des adaptiven Reglers etwas schlechter als die des Komfortreglers, jedoch sind die Federwege des Komfortreglers viel größer, als in der Praxis möglich wäre. Der adaptive Regler hält die Begrenzung des Federweges in jedem Fall ein.

Ein Defizit der Regelung resultiert aus der Offline-Berechnung der Pareto-Optima. Da die Offline-Berechnung nur ein Anregungsspektrum berücksichtigt, ist die Wahl der Reglerparameter bzgl. der tatsächlichen Anregungsspektren nicht optimal. Dies macht jedoch wiederum deutlich, dass mit einer Erweiterung der Online-Optimierung weitere Verbesserungen möglich sind, und ist somit ein weiteres Argument für Selbstoptimierung in mechatronischen Systemen.

[32] In [HO03] wird darüber hinaus noch die Wankbeschleunigung betrachtet.

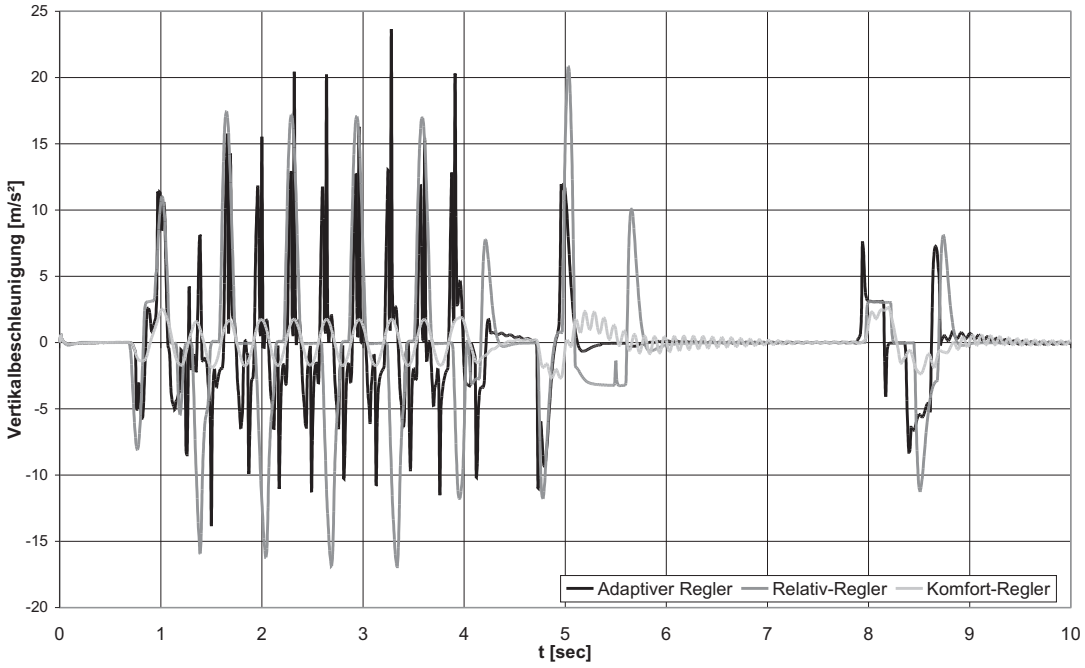

Abbildung 6.16: Aufbaubeschleunigung bei verschiedenen Reglerkonfigurationen

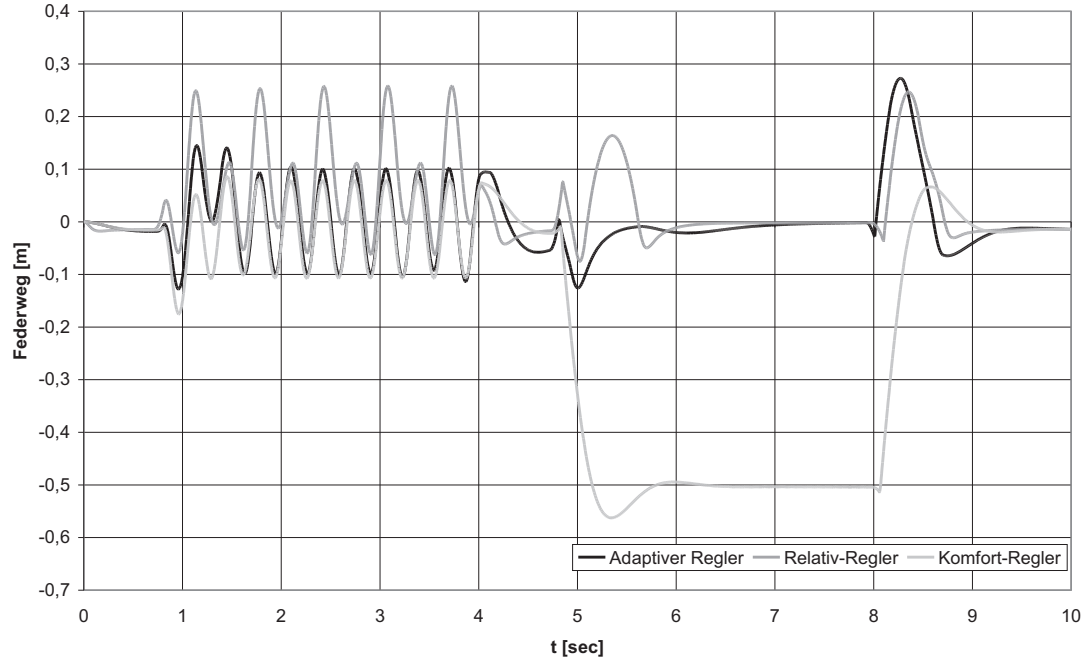

Abbildung 6.17: Federweg bei verschiedenen Reglerkonfigurationen

6.3 Ein Beispiel für verteilte Optimierung: Trajektorienoptimierung bei schienengebundenen Fahrzeugen

Grundlage der meisten Optimierungsverfahren sind das Durchlaufen eines Test- oder Anregungsfalls mit anschließender Bewertung und eine darauf folgende gezielte Veränderung der Systemparameter. In der Simulation werden dafür häufig Anregungsfunktionen verwendet, die z. B. einer bestimmten zu reduzierenden Frequenz entsprechen. Jedoch ist das Aufschalten von Anregungsfunktionen nicht in jedem Fall möglich. Dies gilt insbesondere für Hardware-in-the-Loop-Simulationen am Prüfstand und den realen Betrieb auf der Strecke. Die Optimierung am realen System kann jedoch signifikante Verbesserungen gegenüber einer rein auf Simulation gestützten Optimierung bringen, insbesondere dann, wenn die Systemparameter nicht hinreichend bekannt sind oder sich im Betrieb auf nicht vorhersehbare Weise verändern [DO00, DOM01].

Eine solche Form der Selbstoptimierung ist bei wiederkehrenden oder vergleichbaren Anregungsfällen besonders zweckmäßig. Noch weiter geht der Ansatz, vergleichbare Systeme, auf die außerdem vergleichbare Anregungszustände wirken, gemeinsam zu optimieren. Werden die Ergebnisse jedes einzelnen Systems gemeinsam genutzt, kann von einer verteilten Optimierung gesprochen werden.

In diesem Beispiel wird ein Szenario für eine verteilte Optimierung der Sollbahnvorgabe für aktive Federungen vorgestellt. Als Anwendungs-Szenario dient das *RailCab*-System der *Neuen Bahntechnik Paderborn*, das wegen seines vollständig in sechs Achsen aktiven Fahrwerks und der entsprechenden Leit- und Regelungstechnik besonders geeignet ist.

6.3.1 Technischer Aufbau des Modellsystems

Das Fahrzeug entstammt der Forschungsinitiative *Neue Bahntechnik Paderborn*. Es handelt sich dabei um ein ca. 1,0 t schweres und 2,5 m langes Versuchsfahrzeug, das für die Teststrecke der Universität Paderborn entwickelt worden ist. Die Fahrzeuge sind autonom selbstfahrend und werden durch einen besonderen Linearmotor angetrieben. Ein zentraler Teil des Gesamtkonzepts ist das aktive Federungssystem, mit dem alle sechs Aufbaufreiheitsgrade vollständig kontrolliert werden können [HES03].

Damit ist eine Kontrolle von Vertikal- und Querrichtung sowie Wende-, Wank- und Nickbewegungen des Fahrzeugs möglich. Die Längsfreiheitsgrade werden durch vier weitere Zylinder beeinflusst, die in horizontaler Ebene angeordnet sind.

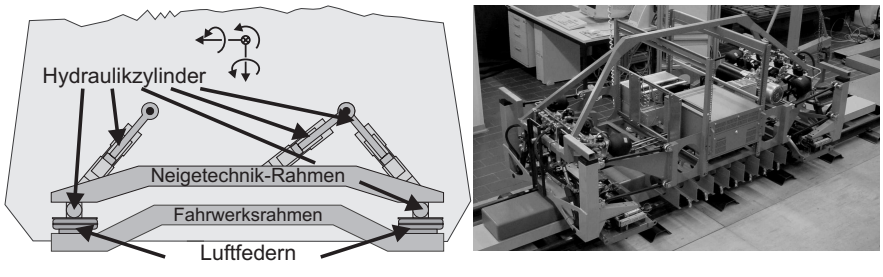

Abbildung 6.18: Aktive Federung des RailCab

Abbildung 6.18 zeigt links den schematischen Aufbau des aktiven Fahrwerks. Wesentlich für die Aufbaukontrolle in vertikaler Richtung sind die Hydraulikzylinder. Je drei Zylinder sind vorn wie hinten mit Aufbau und Neigetechnikrahmen verbunden. Die Führung in der Ebene wird von einer 4-Lenker-Konstruktion übernommen. Die Luftfedern sorgen für eine Entkopplung der Rahmenkonstruktion. Die Dämpfung des Aufbaus erfolgt aktiv durch die Hydraulikzylinder.

Für die Regelung der Federung erfassen Sensoren die Beschleunigung des Aufbaus. Sensoren an den Zylindern und den Luftfedern ermitteln jeweils die Auslenkung. Die Messwerte der Wegsensoren können für eine Dämpfung zwischen Aufbau und Fahrwerk mittels Fußpunktverstellung genutzt werden. Aus den Beschleunigungswerten des Aufbaus kann mittels einer so genannten *Skyhook-Dämpfung* [Mit97] die Absolutbewegung des Aufbaus unterdrückt werden.

Für die Regelung der sechs Starrkörper-Freiheitsgrade muss als Aufbauregler ein Mehrkörper-Regler eingesetzt werden. In [HES03] wird gezeigt, dass die dynamische Regelung des Aufbaus mit Hilfe der Modaltransformation auf die Regelung von Einmassen-Schwingern zurückgeführt werden kann. Für eine nähere Erläuterung wird auf [HES03] und [HMO04] verwiesen.

6.3.2 Sollbahnoptimierung

Das Konzept der Bahnoptimierung basiert auf der Vorgabe von Führungsdaten für die Regelung des Fahrwerks für einen bestimmten Streckenabschnitt. Dieser Steckenabschnitt wird vom Fahrzeug durchfahren und anschließend von ihm auch bewertet. Dabei wird angenommen, dass das dynamische Verhalten der Fahrzeuge in etwa gleich ist. Somit kann davon ausgegangen werden, dass eine Bewertung auch durch verschiedene Fahrzeuge möglich ist. Dabei erhält der jeweilige Streckenabschnitt die besondere Aufgabe, die Daten zu sammeln, zu optimieren und den Shuttles neue, optimierte Führungsvorgaben zur Verfügung zu stellen. Diese verteilte Architektur entspricht in ihrer Wirkweise einer Optimierung über viele Individuen.

Die Führungsvorgaben können auf verschiedene Weise dargestellt werden. Besonders geeignet ist die Abbildung als Trajektorie $\dot{z}_{soll} = f(t)$, die kompakt durch einen kubischen Spline angenähert werden kann. Aus praktischen Erwägungen, wie der Berücksichtigung von Randbedingungen der Stellwegbegrenzungen des Fahrwerks, wird zur Optimierung eine Trajektorie $z = f(x)$ verwendet[33]. Die Trajektorie $\dot{z}$ kann wegen der Darstellung als kubischer Spline besonders leicht aus $\dot{z} = v \cdot dz/dx$ berechnet werden.

Zu einer weiteren Reduzierung der Daten führt die Festlegung eines bestimmten gleichen Abstands zwischen den Stützpunkten. Bewertet wird das Integral der quadratischen Beschleunigung $\phi = \int \ddot{z}^2 dt$, das um den jeweiligen Stützpunkt gebildet wird. Dadurch erhält jeder Stützpunkt quasi eine einzige Maßzahl als Bewertung für den lokalen Komfort.

Der Raum um einen Stützpunkt ist in sogenannte *Segmente* eingeteilt, die den Einflussbereich eines Punktes auf die bewertete Beschleunigung festlegen. Diese Vereinfachung vernachlässigt die Beeinflussung benachbarter Punkte.

In Abbildung 6.19 ist das Prinzip der Trajektorienoptimierung schematisch dargestellt. Die Optimierung erfolgt durch eine Veränderung der Stützpunkte. Bei der

[33] x entspricht der Bahnposition innerhalb eines Streckenabschnitts.

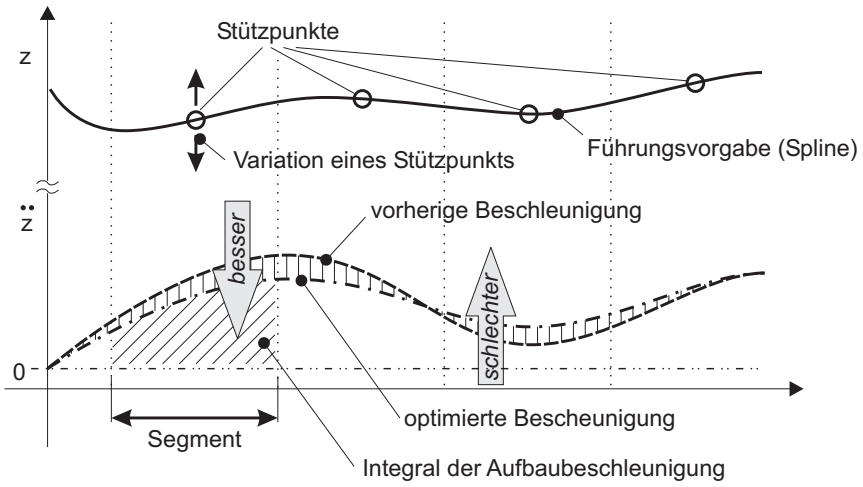

Abbildung 6.19: Optimierung der Trajektorie

anschließenden Durchfahrt wird die Aufbaubeschleunigung aufgezeichnet und in Bezug auf die jeweiligen Stützstellen bewertet. Somit wird jedem Segment und so jeder Stützstelle eine Bewertungszahl ϕ_i nach Gleichung 6.3.1 zugeordnet:

$$\phi_i = \int_{G_v}^{G_h} \ddot{z}^2 dt; \text{ mit } G_v \text{ / } G_h - \text{vordere/hintere Segmentgrenze} \tag{6.3.1}$$

Als Ergebnis werden Stützpunkte[34] des Splines (z_i) von der Strecke an das Fahrzeug gesendet, die den Verlauf der Trajektorie beschreiben. Nach der Durchfahrt wird quasi als Antwort die Bewertung ϕ_i, die mit den Spline-Stützpunkten korrespondiert, von der Strecke zum Shuttle zurückgesendet.

Für eine Begrenzung des Optimierungsproblems auf einen lokalen Bereich wird die Strecke in verschiedene logische Abschnitte (Bereiche) unterteilt, die separat betrachtet werden. Jedem Bereich wird dabei eine sogenannte *Bereichskontrolle* zugeordnet. Von Kontrolle muss deshalb gesprochen werden, da der jeweiligen Instanz die wichtige Aufgabe der Vorgabe von Führungsdaten für ein durchfahrendes Shuttle zukommt. Die Bereichskontrolle führt dabei selbstständig auf der Basis bekannter und neu erfasster Bewertungsdaten sowie bestehender Vorgabedaten eine Optimierung durch. Darüber hinaus muss sie mit benachbarten Bereichskontrollen die Optimierungsergebnisse abstimmen, damit es beispielsweise nicht zu Sprüngen oder Unstetigkeiten an den Rändern der Splines kommt. Deshalb ist es naheliegend, diesen Kontrollbaustein des Systems als kognitiven Agenten zu modellieren. Im Sinne der vorgestellten Modellierungsansätze entspricht dies einem OCM, das im Wesentlichen nur aus einem kognitivem Operator besteht.

Der Ansatz der Aufteilung in Bereiche und die Zuordnung der Informationsverarbeitung zeigen die konsequente Orientierung der Strukturierung der Informationsverarbeitung an dem technischen System.

Die Grundstruktur ist in Abbildung 6.20 dargestellt. Die Bereichskontrolle (Mitte) verwaltet und optimiert die Führungstrajektorie für einen bestimmten, fest zugeordneten Streckenabschnitt. Ein in den Bereich einfahrendes Shuttle meldet sich vor

[34]Eine Darstellung als geordnete Paare (x_i, z_i) ist nicht notwendig, wenn von Stützpunkten mit fester Anzahl und festem Abstand ausgegangen wird.

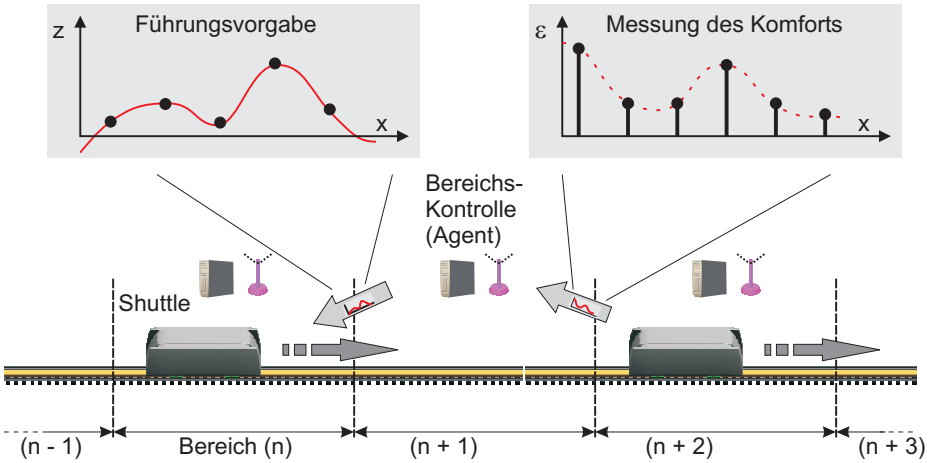

Abbildung 6.20: Grundstruktur der verteilten Optimierung

der Durchfahrt an. Als Antwort sendet die Bereichskontrolle die Führungstrajektorie für die Aufbauregelung in Form von Stützstellen eines kubischen Splines. Nach der Ausfahrt aus dem Bereich sendet das Shuttle Bewertungsdaten über das Aufbauverhalten während der Durchfahrt in Form von Kennzahlen für alle Spline-Stützpunkte. In diesem Beispiel wird der Komfort in vertikaler Richtung betrachtet. Die als Agent organisierte Kontrolle führt aufgrund der neuen Daten einen Bewertungsschritt aus und berechnet eine neue Führungsvorgabe.

Um einen sicheren Betrieb zu gewährleisten, muss es grundsätzlich möglich sein, ohne Führungsvorgabe sicher durch den Bereich zu fahren. Falls keine Daten zur Verfügung stehen, schaltet die Regelung auf einen Relativregler um (Abbildung 6.21, Relativ-Anteil).

Das hier vorgestellte und getestete Szenario betrachtet nur die vertikale Aufbaubewegung des Shuttles. Grundsätzlich ist das Verfahren aber für alle Freiheitsgrade denkbar, insbesondere für die Querrichtung, da hier starke, wiederkehrende Anregungen bei Durchfahrt durch eine Weiche auftreten können. Auch darüber hinausgehende Analysen des Durchfahrtsverhalten sind denkbar. So können stark abweichende Bewertungen auf ein Problem des Fahrzeugs (Fahrwerk, Räder etc.) oder des Schienenstrangs (Versatz/Verzug bei hohen/niedrigen Temperaturen etc.) hindeuten.

6.3.3 Regelung des Aufbaus mit Führungsvorgabe

Der Aufbau der Regelung für das betrachtete Beispiel soll nun kurz dargestellt werden. Eine ausführlichere Diskussion findet sich in [HMO04]. An dieser Stelle sollen nur die Grundlagen aufgezeigt werden, die für den Ansatz der Selbstoptimierung und das Verständnis der Grundidee notwendig sind.

Ein *klassischer* Ansatz für die Regelung aktiver Fahrwerke ist der sogenannte *Skyhook-Regler*. Dieser Ansatz, der sich vor allem durch seine Einfachheit und Anschaulichkeit auszeichnet, findet sich in vielen Anwendungen wieder. Die prinzipielle Struktur ist in Abbildung 6.21 dargestellt.

Aus dem Abstand zwischen Aufbau und Fahrwerk z_{rel}, auch als Fahrzeugniveau bezeichnet, und der Aufbaubeschleunigung $\ddot{z}_a$ berechnet der Regelungsalgorithmus eine Zusatzkraft F_{aktiv}. Durch einen nachgeschalteten Tiefpass-Filter wird die Zusatzkraft auf den relevanten Frequenzbereich beschränkt. Unter Umständen kann

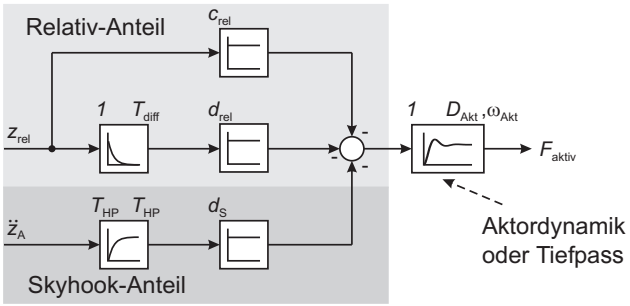

Abbildung 6.21: Klassischer Regelungsansatz mit Relativ- und Skyhook-Anteil

hier die Tiefpass-Charakteristik der Aktorik genutzt werden, wie in Abbildung 6.21 recht zu sehen ist. In diesem Fall wird die Charakteristik durch ein PT_2-Glied mit der Eigenfrequenz ω_{Akt} und der Lehrschen Dämpfung D_{Akt} dargestellt.

Der Relativanteil der Federung ist im oberen Teil von Abbildung 6.21 zu sehen. Er hat die Wirkung eines zusätzlichen Federbeins mit der Federsteifigkeit c_{rel} und der Dämpfungskonstante d_{rel}. Im unteren Teil der Abbildung 6.21 ist der Skyhook-Anteil abgebildet. Er wirkt wie ein Dämpfer zur Umgebung mit der Dämpfungskonstante d_s, also quasi wie ein Dämpfer zwischen Aufbau und Himmel – daher die Bezeichnung *Skyhook*. Da die Relativgeschwindigkeit $\dot{z}_{rel}$ in der Regel nicht zur Verfügung steht, wird sie durch Differentation, mit der Zeitkonstante T_{diff}, aus dem Niveau berechnet. Die Aufbaugeschwindigkeit $\dot{z}_A$ wird ebenfalls berechnet. Dazu dient eine hochpassgefilterte Integration mit der Zeitkonstante T_{HP}.

Ein prinzipbedingtes Problem des Skyhook-Reglers ist der durch die Dämpfung zur Umgebung auftretende, sogenannte *Rampenfehler*. Dieser Fehler tritt aufgrund des Widerspruchs auf, dass sich auf der einen Seite der Aufbau möglichst nicht (vertikal) bewegen soll, aber andererseits der Stellweg der Federung begrenzt ist und auf der zu fahrenden Strecke Höhenunterschiede zu überwinden sind. Mit Hilfe einer Hochpassfilterung lässt sich der Rampenfehler zwar nach einiger Zeit auf 0 verringern, wird aber nicht sofort ausgeglichen. Daher stellt sich die Frage, welcher Bahn der Aufbau bei einer gegebenen Strecke *idealerweise* folgen muss, um einen guten Kompromiss zwischen Komfort und reduziertem Rampenfehler zu liefern.

Die zentrale Idee des hier vorgestellten Ansatzes ist die Vorgabe einer Sollbahn, an der entlang das Fahrzeug geführt wird (vgl. 6.3.2). Da bei diesem Ansatz der Stellweg der Federung mit in die Berechnung der Sollbahn bzw. Führungstrajektorie einfließt, kann hier auf eine relative Dämpfung verzichtet werden[35].

Für eine hinreichend genaue Ausregelung des Fahrwerks gegenüber der Strecke werden zum einen die Vorgabe der idealen Aufbaubewegung benötigt, zum anderen die Störanregung der Strecke. Die Vorgabe erfolgt durch den Sollgeschwindigkeitsverlauf $\dot{z}_{soll}$, der durch einen Spline abgebildet wird. $\dot{z}_{soll}$ lässt sich mit Hilfe der Gleichung $\dot{z}_{soll} = dz/dx \cdot dx/dt = \acute{z}_{soll} \cdot v$ bestimmen. $\acute{z}_{soll}$ ist hier die von der Zeit abhängige Vorgabe, im Gegensatz zu $\dot{z}_{soll}$, die von der Streckenposition abhängt. Dabei wird vereinfachend vorausgesetzt, dass das Shuttle in einem Streckenabschnitt seine Geschwindigkeit beibehält. Eine Vorgabe unter Berücksichtigung einer veränderlichen Geschwindigkeit ist noch Teil der Diskussion.

[35]Ein Verfahren, wie mit Hilfe von verhaltensbasierten Ansätzen die maximale Auslenkung des Fahrwerks begrenzt werden kann, wird im Abschnitt 6.2 dargestellt.

Aus Gründen der Vereinfachung der Optimierung können die Größen $\acute{z}_{soll}$ und $\tilde{z}_0$ zu $\acute{\tilde{z}}_0$ zusammengefasst werden [HMO04]. Dies ist vor allem dann sinnvoll, wenn sowohl die ideale Bahnvorgabe als auch die Streckenanregung zunächst als nicht bekannt angenommen werden.

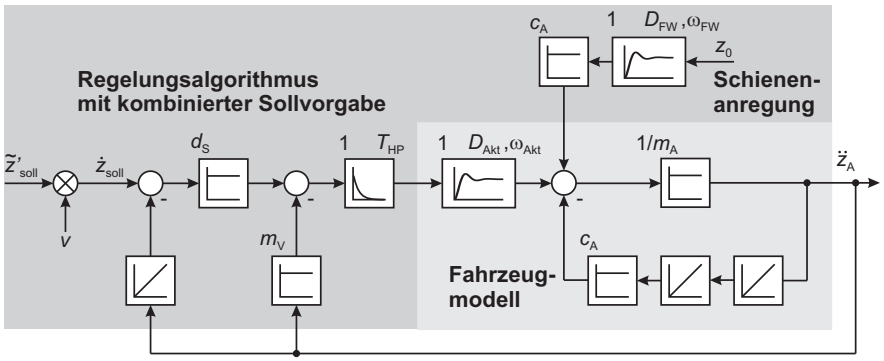

Abbildung 6.22: Regelungsstruktur mit kombinierter Sollgeschwindigkeit

Abbildung 6.22 zeigt die verwendete Regelung mit dem Fahrzeugmodell. Der linke Teil zeigt die vereinfachte Regelungsstruktur mit kombinierter Sollvorgabe.

6.3.4 Optimierungsverfahren

Grundlage der Optimierung der Bahnvorgabe ist die Verschiebung der Stützpunkte der Splines der Vorgabetrajektorie in vertikaler Richtung. Daher können die Positionen der Stützpunkte als freie Parameter der Optimierung angesehen werden. Zentral ist dabei die Frage, wie stark die Stützstellen voneinander entkoppelt sind. Davon hängt ab, inwieweit Segmente für eine Teiloptimierung zusammengefasst werden können. Da viele Segmente einen Abschnitt bilden, kann bei hinreichender Entkopplung die Optimierung parallelisiert werden. Deshalb müssen zunächst die Entkoppelbarkeit und der Einflussbereich einer Stützpunktverschiebung betrachtet werden.

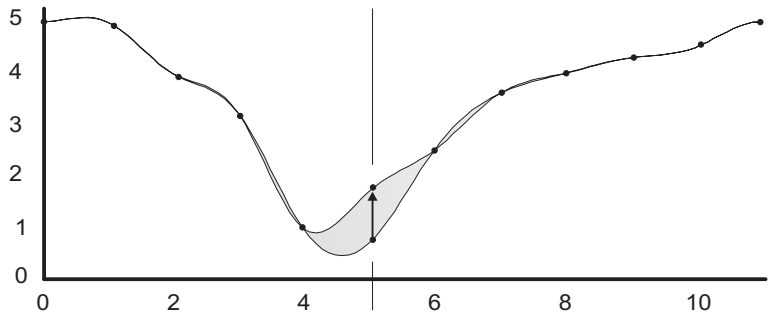

Abbildung 6.23: Änderung einer Splinekurve durch Verschiebung einer Stützstelle

In Abbildung 6.23 ist die Verschiebung eines Stützpunktes dargestellt. Wie die grau hinterlegte Fläche andeutet, ist dies die Differenz zwischen den beiden Kurven als Konsequenz der Verschiebung. Die größten Unterschiede treten in der Nähe des Punkts 5 auf. Weitere deutliche Verschiebungen sind in der Umgebung von zwei Splinepunkten links und rechts zu sehen. Augenscheinlich ist die Veränderung darüber hinaus verschwindend gering. Jede Verschiebung einer Stützstelle hat theoretisch

Einfluss auf den gesamten Spline. Jedoch ist für die Praxis eine genauere Untersuchung nicht angebracht, da der Einfluss spätestens durch die numerische Abbildung der Zahlen ganz verschwindet.

Weiterhin muss der Einfluss auf die erste (Abbildung 6.24) und die zweite (Abbildung 6.25) Ableitung untersucht werden. Auch hier zeigt sich ein ähnliches Bild. Es sind Veränderungen in der Umgebung von bis zu 4 Stützpunkten zu erkennen.

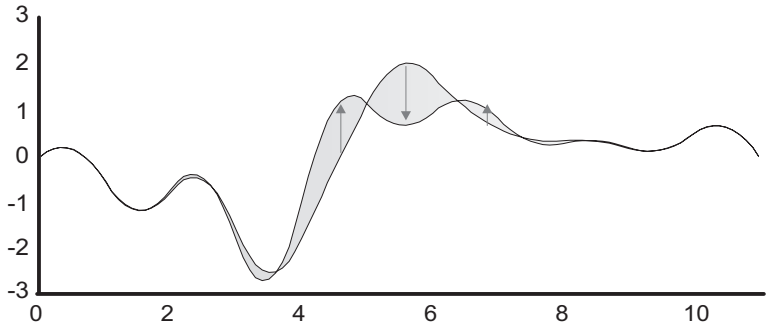

Abbildung 6.24: Änderung der ersten Ableitung durch Verschiebung einer Stützstelle

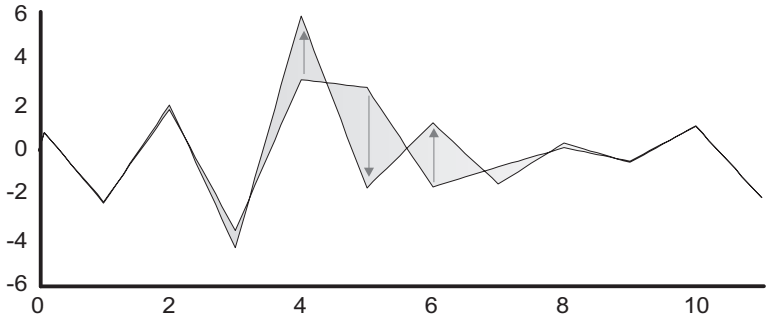

Abbildung 6.25: Änderung der zweiten Ableitung durch Veränderung einer Stützstelle

Damit ist deutlich, dass aus praktischer Sicht von einer Entkopplung weiter entfernt liegender Stützpunkte ausgegangen werden kann. Dies lässt sich weiter im Optimierungsalgorithmus nutzen. Dazu wird der Spline in Partitionen P_i eingeteilt, die genau einer zu optimierenden Stützstelle x_i zugeordnet sind. Die Aufteilung der Partitionen erfolgt so, dass keine Überschneidungen auftreten. Für die Auswahl der zu optimierenden Partitionen werden bei jedem Optimierungsschritt die Bewertungszahlen ϕ_i herangezogen. Dabei werden hohe Bewertungszahlen bevorzugt. Eine Optimierung erfolgt somit gezielt an den Stellen, wo ein erhöhter Optimierungsbedarf besteht.

Die Optimierung läuft nach dem Prinzip eines *Hill-Climbing*-Verfahrens ab. Hill-Climbing gehört zu den Methoden der nichtlinearen globalen Optimierung [RN03]. Das Verfahren ist sehr anschaulich und leicht zu implementieren, neigt jedoch zum Verharren in lokalen Minima. Es ist hier gut geeignet, da der Gesamtalgorithmus ein Verharren in lokalen Optima verhindert. Die hier eingesetzte Variante ist das Gradientenverfahren, das eine Suchrichtung (Richtung des steilsten Abstiegs oder

Anstiegs) festlegt. Diese Methode konvergiert besser als das ursprüngliche Hill-Climbing-Verfahren [Mün03].

Bei der Optimierung der Trajektorie werden aufgrund der beschriebenen Entkopplung mit Hilfe des Hill-Climbing-Verfahrens lokale Partitionen des Splines parallel optimiert. Zielgrößen sind die Bewertungszahlen ϕ_i, die einer lokalen Aufbaubeschleunigung an der Stelle entsprechen. Verändert wird jeweils nur eine Stützstelle z_i; somit lässt sich die Optimierung des Gesamtproblems auch auf die Optimierung mehrerer eindimensionaler Probleme zurückführen. Die Kopplung der Teilprobleme erfolgt indirekt durch den Spline und die Verschiebung bzw. Neueinteilung der Partitionen. Auf diese Weise kann ein Gesamtoptimum angenähert werden. Ein *Festfahren* des Algorithmus an einem dominierenden, aber nicht weiter zu verbessernden Punkt wird durch eine sogenannte *Tabu-Liste* erreicht, in die alle Stützpunkte bzw. die daraus resultierenden Partitionen aufgenommen werden, die nach mehreren Schritten keine Verbesserung mehr zeigen. Nach einer bestimmten weiteren Anzahl von Schritten wird der Eintrag wieder aus der Liste entfernt. Der vollständige Optimierungsalgorithmus wird in [HMO04] beschrieben.

Insgesamt lässt sich sagen, dass der Algorithmus nur für die Optimierung lokaler Probleme geeignet ist. Ein globales Optimum für einen Streckenabschnitt, wie z. B. Energieverbrauch, ist mit diesem Verfahren nicht zu erreichen, da nur lokale Ziele verfolgt werden. Wie sich der Gesamtansatz auf globale Optimierungsziele übertragen lässt, muss noch untersucht werden.

6.3.5 Struktur der Informationsverarbeitung

Die Informationsverarbeitung eines mechatronischen Systems muss eine Vielzahl von Funktionen erfüllen: eine quasi-kontinuierlich arbeitende Regelung kontrolliert die Dynamik der Strecke; Fehlfunktionen müssen erfasst und behandelt werden; Adaptionsalgorithmen passen die Regelung an veränderte Umgebungszustände an, um nur ein paar Funktionen zu nennen. Darüber hinaus hat bei verteilten Systemen die Kommunikation eine wichtige Funktion, insbesondere im Falle einer verteilten Optimierung. Neben funktionellen Aspekten spielt aber auch der Entwurf eine Rolle. Die Inbetriebnahme erfordert einen modularen Aufbau. Dies erlaubt zum einen eine getrennte Betrachtung der Optimierung und der vitalen Informationsverarbeitung (Regelung), zum anderen eine erhöhte Sicherheit, wenn zunächst Fehler bei den sicherheitsrelevanten, die Aktorik beeinflussenden Teilen der Informationsverarbeitung ausgeschlossen werden können.

Eine diesen Anforderungen besonders gerecht werdende Architektur stellt das Operator-Controller-Modul (OCM) dar, das als Entwurfsschema für die Informationsverarbeitung in selbstoptimierenden Systemen verwendet werden kann. Im Sinne einer selbsttätigen Handlung auf Ebene der Optimierung kann auch das Entwurfsparadigma der Agententechnik berücksichtigt werden. Im Sinne der Agententechnik stellt das OCM die *Mikrostruktur* eines vernetzten Multiagentensystems dar, das sich auf Ebene der kognitiven Informationsverarbeitung (vgl. 3.4.2) aus kommunizierenden OCMs ergibt (Fahrzeuge und Strecke).

Die Struktur der Informationsverarbeitung der Streckenkontrolle ist grundsätzlich der der Regelung ähnlich, jedoch wird die Ebene der motorischen Informationsverarbeitung (Controller) nicht verwendet, da kein technisches System direkt angekoppelt ist. Wird das OCM als akzeptierte Struktur innerhalb eines Multiagentensystems angesehen, ist es sinnvoll, trotz fehlenden Controllers weiterhin an dem

grundsätzlichen Aufbau fest zu halten. Auf der Ebene der kognitiven Informationsverarbeitung verwischen die Grenzen zwischen Agent und OCM; somit ist eine strenge Unterscheidung nicht notwendig oder sinnvoll.

Im Sinne der Agententechnik wird die globale Kommunikationsstruktur auch als *Makrostuktur* bezeichnet. Die Struktur orientiert sich an der Topologie des technischen Systems.

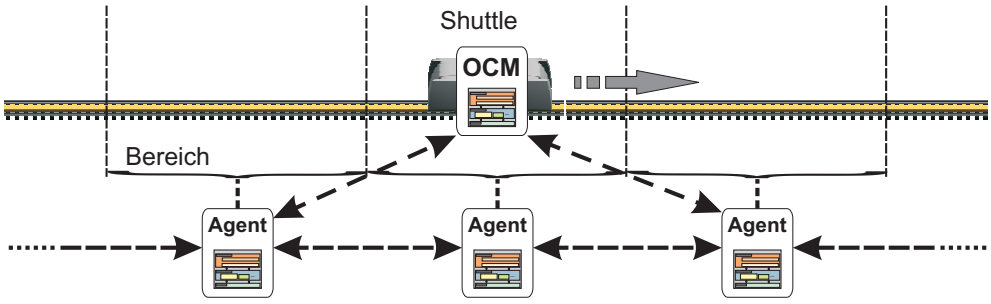

Abbildung 6.26: Makrostruktur der Informationsverarbeitung

Die schematische Struktur des Multiagentensystems ist in Abbildung 6.26 dargestellt. Die Strecke ist in logische Bereiche eingeteilt, die jeweils von einem Agenten verwaltet werden. Das OCM des Shuttles kann hier als *mobiler Agent* aufgefasst werden, der von Streckenabschnitt zu Streckenabschnitt weitergereicht wird. Die Kontaktaufnahme erfolgt dabei über den kognitiven Teil des OCM. Ist die Verbindung etabliert, werden die Daten für die Bahnvorgabe bzw. die Bewertungsdaten zwischen den Operatoren ausgetauscht. Ist die Übertragung der Vorgabedaten abgeschlossen, werden diese Daten dem reflektorischen Operator übergeben, der diese dann zum entsprechenden Zeitpunkt (bei der Einfahrt in den nächsten Bereich) in der Regelung aktiviert. Die Aufzeichnung der Bewertungsdaten erfolgt im reflektorischen Operator, Weiterverarbeitung und Übertragung zum Streckenagenten jedoch im kognitiven Operator. Während der Durchfahrt wird mit dem jeweiligen Streckenagenten nicht kommuniziert. Prinzipiell ist auch eine Abhandlung des Datenaustauschs direkt zwischen den reflektorischen Operatoren denkbar, da das Protokoll streng formalisiert werden kann. In diesem Fall wird die Planungskomponente nur zur Suche des nächsten Streckenagenten benötigt.

Im Gegensatz zum OCM der Shuttles, bei denen alle drei Ebenen des OCM implementiert sind, werden für die Streckenagenten nur die Operatoren der OCM-Architektur benötigt, da keine angekoppelte Strecke vorhanden ist. Jedoch ist es schon allein aus Gründen der Übersichtlichkeit sinnvoll, an der OCM-Struktur festzuhalten, um eine homogene Architektur zur erhalten.

Zwischen den einzelnen Strecken-Agenten wird zur Abstimmung der Vorgabedaten eine direkte Kommunikation benötigt. Die Verbindungen zwischen benachbarten Agenten können bereits zum Initialisierungszeitpunkt fest etabliert werden und ergeben sich aus der Topologie des Schienensystems.

6.3.6 Simulationsszenario und -ergebnisse

Grundlage des Szenarios ist die Teststrecke der *Neuen Bahntechnik Paderborn*. Die Höhendaten der Teststrecke wurden nach Konstruktionsdaten in einem Kennfeld

aufgetragen. Als zusätzliche Anregung wurde bei 30 m ein 10 mm starker Schienenversatz eingefügt. Dies ist gegenüber der Realität zwar deutlich übertrieben, erlaubt aber eine genauere Betrachtung des Konvergenzverhaltens bei der Entwicklung verschiedener Algorithmen, die hier nicht weiter betrachtet werden.

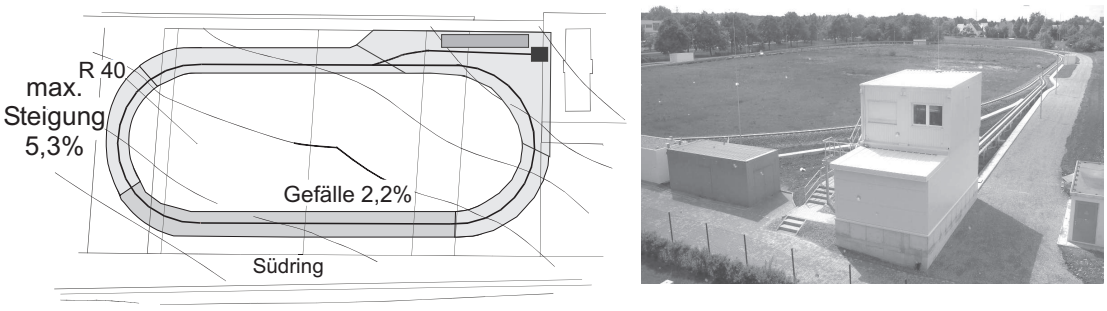

Abbildung 6.27: Teststrecke der *Neuen Bahntechnik Paderborn*

Die Teststrecke ist in Abbildung 6.27 zu sehen. Die Ansicht links zeigt den schematischen Streckenverlauf in der Draufsicht. Die Ansicht rechts zeigt das Leitstellengebäude mit der darunter liegenden Wartungshalle, die am Ende des Bahnhofsabzweigs aufgestellt ist.

Aus den vorliegenden Streckendaten und dem bereits beschriebenen künstlich eingebrachten Schienenversatz wurden Eingangsdaten für die Simulation erzeugt. Besonders interessant ist dabei der zeitliche Verlauf im Bereich des Schienenversatzes.

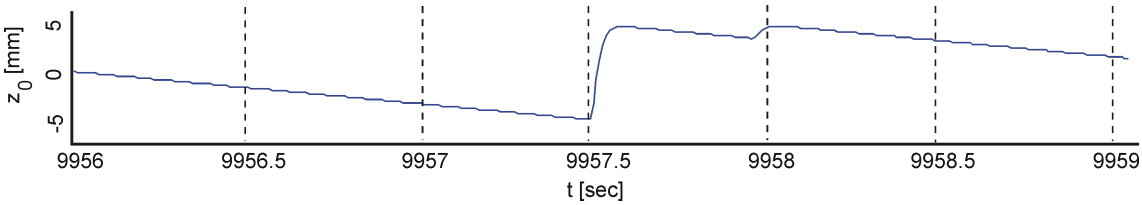

Abbildung 6.28: Vertikalverlauf der Schiene

Abbildung 6.28 zeigt den relevanten Bereich. Das Fahrzeug erreicht den Sprung zum Zeitpunkt $t = 9957,5$ sec. Dies ist bereits eine fortgeschrittene Zeit, d. h. die Simulation konnte bereits einige Optimierungsläufe durchführen. Die durch die Optimierung erzeugte Vorgabetrajektorie ist in Abbildung 6.29 zu sehen. An dieser Stelle ist es wichtig, darauf hinzuweisen, dass die Vorgabetrajektorie bei einer *reinen* Skyhook-Federung hier einer *waagerechten* Linie entspricht ($v_{z,soll}(t) = 0$), was den Fortschritt des neuen Ansatzes gegenüber der klassischen Skyhook-Federung klar erkennbar macht.

Interessant ist, dass die Optimierung als Reaktion auf die Störung die Kurve zunächst ansteigen und dann stark abfallen lässt. Das Ergebnis ist eine teilweise Kompensation des Sprunges der Schiene. Dieses Verhalten findet sich in abgeschwächter Form auch beim zweiten, kleineren Sprung.

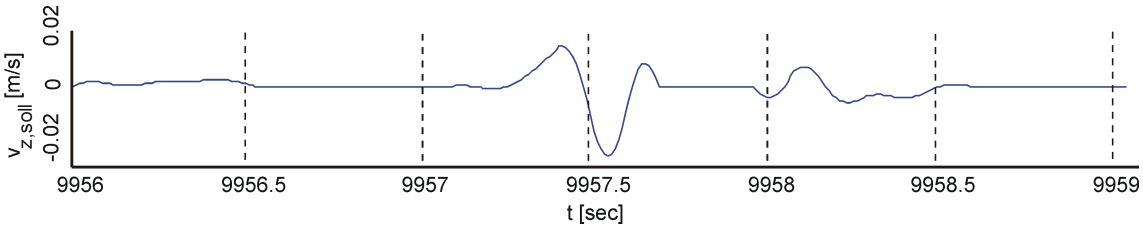

Abbildung 6.29: Geschwindigkeitsvorgabe des Shuttleaufbaus

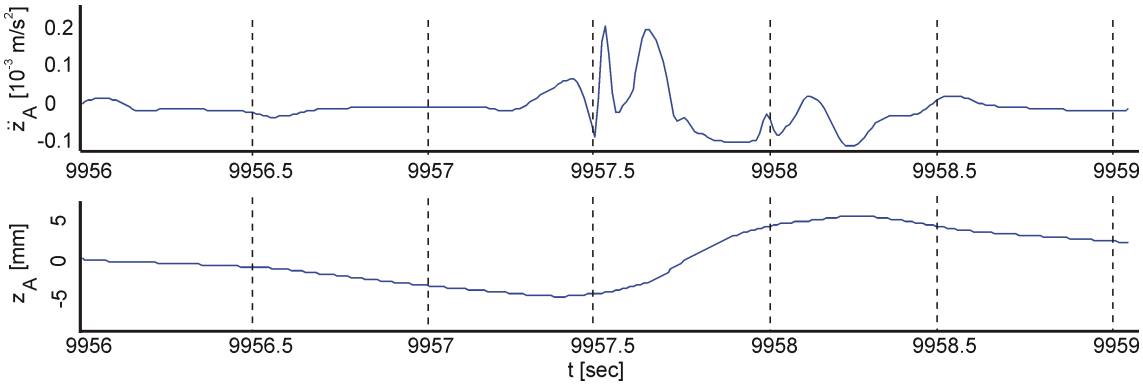

Abbildung 6.30: Zeitantwort des Shuttleaufbaus

Die Zeitantwort des Shuttles ist in Abbildung 6.30 zu sehen. Auch hier zeigt sich die Beschleunigung des Shuttles in negativer z-Richtung kurz vor dem ersten Sprung in der Schiene. Offensichtlich bewirkt dies einen betragsmäßig kleineren Anstieg der Beschleunigung beim Durchlaufen des Sprungs.

Besonders hervorzuheben ist darüber hinaus die Entwicklung des Zielgrößenverlaufs während der Optimierung (Abbildung 6.31). Die Abbildung zeigt den Zielgrößenverlauf für $\phi_{\ddot{z}}$ im betrachteten Bereich 2. Wie zu sehen ist, konnten die Zielgrößenwerte um die Stützstellen 10, 40, 70 und 100, an denen jeweils die beschriebenen Schienenversätze eingebaut wurden, um den Faktor 5 bis 6 reduziert werden. Die Aufbaubeschleunigung des Shuttles reduzierte sich dabei um die Hälfte.

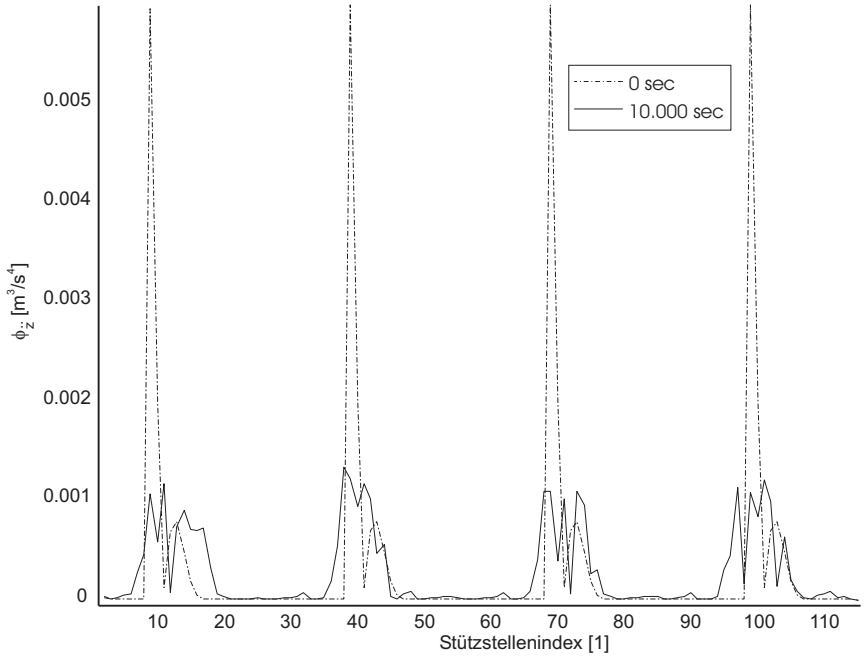

Abbildung 6.31: Verlauf der Zielgrößen bei Streckenabschnitt 2 zu Beginn und nach 10.000 sec

Zusammenfassend lässt sich sagen, dass sich durch die verteilte Online-Optimierung neue Möglichkeiten für Selbstoptimierung in mechatronischen Systemen ergeben. Gerade bei häufig wiederkehrenden Ereignissen, die bei unterschiedlichen mechatronischen Systemen auftreten, was im Bereich Transport und Verkehr besonders häufig zu erwarten ist, lässt sich das Potential voll ausschöpfen. Ein wichtiger Lösungsansatz ist dabei die Zusammenführung der Optimierungsdaten. Ein der Fahrstrecke zugeordnetes Überwachungselement (hier: Bereichskontrolle) löst auch das Problem, dass zur Optimierung die Daten gesammelt und verglichen werden müssen. Eine rein dezentrale Lösung, die nur die Fahrzeuge einbezieht, benötigt deutlich mehr Kommunikation, da für jeden Optimierungsschritt zunächst Fahrzeuge gesucht werden müssen, die den gleichen Abschnitt in vergleichbarer Konfiguration überfahren haben.

Das Szenario ist auf andere Problemfälle übertragbar. Für Anwendungen, die ein, zumindest abschnittsweises, Optimum suchen, wie beispielsweise Energieverbrauch für einen größeren Streckenabschnitt, sind allerdings andere Optimierungsalgorithmen erforderlich.

Kapitel 7

Zusammenfassende Diskussion und Ausblick

Die höchsten Türme fangen beim Fundament an (Thomas Alva Edison)

Selbstoptimierende mechatronische Systeme sind nicht neu – sie sind die Summe von vorhandenen Technologien und bauen darauf auf; darin liegt ihre Stärke. Methoden zur Verbesserung von Systemen sind zahlreich; jede Domäne der Mechatronik, von der Mechanik bis zur Informatik, bietet eine Vielzahl von erprobten Verfahren, die darauf warten in einem mechatronischen System vereint zu werden. Viele Ansätze sind dabei sehr ähnlich. Ob nichtlineare Optimierung oder verhaltensbasierte Agentensysteme – die mathematischen Grundlagen zur Suche nach einem Optimum sind oft gleich.

Was selbstoptimierende Systeme von bisherigen Ansätzen unterscheidet, ist die *Ganzheitlichkeit*. Der Entwurf solcher Systeme setzt voraus, von Anfang an das Gesamte im Blick zu haben, ohne frühzeitig in einzelne Domänen zu flüchten. Der Gedanke, den die Mechatronik vorgezeichnet hat, wird damit weitergeführt.

Der Entwurf spielt eine zentrale Rolle für zukünftige Entwicklungen. Systeme, die sich in ihrer Struktur – und sei es nur in der Struktur ihrer Regelung – zur Laufzeit verändern können, benötigen neue Ansätze zur Modellierung. Entscheidend dabei ist, bekannte Methoden für den Entwurf, sowohl des technischen Systems als auch der Informationsverarbeitung, weiterhin nutzen zu können. Klassische, nicht rekonfigurierbare Systeme müssen sich mit den gleichen Werkzeugen und Methoden entwickeln lassen wie selbstoptimierende Systeme.

Modellbildung bildet die Grundlage für den Entwurf komplexer Systeme. Simulierte Systeme dienen heute der Auslegung und dem Entwurf. Es wird viel Aufwand getrieben, um solche Modelle zu erstellen. Die Selbstoptimierung zeigt einen Weg, durch die modellbasierte Online-Optimierung dieses ausführbare Wissen im laufenden Betrieb zu nutzen.

Die Ausführung der Informationsverarbeitung erfordert für selbstoptimierende mechatronische Systeme höhere Rechenleistungen als für konventionelle Systeme. Jedoch ist nicht allein der Rechenaufwand an sich die Herausforderung, sondern vielmehr das *Wie*. Wie muss die Informationsverarbeitung beim Entwurf strukturiert sein, um den Anforderungen an den Betrieb des technischen Systems gerecht zu werden? Welche Anforderungen stellen sich an die numerische Verarbeitung?

Zukünftige Arbeiten müssen die beschriebenen Ansätze verfeinern und konkretisieren. Ein wesentlicher Schritt wird sein, ein Werkzeug für die Modellierung von

rekonfigurierbaren Systemen zu entwickeln. Aber auch im Bereich der Informationsverarbeitung ist noch viel zu leisten. Dabei stellt sich auch die Frage nach der richtigen Sicht für den Entwickler. Wie werden solche komplexen Systeme in Zukunft beschrieben, und wie kann die Funktion sichergestellt werden?

Laufzeitumgebungen wie IPANEMA können als Plattformen für den Entwurf, wie auch für die Ausführung dienen. Die entworfenen Systeme können dadurch direkt realisiert werden. Eine Neuimplementierung nach der Laborphase ist dann nicht mehr nötig, was die sogenannte *time-to-market* erheblich reduzieren kann.

Inzwischen wurden bereits verschiedene Ansätze, die in dieser Arbeit zusammengefasst sind, im Sonderforschungsbereich 614 der Universität angewendet und verfeinert. Insbesondere sind hier die Arbeiten der Herren Gambuzza, Vöcking und Münch unter Leitung von Herrn Prof. Dr.-Ing J. Lückel hervorzuheben[36]. In verschiedenen Arbeiten wird die Online-Optimierung der im Abschnitt 6.3 dargestellten Trajektorien-Optimierung, verbunden mit den verhaltensbasierten Ansätzen aus Abschnitt 6.2 weiter entwickelt [LMH05], [MVH05]. Neue Optimierungsansätze kommen hinzu.

Darüber hinaus ist die Entwicklung der Informationsverarbeitung weiter fortgeschritten. Ein Meilenstein ist hier sicher die Integration der Werkzeuge CAMel-View und Fujaba[37], die sowohl die Ansätze der rekonfigurierbaren Systeme (vgl. Abschnitt 3.2), als auch der hybriden Komponenten [Bur06] in einem ersten Schritt verbindet [BGH+07], [BGM+08].

Ich bin glücklich und stolz meinen Beitrag für diese großen Entwicklungen geleistet zu haben.

[36] Inzwischen werden die Arbeiten dieser Gruppe unter neuer Leitung von Herrn Prof. Dr.-Ing. A. Trächtler weitergeführt.

[37] http://www.fujaba.de/

Anhang A

Literaturverzeichnis

[ACH+95] Alur, R., C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis und S. Yovine: *The algorithmic analysis of hybrid systems.* Theoretical Computer Science, 138(1):3–34, 1995.

[Ark98] Arkin, R. C.: *An Behavior-based Robotics.* MIT Press, Cambridge, MA, USA, 1998.

[Att00] Attoui, A.: *Real-Time and Multi-Agent Systems.* Springer, Berlin, 1. Auflage, 2000.

[AW94] Astrom, K.J. und B. Wittenmark: *Adaptive Control.* Addison-Wesley Longman, 1994.

[Azi90] Azizi, S.A.: *Entwurf und Realisierung digitaler Filter.* Oldenbourg, München, 5. Auflage, 1990.

[Bec03] Becker, M.: *Mechatronischer Entwurf eines reversierenden, hydraulischen Antriebs-aktors für die aktive Fahrzeugfederung.* Dissertation, Universität Paderborn, VDI-Verlag, Düsseldorf, 2003.

[BFG+97] Bonasso, R.P., R. Firby, E. Gat, D. Kortenkamp, D. Miller und M. Slack: *Experiences with an architecture for intelligent, reactive agents.* Journal of Experimental and Theoretical Artificial Intelligence (JETAI), 9:237–256, 1997.

[BGGO04] Burmester, S., H. Giese, A. Gambuzza und O. Oberschelp: *Partitioning and Modular Code Synthesis for Reconfigurable Mechatronic Software Components.* In: *European Simulation and Modelling Conference (ESMc'2004)*, Seiten 66–73, Paris, France, October 2004.

[BGH+07] Burmester, S., H. Giese, S. Henkler, M. Hirsch, M. Tichy, A. Gambuzza, E. Munch und H. Vocking: *Tool Support for Developing Advanced Mechatronic Systems: Integrating the Fujaba Real-Time Tool Suite with CAMeL-View.* In: *29th International Conference on Software Engineering*, Seiten 801–804. IEEE Computer Society Washington, DC, USA, 2007.

[BGM+08] Burmester, Sven, H. Giese, E. Münch, O. Oberschelp, F. Klein und P. Scheideler: *Tool Support for the Design of Self-Optimizing Mechatronic Multi-Agent Systems.* International Journal on Software Tools for Technology Transfer (STTT), 8(4):1–16, December 2008. (to appear).

[BGO04a] Burmester, S., H. Giese und O. Oberschelp: *Hybrid UML Components for the Correct Design of Self-optimizing Mechatronic Systems.* Technischer Bericht tr-ri-03-246, University of Paderborn, Paderborn, Germany, January 2004.

[BGO04b] Burmester, S., H. Giese und O. Oberschelp: *Hybrid UML Components for the Design of Complex Self-optimizing Mechatronic Systems.* In: *1st International Conference on Informatics in Control, Automation and Robotics (ICINCO 2004)*, Seiten 222–229, Setubal, Portugal, August 2004.

[BGO06] Burmester, S., H. Giese und O. Oberschelp: *Hybrid UML Components for the Design of Complex Self-optimizing Mechatronic Systems.* In: Braz, J., H. Araújo, A. Vieira und B. Encarnaçao (Herausgeber): *Informatics in Control, Automation and Robotics I*, Seiten 281–288. Springer, Berlin/Heidelberg, 2006.

[Blu78] Blum, H.: *Numerical Integration of Large Scale System using Separate Step-sizes.* International Symposium on Simulation Software and Numerical Methods for Differential Equations, Blacksburg, Va, Seiten 19–23, 1978.

[Boe81] Boehm, B. W.: *Software Engineering Economics.* Prentice Hall, Englewood Cliffs, NJ, 1981.

[Boi05] Boiko, V.: *Erweiterung des CAE-Werkzeugs CAMeL-View um eine modulare Codegenerierung zur Unterstützung hybrider UML Komponenten.* Studienarbeit, Universität Paderborn, 2005.

[Bom07] Bombardier Transportation: *Mechatronic Bogie: New boost for the intelligent bogie by Bombardier.* http://www.bombardier.com/, 2007.

[Bra87] Bratman, M.: *Intention, plans, and practical reason.* Harvard University Press, Cambridge, MA, 1987.

[Bro71] Brockhaus, F. A.: *Brockhaus Enzyklopädie.* F. A. Brockhaus, Wiesbaden, 1971.

[BSMM97] Bronstein, I. N., K. A. Semendjajew, G. Musiol und H. Mühlig: *Taschenbuch der Mathematik.* Verlag Harri Deutsch, Frankfurt am Main, 3. Auflage, 1997.

[Bur06] Burmester, S.: *Model-Driven Engineering of Reconfigurable Mechatronic Systems.* Dissertation, Universität Paderborn, Logos Verlag, Berlin, 2006.

[But87] Butcher, J. C.: *The numerical analysis of ordinary differential equations: Runge Kutta and general linear methods.* John Wiley & Sons, Chichester, 1987.

[CB92] Chen, S. und S. A. Billings: *Neural networks for nonlinear dynamic system modelling and identification.* International Journal of Control, 56(2):319–346, 1992.

[Clö02] Clössner, Wolfgang: *Rennbahn mit Hindernissen. NBS Köln - Frankfurt.* LOK MAGAZIN, 41(252):6–9, 2002. GeraNova Zeitschriftenverlag.

[CWM98] Conrad, M., M. Weber und O. Muller: *Towards a methodology for the design of hybrid systems in automotive electronics.* Proc. of ISATA, 98, 1998.

[Deu07] Deutsche Forschungsgemeinschaft: *Geförderte Projekte: KONDISK.* http://gepris.dfg.de (Internetdatenbank der Deutschen Forschungsgemeinschaft), 2007.

[DFG07] DFG-Schwerpunkt 1125: *Internetauftritt des DFG-Schwerpunkts 1125 – Kooperierende Teams mobiler Roboter in dynamischen Umgebungen.* http://www.ais.fraunhofer.de/dfg-robocup/index.html, 2007.

[DFL02] Dylla, F., A. Ferrein und G. Lakemeyer: *Acting and Deliberating using Golog in Robotic Soccer – A Hybrid Architecture.* In: *3rd International Cognitive Robotics Workshop*, Edmonton, Alberta, Canada, 2002.

[DK07] DB-Konzern: *Internetseite: Die Neubaustrecke Köln – Rhein/Main.* http://www.db.de/(...)/koeln__rhein__main.html, 2007.

[DO00] Deppe, M. und O. Oberschelp: *Real-Time Support for Online Controller Supervision and Optimisation.* In: *IFIP WG10. 3/WG10. 4/WG10. 5 International Workshop on Distributed and Parallel Embedded Systems: Architecture and Design of Distributed Embedded Systems*, Seiten 121–130. Kluwer, BV Deventer, The Netherlands, 2000.

[DOM01] Deppe, M., O. Oberschelp und E. Münch: *Echtzeit-Parameter-Optimierung und -Überwachung in mechatronischen Systemen.* In: *5. Magdeburger Maschinenbautage - Entwicklungsmethoden und Entwicklungsprozesse im Maschinenbau*, Seiten 355 – 364, Berlin, September 2001. Otto-von-Guericke-Universität, Magdeburg, Logos Verlag.

[Dür99] Dürr, R.: *Kopplungsansätze mechatronischer Systeme.* Dissertation, Universität Stuttgart, VDI-Verlag, Düsseldorf, 1999.

[Ehr95] Ehrlenspiel, K.: *Integrierte Produktentwicklung.* Hanser Verlag, München, 1995.

[Fei90] Feindt, E.-G.: *Regeln mit dem Rechner - Abtastregelungen mit besonderer Berücksichtigung der digitalen Regelungen.* Oldenbourg Verlag, München, 1990.

[Fer99] Ferber, J.: *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence.* Addison Wesley, Edinburgh, 1999.

[Fer02] Ferreira, C.: *Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence.* Springer, Berlin/Heidelberg, 2002.

[FGH+04] Fischer, R., R. Gscheidle, U. Heider, B. Hohmann, W. Keil, J. Mann, W. Pichler, B. Schlögl, P. Siegmayer, A. Wimmer und G. Wormer: *Fachkunde Kraftfahrzeugtechnik.* Europa Lehrmittel, Haan-Gruiten, 28. Auflage, 2004.

[FGK+04] Frank, U., H. Giese, F. Klein, O. Oberschelp, A. Schmidt, B. Schulz, H. Vöcking und K. Witting: *Selbstoptimierende Systeme des Maschinenbaus - Definitionen und Konzepte*, Band 155. HNI-Verlagschriftenreihe, Paderborn, 2004.

[Föl94] Föllinger, Otto: *Regelungstechnik.* Hüthig, Heidelberg, 1994.

[FNW99] Friesen, V., A. Nordwig und M. Weber: *Toward an object-oriented design methodology for hybrid systems.* Object-oriented technology and computing systems re-engineering, Seiten 1–16, 1999.

[Fri97] Friesen, V.: *Objektorientierte Spezifikation hybrider Systeme.* Dissertation, Technische Universität Berlin, 1997.

[Gag80] Gagné, Robert M.: *Die Bedingungen des menschlichen Lernens.* Schroedel Verlag, Hannover, 1980.

[Gam02] Gambuzza, A.: *Konzept zur verteilten modularen Simulation mechatronischer Systeme.* Diplomarbeit, Universität Paderborn, MLaP, 2002.

[GBSO04] Giese, H., S. Burmester, W. Schäfer und O. Oberschelp: *Modular Design and Verification of Component-Based Mechatronic Systems with Online-Reconfiguration.* In: *12th ACM SIGSOFT Foundations of Software Engineering 2004 (FSE 2004)*, Seiten 179 – 188, Newport Beach, November 2004.

[GCJK97] Guzzoni, D., A. Cheyer, L. Julia und K. Konolige: *Many robots make short work.* AI Magazine, 18(1):55–64, 1997.

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

[GL00] Gausemeier, J. und J. Lückel: *Entwicklungsumgebungen Mechatronik*, Band 80. HNI-Verlagsschriftenreihe, Paderborn, 2000.

[GLR01] Gausemeier, J., J. Lückel und F. Rammig: *Sonderforschungsbereich 614 - Selbstoptimierende Systeme des Maschinenbaus.* Universität Paderborn, Paderborn, 2001. Finanzierungsantrag 2002-2005.

[GO03] Gambuzza, A. und O. Oberschelp: *Distributed Modular Simulation of Mechatronic Systems.* In: *ESMc'2003 – European Simulation and Modelling Conference*, Neapel, Oktober 2003. University of Naples II.

[GOD03] Gambuzza, A., O. Oberschelp und M. Deppe: *Verteilte modulare Simulation mechatronischer Systeme.* VDI-Berichte, 1753:291–309, 2003.

[GP89] Girosi, F. und T. Poggio: *Networks and the Best Approximation Property.* Technischer Bericht AIM-1164, Massachusetts Institute of Technology, Cambridge, Ma, 1989.

[GTB+03] Giese, H., M. Tichy, S. Burmester, W. Schäfer und S. Flake: *Towards the Compositional Verification of Real-Time UML Designs.* In: *European Software Engineering Conference (ESEC)*, Seiten 38–47, Helsinki, Finland, September 2003. ACM press.

[GW84] Gear, C. W. C. und D. R. C. Wells: *Multirate Linear Multistep Methods.* BIT Numerical Mathematics, 24(4):484–502, 1984.

[Hah99] Hahn, M.: *OMD – Ein Objektmodell für den Mechatronikentwurf.* Dissertation, Universität Paderborn, VDI-Verlag, Düsseldorf, 1999.

[Hau83] Haug, E. J. (Herausgeber): *Computer Aided Analysis and Optimization of Mechanical System Dynamics*, Berlin, August 1983. NATO Advanced Study Institute, Springer-Verlag.

[HBN01] Hestermeyer, T., M. Becker und N. Neuendorf: *Nichtlineare ABC-Regelungen mit Operator-Controller-Struktur, abgestimmt auf Führung und Störung der Straße.* In: *Driveability, Haus der Technik*, Essen, 2001.

[HD99] Hagan, M.T. und H. Demuth: *Neural Networks for Control.* American Control Conference, 3:1642–1656, 1999.

[Hel95] Helmersson, A.: *Methods for robust gain scheduling.* Dissertation, Department of Electrical Engineering, Linköping University, 1995.

[Hen97] Henrichfreise, H.: *Prototyping of a LQG Compensator for a Compliant Positioning System with Friction.* In: Gausemeier, Jürgen (Herausgeber): *Workshop TransMechatronik – Entwicklung und Transfer von Entwicklungssystemen der Mechatronik*, Band 23, 1997.

[Hes00] Hestermeyer, Th.: *Entwurf, Implementierung und Realisierung einer modellgestützten Regelung für ein Fahrzeug mit volltragendem aktivem Fahrwerk und Motorpumpeneinheiten.* Diplomarbeit, Universität Paderborn, MLaP, 2000.

[HES03] Hestermeyer, T., C. Ettingshausen und P. Schlautmann: *Aktive Federung für Schienenfahrzeuge – Systemaufbau, Regelung und Realisierung.* In: *5. VDI-Mechatroniktagung 2003 – Innovative Produktentwicklung*, Fulda, 2003.

[Hes06] Hestermeyer, T.: *Strukturierte Entwicklung der Informationsverarbeitung für die aktive Federung eines Schienenfahrzeugs.* Dissertation, Universität Paderborn, VDI-Verlag, Paderborn, 2006.

[HGP98] Heimann, B., W. Gerth und K. Popp: *Mechatronik: Komponenten - Methoden - Beispiele.* Fachbuchverlag Leipzig im Carl Hanser Verlag, München, Hannover, 1998.

[HHK+03] Hengesbach, K., P. Hille, F. Koch, J. Lehberger, D. Müser, G. Pyzalla, W. Quadflieg, W. Schlilke und J. Schmidt: *Berufsfeld Metall - Industriemechanik.* Bildungsverlag E1NS-Stam, Troisdorf, 3. Auflage, 2003.

[HHWT95] Henzinger, T. A., P.-H. Ho und H. Wong-Toi: *HyTech: The Next Generation.* In: *IEEE Real-Time Systems Symposium*, Seiten 56–65, 1995.

[Hil01] Hillermeier, C.: *Nonlinear Multiobjective Optimization, A Generalized Homotopy Approach.* Birkhäuser Verlag, Basel, 2001.

[HJW02] Henrichfreise, H., J. Jusseit und M. Weller: *Der mechatronische Entwicklungsprozess am Beispiel der Regelung eines elektromechanischen Positioniersystems. 9.* IIR F&E-Zukunftsforum, Rüsselsheim, 10, 2002.

[HMO04] Hestermeyer, Thorsten, E. Münch und O. Oberschelp: *Sollbahn-Planung für schienengebundene Fahrzeuge.* In: *VDI–Tagung Berechnung und Simulation im Fahrzeugbau*, Würzburg, 2004.

[HO03] Hestermeyer, Thorsten und O. Oberschelp: *Selbstoptimierende Fahrzeugregelung – Verhaltensbasierte Adaption.* In: Gausemeier, J., J. Lückel und J. Wallaschek (Herausgeber): *Intelligente Mechatronische Systeme*, Paderborn, 2003. Heinz Nixdorf Institut, Paderborn.

[HOG04] Hestermeyer, T., O. Oberschelp und H. Giese: *Structured Information Processing For Self-optimizing Mechatronic Systems.* In: *1st International Conference on Informatics in Control, Automation and Robotics (ICINCO 2004)*, Setubal, Portugal, August 2004. IEEE.

[Hon98] Honekamp, U.: *IPANEMA – Verteilte Echtzeit-Informationsverarbeitung in mechatronischen Systemen.* Dissertation, Universität Paderborn, VDI-Verlag, Düsseldorf, 1998.

[HP87] Harel, D. und A. Pnueli: *Statecharts: A visual formalism for complex systems.* Science of computer programming, 8(3):231–274, 1987.

[Hör07] Hörner, J.: *Internetauftritt von Mathematik-Online.* http://mo.mathematik.uni-stuttgart.de/, 2007.

[HSŻG92] Hunt, K. J., D. Sbarbaro, R. Żbikowski und P. Gawthrop: *Neural networks for control systems: a survey.* Automatica (Journal of IFAC), 28(6):1083–1112, 1992.

[HW91] Hairer, E. und G. Wanner: *Solving Ordinary Differential Equations II, Stiff and Differential-Algebraic Problems.* Springer-Verlag, Berlin, Heidelberg, 1991.

[HW98] Henrichfreise, H. und C. Witte: *Beobachtergestützte nichtlineare Kompensation trockener Reibung in einem Positionierantrieb.* Automatisierungstechnik, 46, 1998. Herrn Prof. Dr.-Ing. Joachim Lückel zum 60. Geburtstag gewidmet.

[IBM07] IBM (Rational): *Unified Modeling Language (Internetauftritt der Fa. IBM und Rational zu UML).* http://www-306.ibm.com/software/rational/uml/, 2007.

[ILM92] Isermann, R., K.-H. Lachmann und D. Matko: *Adaptive Control Systems.* Prentice Hall, New York, 1992.

[iXt06] iXtronics GmbH, Paderborn: *Internetauftritt CAMeL-View.* http://www.ixtronics.com, 2006.

[JW98] Jennings, N. R. und M. J. Wooldridge: *Agent technology: foundations, applications, and markets.* Springer-Verlag, New York, 1998.

[Kas85] Kasper, R.: *Entwicklung und Erprobung eines instrumentellen Verfahrens zum Entwurf von Mehrgrößenregelungen.* Dissertation, Universität Paderborn, Paderborn, 1985.

[KHH04] Kiguchi, K., H. Henrichfreise und K. Hesseler: *Friction compensation of the electromechanical drive systems using neural networks.* Industrial Electronics Society, 2004. IECON 2004. 30th Annual Conference of IEEE, 2:1758 – 1762, 2004.

[KK07] Kleinjohann, Bernd und L. Kleinjohann: *Script zur Vorlesung: Intelligenz in Eingebetteten Systemen.* Universität Paderborn, 2007.

[KLJS90] Kasper, R., J. Lückel, K.-P. Jäcker und J. Schröder: *CACE tool for multi-input, multi-output systems using a new vector optimization method.* International Journal of Control, 51(5):963–993, 1990.

[KLM96] Kaelbling, L. P., M. L. Littman und A. W. Moore: *Reinforcement Learning: A Survey.* Journal of Artificial Intelligence Research, 4:237–285, 1996.

[Kno01] Knohl, T.: *Anwendung künstlicher neuronaler Netze zur nichtlinearen adaptiven Regelung.* Dissertation, Ruhr Universität Bochum, VDI-Verlag, Düsseldorf, 2001.

[KO04] Koch, M. und O. Oberschelp: *Simulation of Self Optimizing Mechatronical Systems with Expert System Knowledge.* In: *5th Asian Control Conference (ASCC– 2004)*, Band 3, Melbourne, 2004.

[Kon97] Konolige, K.: *The Saphira architecture: a design for autonomy.* Journal of Experimental & Theoretical Artificial Intelligence, 9(2):215–235, 1997.

[KP92] Kesten, Y. und A. Pnueli: *Timed and Hybrid Statecharts and their Textual Representation.* In: Vytopil, J. (Herausgeber): *Formal Techniques in Real-Time and Fault-Tolerant Systems*, Band 571 der Reihe *Lecture Notes in Computer Science*, Heidelberg, 1992. Springer-Verlag.

[KR99] Kværnø, A. und P. Rentrop: *Low Order Multirate Rung-Kutta Methods in Electric Circuit Simulation.* Technischer Bericht, The Norwegian University of Science and Technology, Trondheim, 1999.

[Kvæ00] Kværnø, A.: *Stability of Multirate Runge-Kutta Schemes.* International J. of Differential Equations and Applications, 1(1):97–105, 2000.

[Lam93] Lamport, Leslie: *Hybrid Systems in TLA+.* In: *Hybrid Systems*, Band 736 der Reihe *Lecture Notes in Computer Science (LNCS)*, Seiten 77–102, Berlin, 1993. Springer-Verlag.

[Lat90] Latombe, J.C.: *Robot Motion Planning.* Kluwer Academic Publishers, Dordrecht, 1990.

[Lau95] Lauzi, M.: *Anwendung der Fuzzy-Logik in automatisierungstechnischen Entscheidungsstrukturen*, Band 490 der Reihe *VDI-Fortschrittsberichte.* VDI-Verlag, Düsseldorf, 1995.

[Lau99] Lauzi, M.: *Modellbasierte adaptive Regelverfahren in der Antriebstechnik, Teil 1: ein praxisorientierter Einstieg.* Automatisierungstechnische Praxis (atp), 41(7):43–51, 1999.

[Lüc02] Lückel, J.: *Mechatronik – Eine Einführung zur Vorlesung: Grundlagen der Regelungstechnik.* Ergänzendes Material zur Vorlesung, Universität Paderborn, 2002.

[LCJR92] Lückel, J., G. Castiglioni, K.-P. Jäker und R. Rutz: *Design of an Active Car Suspension System, Combined with an Active Vibration Absorber.* In: *6th National Symposium on the Effect of Vibration in Environment*, Krakau, 1992.

[LDHS01] Liebig, S., S. Dronka, S. Helduser und M. Stüwing: *Simulation gekoppelter mechanischer und hydraulischer Systeme.* In: *ASIM-Workshop der Fachgruppen STS und GMMS*, Dresden, 2001.

[Lev02] Levinson, W. A.: *Henry Ford's lean vision: enduring principles from the first Ford motor plant.* Productivity Press, Florence, KY, 2002.

[LHLH01] Lückel, J., T. Hestermeyer und X. Liu-Henke: *Generalization of the Cascade Principle in View of a Structured Form of Mechatronic Systems. IEEE.* In: *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2001)*, 2001.

[LHLJ00a] Liu-Henke, X., J. Lückel und K.-P. Jäker: *Development of an Active Suspension/Tilt System for a Mechatronic Railway Carriage.* In: *1st IFAC-Conference on Mechatronics Systems (Mechatronics 2000)*, Darmstadt, 2000.

[LHLJ00b] Liu-Henke, X., J. Lückel und K.-P. Jäker: *Ganzheitlicher mechatronischer Entwurf eines aktiven Feder-/Neigemoduls.* In: *VDI-Tagung Mechatronik – Mechanisch/Elektrische Antriebstechnik*, Düsseldorf, März 2000. VDI-Verlag.

[LJ96] Lückel, J. und K.-P. Jäker: *Digitale Steuerungen und Regelungen.* Manuskript zur Vorlesung, Universität Paderborn, 1996.

[LK03] Lückel, J. und T. Koch: *Funktionsorientierter Entwurf mechatronischer Systeme.* Skript zur Vorlesung, Universität Paderborn, 2003.

[LMH05] Lückel, J., E. Münch und H. V. T. Hestermeyer: *Online Optimization of a Preview Controller – Structure and Algorithms.* Journal of Theoretical and Applied Mechanics, 43:575–591, 2005.

[LTB+99] Lückel, J., S. Toepper, M. Becker, K.-J. Jäker, W. Kuhlbusch und W. Moritz: *Innovative Prototypes at the MLaP.* In: Lindemann, U., H. Birkhofer, H. Meerkamm und S. Vanja (Herausgeber): *ICED – Proceedings of the 12th International Conference on Engineering Design*, München, 1999.

[Mat06] MathWorks, Inc.: *Internetauftritt Matlab/Simulink.* http://www.mathworks.de/, 2006.

[MG04] Meinders, R. und T. Gutberlet: *Rückrufaktionen setzen Produktentwickler unter Druck.* VDI-Nachrichten, 4:9, 2004.

[Mit97] Mitschke, M.: *Dynamik der Kraftfahrzeuge*, Band B: Schwingungen. Springer Verlag, Berlin, Heidelberg, 1997.

[Mün01] Münch, E.: *Fortentwicklung und Realisierung eines Verfahrens zur gleichzeitigen Optimierung mehrerer Zielgrößen.* Studienarbeit, Universität Paderborn, MLaP, 2001.

[Mün03] Münch, E.: *Mehrgrößenoptimierung – Algorithmusentwicklung und Anwendung an der Spurführung der NBP (Neue Bahntechnik Paderborn).* Diplomarbeit, Universität Paderborn, Paderborn, 2003.

[MOH+04] Münch, E., O. Oberschelp, T. Hestermeyer, P. Scheideler und A. Schmidt: *Distributed Optimization of Reference Trajectories for Active Suspension with Multi-Agent Systems.* In: *18th European Simulation Multiconference (ESM)*, Magdeburg, Germany, 2004.

[MS00] Muller, O. und T. Stauner: *Modelling and Verification using Linear Hybrid Automata.* Mathematical and Computer Modelling of Dynamical Systems, 6(1):71–89, 2000.

[MSC06] MSC.Software: *Internetauftritt ADAMS.* http://www.mscsoftware.com, 2006. Internetseite von MSC.Software: ADAMS, Dymola etc.

[MVH05] Münch, E., H. Vöcking und T. Hestermeyer: *Self-Learning Disturbance Compensation for Active Suspension Systems.* In: *ICINCO 2005 – 2nd International Conference on Informatics in Control, Automation and Robotics*, Barcelona, 2005.

[Nau00] Naumann, R.: *Modellierung und Verarbeitung vernetzter intelligenter mechatronischer Systeme.* Dissertation, Universität Paderborn, VDI-Verlag, Düsseldorf, 2000.

[Neu06] Neue Bahntechnik Paderborn: *Internetauftritt der Neuen Bahntechnik Paderborn.* http://nbp-www.upb.de/, 2006.

[NR98] Naumann, R. und R. Rasche: *Description and Simulation of Hybrid Mechatronic Systems.* In: *International Workshop on Hybrid Systems: Computation and Control*, Berkeley, CA, April 1998.

[Nye06] Nyenhuis, Marcus: *Strukturierter mechatronischer Entwurf einer SbW-Lenkung.* Dissertation, Universität Paderborn, VDI-Verlag, Düsseldorf, 2006.

[Obe98a] Oberschelp, O.: *Multirate-Integrationsverfahren zur Lösung von Differentialgleichungen bei der Simulation mechatronischer Systeme.* Studienarbeit, Universität Paderborn, MLaP, 1998.

[Obe98b] Oberschelp, Oliver: *Entwurf und Implementierung eines flexiblen Codegenerators zur Prozeßkopplung für verteilte Hardware-in-the-Loop Simulationen.* Diplomarbeit, Universität Paderborn, MLaP, 1998.

[OGBG04] Oberschelp, O., A. Gambuzza, S. Burmester und H. Giese: *Modular Generation and Simulation of Mechatronic Systems.* In: *8th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI)*, Seiten 1–6, Orlando, Fl, Juli 2004.

[OHD+01] Oberschelp, O., C. Homburg, M. Deppe, A. Gambuzza und J. Seuss: *Verarbeitungsorientierte Darstellung verteilter hybrider Systeme der Mechatronik.* In: *5. Magdeburger Maschinenbautage*, Seiten 345–354. Logos Verlag Berlin, September 2001.

[OHG04] Oberschelp, O., T. Hestermeyer und H. Giese: *Strukturierte Informationsverarbeitung für selbstoptimierende mechatronische Systeme.* In: *Zweiter Paderborner Workshop Intelligente Mechatronische Systeme*, Nummer 145 in *HNI-Verlagsschriftenreihe*, Seiten 43–56, Paderborn, 2004.

[OHKK02] Oberschelp, Oliver, T. Hestermeyer, B. Kleinjohann und L. Kleinjohann: *Design of self-optimizing agent-based controllers.* In: *3rd International Workshop on Agent Based Simulation*, Passau, April 2002.

[Opt07] Optimization Technology Center: *Homepage des Optimization Technology Centers.* http://www.ece.northwestern.edu/OTC/, 2007.

[OV04] Oberschelp, O. und H. Vöcking: *Multirate simulation of mechatronic systems.* In: *International Conference on Mechatronics (ICM'04)*, Seiten 404–409, Istanbul, 2004. IEEE.

[Paw55] Pawlow, I.P.: *Ausgewählte Werke.* Akademie-Verlag, Berlin, 1955.

[PB97] Pahl, G. und W. Beitz: *Konstruktionslehre - Methoden und Anwendung*, Band 4. Springer-Verlag, Berlin, Heidelberg, 1997.

[PS92] Pottmann, M. und D. Seborg: *Identification of nonlinear processes using reciprocal multiquadratic functions.* Journal of Process Control, 2(4):189 – 203, 1992.

[Ras05] Rasche, R.: *Kreuzungsmanagement – Informationtechnische Vernetzung autonomer Fahrzeuge als Beispiel für Selbstoptimierung im Maschinenbau.* Dissertation, Universität Paderborn, VDI-Verlag, Düsseldorf, 2005.

[Ray01] Raymond, Eric S.: *The Cathedral and the Bazaar.* O'Reilly and Associates, Sebastopol, CA, 2001.

[RG94] Rentrop, P. und M. Günther: *Partitioning and Multirate Strategies in Latent Electric Circuits.* International Series of Numerical Mathematics. Birkhäuser, Basel, 1994.

[RG95] Rao, A. S. und M. P. Georgeff: *BDI agents: From theory to practice.* In: *First International Conference on Multi-Agent Systems (ICMAS-95)*, Seiten 312–319, San Francisco, Ca, 1995.

[Rit07] Ritter, A.: *Internetauftritt: Digitale Fabrik: Agententechnik (Fraunhofer-Institut für Produktionstechnik und Automatisierung (IPA)).* http://www.ipa.fhg.de/Arbeitsgebiete/digitalefabrik/fabrik/, 2007.

[Rük96] Rükgauer, A.: *Modulare Simulation mechatronischer Systeme mit Anwendung in der Fahrzeugdynamik.* Dissertation, Universität Stuttgart, VDI-Verlag, Düsseldorf, 1996.

[RN03] Russel, S. und P. Norvig: *Artificial Intelligence.* Prentice Hall, NJ, 2003.

[Rob07] Robert Bosch GmbH: *Internetseite der Robert Bosch GmbH zu ESP®.* http://www.bosch-esperience.de/de/language1/index.html, 2007.

[Roy87] Royce, W. W.: *Managing the development of large software systems: concepts and techniques.* In: *ICSE '87: Proceedings of the 9th international conference on Software Engineering*, Seiten 328–338, Los Alamitos, CA, 1987. IEEE Computer Society Press.

[RSW96] Rentrop, P., K. Strehmel und R. Weiner: *Ein Überblick über Einschrittverfahren zur numerischen Integration in der technischen Simulation.* GAMM-Mitteilungen, 1:9–43, 1996.

[SB98] Sutton, R.S. und A. Barto: *Reinforcement Learning: An Introduction.* The MIT Press, Cambridge, Ma, 1998.

[Sch86] Schwarz, H. R.: *Numerische Mathematik.* B. G. Teubner, Stuttgart, 1986.

[Scz95] Sczyrba, C.: *Entwurf und Implementierung einer portablen echtzeitfähigen Laufzeitplattform zur Hardeware-in-the-Loop-Simulation mechatronischer Systeme.* Diplomarbeit, Universität Paderborn, Fachbereich Informatik und MLaP, 1995.

[SJW06] Schäfer, E., K.-P. Jäker und A. Wielenberg: *Entwicklung und Inbetriebnahme einer aktiven Federung für ein geländegängiges Nutzfahrzeug.* In: *3. VDI/VDE-Fachtagung zur Steuerung und Regelung von Fahrzeugen und Motoren (AUTOREG 2006)*, Wiesloch, 7.-8. März 2006.

[Ski53] Skinner, B.F.: *Science and human behavior.* Macmillan New York, 1953.

[SS01] Schiehlen, W. und C. Scholz: *Step Size Control of Simulator Coupling for Multibody Systems.* In: *Co-Simulation for Mechatronic Systems*, Stuttgart, Oktober 2001. DLR.

[Sta01] Stauner, T.: *Systematic Development of Hybrid Systems.* Dissertation, Technische Universität München, Deutsche Nationalbibliothek, Leipzig, Frankfurt am Main, 2001.

[Ste88] Stearns, S. D.: *Digitale Verarbeitung analoger Signale.* Oldenbourg Verlag, München, 4 Auflage, 1988.

[Sto00] Stone, P.H.: *Layered Learning in Multiagent Systems: A Winning Approach to Robotic Soccer.* The MIT Press, Cambridge, Ma, 2000.

[Sto04] Stolpe, R.: *Verteilte kommunizierende mechatronische Funktionsmodule: Von der mechatronisch funktionalen Modularisierung bis zur verteilten HIL-Realisierung.* Dissertation, Universität Paderborn, VDI-Verlag, Düsseldorf, 2004.

[Stu96] Stumpe, M.: *Vergleichende Untersuchung von Integrationsverfahren zur numerischen Lösung von gewöhnlichen nichtlinearen Differentialgleichungen.* Diplomarbeit, Universität Paderborn, MLaP, 1996.

[TGR+03] Teich, J., M. Glesner, F. Rammig, W. Rosenstiel und H. Schmeck: *Internetauftritt des Schwerpunktprogramms (SPP 1148) der Universität Erlangen-Nürnberg.* http://www12.informatik.uni-erlangen.de/spprr/abstract.php, 2003.

[The07] The Object Management Group: *Internetauftritt der Object Management Group: Unified Modelling Language.* http://www.uml.org/, 2007.

[Toe02] Toepper, S.: *Die mechatronische Entwicklung des Parallelroboters TRIPLANAR.* Dissertation, Unversität Paderborn, VDI-Verlag, Düsseldorf, 2002.

[Uni98] Universität Paderborn: *Internetauftritt des Projektes METRO – Einsatz massiv paralleler Rechner beim Entwurf und der Realisierung komplexer mechatronischer Systeme.* http://wwwhni-alt.upb.de/projekte/projekt.php3?id=318, 1998.

[Uni06] Universität Paderborn: *Internetauftritt Sonderforschungsbereich 376: Massive Parallelität: Algorithmen · Entwurfsmethoden · Anwendungen.* http://wwwcs.uni-paderborn.de/SFB376/, 2006.

[Ver87] Verein Deutscher Ingenieure: *Einwirkung mechanischer Schwingungen auf den Menschen – Blatt 2: Bewertung.* Düsseldorf, 1987.

[Ver03] Verein Deutscher Ingenieure: *Entwicklungsmethodik für mechatronische Systeme.* Düsseldorf, 2003.

[Ver04] Verband der Automobilindustrie (VdA): *Wertschöpfung steigt auf 75 Prozent - Netzwerke schaffen neue Qualität - Zulieferer profitieren von globalem Wachstum.* http://www.vda.de, 2004. Online-Pressemitteilung.

[VO04] Vöcking, Henner und O. Oberschelp: *Modelling of Multirate Effects in Mechatronic Systems.* In: *12th Mediterranean Conference on Control and Automation*, Kusadasi, 2004.

[Vöc03] Vöcking, H.: *Multirate-Verfahren und Umschaltstrategien für verteilte Reglersysteme.* Diplomarbeit, Universität Paderborn, MLaP, 2003.

[Wal03] Wallaschek, J.: *Mechatronik.* Skript zur Vorlesung Mechatronik, Universität Paderborn, 2003.

[WC01] Wooldridge, M. und P. Ciancarini (Herausgeber): *Agent-Oriented Software Engineering*, Band 1957 der Reihe *Lecture Notes in Computer Science*, Limerick, Ireland, June 2001. Springer Verlag. Revised Papers.

[Web71] Weber, W.: *Adaptive Regelungssysteme I – Allgemeine Struktur und Erkennungsmethoden.* R. Oldenbourg, München und Wien, 1971.

[Wer07] Werske, A.: *Internetauftritt: Die schnellsten Züge der Welt.* www.hochgeschwindigkeitszuege.com, 2007.

[Wie96] Wieting, R.: *Hybrid High-level Nets.* In: *1996 Winter Simulation Conference*, Seiten 848 – 855, Coronado, CA, 1996.

[Wik06] Wikipedia – Die freie Enzyklopädie: *Optimierung.* http://de.wikipedia.org/wiki/Optimierung, 07 2006. Internetseite zum Begriff Optimierung.

[Wik07] Wikipedia – Die freie Enzyklopädie: *Proaktiv.* http://de.wikipedia.org/wiki/Proaktiv, 08 2007. Internetseite zum Begriff Proaktiv.

[WJ95] Wooldridge, Michael und N. Jennings: *Agent theories, architectures, and languages: a survey.* In: Wooldridge, Michael und N. Jennings (Herausgeber): *Intelligent Agents*, Berlin, Heidelberg, 1995. Springer Verlag.

[ZRL+01] Zanella, M., M. Robrecht, T. Lehmann, R. Gielow, A. de Freitas Francisco und A. Horst: *RABBIT – A Modular Rapid Prototyping Platform for Distributed Mechatronic Systems*. In: *SBCCI 2001 – XIV Symposium on Integrated Circuits and Systems Design*, Brasilia, 2001.

Anhang B

Stichwortverzeichnis