Sonderforschungsbereich 614

Selbstoptimierende Systeme des Maschinenbaus

Selbstoptimierende Systeme des Maschinenbaus

Definitionen und Konzepte

Ursula Frank Holger Giese Florian Klein
Oliver Oberschelp Andreas Schmidt Bernd Schulz
Henner Vöcking Katrin Witting

Dieses Buch ist im Sonderforschungsbereich 614 – Selbstoptimierende Systeme des Maschinenbaus, Universität Paderborn – entstanden und wurde auf seine Veranlassung unter Verwendung der ihm von der Deutschen Forschungsgemeinschaft zur Verfügung gestellten Mittel veröffentlicht.

Vorwort

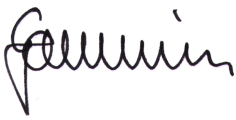
Moderne maschinenbauliche Erzeugnisse beruhen auf dem engen Zusammenwirken von Mechanik, Elektronik, Regelungstechnik und Softwaretechnik, was durch den Begriff Mechatronik zum Ausdruck kommt. Ein wesentlicher Treiber dieser Entwicklung ist die Informationstechnik, die sich fortsetzt und künftig Maschinen mit einer inhärenten Teilintelligenz ermöglichen wird. Diese Perspektive bezeichnen wir als Selbstoptimierung. Unter Selbstoptimierung eines technischen Systems wird die endogene Änderung der Ziele des Systems auf veränderte Umfeldbedingungen und die daraus resultierende zielkonforme autonome Anpassung der Parameter und ggf. der Struktur und somit des Verhaltens dieses Systems verstanden. Damit geht Selbstoptimierung über die bekannten Regel- und Adaptionsstrategien wesentlich hinaus; Selbstoptimierung ermöglicht handlungsfähige Systeme mit inhärenter „Intelligenz“, die in der Lage sind, selbstständig und flexibel auf veränderte Betriebsbedingungen zu reagieren.

Der im Juli 2002 gestartete Sonderforschungsbereich 614 „Selbstoptimierende Systeme des Maschinenbaus“ verfolgt die langfristige Zielsetzung, das Wirkparadigma der Selbstoptimierung für den Maschinenbau zu erschließen und ein Instrumentarium zur Entwicklung derartiger Systeme zu schaffen.

Während es viele Beispiele für den Nutzen der Mechatronik gibt, zeichnen sich die Nutzenpotentiale der Selbstoptimierung von maschinenbaulichen Systemen erst in groben Konturen ab. Offensichtlich ist Phantasie gefragt, Maschinen mit inhärenter Teilintelligenz zu definieren. Eine wesentliche Voraussetzung zur Bewältigung dieser Herausforderung ist die wissenschaftliche Durchdringung und die ingenieurmäßige Aufbereitung des Wirkparadigmas der Selbstoptimierung.

Diese Herausforderung motivierte ein Team, bestehend aus Dipl.-Wirt.-Ing. Ursula Frank, Dr. rer. nat. Holger Giese, Florian Klein, MScIS, Dipl.-Ing. Oliver Oberschelp, Dipl.-Wirt.-Ing. Andreas Schmidt, Dipl.-Ing. Bernd Schulz, Dipl.-Math. Henner Vöcking und Dipl.-Math. Katrin Witting, einen grundlegenden Beitrag für die Gestaltung der Forschungsarbeiten in dem Sonderforschungsbereich 614 zu leisten. Dies ist aus Sicht der Mitglieder des Sonderforschungsbereichs 614 gut gelungen. Im Namen der Mitglieder danke ich dem Team für das vorbildliche Engagement und die geleistete Arbeit. Frau Frank verdient unseren besonderen Respekt; sie hat neben der konzeptionellen Arbeit und der Betreuung ihrer einjährigen Zwillinge zusammen mit Herrn Klein die Redaktionsarbeit geleistet. Ihnen gebührt unser besonderer Dank. Zudem danken wir Dr.-Ing. Norbert Fröhleke, Dr. rer. nat. Rainer Feldmann und Dipl.-Wirt.-Inf. Matthias Tichy für ihre wertvollen Kommentare und Beiträge bei der Erarbeitung dieses Buches.

Paderborn 2004



Prof. Dr.-Ing. Jürgen Gausemeier

Sprecher des Sonderforschungsbereichs 614
„Selbstoptimierende Systeme des Maschinenbaus“
und Herausgeber

Inhaltsverzeichnis

Kapitel 1

Einführung und Motivation

1.1 Ziel und Ausrichtung dieses Buches

Der Sonderforschungsbereich 614 – Selbstoptimierende Systeme des Maschinenbaus der Universität Paderborn beschäftigt sich mit der Erforschung des Paradigmas der Selbstoptimierung und der Entwicklung technischer Systeme, die dieses Paradigma umsetzen. Selbstoptimierende Systeme besitzen die charakteristische Fähigkeit, flexibel auf sich ändernde Umweltbedingungen, Eingriffe des Benutzers oder Einwirkungen des Systems zu reagieren, um autonom ihr Verhalten zu optimieren.

Ziel dieses Buches ist es, den Begriff Selbstoptimierung und damit verbundener Konzepte vorzustellen, wie sie im Sonderforschungsbereich 614 eingesetzt werden. Es fasst damit die Ergebnisse der Diskussion der Interessengruppe Selbstoptimierung zusammen, in der Vertreter aller am Sonderforschungsbereich 614 beteiligten Fachrichtungen gemeinsam eine fachübergreifend einheitliche Terminologie erarbeitet und Potentiale der Selbstoptimierung ausgelotet haben. Ergänzend werden grundlegende Konzepte zur praktischen Umsetzung der dargelegten Prinzipien umrissen und die Konzepte der Selbstoptimierung durch Beispiele illustriert.

1.2 Ein domänenübergreifendes Paradigma

Selbstoptimierende mechatronische Systeme basieren auf komplexen Reglersystemen, deren Funktionalität durch zusätzliche Informationsverarbeitung und Optimierungsverfahren noch erweitert wird. Besondere Bedeutung gewinnt dabei die Vernetzung solcher Reglersysteme, um kollaborative und emergente Selbstoptimierung zu unterstützen. Auf loka-

ler Ebene müssen neben den modell- und verhaltensbasierten Optimierungsverfahren auch Techniken für die Rekonfiguration zur Laufzeit durch geeignete Entwurfsmethoden integriert werden. Eine wesentliche Herausforderung ergibt sich zusätzlich aus dem sicherheitskritischen Charakter der Systeme, der bedingt, dass der Verbund aus Software und technischem System ein vorhersagbar korrektes Verhalten zeigen muss – trotz Vernetzung, Rekonfiguration und der Integration von Techniken der künstlichen Intelligenz.

Selbstoptimierung und damit zusammenhängende Aspekte und Fragestellungen werden in den Domänen Maschinenbau, Regelungstechnik, Elektrotechnik und Informatik, aber auch verwandten Domänen wie Biologie, Chemie, Psychologie oder Volkswirtschaftslehre thematisiert. Die Domänen verwenden dabei jedoch jeweils eigene Terminologien. Gleiche Begriffe sind oft abweichend belegt, während gleichzeitig identische Sachverhalte unterschiedlich ausgedrückt werden. Ziel dieses Buches ist es, die Selbstoptimierung in eine einheitliche Begriffswelt einzuordnen, von ähnlichen Ansätzen abzugrenzen und Gemeinsamkeiten aufzuzeigen. Das Buch hat nicht den Anspruch auf Vollständigkeit, sondern bringt nur so weit wie für die Arbeiten im Sonderforschungsbereich 614 nötig eine Ordnung in die Begriffswelt.

1.3 Aufbau des Buches

Im zweiten Kapitel werden zunächst grundlegende Begriffe wie System, Struktur, Verhalten, Parameter und Ziel eingeführt, die dann die Grundlage der Definition der Selbstoptimierung als abstraktes Prinzip bilden. Anschließend werden Merkmale zur Klassifikation selbstoptimierender Systeme vorgeschlagen.

Das dritte Kapitel beschäftigt sich mit der Frage, wie ein selbstoptimierendes System effizient entworfen und sicher technisch umgesetzt werden kann. Dazu wird mit dem Operator-Controller-Modul (OCM) eine mögliche Architektur für selbstoptimierende mechatronische Systeme erläutert. Hierbei handelt es sich um ein Strukturierungs- und Entwurfskonzept für rekonfigurierbare Reglersysteme mit Schwerpunkt auf der Gliederung der inneren Funktionen. Anschließend wird aufgezeigt, wie mit Hilfe von Lösungsmustern (z.B. Wirkprinzipien, Muster der Softwaretechnik oder Wirkmuster zur Selbstoptimierung) der grundlegende Aufbau und die Funktionsweise selbstoptimierender Systeme durch Wiederverwendung bewährter Lösungsansätze entworfen werden kann.

Das vierte Kapitel gibt einen Überblick zu verwandten Arbeiten und grenzt diese von den Arbeiten im Sonderforschungsbereich 614 ab. Dabei wird aufgezeigt, welche Themen der verwandten Arbeiten für den Sonderforschungsbereich 614 wichtig sind und welche dieser Themen in welcher Art und Weise berücksichtigt werden.

Im fünften Kapitel werden schließlich als Anwendungsbeispiele die Demonstratoren, die im Sonderforschungsbereich 614 entstanden sind, vorgestellt. Für das Antriebs-/Bremsmodul, das Energiemanagement, das Feder-Neige-Modul und die Konvoifahrt wird jeweils umrissen, wie Selbstoptimierung in dem System eingesetzt wird, wie sie abläuft und wie sie unter Verwendung der oben vorgeschlagenen Merkmale klassifiziert werden kann.

Das letzte Kapitel fasst die wichtigsten Ergebnisse zusammen.

Kapitel 2

Definition der Selbstoptimierung

2.1 Grundlegende Begriffe

In diesem Kapitel sollen als Basis für die Definition der Selbstoptimierung in Kapitel 2.2 grundlegende Begriffe definiert werden. Ausgehend von dem Systembegriff werden für selbstoptimierende Systeme wichtige Aspekte wie Parameter, Verhalten, Struktur und Ziele erläutert.

Ein **System** *(system)* ist eine in einem betrachteten Zusammenhang gegebene Anordnung von Elementen, die miteinander in Wechselwirkung stehen. Diese Anordnung wird aufgrund bestimmter Vorgaben gegenüber ihrer Umgebung abgegrenzt (in Anlehnung an DIN 19226 [DIN94]). System

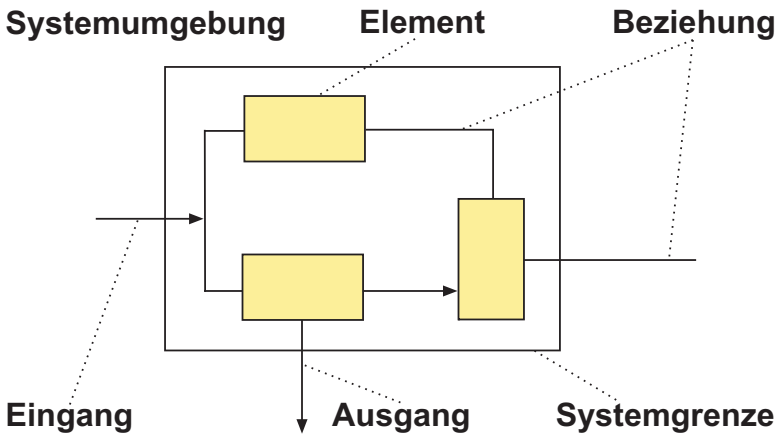


Abbildung 2.1: Definition des Begriffs System

Element

Ein System besteht aus **Elementen** *(element)* (Teilen / Komponenten / Gebilden), welche wiederum als System betrachtet werden können (siehe Abbildung 2.1). Elemente sind durch **Beziehungen** *(relation)* miteinander oder mit Elementen der Systemumgebung verbunden. Dabei wird zwischen gerichteten und ungerichteten Beziehungen unterschieden. Gerichtete Beziehungen zwischen Systemelementen und Elementen der Systemumgebung werden als **Eingang** *(input)* bzw. **Ausgang** *(output)* bezeichnet.

Beziehung

Eingang

Ausgang

Bei den Beziehungen kann es sich beispielsweise um Informationsflussbeziehungen, Lagebeziehungen, Wirkzusammenhänge aber auch um Materialflussbeziehungen handeln. Die Elemente eines Systems und deren Beziehungen zueinander beschreiben als Ganzes die **Struktur** *(structure)* des Systems (gegliederter Aufbau, Gefüge, Anordnungen). Das System wird durch eine **Systemgrenze** *(system border)* (Hüllfläche) von der Systemumgebung abgegrenzt. Die Systemgrenze muss nicht physisch sichtbar sein, es kann sich hier auch um gedankliche Grenzen handeln. Je nach Betrachtungsstandpunkt kann sie unterschiedlich verlaufen. Unter **Systemumgebung** *(environment)* werden Systeme oder Elemente verstanden, die außerhalb der Systemgrenze liegen und Einfluss auf das System nehmen. Die Umwelt, der Benutzer oder andere technische Systeme sind beispielsweise Systeme der Systemumgebung.

Struktur

Systemgrenze

Systemumgebung

Systemparameter

Zustand

Systemparameter *(parameter)* beschreiben die Eigenschaften eines Systems bzw. seiner Elemente und Beziehungen. Ihre gemeinsame Ausprägung zu einem definierten Zeitpunkt wird als **Zustand** *(state)*[1] des Systems bezeichnet. Der Zustand des Systems beinhaltet auch den Zustand seiner Elemente.

Verhalten

Funktion

Aktivität

Das **Verhalten** *(behavior)* eines Systems wird durch dessen Eingangs- und Ausgangsgrößentupel in einem bestimmten Zustand beschrieben. Im Bereich Maschinenbau ist dabei zur detaillierten Verhaltensbeschreibung die Verwendung von Funktionsstrukturen gebräuchlich. Eine **Funktion** *(function)* ist dort als allgemeiner und gewollter Zusammenhang zwischen Eingang und Ausgang eines Systems mit dem Ziel, eine Aufgabe zu erfüllen [PB97] bzw. als lösungsneutral beschriebene Beziehung zwischen Eingangs-, Ausgangs- und Zustandsgrößen [VDI93] definiert. Funktionen beschreiben die Fähigkeiten bzw. das benötigte reaktive Verhalten eines Systems, drücken also eine passive Sicht auf das Systemverhalten aus. Aus Sicht der Informationsverarbeitung wird Systemverhalten durch eine Menge von Prozessen, die als eine Abfolge von **Aktivitäten**

[1] Der Begriff Zustand ist hier nicht gleichbedeutend mit den Begriffen Zustandsgröße oder Zustandsvariable.

(activity) verstanden werden, beschrieben. Eine Aktivität ist ein konkreter, von einem Systemelement durchgeführter Prozessschritt (z.B. „Regelfehler bestimmen“ oder „Sensordaten einlesen“). Im Gegensatz zum Zusammenspiel der Funktionen in einer Wirkstruktur ist die Abfolge der Aktivitäten keinen natürlichen Gesetzmäßigkeiten unterworfen und ergibt sich daher nicht rein reaktiv, sondern muss aktiv gesteuert werden. Bei einer Betrachtung auf Systemebene können verschiedene Aktivitäten einzelner Elemente zu einer **Aktion** *(joint action)* zusammengefasst werden (z.B. „Regeln“). Der Aktionsbegriff wird hier im Sinne eines „gemeinsamen, gezielten Vorgehens“ [Dud00] verstanden. Die Aktion ist somit eine Abstraktion, die eine Beschreibung des Gesamtvorgehens des Systems erlaubt, ohne im Detail darauf einzugehen, welche Schritte von welchen Elementen durchgeführt werden. Ist einem System eine Funktion zugeordnet, so kann die Erfüllung dieser Funktion durch eine Aktion ausgedrückt werden. Es wird zwischen einmaligen und wiederkehrenden Aktionen unterschieden.

Aktion

Ausgewählte Aspekte eines Systems oder eines Systemprozesses werden durch Modelle dargestellt. **Modelle** *(model)* sind Abstraktionen und Vereinfachungen der Realität. Entsprechend des Modellierungszwecks werden die interessierenden Aspekte durch ein anderes beschreibendes oder physikalisches System, d.h. durch ein Modell, abgebildet. Es wird aus der Anwendung bekannter Gesetzmäßigkeiten, einer Identifikation oder auch getroffener Annahmen gewonnen. Das gewählte Modell muss die interessierenden Aspekte des Systems hinreichend genau abbilden.

Modell

Nachfolgend werden die Aspekte **Einflüsse** auf das System, **Ziele** des Systems sowie **Struktur-, Verhaltens- und Parameteranpassungen** vertieft, da sie bei selbstoptimierenden Systemen eine besonders wichtige Rolle spielen.

Einflüsse *(influence)* auf das System können von Systemen aus der Systemumgebung (Umwelt und Benutzer), von anderen technischen Systemen z.B. aus anderen Hierarchieebenen sowie von dem System selbst ausgehen. Einflüsse können den Zweck des Systems unterstützen, behindern oder verhindern. Zweckbehindernde bzw. zweckverhindernde Einflüsse sind Störgrößen. Weiterhin kann zwischen unstrukturierten, nicht vorhersehbaren, nicht geplanten Einflüssen und gezielten, geplanten Einflüssen auf das System unterschieden werden. Letztere werden als Vorgaben bezeichnet. Einflüsse aus der Umwelt, z.B. starker Wind oder Glatteis sind unstrukturiert und oftmals nicht vorhersagbar. Häufig be- bzw. verhindern sie den Zweck des Systems. Der Benutzer gibt dem System Ziele vor, die das System erfüllen soll. Bei einem Transportsys-

Einfluss

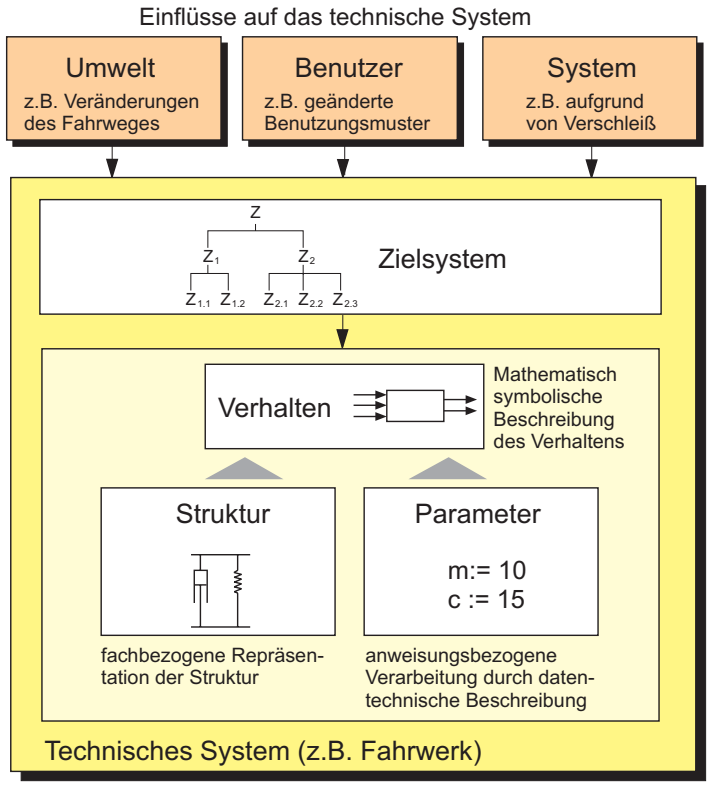


Abbildung 2.2: Die Aspekte Einflüsse, Ziele, Verhalten, Struktur und Parameter eines selbstoptimierenden Systems

tem könnten dieses beispielsweise der Zielbahnhof, die gewünschte Fahrdauer oder der gewünschte Komfort sein. Vorgaben können auch von anderen technischen Systemen, mit denen das betrachtete System direkt in Beziehung steht, erfolgen. Besteht z.B. ein Transportsystem u.a. aus einem Fahrwerk, einem Antriebsmodul und einem Energiemanagementmodul, kann das Energiemanagementmodul den anderen Modulen eine bestimmte Energiemenge zuteilen. Neben den zweckunterstützenden Einflüssen können von dem Benutzer oder von anderen technischen Systemen Störgrößen auf das System einwirken. Einflüsse durch das System selbst werden als innere (endogene) Einflüsse bezeichnet. Verschleiß oder der Bruch einer Feder sind hierfür Beispiele. Neben den Einflüssen auf das System kann es auch zu Nebenwirkungen als ungewollte Ausgangsgrößen aus dem System auf Umwelt, Benutzer und andere technische Systeme kommen (in Abbildung 2.2 nicht dargestellt).

Ein selbstoptimierendes System soll sich entsprechend seines **Zielsystems** *(system of objectives)* (Zielvorgaben) verhalten. Unter einem Zielsystem wird hier ganz allgemein eine Menge zueinander in Beziehung stehender Ziele verstanden. Zielsystem

Ziele *(objectives)* formulieren das geforderte, gewünschte oder zu vermeidende Verhalten eines Systems. Dabei wird zwischen externen, inhärenten und internen Zielen unterschieden. Ziele

Externe Ziele werden von außen durch den Menschen, die Umwelt oder andere technische Systeme an das System herangetragen. Beispielsweise wünscht der Insasse eines Shuttles einen hohen Komfort bei niedrigen Kosten, oder das im Shuttle für die Energieversorgung zuständige Modul gibt dem Feder-Neige-Modul als Ziel einen minimalen Energieverbrauch vor. externe Ziele

Inhärente Ziele spiegeln den Entwurfszweck des Systems wider und sichern seine Grundfunktionalität. Sie werden während der Produktentwicklung festgelegt und dem System vorgegeben. Beispiele für inhärente Ziele sind beispielsweise im Antriebsmodul eines Shuttles das Sichern der Grundfunktionen Antreiben und Bremsen sowie geringer Verschleiß und ein hoher Wirkungsgrad. inhärente Ziele

Interne Ziele sind jene Ziele, die zu einem konkreten Zeitpunkt von einem System verfolgt werden. Im Gegensatz zu den erstgenannten Zielkategorien unterliegen sie der Kontrolle des Systems und können durch Generierung, Auswahl oder Gewichtung externer und inhärenter Ziele oder Anpassung existierender interner Ziele eigenständig festgelegt werden. Hoher Komfort und minimaler Energieverbrauch sind Beispiele für interne Ziele des Feder-Neige-Moduls, wenn das System im Normalbetrieb ist und keine Einschränkungen vorliegen. interne Ziele

Wo im Weiteren – insbesondere in Verbindung mit Optimierung und Anpassung – von den Zielen eines Systems gesprochen wird, bezieht sich dies auf die internen Ziele, sofern nicht explizit anderes angegeben ist. Die verschiedenen Zielklassen werden im Kapitel 2.3.1 noch genauer betrachtet werden.

Veränderungen als Reaktion auf wechselnde Einflüsse spielen für selbstoptimierende Systeme eine zentrale Rolle. Im Weiteren werden zielgerichtete, willentliche Veränderungen als **Anpassungen** *(adaptation)* bezeichnet, während der allgemeinere Begriff **Änderungen** *(variation)* zusätzlich auch extern verursachte und unwillkürliche Veränderungen, z.B. durch Verschleiss, mit einschließt. Anpassung Änderung

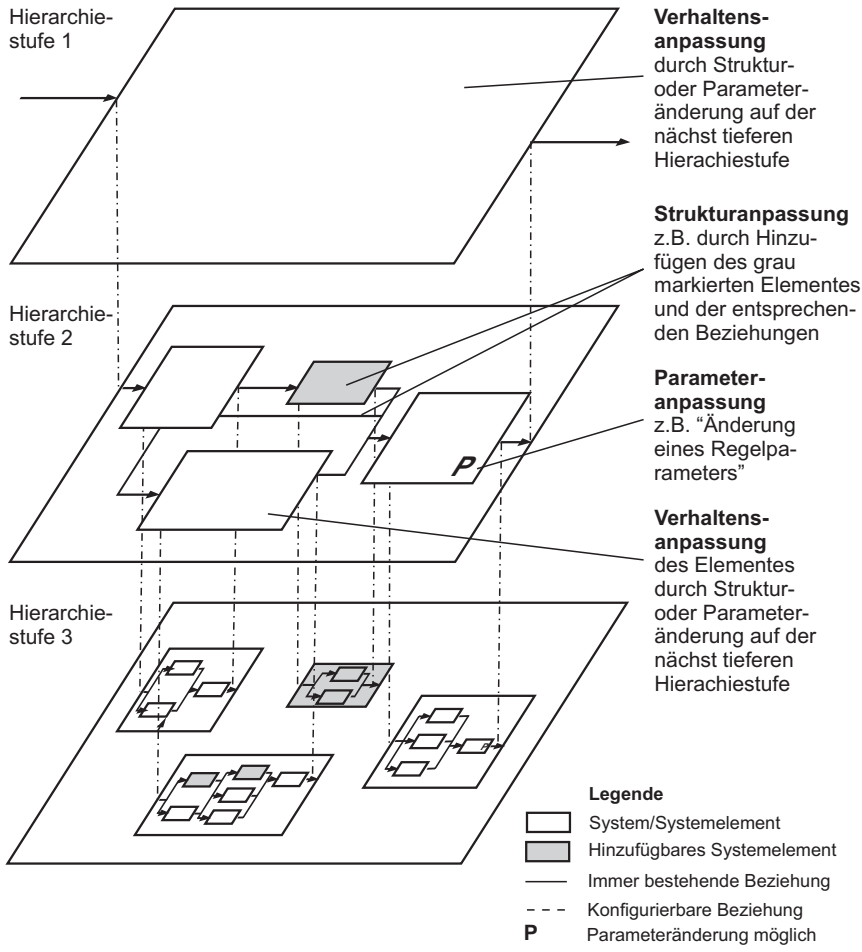


Abbildung 2.3: Struktur-, Verhaltens- und Parameteränderung eines selbstoptimierenden Systems

Zielanpassung

Aufgrund von Einflüssen kann es zunächst zu **Zielanpassungen** *(objective adaptation)* kommen. Dies bedeutet, dass beispielsweise die Gewichtung der Ziele verändert wird, neue Ziele hinzukommen oder vorhandene Ziele nicht mehr verfolgt werden. Als Folge einer Zieländerung wird auch das Verhalten des Systems zielkonform verändert. Die notwendige Verhaltensanpassung wird durch Parameter- oder Strukturanpassungen erzielt (siehe Abbildung 2.3).

Unter einer **Parameteranpassung** *(parameter adaptation)* wird die Anpassung eines Systemparameters verstanden, z.B. das Ändern eines Regelparameters. **Strukturanpassungen** *(structural adaptation)* betreffen die Anordnung und Beziehungen der Elemente eines Systems. Bei strukturellen Anpassungen wird zwischen **Rekonfiguration** *(reconfiguration)*, bei der die Beziehungen einer festen Menge von verfügbaren Elementen verändert wird, und **Kompositionaler Anpassung** *(compositional adaptation)*[MSKC04], bei der neue Elemente in die bisherige Struktur integriert bzw. Elemente aus der Struktur herausgenommen werden, unterschieden. In Abbildung 2.3 ist beispielhaft eine Strukturanpassung dargestellt, in deren Rahmen bisher nicht genutzte Elemente und entsprechende Beziehungen aktiviert und nicht mehr benötigte Beziehungen deaktiviert werden. Da lediglich auf vorhandene Elemente des Systems zurückgegriffen wird und die Menge der das System konstituierenden Elemente gleich bleibt, handelt es sich dabei um ein Beispiel für Rekonfiguration. Eine **Verhaltensanpassung** *(behavior adaptation)* der Elemente wird durch eine Struktur- oder Parameteranpassung der Elemente auf der nächst tieferen Hierarchieebene erreicht.

Parameteranpassung

Strukturanpassung

Rekonfiguration

kompositionale Anpassung

Verhaltensanpassung

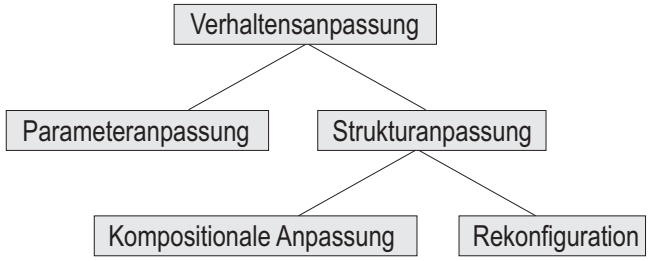


Abbildung 2.4: Arten der verwendeten Verhaltensanpassungen

Abbildung 2.4 stellt nochmals überblicksartig die Klassifikation der verwendeten Anpassungsbegriffe dar.

2.2 Definition der Selbstoptimierung

Selbstoptimierung

Definition 1 *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*

Für die Selbstoptimierung ist wesentlich, dass Ziele durch das *System* situationsbedingt festgelegt oder angepasst werden. Ein System, in dem nur die erste und dritte, nicht aber die zweite Aktion durchgeführt werden, enthält somit *keine* Selbstoptimierung.

Ist-Situation

Die in der ersten Aktion betrachtete **Ist-Situation** *(current situation)* umfasst den Zustand *(state)* des Systems selbst sowie alle möglichen Beobachtungen *(observations)* über seine Umgebung. Dabei können Beobachtungen auch indirekt durch Kommunikation mit anderen Systemen gewonnen werden. Der Zustand eines Systems beinhaltet weiterhin eventuell gespeicherte zurückliegende Beobachtungen. Ein wesentlicher Aspekt der Analyse ist die Prüfung des Erfüllungsgrades der verfolgten Ziele.

Zielbestimmung

Bei der **Zielbestimmung** *(determination of objectives)* in der zweiten Aktion können die neuen Ziele des Systems durch Auswahl, Anpassung oder Generierung dieser gewonnen werden. Hierbei verstehen wir unter Auswahl *(selection)* die Selektion einer Alternative aus einer fest vorgegebenen, diskreten, endlichen Menge von möglichen Zielen. Eine Anpassung *(adaptation)* von Zielen dagegen beschreibt die graduelle Veränderung bestehender Ziele. Von Generierung *(generation)* der Ziele sprechen wir dagagen dann, wenn diese unabhängig von den bisher bekannten neu erzeugt werden.

Systemverhalten

Die Anpassung des **Systemverhaltens** *(system behavior)* ist, wie in Abschnitt 2.1 beschrieben, durch die drei Aspekte Parameter, Struktur, und Verhalten bestimmt. In der dritten Aktion wird die abschließende Rückwirkung des Selbstoptimierungskreislaufes durch Anpassung des Systemverhaltens beschrieben. Die einzelnen Fälle der Anpassung können je nachdem, auf welcher Ebene eines mechatronischen Systems wir uns

befinden[2], sehr unterschiedlich ausfallen. Auch die Domäne, in der die Anpassung umgesetzt wird, spielt hierbei eine wesentliche Rolle.

Optimierung im engeren Sinne, also die explizite Anwendung eines Optimierungsalgorithmus, kann dabei sowohl im Rahmen der zweiten Aktion zur Bestimmung geeigneter Ziele als auch in der dritten Aktion zur Ermittlung möglichst günstiger zielkonformer Anpassungen stattfinden.

Zwischen den einzelnen Aktionen werden lediglich kausale Beziehungen in Form notwendiger Informationsflüsse vorausgesetzt. Die Ergebnisse der Analyse der Ist-Situation bilden die Grundlage der Zielbestimmung, und der Anpassungsschritt hängt wiederum von den dort bestimmten Zielen ab. Ferner bestehen naturgemäß Wirkzusammenhänge zwischen den vorgenommenen Anpassungen und der in Folge wahrgenommenen Ist-Situation.

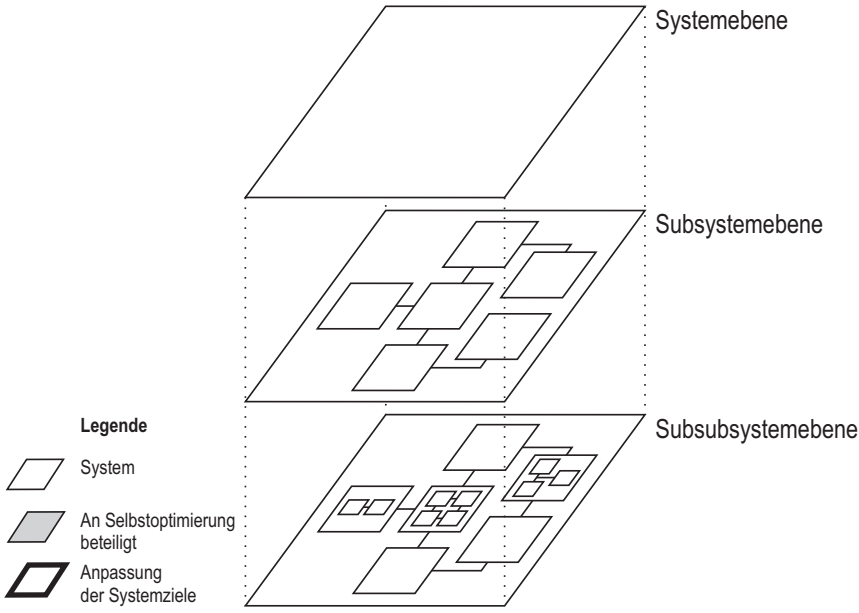


Abbildung 2.5: Systemstruktur auf verschiedenen Abstraktionsebenen

[2]Wir grenzen bei der Beschreibung mechatronischer Systeme drei hierarchische Ebenen begrifflich voneinander ab: **Mechatronische Funktionsmodule (MFM)** *(mechatronic function module (MFM))*, die als Baugruppen in **Autonome Mechatronische Systeme (AMS)** *(autonomous mechatronic system (AMS))* eingehen, die sich wiederum zu **Vernetzten Mechatronischen Systemen (VMS)** *(connected mechatronic system (CMS))* organisieren können.

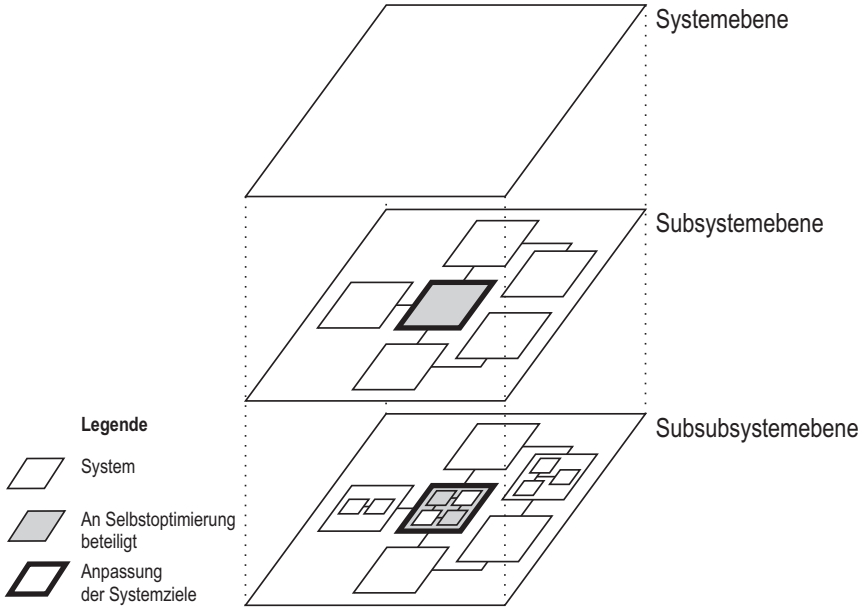


Abbildung 2.6: Nicht selbstoptimierendes System mit einem selbstoptimierenden Subsystem

Der verwendete Systembegriff erlaubt abhängig vom jeweiligen Betrachtungszweck ein flexibles Ziehen der Systemgrenzen. Beim Entwurf einer Anwendung ist es jedoch zweckmäßig, die Grenzen der zur Beschreibung verwendeten Systeme und Elemente verbindlich festzulegen. Im Bereich der Informationsverarbeitung wird die Definition von festen Systemgrenzen zur Strukturierung des Entwurfs bewusst als Mittel zur Komplexitätsreduktion eingesetzt. Die nachfolgenden Definitionen klassifizieren vor dem Hintergrund einer solchen konkreten Festlegung verschiedene Konstellationen von Systemen und Systemelementen (siehe Abbildung 2.5). Dabei ist zu beachten, dass Systemelemente prinzipiell auch immer selbst wieder synonym als Subsysteme verstanden werden können, besonders wenn von der gewählten Abstraktion die interne Struktur in Form enthaltener Elemente erfasst wird. Jedes Element kann somit in seiner Eigenschaft als System auch ein eigenes Zielsystem besitzen.

Läuft ein Selbstoptimierungsprozess ausschließlich gekapselt in den Grenzen eines einzelnen Subsystems ab, so findet innerhalb des übergeordneten Systems zwar zweifellos Selbstoptimierung statt, das System wäre jedoch dennoch gemäß Definition 1 nicht als selbstoptimierend zu bezeichnen (siehe Abbildung 2.6).

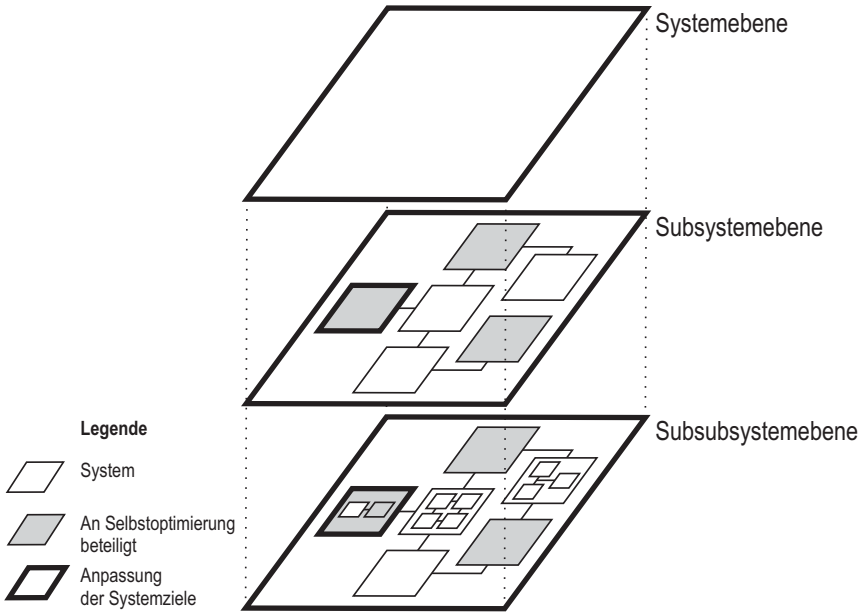


Abbildung 2.7: Selbstoptimierung erfordert Anpassung von Zielen

selbstoptimierendes System

Von einem **selbstoptimierenden System** *(self-optimizing system)* sprechen wir nur dort, wo auch tatsächlich Ziele, die dem betrachteten System direkt zugeordnet werden können, einer Anpassung unterliegen (siehe Abbildung 2.7).

individuell selbstoptimierendes System

Verläuft der wiederkehrende Prozess der Selbstoptimierung dergestalt, dass die Aktivitäten, von denen die drei notwendigen Aktionen Analyse, Zielbestimmung und Anpassung umgesetzt werden, von den Elementen eines einzelnen, isolierten Systems durchgeführt werden, so sprechen wir bei diesem System von einem **individuell selbstoptimierenden System** *(individually self-optimizing system)*. Falls das System Subsysteme mit eigenem Zielsystem enthält, müssen diese für die Selbstoptimierung irrelevant sein, d.h. sie dürfen nicht an den entsprechenden Aktivitäten beteiligt sein (siehe Abbildung 2.8). Einfache reglungstechnische Komponenten auf der Ebene der MFM oder Ansätze aus der klassischen KI mit einer zentralistisch geprägten Inferenzkomponente sind typische Beispiele für entsprechende Systeme.

zusammengesetztes selbstoptimierendes System

Enthält ein selbstoptimierendes System Subsysteme mit eigenen Zielen, die am Prozess der Selbstoptimierung beteiligt sind, so sprechen wir von einem **zusammengesetzten selbstoptimierenden System** *(composed*

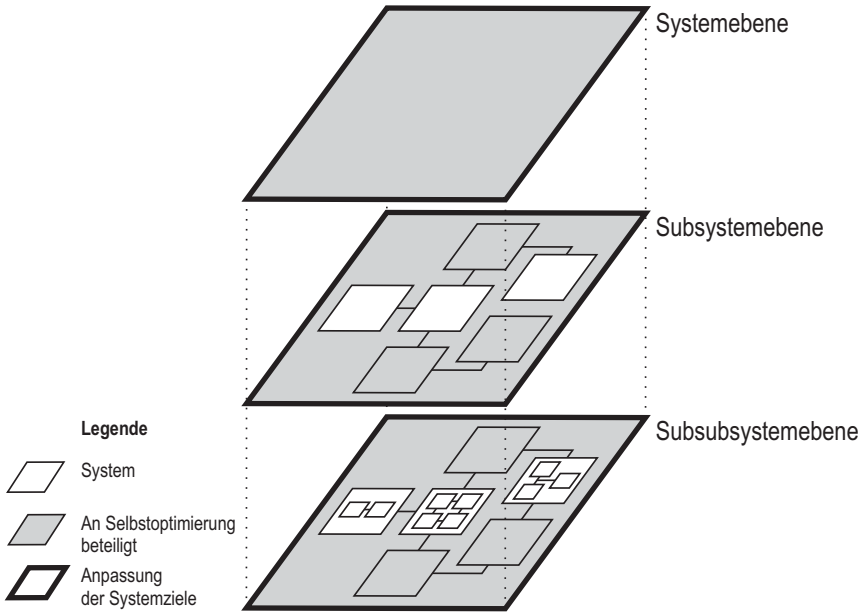


Abbildung 2.8: Individuell selbstoptimierendes System

self-optimizing system) (siehe Abbildung 2.9). Dies kann bedeuten, dass einzelne Subsysteme für bestimmte Aktionen verantwortlich sind; es ist aber gleichermaßen möglich, dass die Aktivitäten einer einzelnen Aktion auf verschiedene zielbehaftete Subsysteme verteilt sind. Beispiele für entsprechende Systeme sind AMS und VMS, die per Definition zusammengesetzt sind.

Dieser Unterscheidung kommt beim Entwurf eines Systems eine große praktische Bedeutung zu. Da die Subsysteme eines zusammengesetzten Systems selbst wieder eigene Ziele besitzen, sind in der Regel komplexere Koordinationsmechanismen notwendig. Zusammengesetzte selbstoptimierende Systeme bieten bezüglich ihrer inneren Struktur ein weites Spektrum von Möglichkeiten: Die Subsysteme selbst können sowohl selbstoptimierend als auch nicht-selbstoptimierend sein. Sie können miteinander kooperieren, konkurrieren oder hierarchische Strukturen bilden. Die Selbstoptimierung, deren Aktionen verteilt auf verschiedene Subsysteme ablaufen, kann zentral koordiniert werden, kann aber auch einfach das Ergebnis der dezentral organisierten Interaktionen sein. Im letzteren Fall sprechen wir von **emergentem Verhalten** *(emergence)*, da sich das selbstoptimierende Verhalten auf Systemebene allein aus dem lokalen

Emergenz

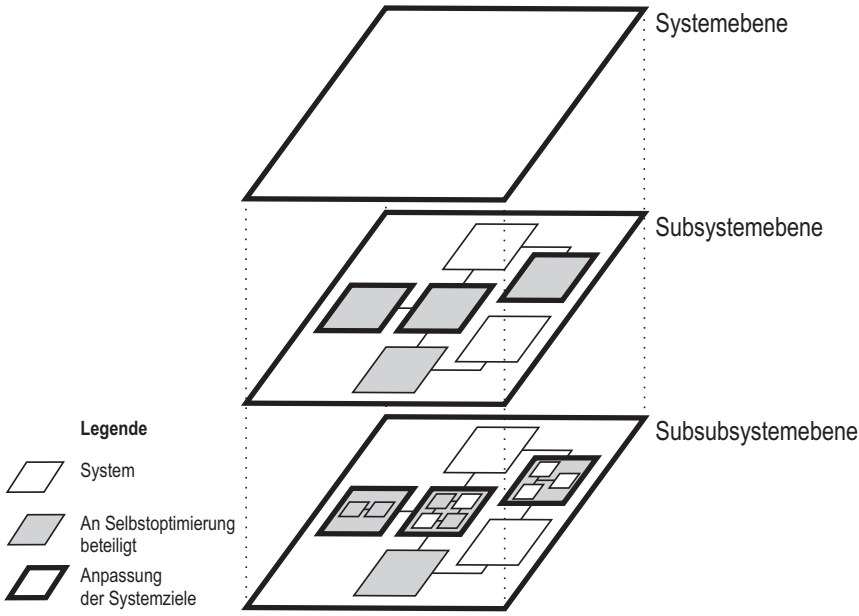


Abbildung 2.9: Zusammengesetztes selbstoptimierendes System

Zusammenspiel der Elemente ergibt. Eine nähere Betrachtung möglicher Koordinationsstrategien findet sich im Rahmen des Klassifikationsschemas für verschiedene Formen der Selbstoptimierung im Abschnitt 2.3.

Die selbständige Festlegung der verfolgten Ziele ist ein essentielles Kennzeichen von Autonomie. Die konkrete Verfolgung dieser Ziele geschieht proaktiv durch die vorgenommenen Anpassungen am Systemverhalten. Die beständige Analyse der Ist-Situation sorgt für angemessene Reaktionen auf Änderungen der Umweltsituation, indem sie den Zielbestimmungsprozess beeinflusst und eine Rückkopplung zum Anpassungsprozess herstellt. Damit erfüllen selbstoptimierende Systeme per Definition die Forderung nach Autonomie *(autonomy)*, Proaktivität *(proactivity)* und Reaktivität *(reactivity)*, wie sie durch den **schwachen Agentenbegriff** *(weak notion of agency)* nach Jennings und Wooldridge [WJ95, Woo00] formuliert werden. Das letze Kriterium, die Fähigkeit zur sozialen Interaktion, ist nur auf die Selbstoptimierung zusammengesetzter Systeme sinnvoll anwendbar, dort aber durch die Forderung nach einer koordinierten Anpassung der Ziele für die Gesamtheit des Systems ebenfalls per Definition gegeben.

schwacher Agentenbegriff

Selbstoptimierende Systeme können somit aus informationstechnischer Sicht generell als Agenten interpretiert werden. Wir bezeichnen daher selbstoptimierende elektronische, mechanische oder informationstechnische Systeme sowie Hierarchien solcher Systeme synonym als **Agenten** *(agent)*, wann immer diese Metapher dem Verständnis dienlich ist oder eine Anlehnung an die Erkenntnisse der Agentenforschung nützlich erscheint. In diesem Sinne kann ein aus mehreren selbstoptimierenden Systemen bestehendes zusammengesetztes System als **Multiagentensystem** *(multi-agent system)* verstanden werden, in dem verschiedene Agenten in Kooperation oder Konkurrenz zueinander handeln.

Agent

Multiagentensystem

2.3 Merkmale

Selbstoptimierung wurde bewusst in Definition 1 als allgemeines Prinzip definiert, das in vielen verschiedenen konkreten Ausprägungen angewendet werden kann. Die Definition der Selbstoptimierung fordert das Vorhandensein der drei Aktionen **Analyse der Ist-Situation**, **Bestimmung der Ziele** und **Anpassung des Verhaltens**, lässt aber große Spielräume bei der Ausgestaltung der einzelnen Aktionen.

Im Folgenden werden verschiedene Dimensionen der Selbstoptimierung herausgestellt, die einen ersten Schritt bei dem Versuch, die ganze Bandbreite denkbarer Ansätze zur Selbstoptimierung zu erfassen, darstellen sollen. Orientiert an verschiedenen, nicht notwendigerweise orthogonalen Achsen werden mögliche Merkmalsausprägungen angegeben, anhand derer sich konkrete Anwendungen einordnen lassen. Der aufgespannte Rahmen soll als ein nützliches Werkzeug zur Einordnung zukünftiger Anwendungen dienen, erhebt dabei aber nicht den Anspruch, ein erschöpfendes Klassifikationsschema für alle erdenklichen Spielarten der Selbstoptimierung darzustellen.

2.3.1 Formen der Optimierung

Der Prozess der Selbstoptimierung kann sehr unterschiedliche Formen annehmen. Wesentliche Faktoren sind dabei die Ziele des Systems und die Optimierungsverfahren, die zu ihrer Erreichung angewendet werden.

Optimierungsziele

Die endogene Zieländerung in der Form von **Generierung**, **Auswahl** oder **Anpassung** wurde im Abschnitt 2.2 als Charakteristikum der Selbstoptimierung herausgestellt. Daher kommt dem **Zielsystem** *(system of objectives)* eines selbstoptimierenden Systems eine entscheidende Bedeutung zu.

Zielsystem

Das Zielsystem kann als **Zielvektor** *(objective vector)*, also als einfache Menge von Zielen, realisiert sein. Häufig werden zwischen den Zielen jedoch Ordnungsbeziehungen bestehen, so dass von einer **Zielhierarchie** *(objective hierarchy)* gesprochen werden kann. Zwei typische Ordnungskriterien sind hier zum einen die Priorisierung der Ziele, also eine Hierar-

Zielvektor

Zielhierarchie

chisierung der Ziele nach ihrer Wichtigkeit, und zum anderen die Dekomposition, also eine Untergliederung von Zielen in die enthaltenen Teilziele. Wechselseitige Abhängigkeiten zwischen Zielen, etwa ob bestimmte Ziele oder Teilmengen von Zielen komplementär oder konträr sind, sind in der Regel nicht mehr durch Hierarchien zu beschreiben, sondern erzeugen ein komplexes Netz von Beziehungen, das durch einen **Zielgraphen** *(objective graph)* beschrieben werden kann.

Zielgraph

Das Zielsystem eines selbstoptimierenden Systems besteht aus seinen **internen Zielen** *(internal objectives)*. Dies sind die von dem System selbst verfolgten Ziele, die seiner Kontrolle unterliegen und im Zuge einer Zielanpassung verändert werden können. Ihnen gegenüber stehen die externen und die inhärenten Ziele, die auf ein System einwirken, seiner Kontrolle aber entzogen sind. Als **externe Ziele** *(external objectives)* werden die Ziele anderer Systeme, eines Benutzers oder der Umwelt bezeichnet. Die **inhärenten Ziele** *(inherent objectives)* spiegeln dagegen den Entwurfszweck des Systems wider und werden ihm durch den Entwickler eingepflanzt. Sie finden sich zwar oft nur implizit in der Wirkstruktur des Systems und den zugrundeliegenden Modellen wieder, sind aber immer dann für die Selbstoptimierung von Bedeutung, wenn der Systemzweck explizit im Optimierungsprozess berücksichtigt werden muss[3] oder Sicherheitsrestriktionen vorliegen.

interne Ziele

externe Ziele

inhärente Ziele

Bezüglich der Herkunft der internen Ziele können zwei Fälle unterschieden werden. Externe und inhärente Ziele können als **angeeignete Ziele** *(appropriated objectives)* von außen in das System der internen Ziele übernommen werden. Welche der an es herangetragenen Ziele ein System in sein internes Zielsystem aufnimmt und welche Priorität es ihnen beimisst, ist anwendungsspezifisch. Es ist denkbar, dass bestimmte Ziele, etwa externe Ziele hierarchisch übergeordneter Systeme oder sicherheitskritische inhärente Ziele, unverändert und mit hoher Priorität übernommen werden müssen; genauso könnte ein System diesbezüglich auch völlig freie Hand haben. Ein wichtiges im Sonderforschungbereich verfolgtes Forschungsziel ist es, hier Richtlinien für geeignete Lösungen herauszuarbeiten. Den angeeigneten Zielen stehen die **generierten Ziele** *(generated objectives)* gegenüber. Dies sind all jene internen Ziele, die eben nicht von außen übernommen, sondern vom System selbst durch Anpassung, Inferenz, Berechnung oder zufälliges Ausprobieren neu generiert wurden.

angeeignete Ziele

generierte Ziele

[3] Etwa wenn einem über seine Ziele räsonierenden System auch die Ausführung seiner eigentlichen Funktion (z.B. Fortbewegung) explizit als Ziel mitgegeben werden muss, um zu verhindern, dass die Verfolgung eines nachgeordneten Ziels (Energieoptimierung) den Systemzweck vereitelt (Stillstand ist energieoptimal).

Die Ziele eines Zielsystems lassen sich bezüglich ihres Inhalts noch zahlreichen weiteren Klassifikationen unterwerfen. Hier seien lediglich drei weitere, für besonders praktisch relevant erachtete Achsen vorgeschlagen: **Quantitative Ziele** *(quantitative objectives)* werden durch einen numerischen Zielwert beschrieben, während **qualitative Ziele** *(qualitative objectives)* kategorisch oder durch ein logisches Prädikat formuliert werden. **Harte Ziele** *(hard objectives)* sind unabänderlich und können nur zur Gänze oder gar nicht erfüllt werden. **Weiche Ziele** *(soft objectives)* sind anpassbar, so dass auch bei ihrer Verletzung ein, wenn auch mit der Stärke der Abweichung abnehmender, Nutzen entsteht. Schließlich können Ziele **obligatorisch** *(mandatory objectives)*, d.h. durch das System unbedingt zu erfüllen, oder **fakultativ** *(facultative objectives)*, d.h. nur möglichst, aber nicht zwangsläufig, einzuhalten, sein.

quantitative Ziele
qualitative Ziele
harte Ziele
weiche Ziele
obligatorische Ziele
fakultative Ziele

Denen im Rahmen einer modellbasierten Optimierung durchgeführten mathematischen Optimierungsverfahren liegt ein enger gefasster als der oben formulierte Zielbegriff zugrunde. Der mathematische Zielbegriff bezieht sich ausschließlich auf quantitative, anpassbare Ziele, die in Zielfunktionen (z.B. Minimierung des Energiebedarfs) umgesetzt werden können. Alle anderen Ziele eines Systems gehen dabei in Nebenbedingungen (z.B. „sei um 17 Uhr in Paderborn “) der mathematischen Optimierung über.

Optimierungsverfahren

Die eigentlichen Optimierungsschritte können modellbasiert, verhaltensbasiert oder in kombinierter Form durchgeführt werden.

Eine **modellbasierte Optimierung** *(model-based optimization)* erfolgt algorithmisch auf Grundlage ausformulierter mathematischer Modelle. Es existieren detaillierte System- und Umweltmodelle und symbolisch mathematische Formalisierungen physikalischer Effekte. Zur Beschreibung der physikalischen Sachverhalte erfolgt eine Verknüpfung von Systemzustand und Umgebungssituation mit einem Modell. Die Optimierung erfolgt mittels dieses physikalisch realistischen Modells der Regelstrecke mit Hilfe von Anregungs- und Bewertungsmodellen. Neu ermittelte Parameter werden in den Regler überblendet, nachdem sie verifiziert wurden.

modellbasierte Optimierung

Die **verhaltensbasierte Optimierung** *(behavior-based optimization)* passt das Systemverhalten an, ohne dabei auf explizite Modelle zurückzugreifen. Aufgrund vorheriger Erfahrungen versucht das System, aus einer Auswahl möglicher Verhaltensweisen die jeweils günstigste auszuwählen,

verhaltensbasierte Optimierung

wobei etwa Techniken des fallbasierten Schließens zum Einsatz kommen können. Sie arbeitet kognitiv, da Modelle über das System und / oder die Umwelt entweder gar nicht vorhanden oder aufgrund der vielfältigen möglichen Einflussgrößen nicht realistisch abbildbar sind. Daher werden in einer Situationsanalyse Veränderungen des Systems und der Umwelt während des Betriebes wahrgenommen und analysiert. Aus der Wissensbasis des Systemelements wird ein Reiz-Reaktionsverhalten selektiert, das in ähnlichen Situationen bereits erfolgreich angewendet wurde. Mit Hilfe dieses Reiz-Reaktions-Verhaltens werden Stellgrößen berechnet, denen das Systemelement folgen soll. Neue Erfahrungen, die das Systemelement durch die Anwendung der Stellgrößen erlebt hat, erweitern seine Wissensbasis.

kombinierte Verfahren

Kombinierte Optimierungsverfahren *(composite methods)* setzen beide Herangehensweisen parallel ein, etwa zur verhaltensbasierten Auswahl eines Verhaltensmusters, dessen Parameter dann modellbasiert angepasst werden können.

Im Rahmen des Sonderforschungsbereich 614 kommt dabei einigen Verfahren eine besondere Rolle zu. Im Bereich der modellbasierten Optimierung existieren Ansätze zur Optimierung mehrerer Zielfunktionen und zur Optimierung nichtkooperativer Agentensysteme, für die verhaltensbasierte Optimierung sind die Verfahren des Case-Based Reasoning (CBR) von herausragender Bedeutung.

Mehrzieloptimierung

Im Zusammenhang mit der Abwägung zwischen mehreren Zielen ist die **Mehrzieloptimierung**(MZO) ein wichtiges mathematisches Hilfsmittel. Dabei werden mehrere Zielfunktionen gleichzeitig betrachtet und optimiert, die als mathematische Funktionen vorliegen müssen (d.h. $f_i(x) = y_i$, $x \in \mathbb{R}^n$, $y_i \in \mathbb{R}$ für $i = 1, \ldots, k$, wobei optimale x-Werte bestimmt werden sollen). Da diese Zielfunktionen typischerweise konfliktär sind, besteht die Lösung eines Mehrzieloptimierungsproblems aus der Menge der optimalen Kompromisse, der sogenannten Paretomenge. Beispiele für solche Zielfunktionen sind bei der Optimierung der Arbeitspunktsteuerung des Linearmotors zwei konfliktäre Wirkungsgrade oder im Falle der Optimierung eines aktiven Federbeins „Komfort“ und „Sicherheit“. Wurde die globale Paretomenge (z.B. wie in [DSH04] beschrieben) berechnet, so kann anhand weiterer problemabhängiger Kriterien geschickt selbstoptimierend auf der Paretomenge gewandert werden (Decision Making). Somit können die aktuellen Systemeinstellungen jeweils in optimaler Art und Weise der jeweiligen Situation angepasst werden. Hat das System eine hierarchische Struktur, können auf jeder Ebene mehrere Zielfunktionen, die gleichzeitig optimiert werden sollen, auftreten. Ziele höherer

Ebenen können auch als Nebenbedingungen in unterliegende Ebenen einfließen.

nichtkooperatives System

Besteht ein System aus mehreren eigenständigen, selbstoptimierenden Agenten, kann sich die Situation ergeben, dass der Nutzen eines Agenten vom Verhalten aller, oder zumindest einiger anderer, Agenten abhängt. In diesem Szenario haben die Agenten private Ziele, während das Gesamtsystem ein globales Ziel hat. Die Agenten sind eigennützig, indem sie durch ihr Verhalten versuchen ihre privaten Zielfunktionen zu optimieren. Eine zentrale Kontrolle würde dagegen versuchen die Zielfunktion des Gesamtsystems zu optimieren und eine global optimale Lösung zu finden. Nash Equilibrien sind stabile Zustände eines Systems eigennütziger Agenten. Sie sind dadurch charakterisiert, dass in einem Nash Equilibrium kein Agent seine Zielfunktion dadurch steigern kann, dass er unilateral von seinem Verhalten im Nash Equilibrium abweicht. Es ist bekannt, dass Nash Equilibrien nicht automatisch global optimale Lösungen sind. In Systemen dieser Art stellt sich für den Systemdesigner das Problem, die privaten Zielfunktionen so zu formulieren, dass von den eigennützigen Agenten im Sinne der globalen Zielfunktion möglichst gute Nash Equilibrien angestrebt werden. Für einen Systemadministrator, der die privaten Zielfunktionen nicht beeinflussen kann, stellt sich das Problem, Regelwerke zu finden, die die Agenten dazu bewegen, diese Regeln einzuhalten und dadurch im Sinne der globalen Zielfunktion möglichst gute Nash Equilibrien anzustreben.

2.3.2 Zeitkonzept

Die Schritte des Selbstoptimierungsprozesses können mit unterschiedlichen Perioden und Geschwindigkeiten durchlaufen werden. Selbstoptimierung kann als ständiger, zeitnaher Vorgang oder nur phasenweise stattfinden. Dabei können die einzelnen Schritte in unterschiedlichem Maße zeitkritisch oder keinen festen Reaktionszeiten unterworfen sein.

Zeitpunkt der Selbstoptimierung

kontinuierlich

Selbstoptimierung kann **kontinuierlich** *(continuously)* im regulären Betrieb vorgenommen werden. Sie kann aber auch nur phasenweise zu bestimmten Zeitpunkten stattfinden. Denkbar ist hier etwa, dass die Selbstoptimierung **zeitgesteuert** *(time-triggered)*, also regelmäßig nach Ablauf eines bestimmten Zeitintervalls, angestoßen wird. Weiterhin kann die

zeitgesteuert

zustandsgesteuert

ereignisgesteuert

lebenszyklusgesteuert

Selbstoptimierung **zustandsgesteuert** *(state-dependent)* nur dann stattfinden, wenn das System bestimmte Zustände annimmt, die ihrem problemlosen und ungestörten Ablauf besonders förderlich sind oder aber eine sofortige Reevaluation der Ziele akut notwendig erscheinen lassen. Zuletzt kann die Selbstoptimierung auch **ereignisgesteuert** *(event-triggered)* ausgelöst werden, üblicherweise ebenfalls, weil die betreffenden Ereignisse eine Neuorientierung der bisherigen Strategie bedingen können. Als Spezialfall der beiden vorhergehenden Varianten kann eine **lebenszyklusgesteuerte** *(life-cycle-dependent)* Selbstoptimierung angesehen werden, die etwa während der Inbetriebnahme oder in Wartungszyklen, also nicht eingebettet in den laufenden Betrieb des Systems, stattfindet. Diese Form der Selbstoptimierung tritt häufig bei hierarchischen Systemen auf, in denen spezialisierte Wartungs- oder Optimierungsagenten flexibel verschiedene Optimierungsaufgaben übernehmen können.

Zeitliche Anforderungen an die Selbstoptimierung

harte Echtzeit

weiche Echtzeit

nicht zeitkritisch

Bezüglich des Ablaufes der Selbstoptimierung kann hinsichtlich der gegebenen Echtzeitanforderungen *(real-time)* unterschieden werden. **Harte Echtzeit** *(hard real-time)* liegt vor, wenn die positive Wirkung der Selbstoptimierung streng an die Einhaltung eines bestimmten Zeitfensters gekoppelt ist. Führt der Optimierungsprozess zu einem Verfehlen dieses Zeitfensters, können negative oder möglicherweise fatale Konsequenzen entstehen. Bei **weicher Echtzeit** *(soft real-time)* tritt eine positive Wirkung ein, wenn die geforderte Zeitigkeit im überwiegenden Fall gegeben ist. Verfehlen des Zeitfensters führt nur zu einer begrenzten Verschlechterung des Optimierungsergebnisses bzw. lediglich zum Ausbleiben positiver Effekte. Tritt ein positiver Effekt unabhängig von einem bestimmten Zeithorizont ein, führt jede tatsächlich vorgenommene, also in endlicher Zeit abgeschlossene, Optimierung zu einer Verbesserung. Die Selbstoptimierung unterliegt somit so schwachen Zeitrestriktionen, dass von einem **nicht zeitkritischen** *(no real-time requirements)* Prozess, der **keinen Echtzeitanforderungen** unterliegt, gesprochen werden kann.

2.3.3 Sicherheitsaspekte

Sicherheitsaspekte spielen in selbstoptimierenden Systemen des Maschinenbaus eine herausragende Rolle und schlagen sich daher an vielen Stellen im Entwurfsprozess nieder. An dieser Stelle sollen spezifische

technische und konzeptionelle Ansätze betrachtet werden, um Ausfälle und unerwünschtes Verhalten zum einen schon im Entstehen zu verhindern und zum anderen zeitnah zu erkennen, um entsprechende Maßnahmen ergreifen zu können. Generell ist anzustreben, dass mechatronische Systeme **fail-safe** *(fail-safe)* sind, d.h. bei Ausfall eines sicherheitskritischen Elements in einen sicheren Zustand überführt werden, bzw. **fail-operational** *(fail-operational)* sind, d.h. auch bei Ausfällen eine notwendige Mindestfunktionalität bewahren, wenn, wie etwa bei einem in der Luft befindlichen Flugzeug, kein trivial sicherer Zustand existiert.

fail-safe
fail-operational

Aufgrund der zentralen Rolle, die die Informationsverarbeitung in selbstoptimierenden Systemen einnimmt, ist die Qualität der verwendeten Software von entscheidender Bedeutung für die Sicherheit des Systems. Da Software selbst keinem Verschleiß unterworfen ist und moderne Hardware extrem zuverlässig arbeitet, hängt ihre Zuverlässigkeit entscheidend von einer korrekten Konzeption und Implementierung der Software ab. Aufgrund der hohen Komplexität der betrachteten Systeme, die durch Parallelität, Verteilung und Nichtdeterminismus geprägt sind, ist Korrektheit ein nichttriviales Ziel, das nur durch einen angepassten Entwicklungsprozess erreicht werden kann. In Abhängigkeit von der Sicherheitsrelevanz, der Risikoklasse und der Komplexität des Systems können beim Entwicklungsprozess unterschiedliche Formalisierungsniveaus eingesetzt werden, die von partiellen Modellen und informellen Tests bis zu einer vollständigen, semantisch präzisen Modellierung der Anforderungen und des Systems mit anschließender formaler Verifikation reichen. Schon frühzeitig können Gefahrenanalysen und probabilistische Modelle potentielle Probleme enthüllen, denen durch den gezielten Einsatz bewährter Prinzipien, Techniken und Entwurfsmuster begegnet werden kann, die die Vorhersagbarkeit, Zuverlässigkeit und Fehlertoleranz des Systems erhöhen.

Inhärente Sicherheit

Den konzeptionell einfachsten Ansatz stellt dabei die **inhärente Sicherheit** *(inherent safety)* dar. Beim Ausfall eines Elements gelangt das System von selbst, ohne Anwendung weiterer Hilfsmittel in einen sicheren Zustand. Für ein selbstoptimierendes System bedeutet dies konkret, dass es bei einem Totalausfall des Selbstoptimierungsprozesses dennoch aufgrund nicht-selbstoptimierender Elemente, etwa klassischer Reglerkomponenten, stabil in einem sicheren Zustand verbleibt.

inhärente Sicherheit

Kontrollierte Sicherheit

kontrollierte Sicherheit

Im Kontext der Selbstoptimierung hat der Ansatz der **kontrollierten Sicherheit** eine herausragende Bedeutung. Eine Überwachung des Selbstoptimierungsprozesses stellt sicher, dass das System nicht in einen unerwünschten Zustand gerät oder dort verbleibt. So können etwa im Sinne eines **Sanity Checks** offensichtlich unsinnige oder schädliche Aktionen erkannt und verhindert werden. Ein **unüberwachtes System** *(unsupervised)* ist nur in unkritischen Fällen vertretbar, jedoch auch untypisch. Selbstoptimierung impliziert ja gerade eine Überprüfung des eigenen Zustandes und der eigenen Ziele, so dass ein gewisses Maß an Selbstreflexion quasi als Voraussetzung anzusehen ist. In welchem Ausmaß sich ein System dabei **intern überwacht** *(internal supervision)*, kann allerdings variieren. Zudem können bestimmte Messgrößen dem System selbst überhaupt nicht zugänglich sein. Daher ist denkbar, dass es zusätzlich **extern überwacht** *(external supervision)* wird, was gleichzeitig zu einer Entkoppelung sicherheitskritischer Aspekte beiträgt.

unüberwacht

intern überwacht

extern überwacht

Redundanz

keine Redundanz

homogene Redundanz

heterogene Redundanz

Die Zuverlässigkeit eines Systems kann durch eine redundante Auslegung in vielen Fällen erhöht werden. **Keine Redundanz** *(no redundancy)* ist nur in nichtkritischen Bereichen oder wenn der Selbstoptimierungsprozess extrem zuverlässig arbeitet empfehlenswert. Ist das verwendete Optimierungsverfahren theoretisch zuverlässig, kann durch die Verwendung mehrerer es implementierender Systeme, also **homogene Redundanz** *(homogenuous redundancy)*, die Ausfallwahrscheinlichkeit verringert werden. Häufig wird allerdings auf **heterogene Redundanz** *(heterogenous redundancy)* zurückgegriffen werden, bei der entweder mehrere selbstoptimierende Module parallel ausgeführt werden oder aber einem selbstoptimierenden Regler ein als theoretisch zuverlässig bekannter klassischer Regler zur Seite gestellt wird, um stets auf eine sichere Alternative zurückfallen zu können.

2.3.4 Verteilungskonzepte für Multiagentensysteme

Eine besondere Klasse selbstoptimierender Systeme stellen solche zusammengesetzten Systeme dar, in denen mehrere klar getrennte Subsysteme verteilt Selbstoptimierung betreiben. Wie oben begründet wor-

den ist, werden diese hier in Anlehnung an die umfangreiche diesbezügliche Literatur als Multiagentensysteme (MAS), in denen mehrere Agenten gemeinsame oder divergente Optimierungsziele verfolgen, interpretiert.

Typisierung

Multiagentensysteme *(multi-agent system)* lassen sich anhand dreier weitgehend unabhängiger Dimensionen grob klassifizieren (siehe Abbildung 2.10). Auch wenn die vorgeschlagene Typisierung nur ausgewählte Aspekte eines MAS beleuchtet, so stellt sie dennoch bereits einige Eigenschaften von grundlegender Bedeutung heraus, die eine erste Einordnung und Identifikation verwandter Ansätze erlauben.

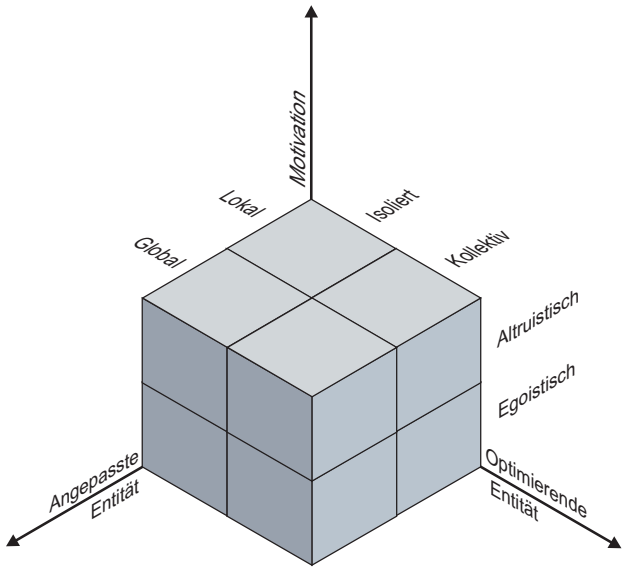


Abbildung 2.10: Drei Dimensionen zur Klassifikation

Die erste Dimension betrifft die **optimierende Entität** *(optimizing entity)* und unterscheidet zwischen isolierter (Typ I) und kollektiver (Typ K) Optimierung. optimierende Entität

Bei **isolierter Optimierung** *(isolated optimization)* führen die Agenten Optimierungsverfahren lokal, ohne diesbezügliche Interaktion mit anderen Agenten, aus. Zentraloptimierung durch einen hochentwickelten Agenten, der auf Grundlage einer Analyse des Gesamtsystems einen op- isolierte Optimierung

timalen Plan zu entwickeln versucht, ist ein typisches Beispiel für isolierte Optimierung. Nichtkooperative Agenten, die quasi als Einzelkämpfer versuchen, ohne Absprache mit anderen ihre eigenen Ziele zu optimieren, fallen ebenfalls in diese Klasse.

kollekive Optimierung

Kollekive Optimierung *(collective optimization)* findet dagegen in Kooperation zwischen mehreren Agenten statt und bietet Raum für eine Vielzahl weiterer Unterklassifizierungen. Relevant ist etwa, ob die Zusammensetzung und Größe des Kollektivs sowie die Art und Verteilung der Rollen dynamisch oder statisch sind. Weiterhin bedingt Kooperation notwendigerweise Kommunikation, sei es nun direkt oder nur indirekt durch Veränderungen der Umgebung. Der gesamte Bereich der Algorithmen zur verteilten Problemlösung fällt unter den Begriff der kollektiven Optimierung.

angepasste Entität

Die zweite Dimension betrifft die **angepasste Entität** *(manipulated entity)*, also quasi die Stellgröße, die verändert wird, und beschreibt, welche Möglichkeiten die Agenten überhaupt zur Umsetzung ihrer Entscheidungen haben. Hierbei wird zwischen lokalen (Typ L) und globalen (Typ G) Anpassungen unterschieden.

lokale Anpassung

Lokale Anpassungen *(local adaptation)* kann ein Agent direkt an sich selbst oder seiner unmittelbaren Umwelt vornehmen. Änderungen am Agenten selbst wirken direkt, etwa als Umparametrisierungen oder Zielvektoränderungen wie bereits für Einzelagenten beschrieben, Änderungen an der Umwelt manifestieren sich zwar auch direkt, etwa wenn der verantwortliche Agent eine Weiche umstellt, wirken sich oft aber nur indirekt und durch scheinbar chaotische, kumulative Effekte auf das Gesamtsystem aus.

globale Anpassung

Globale Anpassungen *(global adaptation)* wirken sich auf das Verhalten auf Systemebene aus, entweder indirekt durch die Änderung von Umweltparametern oder Anreizsystemen, oder direkter durch die Einführung oder Änderung von Verhaltensregeln. Globale Änderungen können entweder unmittelbar wirken und daher genau vorhersagbare Auswirkungen haben, oder auf emergentes Verhalten abzielen und nur einen Rahmen abstecken, in dem sich das konkrete Verhalten dynamisch entwickelt.

Motivation

Die letzte Dimension betrifft die **Motivation** *(motivation)*, aus der heraus die Agenten handeln. In Anlehnung an die in der Multiagentenforschung gebräuchliche Terminologie wird hierbei zwischen egoistischen (Typ E) und altruistischen (Typ A) Agenten unterschieden (vgl. z.B. [Fer01]).

egoistischer Agent

Egoistische Agenten *(egoistic agent)* verfolgen ihre lokalen Ziele auch in Konkurrenz zu anderen Agenten, wie sie etwa in Marktmodellen auftritt.

Im Gegensatz dazu sind **altruistische Agenten** *(altruistic agent)* bereit, die eigenen Ziele zugunsten des Wohls des Gesamtsystems gegebenenfalls zurückzustellen.[4] Diese Unterscheidung ist beim Entwurf eines Multiagentensystems von weitreichender Bedeutung, da sich viele Konsequenzen daraus ableiten, etwa ob davon ausgegangen werden kann, dass Agenten kooperieren, die Wahrheit sagen oder sich an gegebene Regeln halten. Beide Typen erfordern im Fall von Zielkonflikten entsprechende Lösungsstrategien, die Fairness zu garantieren helfen. Systeme egoistischer Agenten sind jedoch in ihrem Entwurf komplexer, da sie mögliches Fehlverhalten berücksichtigen müssen. Zudem können sie in der Regel nur suboptimale Equilibrien erreichen, während altruistische Agenten im Prinzip das theoretische Optimum erreichen können. Allerdings ist die Annahme egoistischen Verhalten, besonders in offenen Systemen, in der Regel realistischer.[5]

altruistischer Agent

Exemplarische Beispiele für entsprechende Typisierungen sind Schwarmverhalten [KE01], wo durch einfache Interaktion lokales Verhalten angepasst und dadurch ein gemeinsames Navigationsziel erreicht wird (Typ KLA, Abbildung 2.11), oder ein zentraler Optimierer, der versucht, Regeln festzulegen, die eine Population egoistischer Agenten in ein möglichst günstiges Nash-Gleichgewicht führen (Typ IGE, Abbildung 2.12).

[4] Altruismus oder Selbstlosigkeit ist hier fallbezogen zu verstehen, also in dem Sinn, dass ein Agent, der durchaus auch eigene Ziele verfolgt, zu selbstlosen Handlungen bereit ist, in denen er diese zurückstellt. Der Begriff impliziert nicht, dass ein Agent grundsätzlich selbstlos bis zur Selbstaufgabe sein muss.

[5] In diesem Zusammenhang sei angemerkt, dass die Mathematik keine entsprechende „moralische“ Unterscheidung kennt, da aus spieltheoretischer Sicht Agenten prinzipiell eigennützig sind und eine Maximierung ihrer Zielfunktion anstreben. Altruistisches Verhalten ergibt sich in diesem Kontext dadurch, dass die Zielfunktion des Agenten solchermaßen gestaltet ist, dass eben dieses Verhalten für den Agenten selbst optimal erscheint (etwa weil der Gesamtnutzen in die Zielfunktion direkt einfließt). Dennoch sind auch in der Spieltheorie auf der Ebene der Zielfunktionen Unterscheidungen relevant, die den oben aus der Differenzierung zwischen egoistischen und altruistischen Agenten heraus motivierten vergleichbar sind: etwa zwischen **selfish routing** und **globally optimal routing**, zwischen den Systemnutzen einbeziehenden und nur den Agenten selbst beachtenden Zielfunktionen oder abhängig von der Bewertung des Nutzens von Aufrichtigkeit. Nur wenn keinerlei Abhängigkeiten zwischen den Zielen der einzelnen Agenten bestehen, wird die Unterscheidung zwischen Egoismus und Altruismus irrelevant – dann liegt allerdings auch kein spieltheoretisches Problem mehr vor.

In der Spieltheorie spielt häufig die Differenzierung zwischen kooperativen und nichtkooperativen Agenten eine wichtige Rolle. Da diese sich wiederum auf die verwendete Zielfunktion zurückführen lässt und auch ein rein egoistischer Agent zur Kooperation bereit sein kann, wenn es ihm selbst nützlich ist, ist diese Unterscheidung von der oben betrachteten Frage nach einer egoistischen oder altruistischen Motivation weitgehend entkoppelt und wo relevant als eigene Dimension zu betrachten.

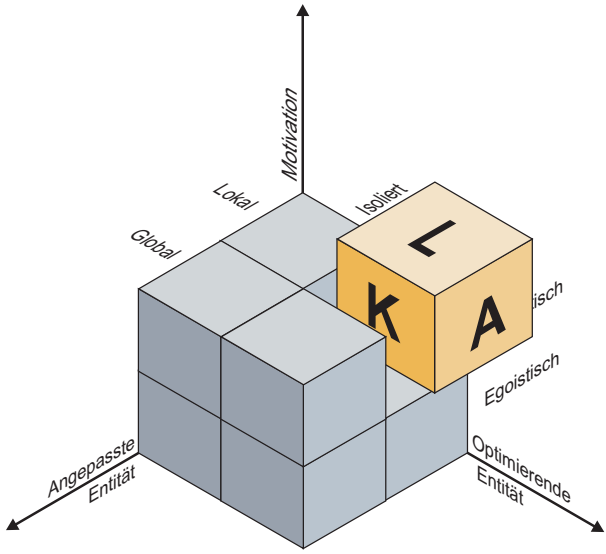


Abbildung 2.11: Einordnung von Schwarmintelligenz

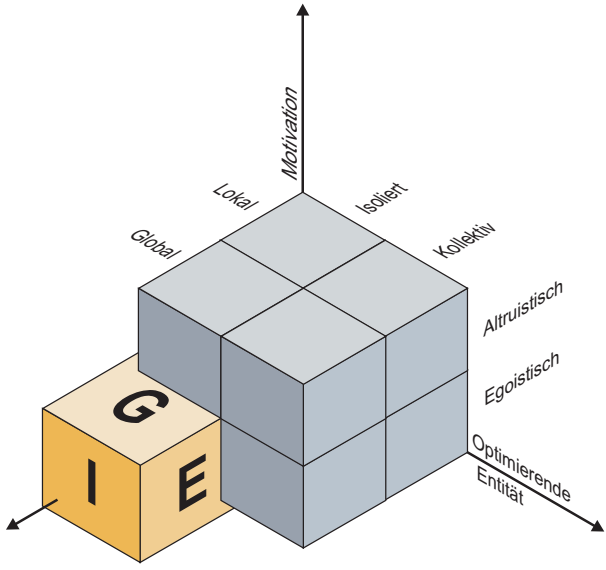


Abbildung 2.12: Einordnung eines spieltheoretischen Ansatzes

Struktur

Die Struktur eines MAS kann in unterschiedlichem Maße dynamisch sein. Es ist auf den drei Ebenen der Agenteninstanzen, der Agententypen und der übernommenen Rollen zwischen statischen, geschlossenen und offenen Systemen zu unterscheiden. Ein auf Instanzebene **statisches System** *(static system)* besteht aus einer eindeutigen Menge von konkreten Agenten, die sich zur Laufzeit nicht ändert. In einem **geschlossenen System** *(closed system)* liegt die maximale Menge der teilnehmenden Agenten zwar eindeutig fest, die konkrete Zusammensetzung kann sich aber dynamisch ändern, es findet also Rekonfiguration statt. Ein **offenes System** *(open system)* erlaubt es neuen **Instanzen** *(instance)*, dem System beizutreten, sodass kompositionale Anpassung möglich ist. Möglicherweise werden in den letzteren Fällen Anforderungen formuliert, dass bestimmte Instanzen zwingend vorhanden sein müssen.

statisches System

geschlossenes System

offenes System

Instanz

Für auf Instanzebene offene und eingeschränkt auch für geschlossene Systeme stellt sich die Frage nach der Typebene, also den Klassen, zu denen die Agenteninstanzen gehören dürfen. Auch hier kann unterschieden werden zwischen Systemen, in denen nur ganz bestimmte, nur in einer Obermenge enthaltene oder beliebige **Typen** *(type)* vorkommen dürfen.

Typ

Unabhängig von den vorhergehenden Aspekten können in einem System **Rollen** *(role)* existieren, die einem Agenten jeweils eine bestimmte Menge von Verhaltensmustern vorschreiben. Die Agenten in einem System können entweder feste Rollen spielen, bestimmte Rollen aus einer Liste unterstützter Rollen übernehmen oder frei aus den verfügbaren Rollen auswählen, wobei wiederum bestimmte Rahmenbedingungen, etwa das Vorhandensein oder die Eindeutigkeit bestimmter Rollen, definiert werden können.

Rolle

2.3.5 Zusammenfassung

Abschließend soll nochmals ein Überblick über die verschiedenen Merkmalsdimensionen gegeben werden.

Erstes Trennungskriterium ist die Unterscheidung zwischen individueller und zusammengesetzter Selbstoptimierung.

Form der Optimierung

Bezüglich der **Form der Optimierung** wird hinsichtlich der von der Selbstoptimierung betroffenen Ziele und den zu ihrer Bestimmung und Erreichung verwendeten modellbasierten, verhaltensbasierten und hybriden Optimierungsmethoden unterschieden.

Zeitpunkt und Ablauf

Hinsichtlich **Zeitpunkt und Ablauf** wird unterschieden, wann die Selbstoptimierung durchgeführt wird und welchen Echtzeitanforderungen sie unterworfen ist.

Sicherheitsaspekte

Die **Sicherheitsaspekte** betreffen zum einen die Sicherheitsanforderungen an ein System, und zum anderen Maßnahmen wie Redundanz und Überwachung des Systemverhaltens zu ihrer Erfüllung.

Typisierung und Struktur

Die **Typisierung und Struktur** von Multiagentensystemen beschreibt die verschiedenen Formen des Zusammenspiels mehrerer Agenten und die Dynamik der Systemstruktur.

Kapitel 3

Entwicklung selbstoptimierender Systeme

Im vorhergehenden Kapitel wurde Selbstoptimierung als abstraktes Prinzip definiert; charakteristische Merkmale selbstoptimierender Systeme wurden beschrieben und klassifiziert. Dabei wurde bereits deutlich, dass Selbstoptimierung in der Regel zu erheblich komplexeren Systemen führt und daher besondere Anforderungen an die Entwurfsmethodik stellt. Wie also können sichere und zuverlässige selbstoptimierende Systeme schnell und kosteneffizient entwickelt werden?

Im Folgenden werden Ansätze zur Lösung dieses Problems vorgeschlagen. Dabei wird zum einen das Problem der Strukturierung der Informationsverarbeitung selbstoptimierender Systeme näher betrachtet, da eine geeignete Architektur entscheidend für die Sicherheit und Beherrschbarkeit des Systems ist. Zum anderen wird der Einsatz von Mustern als Mittel zur Wiederverwendung erfolgreicher Lösungen diskutiert. Dabei werden insbesondere Wirkmuster der Selbstoptimierung als Hilfsmittel zur Konzipierung und Koordinationsmuster als Formalismus zu ihrer softwaretechnischen Umsetzung vorgestellt.

3.1 Architektur

Komplexe mechatronische Systeme, und damit in besonderem Maße selbstoptimierende Systeme, werden erst durch eine konsequente Strukturierung der regelnden Informationsverarbeitung beherrschbar. Insoweit stellt die Existenz einer geeigneten Architektur eine wesentliche Voraussetzung für die Realisierbarkeit selbstoptimierender Systeme dar.

Bereits in der aktuellen Generation mechatronischer Systemen brachte die erweiterte Funktionalität softwarebasierter Reglersysteme auch eine

deutliche Erhöhung des Anteils der durch Software verursachten Fehler mit sich. Mit weiter zunehmender Komplexität der Informationsverarbeitung ist mit einer deutlichen Zunahme dieser Probleme zu rechnen, wenn dem nicht durch eine klare Systemstrukturierung mit geeigneter Softwareunterstützung, Codegenerierung und automatischen Verifikationsverfahren entgegengewirkt wird.

Dies gilt umso mehr für selbstoptimierende Systeme, in denen veränderliche Umwelteinflüsse und Veränderungen im eigenen Verhalten bei der Regelung berücksichtigt werden, wobei insbesondere die Rekonfiguration von Regelungskomponenten von zentraler Bedeutung ist. Der sicherheitskritische Charakter mechatronischer Systeme [Sto96] führt dazu, dass die Probleme einer Integration kognitiver Fähigkeiten (etwa durch die Verwendung von Techniken der künstlichen Intelligenz) und der Vernetzung rekonfigurierbarer Subsysteme noch verschärft werden. Ein systematisches Strukturierungskonzept mit geeigneten Entwurfstechniken ist hier zwingend erforderlich.

Im Folgenden wird ein solches Strukturierungskonzept für die Informationsverarbeitung intelligenter mechatronischer Systeme vorgestellt, das in [HOG04, OHG04] detailliert beschrieben wird. Zentral ist dabei der Ansatz zur Handhabung von Strukturveränderungen, die eine Grundvoraussetzung dynamisch veränderbarer Systeme im Allgemeinen und der Selbstoptimierung im Speziellen sind. Die konkrete Umsetzung dieses Konzeptes ist das **Operator-Controller-Modul** *(Operator-Controller-Module)* (**OCM**), das als Grundlage für eine strukturierte Realisierung selbstoptimierender Systeme dienen kann.

Operator-Controller-Modul (OCM)

3.1.1 Operator-Controller-Modul (OCM)

Die Informationsverarbeitung eines mechatronischen Systems muss eine Vielzahl von Funktionen erfüllen: quasi-kontinuierlich arbeitender Regelungscode regelt die Bewegungen der Strecke, Fehleranalyse-Software überwacht die Strecke auf auftretende Fehlfunktionen, Adaptionsalgorithmen passen die Regelung an veränderte Umgebungszustände an, unterschiedliche Systeme werden vernetzt – um nur einige der Funktionen zu nennen.

Diese unterschiedlichen Aufgaben greifen mehr oder weniger direkt auf die Aktoren der Strecke zu. Abbildung 3.1 schlägt eine neue, aus praktischer Erfahrung mit Anwendungen erwachsene Struktur für die Informationsverarbeitung eines mechatronischen Funktionsmoduls vor, wel-

che die in [NR98] und [Nau00] vorgestellten Ansätze aufgreift und mit Vorstellungen aus [HBN01] und [OHKK02] verbindet.

Grundsätzlich orientiert sich die OCM-Einteilung an der Art des Durchgriffs auf das technische System:

- Auf der untersten Ebene des OCM liegt der **Controller** *(controller)*. Dieser innerste Regelkreis verarbeitet in direkter Wirkkette die Messsignale, ermittelt Stellsignale und gibt diese aus. Er wird daher als *motorischer* Kreis bezeichnet werden. Die Software-Verarbeitung auf dieser Ebene arbeitet quasi-kontinuierlich, d. h. Messwerte werden kontinuierlich eingelesen, verarbeitet und unter harten Echtzeitbedingungen wieder ausgegeben. Dabei kann sich der Controller aus mehreren Reglern zusammensetzen, zwischen denen umgeschaltet werden kann. Diese Umschaltung erfolgt dabei in einem Schritt; Überblendungsmechanismen u. ä. sind wiederum in einem eigenen Reglerelement zusammengefasst.

  Controller

- Der **reflektorische Operator** *(reflective operator)* überwacht und steuert den Controller. Er greift dabei nicht direkt auf die Aktorik des Systems zu, sondern modifiziert den Controller, indem er Parameter- oder Strukturänderungen initiiert. Bei Strukturänderungen – wie beispielsweise Rekonfigurationen – werden nicht nur die Regler ausgetauscht, sondern es werden auch entsprechende Kontroll- bzw. Signalflüsse im Controller umgeschaltet. Kombinationen aus Reglern, Schaltelementen und zugehörigen Kontroll- bzw. Signalflüssen werden als Controllerkonfigurationen bezeichnet. In Abbildung 3.1 sind Controllerkonfigurationen im Controller durch die Blöcke A, B und C angedeutet. Die Konfigurationssteuerung – realisiert durch eine Zustandsmaschine – definiert, bei welchem Systemzustand welche Konfigurationen gültig ist sowie wie und unter welchen Bedingungen zwischen den Konfiguration umgeschaltet wird. In welcher Reihenfolge und unter welchen Zeitrestriktionen der Rekonfigurationsprozess durchgeführt wird, wird durch eine Ablaufsteuerung bestimmt und kontrolliert. Eine weitere wichtige Funktion des reflektorischen Operators ist die kontinuierliche Zustandsüberwachung des Controllers, z.B. durch Watch-Dogs. Watch-Dogs erwarten zu bestimmten Zeitpunkten einen Wert von dem zu überwachenden System. Bleibt dieser Wert aus, veranlassen sie eine entsprechende Notfall-Routine. Diese Notfall-Routine schaltet beispielsweise die Selbstoptimierung aus und veranlasst das System, nach klassischen Verfahren zu arbeiten. Die Implementierung des reflektorischen Operators arbeitet überwiegend ereignisorientiert.

  reflektorischer Operator

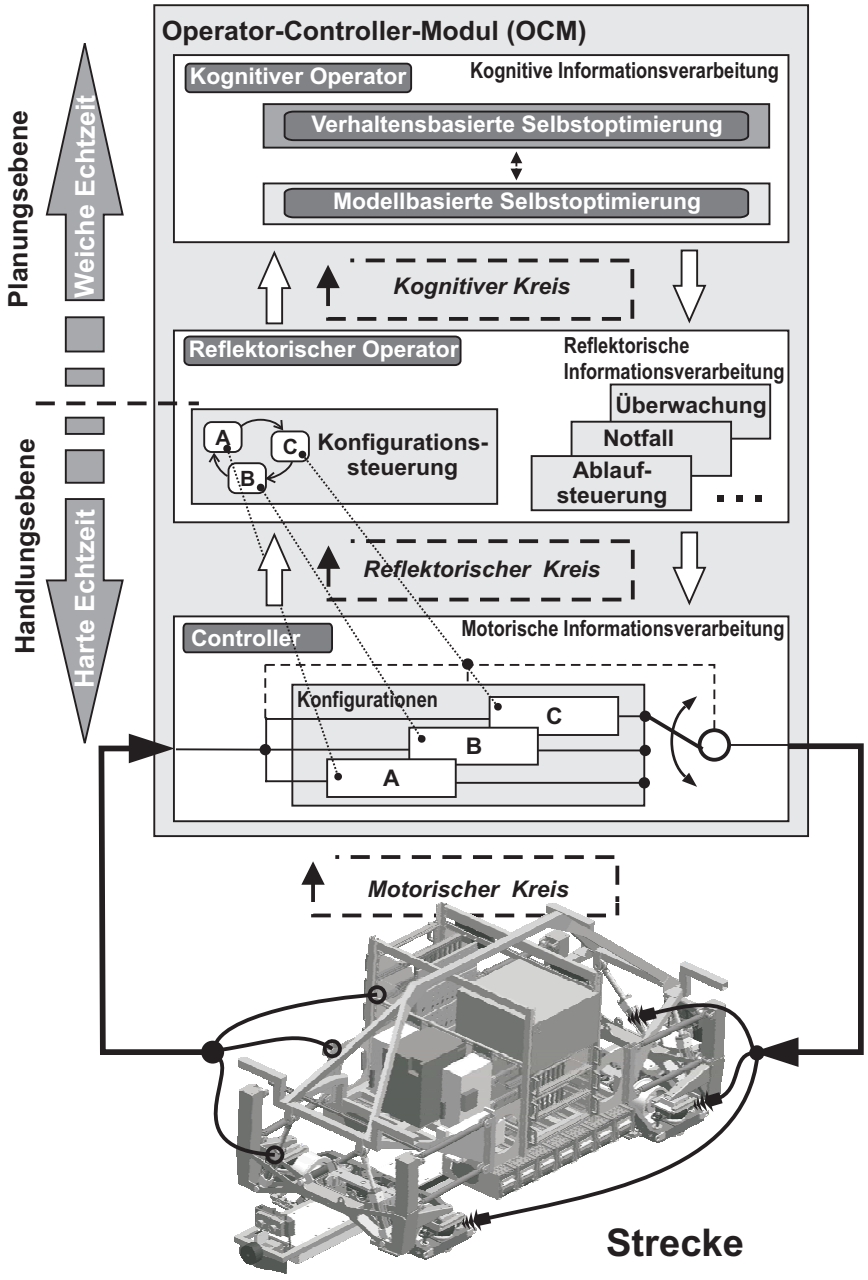


Abbildung 3.1: Struktur des Operator-Controller-Moduls

Die enge Verknüpfung mit dem Controller erfordert eine Abarbeitung in harter Echtzeit. Als Verbindungselement zur kognitiven Ebene des OCM bietet der reflektorische Operator ein Interface zwischen den nicht echtzeitfähigen bzw. mit weicher Echtzeit arbeitenden Elementen und dem Controller. Er filtert die ankommenden Signale und bringt sie in die unterlagerten Ebenen ein. Der reflektorische Operator ist weiterhin für die Echtzeitkommunikation zwischen mehreren OCMs verantwortlich, die gemeinsam ein zusammengesetztes selbstoptimierendes System bilden, wie es etwa in [GTB+03] beschrieben wird.

- Die oberste Ebene des OCM bildet der **kognitive Operator** *(cognitive operator)*. Abbildung 3.2 stellt diese detaillierter dar. Auf dieser Ebene kann das System durch Anwendung vielfältiger Methoden (etwa Lernverfahren, modellbasierte Optimierungsverfahren oder den Einsatz wissensbasierter Systeme) Wissen über sich und die Umgebung zur Verbesserung des eigenen Verhaltens nutzen. Der Schwerpunkt liegt hier auf den kognitiven Fähigkeiten zur Durchführung einer individuellen Selbstoptimierung. Modellbasierte Verfahren erlauben eine vorausschauende und vom realen System zeitlich entkoppelte Optimierung, während verhaltensbasierte Optimierung Funktionen zur Planung und zur Bewertung der aktuellen Zielvorgaben liefert (siehe [OHKK02] und [HO03]). Während sowohl Controller als auch reflektorischer Operator harten Echtzeitanforderungen unterliegen, kann der kognitive Operator auch asynchron zur Realzeit arbeiten. Dabei ist aber selbstverständlich auch eine Antwort innerhalb eines gewissen Zeitfensters erforderlich, da die Selbstoptimierung aufgrund veränderter Umgebungsbedingungen sonst zu keinen verwertbaren Ergebnissen käme. Der kognitive Operator unterliegt folglich weicher Echtzeit.

kognitiver Operator

Zusammenfassend lassen sich zwei deutliche Trennungsebenen erkennen: Einerseits zerfällt die Informationsverarbeitung in einen direkt auf das System wirkenden und in einen nur indirekt darauf wirkenden Kreis. Diese Einteilung entspricht der Einteilung in Operator und Controller. Andererseits lässt sich nach harter und weicher Echtzeitanforderung trennen. Diese Einteilung führt zu einer Trennung zwischen kognitivem Operator einerseits und reflektorischem Operator und Controller andererseits. Da die Wahl der Trennungsebene von der Aufgabenstellung abhängt, wurden hier drei einzelne Elemente vorgesehen.

In der Informatik gibt es eine Reihe unterschiedlicher Einteilungen, die Ähnlichkeiten mit der hier vorgestellten Systematik haben. So spricht

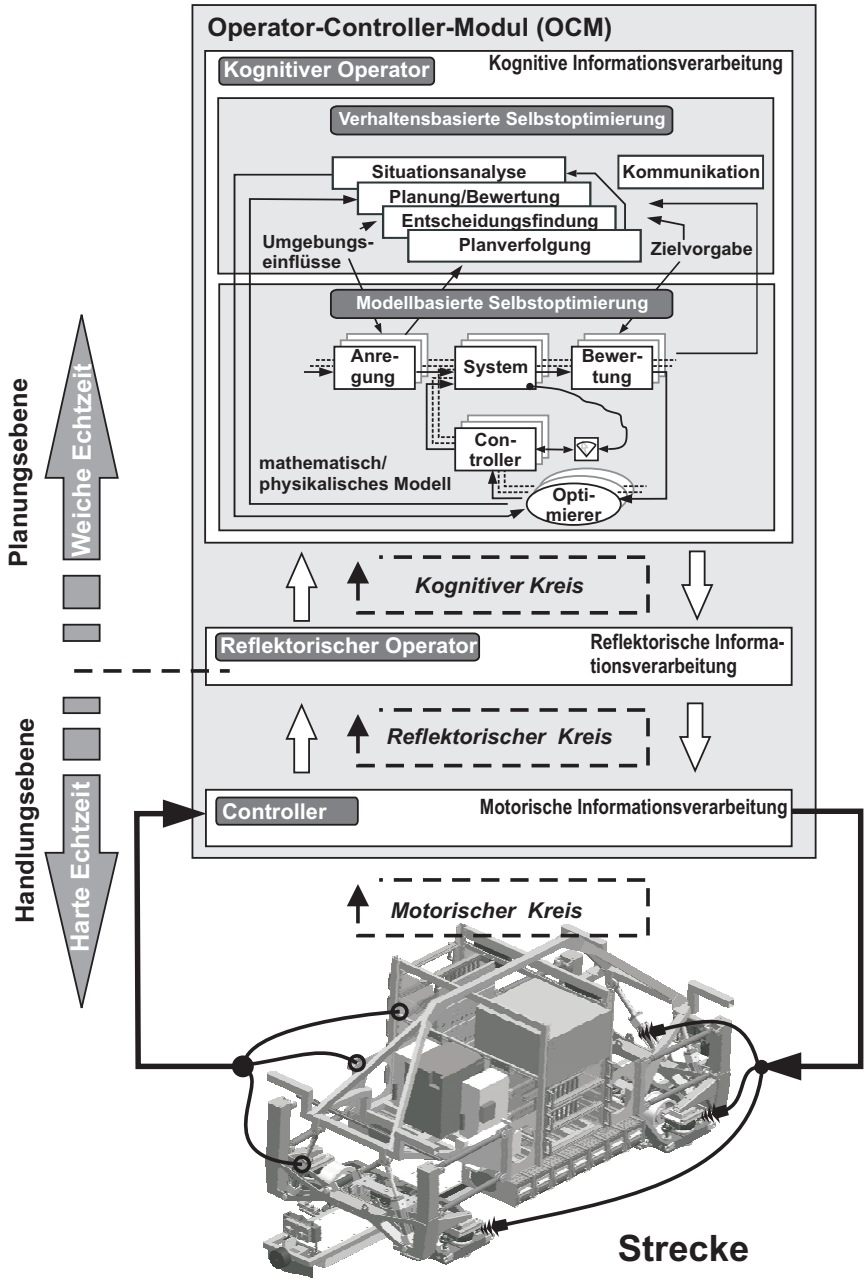


Abbildung 3.2: Der kognitive Operator des Operator-Controller-Moduls

Schmidt in [MS04] von einer Einteilung für Verhaltenssteuerungen, die der hier vorgestellten Einteilung in motorische, reflektorische und kognitive Informationsverarbeitung sehr nahe kommt, jedoch für Anwendungen des Maschinenbaus weniger geeignet erscheint. Dabei wird zwischen reaktiver, deliberativer und reflexiver Ebene unterschieden. Die reaktive Ebene entspricht näherungsweise einer Kombination aus den hier vorgestellten Ebenen Controller und reflektorischer Operator; deliberative und reflexive Ebene finden sich in der kognitiven Ebene wieder.

Das Operator-Controller-Modul (OCM) weist weiterhin Parallelen zu bekannten MAS-Schichtenarchitekturen auf. Seine Architektur kann als eine vertikale Schichtung mit entkoppelter 2 Phasen-Architektur für den Informationsfluss der Wahrnehmung aufgefasst werden. Die *Subsumption Architekture* von Brooks [Bro91] führte zuerst eine hierarchische Struktur aus repräsentationslosen, rein reaktiven Schichten ein, die aufbauend auf der untersten Schicht, die lediglich reflexartiges Verhalten hervorbringt, durch Überlagerung, Verstärkung und Inhibition höhere, komplexere Verhaltensweisen zu erzeugen in der Lage ist.

Die TouringMachine Architektur [Fer92] arbeitet nicht repräsentationslos, nimmt aber den Gedanken einer reaktiven Schicht auf, die durch abstraktere, kognitive Schichten ergänzt wird. Das OCM lässt sich zu dieser Architektur wie folgt in Beziehung stellen: Der Controller entspricht der reaktiven Schicht, der reflektorische Operator der Planungsschicht, wobei er zusätzlich die Aufgabe übernimmt, das Verhalten der Schichten geeignet zu kontrollieren, und der kognitive Operator reflektiert wie die Modellierungsschicht der TouringMachines mittels model-basierter oder verhaltens-basierter Verfahren über die aktuelle Situation.

Da die Schichten des OCMs gewissen strukturellen Einschränkungen genügen müssen, ermöglicht die OCM Architektur im Gegensatz zu üblichen Agentenarchitekturen die benötigten Vorhersagen über kritische Verhaltensaspekte (Stabilität, Zeitinkonsistenzen, Verklemmungen) des Systems, damit ein sicherer Betrieb des Systems gewährleistet werden kann (siehe Kapitel 3.1.3).

Im Rahmen der OCM-Architektur können die drei Aktionen der Selbstoptimierung – Ist-Analyse, Zielbestimmung und Verhaltensanpassung – auf vielfältige Art und Weise durchgeführt werden. Wenn die selbstoptimierende Anpassung Echtzeitanforderungen genügen muss, werden in einem individuell selbstoptimierenden System alle drei Aktionen im reflektorischen Operator durchgeführt. Systeme, die die Selbstoptimierung nicht in Echtzeit durchführen müssen, können hier aufwändigere Verfahren einsetzen, die im kognitiven Operator angesiedelt werden. Die Verhaltensan-

passung erfolgt in diesem Fall indirekt unter Vermittlung durch den reflektorischen Operator, der die Anweisungen zur Verhaltensanpassung auf geeignete Weise mit dem Echtzeitablauf des Controllers synchronisieren muss. Daneben können innerhalb eines einzelnen OCM auch Mischformen auftreten, bei denen die beiden beschriebenen Formen der Selbstoptimierung parallel und zueinander asynchron ablaufen. Bei zusammengesetzten selbstoptimierenden Systemen, die mehrere OCMs umfassen, kann die notwendige Kommunikation wiederum über den reflektorischen oder den kognitiven Operator stattfinden. Echtzeitschranken können allerdings nur garantiert werden, wenn die Koordination über die reflektorischen Operatoren realisiert wird.

3.1.2 Komposition

Systeme sind neben der Hierarchie auch durch den sich für jede Konfiguration ergebenden Wirkfluss gekennzeichnet. Der Wirkfluss der Elemente des Integrationsmodells kann über Ereigniskanäle, diskrete Signalkanäle oder quasi-kontinuierliche Signalkanäle erfolgen und ist im Interesse einer möglichst allgemeinen Darstellung hier nicht im Modell mit aufgeführt. Dieser Wirkfluss führt im Weiteren zu gerichteten und bidirektionalen Verbindungen zwischen den Ein- und Ausgängen der Subsysteme.

Ein OCM repräsentiert beim Entwurf selbstoptimierender mechatronischer Systeme jeweils einem System, also einer zusammenhängenden Menge von eindeutig zugeordneten Elementen. Ein hierarchisches Gesamtsystem entspricht dementsprechend einem Baum von OCM. Die Einordnung in die verschiedenen Bestandteile des OCM ist nur dann eine wohlstrukturierte Architektur, wenn die folgenden Regeln gelten:

- Alle Elemente, 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 und der reflektorische Operator immer auch ohne agierenden kognitiven Operator insoweit autonom handlungsfähig bleiben muss, dass sein Verhalten ausreichend für den sicheren Betrieb des OCM ist.

- Für die Verbindungen zwischen den Bestandteilen der verschiedenen OCM muss darüber hinaus gelten, dass Controller nur mit Controllern des übergeordneten und der untergeordneten OCM verbunden sind und reaktive Operatoren nur mit den reaktiven Operatoren des übergeordneten und der untergeordneten reaktiven Operatoren. Für die kognitiven Operatoren ist eine ähnliche Struktur sinnvoll, aber nicht zwingend erforderlich.

Die oben beschriebene Einschränkung auf wohlstrukturierte Architekturen garantiert, dass die Entkopplung zwischen Controller und reflektorischem Operator sowie zwischen reflektorischem und kognitivem Operator der in Abschnitt 3.1.1 ausgeführten Architektur entsprechen und es somit zu keinem unüberwachten, direkten Zugriff der kognitiven Funktionen auf den Controller kommen kann.

3.1.3 Verhaltensanpassung

Im Rahmen dieser Architektur kann nun sicher das Verhalten des Systems angepasst werden, wobei neben den hier besonders herausgehobenen Strukturanpassungen auch eine einfach Anpassung von Systemparametern möglich ist.

Parameteranpassung

Zunächst ist innerhalb des Controllers weiterhin eine Parameteranpassung in harter Echtzeit im Sinne einer klassischen adaptiven Regelung denkbar. Interessanter ist in Zusammenhang mit der OCM-Architektur jedoch die Möglichkeit, aus dem reflektorischen Operator heraus die Parameter des Controllers anzupassen. Da diese Anpassungen ebenfalls in harter Echtzeit ablaufen, ist trotz der Verwendung potentiell komplexer Anpassungsregeln auch hier eine rechtzeitige und sichere Reaktion auf sich ändernde Einflüsse garantiert. Weiterhin kann der kognitive Operator die Parameter des reflektorischen Operators beeinflussen und das verwendete Regelsystem anpassen, während ihm der direkte Durchgriff auf den Controller aber prinzipiell verwehrt ist. Konzepte zur Beschreibung entsprechender Parameteranpassungen sind in [GBK+03] beschrieben.

Rekonfiguration

Der sicherheitskritische Charakter mechatronischer Systeme [Sto96] bedingt, dass nur solche rekonfigurierbaren Systeme adäquat sind, die einer wohlstrukturierten Architektur genügen und ein vorhersagbar sicheres Verhalten zeigen. Über die Wohlstrukturiertheit hinaus sind die folgenden Eigenschaften zu betrachten:

1. Eine Konfiguration ist strukturell korrekt, wenn keine unverbundenen Eingänge oder Ausgänge existieren. Ist dies nicht der Fall, können die kontinuierliche Auswertung oder fehlende Ereignisse zu undefiniertem Verhalten führen. Sind alle Konfigurationen, die sich durch Kombination aus den einzelnen, durch die Zustände der Elemente beschriebenen Teilkonfigurationen ergeben, strukturell korrekt, so sprechen wir von **allgemein korrekter Rekonfiguration** *(generically correct reconfiguration)*. Sind dagegen nur die durch die Interaktion des Systems konkret erreichbaren Konfigurationen strukturell korrekt, so sprechen wir nur von **spezifisch korrekter Rekonfiguration** *(specifically correct reconfiguration)*.

allgemein korrekte Rekonfiguration

spezifisch korrekte Rekonfiguration

2. Die durch den Wirkfluss beschriebene, kontinuierliche Regelung durch die Controller muss für alle erreichbaren Konfigurationen und jeden möglichen Übergang zwischen ihnen die notwendige regelungstechnische Stabilität aufweisen.

3. Das über Ereignisse und diskrete Signale gekoppelte Echtzeitverhalten der reflektiven Operatoren, welches das Notfallverhalten (fail safe und/oder fail operational) einschließt, muss korrekt arbeiten, d.h. es muss notwendige Sicherheitseigenschaften erfüllen und nachweislich frei von Verklemmungen sein.

Während das Problem (1) im Wesentlichen durch eine entsprechende Komponentenarchitektur mit diskreten und kontinuierlichen Anteilen gelöst werden kann [BGO04], stellen die Probleme (2) und (3) eine im Allgemeinen kaum lösbare Herausforderung dar. Durch Ausnutzung der spezifischen Einschränkungen mechatronischer Systeme konnte für Problem (3) bereits eine erste Lösung mittels kompositionalen Modelcheckings [GTB$^+$03] entwickelt werden. Ein Lösungsansatz, der das aus (1) und (3) kombinierte Gesamtproblem behandelt, wird in [GBSO04] beschrieben. Für Problem (2) werden zur Zeit im Wesentlichen Überblendtechniken [Vöc03] sowie Ansätze zum Model-checking hybrider Systeme untersucht.

Kompositionale Anpassung

kompositionale Anpassung

Neben den im letzen Abschnitt behandelten Strukturveränderungen durch Rekonfiguration ist auch eine **kompositionale Anpassung** denkbar, bei der über das Ein- und Ausblenden vorhandener Elemente einer Hierarchie hinaus auch der Zusammenschluss vorher nicht in einer festen Hierarchie angeordneter Elemente möglich ist.

Für die hier betrachteten technischen Systeme ist ein derart flexibles Vorgehen nur dann akzeptabel, wenn dabei nicht die vorhersagbare Sicherheit des Systems leidet. Im Sonderforschungsbereich 614 werden deswegen die folgenden zwei Varianten für die Modellierung solcher dynamischen Strukturveränderungen betrachtet:

1. Dynamische Strukturveränderungen für nicht hierarchische Systeme, die nur über Nachrichten auf der Ebene diskreten Echtzeitverhaltens gekoppelt werden.
2. Dynamische Strukturveränderungen für spontan etablierte quasi-kontinuierliche hierarchische Systeme, die zur Laufzeit aus hierarchischen Teilsystemen kombiniert werden.

Für den ersten Fall wurde im Rahmen des Sonderforschungsbereich 614 bereits ein UML-basierter Entwurfsansatz entwickelt, der mittels Komponenten und Koordinationsmustern die kompositionale Verifikation der Echtzeitkoordination für beliebige Strukturen erlaubt [GTB+03], wenn diese syntaktisch korrekt aus Komponenten und Koordinationsmustern zusammengesetzt sind.

Als Lösung für den zweiten Fall sollen quasi-kontinuierliche hierarchische Systeme mittels diskreter Echtzeitkoordination spontan, aber sicher etabliert werden. Dazu muss die Rekonfiguration mehrerer OCM so koordiniert werden, dass ihre kontinuierlichen Schnittstellen synchronisiert und auf zueinander kompatible Varianten umschalten, um undefinierte Eingabewerte zu vermeiden. Auf Grundlage der Verfahren für rein diskrete Systeme kann dann korrektes Echtzeitverhalten verifiziert werden.

3.2 Entwurf mit Lösungsmustern

Für die effiziente Entwicklung selbstoptimierender Systeme spielt die Beherrschung des komplexen Zusammenspiels zwischen den drei Aktionen der Selbstoptimierung eine zentrale Rolle. Hier ist schon in den frühen Phasen des Entwicklungsprozesses eine systematische Unterstützung durch geeignete Hilfsmittel notwendig.

Muster

In der Produktentwicklung treten häufig Probleme auf, die zuvor bereits erfolgreich gelöst worden sind. In der Architektur wurde in diesem Zusammenhang die Idee formuliert [AIS+77], dass sich der Kern einer Lösung für ein spezifisches Problem als **Muster** *(pattern)* beschreiben lässt, das in analogen Problemsituationen herangezogen werden kann, um wiederkehrende Probleme nicht jedes Mal von Grund auf neu lösen zu müssen. Dieser Gedanke wurde in die Domäne der Informationsverarbeitung übertragen und dort durch das Konzept des **Entwurfsmusters** *(design pattern)* [GHJV95] systematisiert. Das gemeinsame Prinzip ist, dass jeweils ein spezifisches Problem und die charakteristischen Elemente möglicher Lösungen und ihr Zusammenwirken, also Struktur und Verhalten der Lösungen, in verallgemeinerter Form formal beschrieben werden. Mit den **Wirkprinzipien** existiert im Maschinenbau und in der Elektrotechnik ein ähnliches Konzept. Beschreibungen dieser Art bezeichnen wir im Weiteren als **Lösungsmuster** *(solution pattern)*, wobei wir vereinfachend auch **Muster** als Synonym verwenden.

Entwurfsmuster

Wirkprinzip

Lösungsmuster

Wirkmuster zur Selbstoptimierung

Die hier vorgestellten **Wirkmuster zur Selbstoptimierung** (WM_{SO}) sollen dem Entwickler helfen, vorauszudenken, wie die charakteristischen Aktionen der Selbstoptimierung – Analyse der Ist-Situation, Bestimmung der Systemziele und Anpassung des Systemverhaltens – im System umgesetzt werden können. Dabei sind einerseits konkrete Lösungsmuster für Struktur und Ablauf des gesamten Selbstoptimierungsprozesses von Interesse, die abstraktere Konzepte wie die im vorherigen Kapitel beschriebene individuelle und zusammengesetzte Selbstoptimierung umsetzen. Gleichzeitig bietet es sich an, auch bei den informationsverarbeitenden Prozessen im Rahmen der einzelnen Aktionen der Selbstoptimierung auf wiederverwendbaren Lösungsmustern aufzubauen. Der neu eingeführte Begriff des Wirkmusters zur Selbstoptimierung (siehe auch [GFG+05]) soll diese beiden Arten von Lösungsmustern zusammenfassen.

Im Weiteren werden zuerst das verwendete Vokabular eingeführt und die verschiedenen Musterbegriffe genauer definiert und zueinander in Bezie-

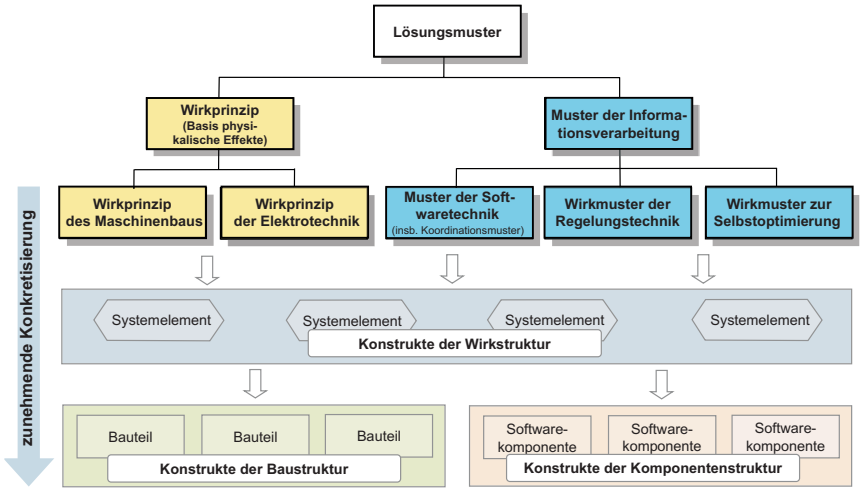


Abbildung 3.3: Klassifizierung von Lösungsmustern

hung gesetzt. Danach wird gezeigt, wie Lösungsmuster während der Produktentwicklung zur Bestimmung und Konkretisierung des grundlegenden Aufbaus und der Funktionsweise selbstoptimierender Systeme eingesetzt werden können. Anschließend werden die wichtigsten Musterklassen ausführlich vorgestellt. Abschließend wird anhand des Anwendungsszenarios „Kooperatives Lernen beim Befahren einer Strecke“ die Verwendung von Wirkmustern zur Selbstoptimierung und ihre Umsetzung in softwaretechnische **Koordinationsmuster** *(coordination pattern)* exemplarisch vorgeführt.

Koordinationsmuster

3.2.1 Begriffsdefinitionen

Die unterschiedlichen Musterbegriffe sollen nun gegeneinander abgegrenzt und in einer hierarchischen Struktur eingeordnet werden. Abbildung 3.3 stellt überblicksartig die Begriffe dar, die im Folgenden genauer definiert werden:

Wirkprinzipien sind Lösungsmuster des Maschinenbaus und der Elektrotechnik entsprechend der Definition nach Pahl/Beitz [PB97], d.h. sie beschreiben den Zusammenhang von physikalischem Effekt und stofflichen und geometrischen Merkmalen (Wirkgeometrie, Wirkbewegung und Werkstoff).

Wirkprinzip

Muster der Softwaretechnik

Als **Muster der Softwaretechnik** bezeichnen wir Problem/Lösungs-Paare, welche erprobtes Software-Engineering-Wissen für neue Problemkontexte anwendbar machen. Die Lösungsbeschreibung besteht aus einer Struktur, sowie dem partiellen Verhalten der einzelnen Strukturelemente.

Wirkmuster der Regelungstechnik

Wirkmuster der Regelungstechnik beschreiben, wie eine Regelstrecke nachgebildet und beeinflusst oder Größen einer Strecke gemessen oder beobachtet werden können.

Wirkmuster zur Selbstoptimierung

Wirkmuster zur Selbstoptimierung (WM_{SO}) erfüllen Funktionen zur Selbstoptimierung wie autonomes Planen, Kooperieren, Handeln und Lernen. Das Spektrum der WM_{SO} umfasst den gesamten Selbstoptimierungsprozess (Analyse der Ist-Situation, Anpassung der Systemziele, Anpassung des Systemverhaltens) oder nur Teile davon.

Muster der Informationsverarbeitung

In der Entwicklung selbstoptimierender Syteme des Maschinenbaus wird Selbstoptimierung zu einem großen Teil durch die Informationsverarbeitung realisiert. Unter dem allgemeinen Begriff **Muster der Informationsverarbeitung** fassen wir daher Wirkmuster der Regelungstechnik, Wirkmuster zur Selbstoptimierung sowie Muster der Softwaretechnik zusammen.

Bauteil

Gestaltbehaftete Systemelemente werden während der Produktentwicklung zu gestaltbehafteten **Bauteilen** konkretisiert und zu Baugruppen aggregiert. Bauteile und Baugruppen sind Entitäten der Baustruktur[1]. Die rechnerinterne Repräsentation eines Bauteils bzw. einer Baugruppe besteht aus unterschiedlichen Aspekten wie Verhalten und Gestalt.

Softwarekomponente

Nach [Szy99] ist eine **Softwarekomponente** *(software component)* ein zusammenhängendes Softwareelement mit spezifizierten Schnittstellen. Ein Softwaresystem wird bei der Systemkomposition durch Zusammensetzen von Softwarekomponenten erstellt. Softwaretechnische Systemelemente werden während der Konzipierung definiert und in den folgenden Prozessschritten zu Softwarekomponenten konkretisiert.

Systemelement

Ein **Systemelement** repräsentiert einen Teil eines Systems, der noch nicht endgültig ausgeprägt ist. Es kann ein oder mehrere Lösungsmuster umsetzen. Systemelemente werden zur Modellierung der Wirkstruk-

[1] Baustruktur: In Anlehnung an Pahl/Beitz [PB97] handelt es sich um den Bauzusammenhang eines maschinenbaulichen Systems, d.h. um die Anordnung der gestaltbehafteten Bauteile im Raum und ihre logische Aggregation zu Baugruppen und Erzeugnissen. Diese Aggregation kann nach unterschiedlichen Gesichtspunkten wie Funktionalität, Montage, Service etc. erfolgen.

tur[2] verwendet. Während der Produktentwicklung werden die Systemelemente konkretisiert und können zu Modulen, Bauteilen/Baugruppen oder Softwarekomponenten zusammengefasst werden. Die rechnerinterne Repräsentation eines Systemelements besteht aus unterschiedlichen Aspekten wie Verhalten und Gestalt. Während der Produktentwicklung weisen die Aspekte unterschiedliche Konkretisierungen auf, die den Phasen des Entwicklungsprozesses entsprechen. Der Aspekt Gestalt enthält zunächst grobe Festlegungen für die Bestimmung der Prinziplösung[3] und später weitergehende Festlegungen für die Bestimmung der Baustruktur. Der Aspekt Verhalten weist für den Fall von Software beispielsweise für die frühen Entwicklungsphasen Zustands- und Interaktionsdiagramme und für die spätere Entwicklungsphase Code auf.

Lösungselement

Ein **Lösungselement** ist eine realisierte und bewährte Lösung zur Erfüllung einer oder mehrerer Funktion/-en. Dabei handelt es sich im Allgemeinen um ein Modul, ein Bauteil, eine Baugruppe oder eine Softwarekomponente, das/die auf einem oder mehreren Lösungsmustern (Wirkprinzip, Softwaremuster, Wirkmuster der Regelungstechnik oder Wirkmuster zur Selbstoptimierung) beruht. Bauteile bzw. Baugruppen sind gestaltbehaftete Lösungselemente und Teil der Baustruktur. Rein softwaretechnische Lösungselemente bestehen aus Softwarekomponenten und sind Teil der Komponentenstruktur[4]. Module sind in der Regel eine Kombination aus Baugruppen und Softwarekomponenten. Die rechnerinterne Repräsentation eines Lösungselementes besteht aus unterschiedlichen Aspekten wie Verhalten und Gestalt.

Lösungsmuster werden während der Produktentwicklung zur Erarbeitung von Produktkonzepten, Entwürfen und deren Realisierung/Implementierung eingesetzt, d.h. Lösungsmuster resultieren letztendlich in Bauteilen und Softwarekomponenten. Wirkprinzipien werden durch Bauteile und Muster der Informationsverarbeitung in der Regel durch Softwarekomponenten umgesetzt, wobei in Ausnahmefällen ein

[2] Wirkstruktur: Sie bildet die Lösungsmuster repräsentierenden Systemelemente, deren Eigenschaften sowie die Beziehungen der Systemelemente zueinander ab. Sie beschreibt den grundsätzlichen Aufbau und die Wirkungsweise des zu entwickelnden Systems.

[3] Prinziplösung (synon. prinzipielle Lösung): Grundsätzliche Lösung für eine Entwicklungsaufgabe, die Festlegungen zur physikalischen Wirkungsweise und zur Art und Anordnung der Komponenten trifft, ohne diese aber im Detail zu definieren [PB97]. Basis für die Prinziplösung ist die Wirkstruktur, die eine Kombination von Lösungsmustern/Systemelementen zur Beschreibung einer Lösung ist. Häufig wird die Wirkstruktur um erste Berechnungen und grobe Festlegungen der Geometrie ergänzt, damit die prinzipielle Lösung deutlich genug erkennbar wird.

[4] Komponentenstruktur: Nach Oestereich [Oes98] handelt es sich um das Gefüge der Komponenten (bzw. synonym Softwarekomponenten), das mit einem Komponentendiagramm dargestellt wird.

Wirkmuster der Regelungstechnik auch durch mechanische Regler (z.B. Fliehkraftregler) realisiert werden kann. Moderne technische Systeme des Maschinenbaus, etwa ein mechatronisches Funktionsmodul, beruhen auf einer Kombination aus physikalischen Effekten und Informationsverarbeitung. Entsprechend bestehen sie aus einer Baustruktur und einer Komponentenstruktur. Diese beiden Strukturen gehen aus der Wirkstruktur hervor. Zur Modellierung der Wirkstruktur haben wir das Konstrukt Systemelement eingeführt. Systemelemente bilden im Zuge der Konkretisierung einen Zwischenschritt zwischen Lösungsmustern und Bauteilen bzw. Softwarekomponenten. Die Beziehungen zwischen den Konstrukten zweier Konkretisierungsebenen haben den Typ n:m. Es ist während der Produktentwicklung ein grundsätzliches Ziel, auf realisierte und bewährte Lösungen zurückzugreifen. Diese bezeichnen wir als Lösungselemente. Häufig werden Lösungselemente auch als Katalogteile, Zulieferkomponenten etc. bezeichnet. Lösungselemente können Bauteile wie Standardgehäuse oder Kugellager, Softwarekomponenten oder eine Kombination aus Bauteilen und Softwarekomponenten wie intelligente Motoren mit Steuerung sein.

3.2.2 Lösungsmuster in der Produktentwicklung

Der Einsatz der unter dem Oberbegriff Lösungsmuster vorgestellten Konstrukte und ihr Zusammenwirken wird in Abbildung 3.4 veranschaulicht. Hier sind wichtige während der Konzipierung[5] zu entwickelnde und in den folgenden Phasen zu konkretisierende Partialmodelle[6] entsprechend ihres Konkretisierungsgrads bzgl. der Systembeschreibung aufgeführt. Es werden u.a. die Funktionshierarchie und die Wirkstruktur eines Systems entwickelt, sowie Teile der Baustruktur und ggf. Ausschnitte der Komponentenstruktur vorausgedacht. In den anschließenden Entwicklungsphasen werden auf Basis der Wirkstruktur die Baustruktur und die Komponentenstruktur konkretisiert.

[5] Die Konzipierung ist eine Phase im Produktentwicklungsprozess. Hier werden die wesentlichen Weichenstellungen für das Produkt getroffen: In einem iterativen Prozess wird die Wirkstruktur, d.h. die aus den Lösungsmustern konkretisierten Systemelemente sowie deren Zusammenwirken, und die grobe Baustruktur, d.h. die Gestalt der einzelnen Elemente und deren räumliche Anordnung, festgelegt. Das Verhalten des Systems wird durch Ablauf- und Anpassungsprozesse beschrieben, inkl. der ausgewählten Konzepte der Selbstoptimierung. Das Resultat ist die Prinziplösung.

[6] Partialmodelle sind verschiedene Sichten auf das zu entwickelnde System. Während der Phase Konzipierung sind die Beschreibungen weniger konkret als in den folgenden Phasen Grobgestaltung, Reglerentwurf, Integration der Informationsverarbeitung, virtuelles Prototyping und Ausarbeitung.

Den Anfangspunkt markiert die Funktionshierarchie, die primär aus der Analyse der Anforderungen entsteht. Für die einzelnen Funktionen werden Lösungen ermittelt. Dies können Wirkprinzipien, Softwaremuster, Wirkmuster der Regelungstechnik, WM_{SO} oder – wenn bekannt – auch Lösungselemente sein.

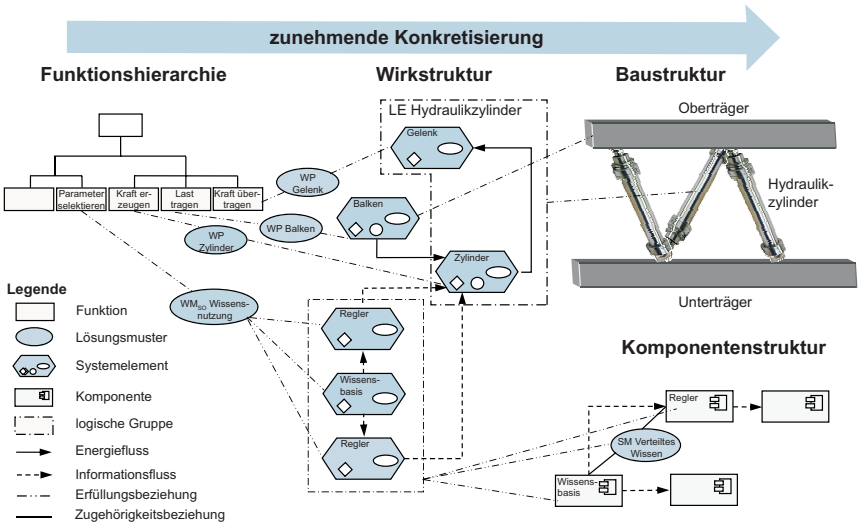


Abbildung 3.4: Konstrukte zur Beschreibung der Partialmodelle Funktionshierarchie, Wirkstruktur, Baustruktur und Komponentenstruktur

Die Lösungsmuster werden zu Systemelementen konkretisiert. Die Verknüpfung der Systemelemente mit Energie-, Stoff-, und Informationsflüssen führt zur Wirkstruktur. Bereits bekannte Lösungselemente werden in der Wirkstruktur als Systemelemente behandelt. Mit der Auswahl geeigneter Lösungsmuster und Lösungselemente sowie deren Umsetzung in Systemelemente und deren Beziehungen ergibt sich Schritt für Schritt die Wirkstruktur. Gleichzeitig entstehen Vorstellungen über die Gestalt des Systems. Dazu werden gestaltbehaftete Systemelemente zu Bauteilen konkretisiert und unter Berücksichtigung gestaltbestimmender Anforderungen im Raum positioniert. So können Angaben über Anzahl, Form, Lage, Anordnung sowie Art der Wirkflächen und der Wirkorte des selbstoptimierenden Systems gemacht werden. Daraus entsteht die Baustruktur. Parallel dazu werden die Systemelemente zur Informationsverarbeitung modelliert, zu Softwarekomponenten zusammengefasst und in Komponentendiagrammen dargestellt. Dabei werden in der Wirkstruktur Anwendungsmöglichkeiten für Softwaremuster identifiziert. Ein Identifizierungskriterium ist die Nutzung eines WM_{SO} zur Erarbeitung der Wirkstruktur.

Die durch ein WM_{SO} semiformal spezifizierten informationstechnischen Strukturen und Verhaltensbeschreibungen werden durch Softwaremuster konkretisiert und domänenspezifisch beschrieben. Die Softwaremuster werden durch bereits existierende oder neu zu entwickelnde Softwarekomponenten umgesetzt. Mit Hilfe von Verteilungsdiagrammen kann eine Zuordnung von Komponenten auf Hardware-Einheiten (Bauteile der Baustruktur) beschrieben werden.

3.2.3 Wesentliche Lösungsmusterklassen

Im Folgenden werden Wirkprinzipien, Muster der Informationsverarbeitung und hier insbesondere WM_{SO} näher erläutert.

Wirkprinzipien

Muster bzw. Lösungsmuster des Maschinenbaus und der Elektrotechnik sind Wirkprinzipien entsprechend der Definition nach Pahl/Beitz, d.h. sie beschreiben den Zusammenhang vom physikalischen Effekt sowie geometrischen und stofflichen Merkmalen (Wirkgeometrie, Wirkbewegung und Werkstoff) [PB97]. Klassische Wirkprinzipien aus dem Bereich Maschinenbau sind beispielsweise auf Reibung basierende Passungen oder auf dem Hebeleffekt beruhende Seilwinden. Beispiele für Wirkprinzipien aus dem Bereich Elektrotechnik sind Kondensatoren und Elektromotoren, die die physikalischen Effekte elektrisches Feld und Lorentzkraft nutzen. Abbildung 3.5 enthält einige Beispiele für Wirkprinzipien der Domäne Elektrotechnik.

Muster der Informationsverarbeitung

Muster der Informationsverarbeitung

In der Entwicklung selbstoptimierender Syteme des Maschinenbaus wird Selbstoptimierung zu einem großen Teil durch die Informationsverarbeitung realisiert. Wirkmuster der Regelungstechnik, Wirkmuster zur Selbstoptimierung sowie Muster der Softwaretechnik fassen wir unter dem allgemeinen Begriff **Muster der Informationsverarbeitung** zusammen. Diese Aufteilung ist nicht disjunkt, da zum Beispiel ein Feedback-Controller sowohl als Architekturmuster in der Softwaretechnik als auch als Wirkmuster der Regelungstechnik bekannt ist. Die unterschiedlichen Musterbegriffe unterscheiden sich auch hinsichtlich der Nutzungsphase und dem

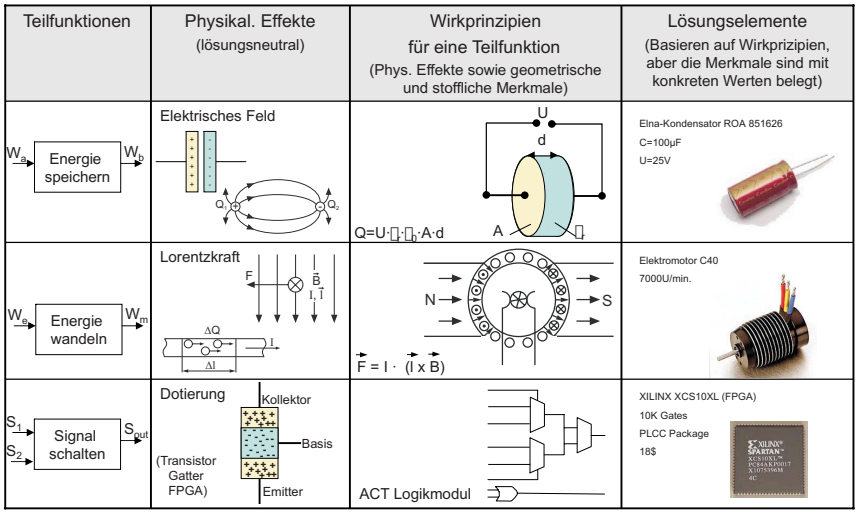


Abbildung 3.5: Beispiele für Wirkprinzipien der Domäne Elektrotechnik

Abstraktionsgrad. Wirkmuster zur Selbstoptimierung werden auf einer abstrakteren Ebene beschrieben und typischerweise in den frühen Phasen bei der Beschreibung der Wirkstruktur genutzt, während Wirkmuster der Regelungstechnik und Muster der Softwaretechnik eher später genutzt werden und weniger abstrakt beschrieben werden. Unter einem **Muster der Softwaretechnik** verstehen wir ein Problem/Lösungs-Paar, welches erprobtes Software-Engineering-Wissen für neue Problemkontexte anwendbar macht. Die Lösungsbeschreibung besteht aus einer Struktur, sowie dem partiellen Verhalten der einzelnen Strukturelemente. Es enthält Hinweise, wie es in neuen Situationen genutzt und umgesetzt werden kann. Ferner wird erklärt, welcher Nutzen erzielt werden kann und welche Kompromisse dabei eingegangen werden müssen. Eine Softwarekomponente kann ein oder mehrere Muster umsetzen. Dabei wird die Softwarekomponente mit einem oder mehreren Elementen aus dem Muster in Beziehung gesetzt und übernimmt das partielle Verhalten dieser Elemente. In der Softwaretechnik existieren Muster für verschiedenste Einsatzzwecke [Ris00]. Die Softwaretechnik kategorisiert Muster häufig nach dem betrachteten Abstraktionsgrad (z.B. Architekturmuster, Idiome etc.) oder Zweck (Analysemuster, Entwurfsmuster, Koordinationsmuster etc.).

Muster der Softwaretechnik

Entwurfsmuster nach [GHJV95] bestehen aus den grundlegenden Elementen: Name, Problembeschreibung, abstrakte Lösung und Implementierungsvarianten. Die abstrakte Lösung setzt sich dabei aus einer Struk-

turbeschreibung der beteiligten Klassen (Participants) sowie einer Beschreibung des jeweiligen Teilverhaltens (Collaborations) dieser Klassen zusammen. Entwurfsmuster sind aufgrund der abstrakten Lösung beim Entwurf beliebiger Software nutzbar. Die Spezifizierung der Entwurfsmuster erfolgt mit Spezifikationstechniken wie der UML (Unified Modelling Language).

Bisherige Spezifikationen von Mustern der Softwaretechnik sind für den Entwurf selbstoptimierender Systeme mit hohen Anforderungen bezüglich Sicherheit und Echtzeitverhalten nicht ausreichend. Insbesondere muss hier sichergestellt werden, dass das Koordinationsverhalten im reflektorischen Operator keine Fehler aufweist. Ein Fehler im reflektorischen Operator bzgl. der Koordination kann zu Unfällen führen, während ein Fehler im kognitiven Operator nur zu schlechterem, weniger optimalen Verhalten führen kann. Für die Entwicklung der Software des reflektorischen Operators haben wir **Koordinationsmuster** *(coordination pattern)* entwickelt, die diesen besonderen Anforderungen genügen [GTB+03], [BTG04]. Demnach beschreiben Koordinationsmuster das partielle Teilverhalten mehrere Elemente bzgl. einer bestimmten Koordination. Die Struktur eines Koordinationsmusters besteht aus mehreren Rollen, die über Konnektoren miteinander verbunden sind, sowie den zu verifizierenden Sicherheitseigenschaften. Des Weiteren wird jeder Rolle ein Verhalten in Form von Real-Time-Statecharts zugeordnet [GTB+03], [BTG04]. Dieses Koordinationsverhalten wird mittels formaler Methoden bzgl. Sicherheitseigenschaften verifiziert und nach erfolgreicher Verifikation in einer Musterbibliothek abgelegt. Später beim Entwurf der Softwarekomponente des reflektorischen Operators werden passende Koordinationsmuster aus der Bibliothek entnommen und wiederverwendet. Hierbei kann das Verhalten speziell für den Anwendungsfall in engen Grenzen, die nicht die Sicherheit einschränken, angepasst werden. Diese Anpassung des Verhaltens nennt man Verfeinerung. Das informationstechnische System besteht demnach aus einer Komposition von Softwarekomponenten, die Systemelemente aus der Wirkstruktur und Softwaremuster umsetzen. Softwarekomponenten lassen sich analog zu den Lösungselementen der Mechanik und Elektrotechnik in verschiedenen Systemumgebungen einsetzen und wieder verwenden. In Abbildung 3.6 wird das Koordinationsmuster „Verteilte Wissensnutzung“ und dessen Anwendung bei der Entwicklung des reflektorischen Operators eines Shuttles dargestellt.

Koordinationsmuster

Auch in der Domäne Regelungstechnik gibt es Lösungsmuster, nach deren Prinzip Teilfunktionen erfüllt werden (Abbildung 3.6). Diese bezeichnen wir als **Wirkmuster der Regelungstechnik**. Sie beschreiben wie eine Regelstrecke nachgebildet, beeinflusst oder Größen einer Strecke

Wirkmuster der Regelungstechnik

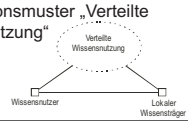
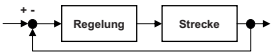
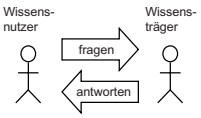
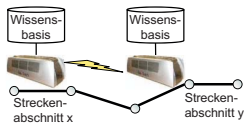
Teilfunktionen	Muster der Informationsverarbeitung für eine Teilfunktion	Lösungselemente / Softwarekomponenten (Basieren auf Lösungsmustern, aber die Werte sind konkret ausgeprägt)
Wissen nutzen	Koordinationsmuster „Verteilte Wissensnutzung“ 	
Größe beeinflussen	Wirkmuster der Regelungstechnik „Regelung“ 	• P-Regler • PI-Regler (para.) • PD-Regler (n. para.) • Riccatti-Regler • Allg. Zustandsregler
Vorgabebahn bestimmen	Wirkmuster zur Selbstoptimierung „Erfahrung nutzen“ 	Trajektorien mit Hilfe von Erfahrungswissen bestimmen 

Abbildung 3.6: Beispiele für Muster der Informationsverarbeitung

gemessen oder beobachtet werden können. Lösungselemente wie der P-Regler oder der Ricatti-Regler basieren auf dem Wirkmuster „Regelung“ und Lösungselemente wie der Kalman-Filter auf dem Wirkmuster „Beobachter“.

Wirkmuster zur Selbstoptimierung

Wirkmuster zur Selbstoptimierung (WM_{SO}) erfüllen Funktionen zur Selbstoptimierung[7] wie autonomes Planen, Kooperieren, Handeln und Lernen. WM_{SO} stellen Schemata zur Verfügung, mit deren Hilfe die Wirkstruktur und insbesondere das Verhalten des selbstoptimierenden Systems spezifiziert werden können. In Anlehnung an die Entwurfsmuster der Softwaretechnik beinhaltet das WM_{SO} die Bestandteile (Aspekte) Prinzipbeschreibung, Anwendungsszenario, Struktur, Verhalten und Verfahren (Abbildung 3.7). WM_{SO} beschreiben die genannten Aspekte semiformal. Sie werden mit Hilfe der Spezifikationstechnik zur Beschreibung

[7] Neben klassischen maschinenbaulichen Funktionen „Kraft übertragen“ oder „Energie wandeln“ werden neuartige Funktionen wie autonomes Planen, Kooperieren, Handeln oder Lernen – so genannte Funktionen zur Selbstoptimierung – zur Beschreibung der Funktionalität selbstoptimierender Systeme genutzt.

der Prinziplösung selbstoptimierender Systeme[8] modelliert, so dass ein domänenübergreifendes Verständnis des WM_{SO} erreicht wird [GFSV04], [GFRS04]. Sie unterstützen in der domänenübergreifenden Phase Konzipierung das Finden von Lösungen für den grundlegenden Aufbau und die Funktionsweise des Systems sowie die Umsetzung dessen in eine Prinziplösung. Softwaremuster konkretisieren die in den WM_{SO} enthaltenen softwarespezifischen Strukturen sowie Verhaltensbeschreibungen. WM_{SO} umfassen den gesamten Selbstoptimierungsprozess (Analyse der Ist-Situation, Bestimmung der Systemziele, Anpassung des Systemverhaltens) oder nur Teile davon. Wesentlich ist, dass Zustandsübergänge des Systems durch das in einem WM_{SO} spezifizierte autonome, intelligente Verhalten ausgelöst, unterstützt und/oder durchgeführt werden. Im Folgenden werden die Bestandteile von WM_{SO} erläutert.

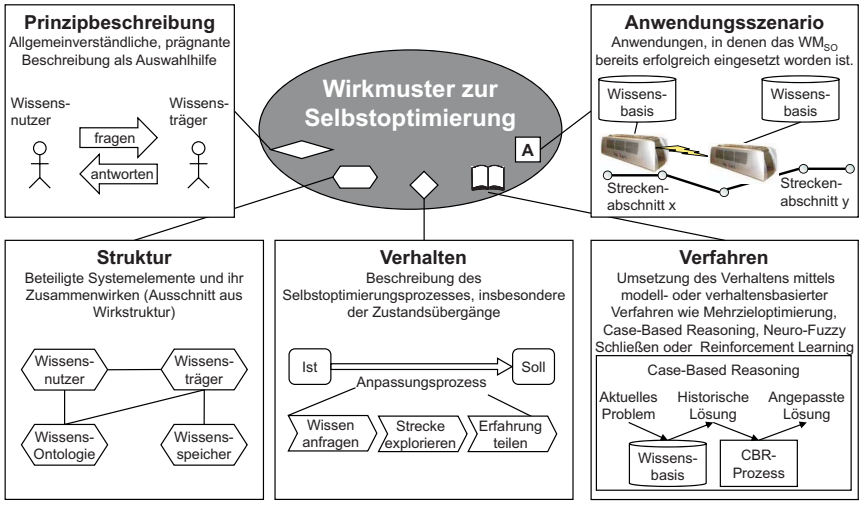


Abbildung 3.7: Bestandteile eines Wirkmusters zur Selbstoptimierung mit beispielhafter Ausprägung

Prinzipbeschreibung

Die **Prinzipbeschreibung** charakterisiert den grundlegenden Gedanken, der hinter dem Wirkmuster steht. Sie dient dazu, dem Entwickler ohne weitere formale Detaillierung einen intuitiven Zugang zum Wirkmuster zu ermöglichen.

[8] Die Spezifikation der Prinziplösung eines s.o. Systems führt zu einem kohärenten System von den sieben Partialmodellen Anforderungen, Umfeld, Ziele, Funktionen, Wirkstruktur, Gestalt und Anwendungsszenarien sowie einem Set von n Partialmodellen, die das Verhalten beschreiben.

Anwendungsszenario

Ein **Anwendungsszenario** spiegelt eine Situation wieder, in denen das Wirkmuster in der Vergangenheit bereits erfolgreich eingesetzt wurde. Diese Szenarien sollen dem Entwickler bei der Selektion eines für die jeweilige Aufgabe relevanten Wirkmusters helfen.

Struktur

Der Aspekt **Struktur** beschreibt, welche Systemelemente grundsätzlich zur Umsetzung des WM_{SO} notwendig sind, sowie welche Beziehungen die Systemelemente zueinander haben. Diese Struktur dient als Schema für einen Ausschnitt der Wirkstruktur des zu entwickelnden Systems. Die Systemelemente nehmen im Kontext des betrachteten WM_{SO} jeweils eine für dieses WM_{SO} spezifische Rolle ein. Die Struktur wird mit der im Sonderforschungsbereich entwickelten semiformalen Spezifikationstechnik zur Beschreibung der Wirkstruktur modelliert [GFRS04].

Verhalten

Der Aspekt **Verhalten** beschreibt das grundsätzlich zu erzielende (Gesamtsystem-)Verhalten, d.h. die Selbstoptimierungsprozesse oder Teile davon. Es geht also um die Modellierung des autonomen, intelligenten Verhaltens, das den Zustandsübergang auslöst, unterstützt und/oder durchführt. Bei der Modellierung eines Zustandsübergangs wird aufgezeigt, wie sich ein System von einem gegebenen Ist-Zustand mit Hilfe eines bestimmten Verfahrens in einen gewünschten Soll-Zustand überführen lässt. Den Zuständen sind jeweils ein Zielsystem, eine Wirkstruktur und Parameter zugeordnet. Neben dem rein physikalischen Zustand – z.B. Zylinder A ist um 10 cm ausgefahren oder Zylinder A ist mit Ventil 1 verbunden – besitzt ein selbstoptimierendes System auch mentale Zustände[9] – beispielsweise Wissen über die Ausprägung des aktuellen Zielsystems oder Vorstellungen des Systems über sein Umfeld. Diese mentalen Zustände können sich ebenfalls während des Betriebes eines selbstoptimierenden Systems ändern, z.B. kann das System aufgrund neuer Sensordaten oder aufgrund von Erfahrungsaustausch mit anderen selbstoptimierenden Systemen neue Vorstellungen über das Umfeld bekommen. Somit beinhalten selbstoptimierende Systeme physikalische und mentale Zustandsübergänge, die durch Anpassungsprozesse beschrieben werden. Da wir von verteilten Systemen ausgehen, kann ein Anpassungsprozess zum einen zwischen verschiedenen Systemen stattfinden. Ein solcher systemexterner Anpassungsprozess kann z.B. durch Kollaboration und Kommunikation erreicht werden. Zum anderen kann ein Anpassungsprozess auch innerhalb eines Systems zwischen Systemelementen ablaufen. Hierbei handelt es sich um einen systeminternen Anpassungsprozess, der z.B. durch Rekonfiguration realisiert werden kann.

[9] Der Begriff des mentalen Zustandes wird auf dem Gebiet der Epistemologie und Künstlichen Intelligenz (KI) genutzt, um die Abbildung von Denkprozessen wie Planen und Problemlösen zu modellieren, siehe u.a. [MA02] [RN03]

Verfahren

Verfahren dienen zur Umsetzung der Selbstoptimierungsprozesse, insbesondere der Anpassung von Zielen und Verhalten (als Folge von Parameter- und ggf. Strukturanpassungen). Sie können verhaltens- oder modellbasiert sein. Beispiele von Verfahren sind Mehrzieloptimierung zur Bestimmung von Zielen, Neuro-Fuzzy Regelschließen zur Anpassung interner Ziele und Case-Based Planning zur erfahrungsunterstützten Verhaltensanpassung.

Ein WM_{SO} kann den gesamten Selbstoptimierungsprozess oder nur Teile davon schematisch beschreiben. Folgendes Beispiel zeigt auf, wie durch mehrere WM_{SO} ein Selbstoptimierungsprozess spezifiziert und autonomes intelligentes Verhalten erreicht werden kann. Dazu stellt Abbildung 3.8 exemplarisch einen Selbstoptimierungsprozess vor und zeigt auf, an welchen Stellen in dem Prozess WM_{SO} zum Einsatz kommen können und ein autonomes, intelligentes Verhalten zur Durchführung eines Zustandsübergangs realisieren. Während der Situationsanalyse werden Erkenntnisse über das Umfeld erlangt. Bei unzureichend bekanntem Umfeld kann das System mit Hilfe des WM_{SO} „Exploration“ dieses erkunden. Aus den Einflüssen auf das System und den aktuellen Zielen kann mit Hilfe des WM_{SO} „Schließen“ auf interne Ziele geschlossen werden (Zielinterpretation). Die so bestimmten Ziele zusammen mit der Zielkonsistenzmatrix ergeben eine in der aktuellen Situation gültige Zielagenda. Aus der Zielagenda wird eine zeitliche Anordnung der zu verfolgenden Ziele gewonnen und in einem Ziel-Schedule dargestellt. Zu einem bestimmten Zeitpunkt sind gewisse vom System verfolgte Ziele aktiv. Es handelt sich entsprechend unserer Systematik um interne Ziele. Diese bilden das Zielsystem. Im Schritt der Verhaltensanpassung wird zunächst aus dem Zielsystem ein zu erreichender Sollzustand abgeleitet. Die Bestimmung des Sollzustandes kann beispielsweise ebenfalls mit Hilfe des WM_{SO} „Schließen“ erfolgen. Danach wird der Anpassungsprozess - d.h. der Weg vom Ist-Zustand zum Soll-Zustand - ermittelt. Die Ermittlung des Anpassungsprozesses kann z.B. auf dem WM_{SO} „Planen“ beruhen. Dabei werden Teilpläne, die bereits in der Vergangenheit in ähnlichen Anpassungsprozessen genutzt wurden, wiederverwendet. Dieses Beispiel zeigt auf, wie intelligentes, autonomes Verhalten durch die Kombination mehrerer WM_{SO} erreicht werden kann.

3.2.4 Anwendung

Der Einsatz von Lösungsmustern wird im Folgenden anhand des Anwendungsszenarios „Kooperatives Lernen beim Befahren einer Strecke“

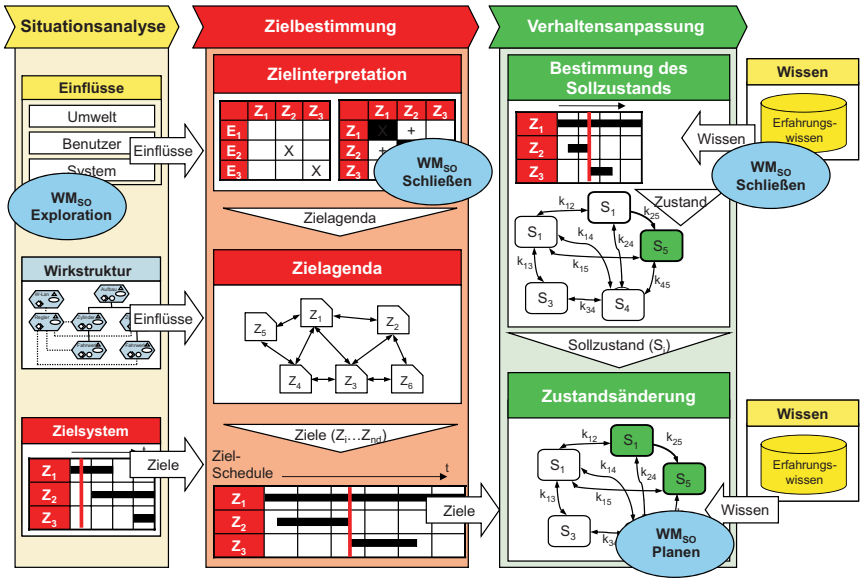


Abbildung 3.8: Beispiel eines Selbstoptimierungsprozesses und Darstellung, wie intelligentes, autonomes Verhalten durch die Kombination mehrerer WM_{SO} erreicht werden kann.

näher erläutert. Beim Befahren eines Streckenabschnitts verfolgt ein Shuttle das externe Ziel, einen hohen Komfort für die Insassen bereitzustellen, das gegen seine inhärenten Ziele, geringer Energieverbrauch und Verschleiß, abgewogen werden muss. Der Komfort hängt von den Bewegungen des Shuttleaufbaus und den dabei wirkenden Beschleunigungen ab, die durch entsprechende Kompensation minimiert werden sollen. Die klassische Regelung verwendet zur Dämpfung der Aufbaubewegungen einen „Skyhook“-Ansatz, bei dem das aktive Feder-Neige-Modul sich so an das Gleisprofil anpasst, dass der Aufbau sich auf einer fest vorgegebenen geglätteten Bahn bewegt. Bei klassischen Lösungen können jedoch im laufenden Betrieb in der einmal eingestellten Vorgabebahn keine auftretenden Änderungen des Gleisprofils berücksichtigt werden, wie sie sich etwa durch eine Abnutzung der Gleise durch wiederholtes Befahren mittelfristig zwangsläufig ergeben. Ziel einer selbstoptimierenden Lösung ist es, den Fahrzeugen im laufenden Betrieb aktuelle erfahrungsbasierte Vorgabebahnen zur Verfügung zu stellen und diesen Ansatz durch Kooperation effizient und wirtschaftlich zu machen.

Wirkmuster zur Selbstoptimierung

Dieses Anwendungsszenario wird während der Produktentwicklung wie folgt ausgearbeitet: Ausgangspunkt ist die Funktion „Vorgabebahn bestimmen“ (Abbildung 3.9). Hierzu wird das WM$_{SO}$ „Erfahrung nutzen“ ausgewählt. Die Auswahl wird durch die im WM$_{SO}$ beschriebenen Prinzipbeschreibungen und Anwendungsszenarien unterstützt. Entsprechend dem Aspekt Struktur sind zur Umsetzung des WM$_{SO}$ mindestens ein Wissensträger und ein Wissensnutzer notwendig. In der Wirkstruktur wird festgelegt, dass für dieses Anwendungsszenario Shuttle Sh_2 der Wissensnutzer und andere Shuttles sowie Streckenabschnitte Wissensträger sind. Mit dem Aspekt Verhalten wird ein Zustandsübergang beschrieben. Der Zustandsübergang wird durch die Aktivitäten „Wissen anfragen“, „Strecke explorieren“ und „Erfahrung teilen“ erreicht. In diesem Fall handelt es sich um einen mentalen Zustandswechsel, der durch einen Zuwachs an Wissen gekennzeichent ist, d.h das Shuttle Sh_2 wechselt aufgrund des Zugriffs auf Erfahrungen anderer seinen Zustand. Als Verfahren zur Umsetzung des Anpassungsprozesses wird Case-Based Reasoning vorgeschlagen. Dieses Verfahren erlaubt die multikriterielle Suche nach ähnlichen Problemstellungen sowie die Anpassung historischer Lösungen an die aktuelle Situation. So gestaltet sich die Verhaltensanpassung von Shuttle Sh_2 derart, dass das Wissen von Shuttle Sh_1 in Form einer in der Vergangenheit erfolgreich eingesetzten Vorgabebahn - Zustand S_0 - für die aktuelle Situation als Ausgangsbasis für die angestrebte Exploration eines optimalen Bahnverlaufs genutzt wird. Dieser Explorationsschritt mündet in den Soll-Zuständen S_1 bzw. S_2. Auf die semiformale Spezifikation des Verhaltens von Sh_2 und das zu nutzende Verfahren werden in der Wirkstruktur durch Verweise hingewiesen. Analog werden zu allen Funktionen der Funktionshierarchie Lösungsmuster ermittelt und diese durch Systemelemente und deren Beziehungen in der Wirkstruktur umgesetzt. Auf diese Weise wird Schritt für Schritt die Wirkstruktur erstellt, aus der sich wiederum die Baustruktur des Systems und die Struktur der Informationsverarbeitung ergeben.

Koordinationsmuster

Zur softwaretechnischen Umsetzung des WM$_{SO}$ „Erfahrung nutzen“ müssen zuerst die informationstechnisch relevanten Systemelemente extrahiert und eine geeignete Komponentenstruktur sowie das zugehörige Verhaltensmodell abgeleitet und formalisiert werden. Das so gewonnene allgemeine Koordinationsmuster kann dann im Anschluss oder im

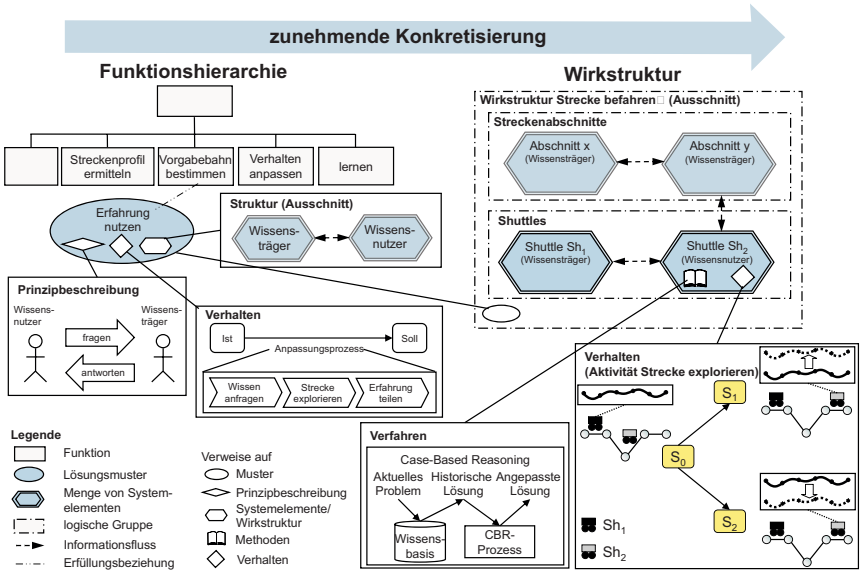


Abbildung 3.9: Von der Funktion über das Wirkmuster zur Wirkstruktur

Rahmen einer späteren Wiederverwendung zur Modellierung konkreter Systeme verwendet werden. Zur Identifikation der beteiligten Elemente werden die Interaktionen im Rahmen des Wirkmusters zunächst durch Erstellen eines oder mehrerer grundlegender Ablaufszenarien dokumentiert. Die Szenarie beschreiben in idealisierter Form die vorgesehenen Abläufe und die daran beteiligten Rollen. Um eine weitgehende Wiederverwendung der Koordinationsmuster zu erlauben, wird dabei möglichst von fallspezifischen Details abstrahiert und ein allgemeingültiges Prinzip extrahiert.

Zur Realisierung des selbstoptimierenden Ansatzes zur Regelung des Fahrzeugaufbaus wird die Vorgabebahn als mathematische Funktion (Trajektorie) kodiert, deren Parameter im laufenden Betrieb variiert werden können, um sie an das reale Gleisprofil oder eine geänderte Gewichtung der Ziele anpassen zu können. Die verwendete Vorgabebahn soll in diesem Fall auf Erfahrungswissen basieren. Dazu werden Informationen über Störungen in der Gleislage eines Streckenabschnitts, über gemachte Erfahrungen der Shuttles bzgl. einzustellender Parameter und des damit erzielten Komforts vom Streckenabschnitt gesammelt und nachfolgenden Shuttles zur Verfügung gestellt. Zur Realisierung dieser Idee soll folgender verteilter Selbstoptimierungsprozess stattfinden: Shuttle Sh1 analysiert seinen gegenwärtigen Status bezüglich Zielgewichtung, Zula-

dung und Energiereserven (Ist-Analyse/1) und übermittelt diese Daten an den Steuerungsrechner des Streckenabschnitts. Auf Basis des verfügbaren Erfahrungswissens berechnet dieser nun einen geeigneten Zielkompromiss und eine entsprechend optimierte Trajektorie (Zielanpassung), die an das Shuttle übermittelt wird. Das Shuttle passt die Vorgabebahn für das Feder-Neige-Modul entsprechend an (Verhaltensanpassung) und durchfährt den Streckenabschnitt. Die dabei gewonnenen Erfahrungen, insbesondere der mit der verwendeten Trajektorie erzielte Komfort und Energieverbrauch, werden erfasst, analysiert und an den Streckenabschnitt zurückübermittelt (Ist-Analyse/2). Der Streckenabschnitt nimmt die Erfahrungen in seine Wissensbasis auf und verwendet sie im Rahmen der Zielanpassung für nachfolgende Shuttles. Der Selbstoptimierungsprozess bei der Durchfahrt baut auf einer verteilten Ist-Analyse auf: der aktuellen Selbstanalyse des Shuttles (Ist-Analyse/1) und den Auswertungen der vorhergehenden Shuttles (Ist-Analyse/2).

Das UML-Sequenzdiagramm in Abbildung 3.10 beschreibt das generische Szenario „Verteilte Wissensnutzung“, das aus dem obigen Konzept extrahiert worden ist: Ein „Lokaler Wissensträger“ verfügt über Erfahrungswissen hinsichtlich eines „Erfahrungsgegenstandes“. Ein „Wissensnutzer“ stellt eine seiner gegenwärtigen Situation angepasste Anfrage an den Wissensträger, der auf Basis der Anfrage und seines Erfahrungswissens eine optimierte Antwort zurückliefert. Der Wissensnutzer nutzt diese Antwort im Rahmen seiner Interaktion mit dem Erfahrungsgegenstand, analysiert die Ergebnisse und schickt seine Erfahrungen an den lokalen Wissensträger zurück, um sie auch anderen Wissensnutzern verfügbar zu machen.

Das UML-Diagramm in Abbildung 3.10 ist zum besseren Verständnis mit Anmerkungen in fetter, kursiver Schrift versehen, um die abstrakten Schritte des Ablaufes (Anfrage, Exploration, Lernen etc.) deutlich herauszustellen. Die numerierten Kreise bezeichnen die drei Aktionen der Selbstoptimierung, Ist-Analyse (1), Zielanpassung (2) und Verhaltensanpassung (3). Wie bereits erwähnt findet aufgrund der verteilten Natur des Selbstoptimierungsprozesses die Ist-Analyse an zwei Stellen statt: zunächst als Teil des unmittelbar ablaufenden Selbstoptimierungsprozesses, abschließend als Teil nachfolgender Selbstoptimierungsprozesse bei der späteren Nutzung durch weitere Wissensnutzer.

Auf Basis des Sequenzdiagramms wird nun ein Koordinationsmuster „Verteilte Wissensnutzung“ abgeleitet, das die Interaktion zwischen Lokalem Wissensträger und Wissensnutzer regelt. Es beschreibt exakt die Verhaltensanforderungen an die beiden Rollen, wobei auch Echtzeitaspekte

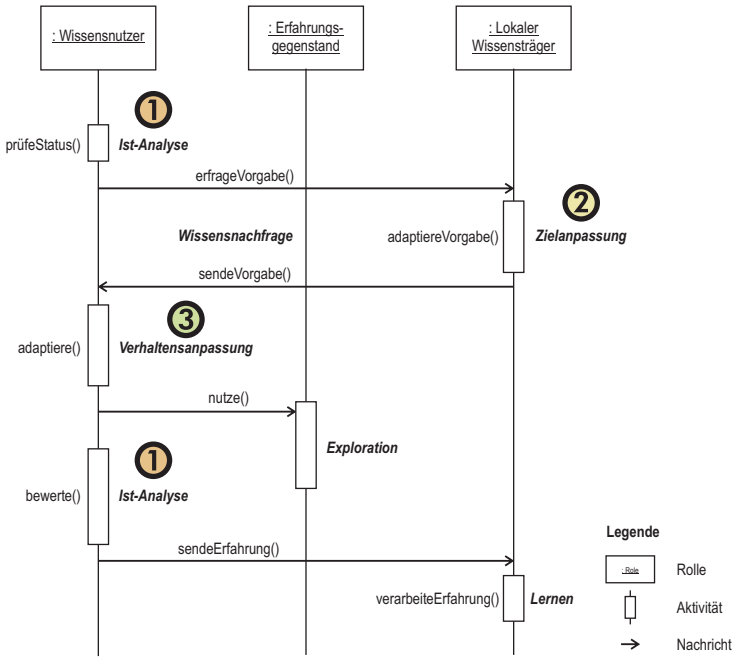


Abbildung 3.10: Das grundlegende Szenario als Sequenzdiagramm

erfasst werden können. Abbildung 3.11 stellt die Struktur des Koordinationsmusters dar. Die beteiligten Rollen sind durch einen Konnektor verbunden, der den Kommunikationskanal zwischen ihnen symbolisiert; sie sind ferner konzeptionell durch das Koordinationsmuster verbunden, das die Kommunikation auf diesem Kanal spezifiziert.

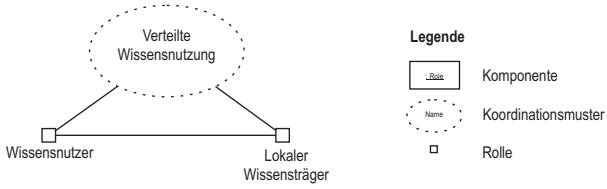


Abbildung 3.11: Formalisierung des Wissensaustausches als Koordinationsmuster

Ausgehend von der Wirkstruktur und unter Verwendung des Koordinationsmusters kann nun ein Komponentendiagramm entwickelt werden. Abbildung 3.12 zeigt einen Teil des Komponentendiagramms, nämlich die Komponenten „Fahrzeug“ und „Streckenabschnitt“. Deren Verhalten wird

partiell durch das abstrakte Koordinationsmuster „Verteilte Wissensnutzung“ beschrieben, das hier konkret zum Austausch der optimierten Vorgabekurven zwischen dem Shuttle und dem Steuerrechner des Streckenabschnitts genutzt wird. Das Fahrzeug übernimmt die Rolle des Wissensnutzers, der Streckenabschnitt fungiert als Lokaler Wissensträger.

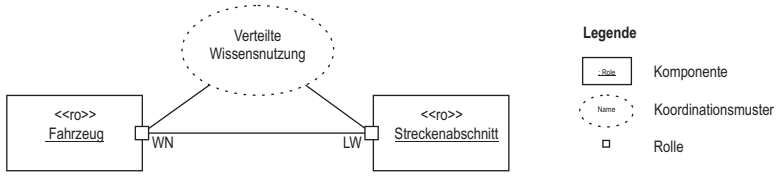


Abbildung 3.12: Ausschnitt aus dem entsprechenden Komponentendiagramm

Das geforderte Verhalten für die beiden Rollen wird in Form von Real-Time Statecharts spezifiziert. Da der Schwerpunkt der Koordinationsmuster auf der präzisen Beschreibung einer sicheren Interaktion zwischen Komponenten liegt, wird dabei von lokalem Verhalten, das für den Verlauf der Kommunikation nicht relevant ist, weitgehend abstrahiert. Daher werden im Beispiel in Abbildung 3.13 konzeptionell wesentliche Schritte wie Bewertung, Zielanpassung oder Lernen nicht explizit erfasst, sondern fließen nur als nichtdeterministisches Verhalten im Rahmen entsprechender Zeitschranken in den Statechart ein. Die Wissensnutzer-Rolle (Fahrzeug) fragt die Wissensträger-Rolle (Streckenabschnitt) nach einer Vorgabekurve. Wenn bis zum Erreichen einer Zeitschranke keine Antwort vorliegt, nimmt das Fahrzeug zur Kenntnis, dass keine Kurve verfügbar ist, und wechselt auf einen robusten Regler, der ohne Vorgabekurve auskommt. Erhält das Fahrzeug hingegen die gewünschten Informationen, nutzt es sie zum Befahren des Streckenabschnitts und schickt die gewonnen Erfahrungen an den Streckenabschnitt zurück. Der Streckenabschnitt (Registry-Rolle) wartet im Wesentlichen auf Anfragen von Fahrzeugen und neue Erfahrungen, die ihm zugesendet werden.

Beim domänenspezifischen Entwurf des reflektorischen Operators werden die beschriebenen Koordinationsmuster nun konkret genutzt. Im vorliegenden Beispiel wird das Koordinationsmuster zweimal angewendet, damit das zu entwerfende Fahrzeug als Wissensnutzer mit zwei Streckenabschnitten - dem aktuellen befahrenen und dem nachfolgenden - als lokalen Wissensträgern gleichzeitig kommunizieren kann, so dass schon bei der Einfahrt in den Streckenabschnitt die Vorgabekurve vorliegen kann. Das Verhalten des Fahrzeugs setzt sich daher aus zwei verfeinerten Shuttle-Rollen zusammen und wird um zusätzliche interne Kommunikati-

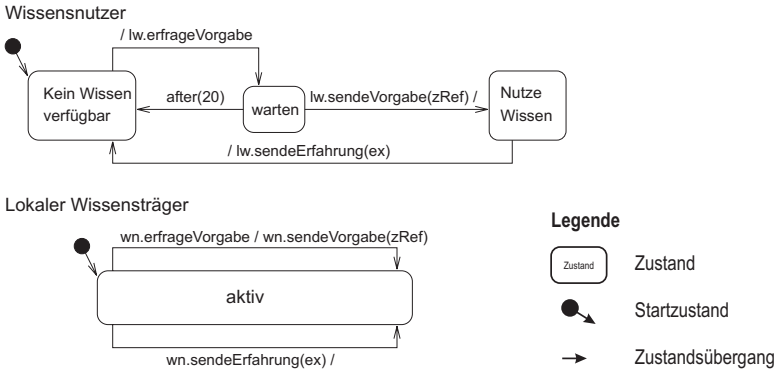


Abbildung 3.13: Verhalten der Rollen Registry und Shuttle

on erweitert (Abbildung 3.14). Zudem enthält das dargestellte Statechart hybrides Verhalten zur Beschreibung der Reglerumschaltung für den Fall, dass keine Vorgabekurve verfügbar ist und auf den robusten Regler gewechselt werden muss [GBSO04].

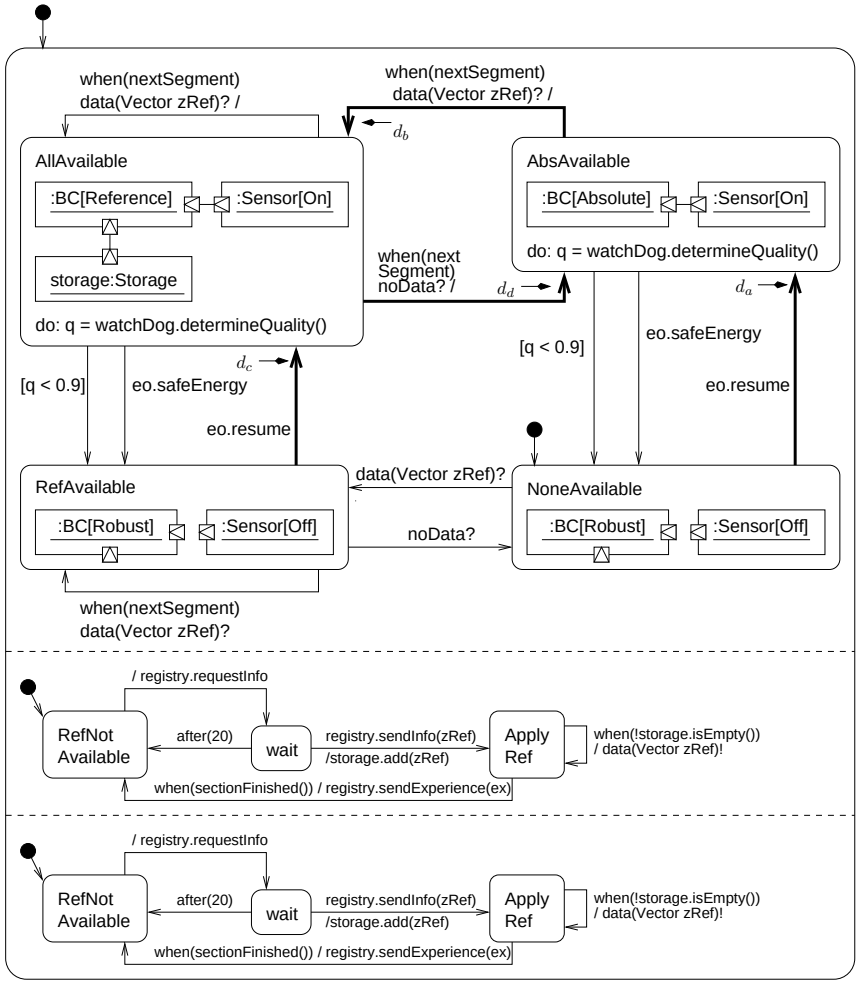


Abbildung 3.14: Struktur des Agentenbasierten Shuttle-Systems

Kapitel 4

Abgrenzung zu verwandten Themen

Wie schon einleitend dargelegt wurde, ist die Selbstoptimierung ein domänenübergreifend relevantes Prinzip. Die allgemeine Frage, wie Systeme selbstständiger, intelligenter und anpassungsfähiger und dadurch leistungsfähiger, effizienter, vielseitiger und robuster gemacht werden können, wird mit unterschiedlichen Schwerpunkten in vielen verschiedenen Fachgebieten beforscht.

Ziel dieses Kapitels ist es, einen Überblick über Themen zu liefern, die den vorgestellten Konzepten und Techniken, insbesondere natürlich dem hier definierten Begriff der Selbstoptimierung, inhaltlich nahestehen. Dabei sollen jeweils entsprechende Forschungsprojekte vorgestellt und von den Arbeiten im Sonderforschungsbereich 614 abgegrenzt werden. Aufgrund ihrer grundlegenden Bedeutung für mechatronische Systeme wird zunächst die Domäne Regelungstechnik betrachtet; daran schließt sich die Domäne Informatik an, in der Multiagentensysteme, künstliche Intelligenz und eine Auswahl interdisziplinärer Arbeiten eingeordnet sind.

Der Begriff „Selbstoptimierung“ selbst findet sowohl im wissenschaftlichen als auch im nichtwissenschaftlichen Bereich eine zunehmende Verbreitung und wird verständlicherweise häufig abweichend von der hier vorgeschlagenen Definition verwendet. Gerade im Marketingbereich werden oft Produkte als selbstoptimierend angepriesen, die auf klassischen, rein adaptiven Reglern basieren. Auch in der Forschung wird Selbstoptimierung zum Teil fundamental anders definiert, etwa als Eigenschaft eines Systems, aus einer geschickt gewählten Ausgangskonfiguration „von selbst“ in einen optimalen Zustand zu streben [Sko00].

Die nachfolgende Zusammenstellung orientiert sich daher primär an Konzepten, nicht an Begrifflichkeiten. Die Darstellung ist notwendigerweise stark selektiv und erhebt keinen Anspruch auf Vollständigkeit, da das Spektrum aktueller Arbeiten, zu denen Bezüge hergestellt werden könn-

ten, dafür zu breit ist. Da die Selbstoptimierung ein noch junges und dynamisches Forschungsthema ist, kann an dieser Stelle auch nur eine Momentaufnahme des Forschungsstandes geboten werden.

4.1 Regelungstechnik

4.1.1 Adaptive Regelungen

Es existiert in der Literatur keine einheitliche Definition zum Begriff der adaptiven Regelung. Weitgehende Einigkeit besteht hinsichtlich der Aussage: Systeme mit adaptiven Regelungen passen ihr Verhalten sich ändernden Eigenschaften und Messwerten eines zu regelnden Prozesses an. Adaptive Regelungen passen sich also selbständig an eine veränderte Umgebung an. Dabei werden feste, schon bei der Implementierung bekannte Adaptionsvorschriften eingesetzt. In der Literatur [ILM92] werden folgende Hauptklassen von adaptiven Regelungen unterschieden:

Feedforward Adaptive Controller

- Beim **Feedforward Adaptive Controller** (z.B. Gain Scheduling) sind die Auswirkungen von messbaren Einflüssen auf das zu regelnde System bekannt. Die gemessenen Einflüsse werden direkt zur Adaption der Reglerparameter verwendet.

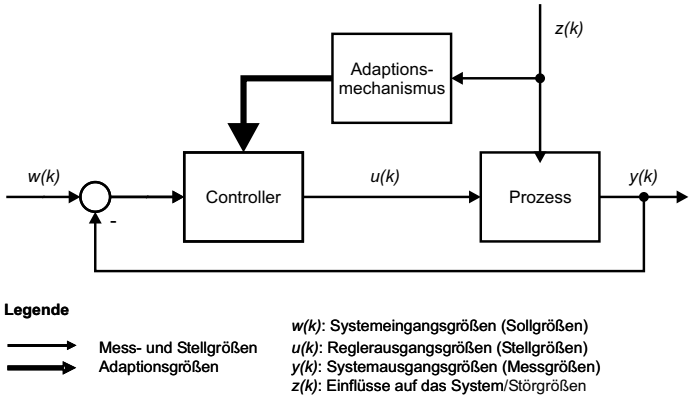


Abbildung 4.1: Feedforward Adaptive Controller [ILM92]

- Der **Feedback Adaptive Controller** beobachtet Systemänderungen durch Messungen von Sytemein- und Ausgängen. Der Adaptionsmechanismus passt die Reglerparameter an die beobachteten Änderungen an.

Feedback Adaptive Controller

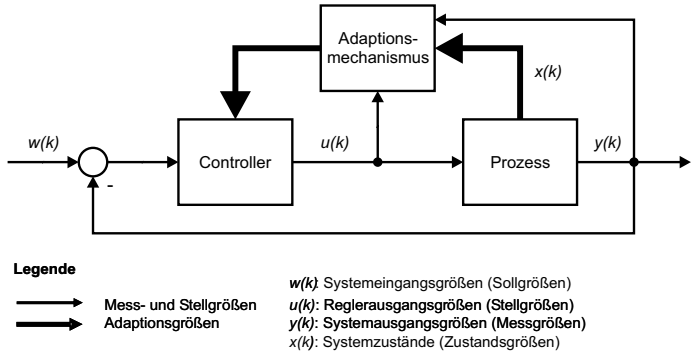


Abbildung 4.2: Feedback Adaptive Controller [ILM92]

Ein spezieller Feedback Adaptive Controller ist der **Model Identification Adaptive Controller** (MIAC), der auch „self-optimizing controller“ genannt wird. Die Vorgehensweise des Adaptionsprozesses in diesen Controllern kommt unserem Verständnis von Selbstoptimierung schon sehr nah. Der wesentliche Unterschied besteht jedoch darin, dass nach unserem Verständnis Selbstoptimierung zusätzlich den Schritt der Anpassung der Ziele enthält.

Model Identification Adaptive Controller

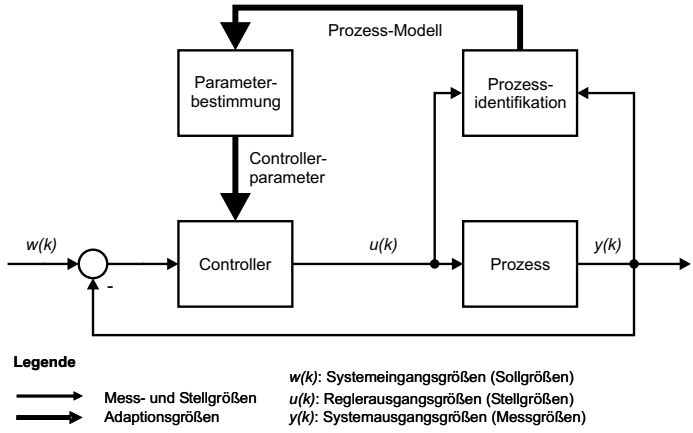


Abbildung 4.3: Model Identification Adaptive Controller (MIAC) oder „self-optimizing controller“ [ILM92]

selbsteinstellende Regelung

- **Selbsteinstellende Regelungen** können als Sonderform der adaptiven Reglung verstanden werden. So wird bei den selbsteinstellenden Reglern zunächst automatisiert die Regelungsstrecke charakterisiert bzw. identifiziert und anschließend eine entsprechende Reglerdimensionierung durchgeführt. Ähnlich wie bei den adaptiven Regelungsverfahren sind die Dimensionierungsvorschriften a priori bekannt, können aber abhängig von der Charakterisierung variieren. Einfache selbsteinstellende Regelungen beruhen auf den oben genannten automatisierten Abläufen. Weitergehende Ansätze nutzen auch wissensbasierte Systeme, die sowohl Simulations- als auch Erfahrungswissen beinhalten. Selbsteinstellende Regelungen haben besonders für die Inbetriebnahme eines Systems eine große Bedeutung und sind bereits in vielen Serienprodukten zu finden.

Abgrenzung zum Sonderforschungsbereich 614 und Bedeutung innerhalb der Regelungstechnik

Adaptive Regelungsverfahren zeichnen sich besonders dadurch aus, dass sie ein klar definiertes vorherbestimmtes Verhalten annehmen und sich dadurch an eine Umwelt- und Anforderungssituation anpassen. Adaptive Regelungskonzepte haben besonders dann eine wesentliche Bedeutung, wenn es um optimale Regelung bezüglich fester Vorgaben und Ziele geht.
Adaptive Verfahren stellen eine wesentliche Basis für selbstoptimierende Systeme dar. Die Erweiterung zur Selbstoptimierung kann als Anpassung von Adaptionsvorschriften verstanden werden.

4.1.2 Fortgeschrittene Regelungstechnik

Einige wesentliche Forschungsprojekte

Ausgehend von den klassischen Methoden der Regelungstechnik wurden und werden eine Vielzahl von fortgeschrittenen regelungstechnischen Verfahren entwickelt. Hier werden einige Forschungsprogramme genannt, die in einigen Teilaspekten wichtige Impulse für die Arbeiten im Sonderforschungsbereich 614 liefern können.

Im DFG-Schwerpunktprogramm KONDISK „Analyse und Synthese kontinuierlich-diskreter Systeme“ wurden wichtige Grundlagen für das

Verständnis gemischter Systeme mit zeitkontinuierlichen und ereignisdiskreten Komponenten erarbeitet z.B. [AL98], [Bus02], [CL99]. Die Entwicklung der Methoden in diesem Bereich hält an und ist noch keinesfalls abgeschlossen.

Die Verfahren zum Entwurf robuster Regelungssysteme unter anderem mittels H-Infinity- Methodik, z.B. [PUS00], [Dul00] haben mittlerweile einen guten Stand erreicht und werden mehr und mehr in Anwendungen tatsächlich eingesetzt. Mit Hilfe von Verfahren der Modellreduktion gelingt es, passable Reglerordnungen zu erreichen. Der Einsatz robuster Regelungsverfahren ist auch im Kontext der Selbstoptimierung als ergänzendes Verfahren für einzelne Komponenten angezeigt, wo ein Optimierungsvorgang oder eine Adaption zu aufwändig erscheint oder zu geringen Nutzen verspricht.

Der Bereich der nichtlinearen Regelungen zeichnet sich weiterhin durch große Methodenvielfalt aus. In den letzten Jahren gelangen aber im Bereich der flachheitsbasierten Regelungen Fortschritte sowohl in der Methodik als auch diese in Anwendungen erfolgreich umzusetzen ([HZ04b], [HZ04a]).

Die Konstruktion von kausalen Rückführreglern für optimale Steuerungen nichtlinearer Systeme (optimales Kontrollproblem) verzeichnet deutliche Fortschritte. Mittlerweile existieren numerische Verfahren, die direkt auch den unendlichen Zeithorizont behandeln, z.B. [KB94], [GW99], [GJ04].

In dem vom US Defense Advanced Research Projets Agency und US Air Force Research Laboratory durchgeführten Forschungsprogramm „Software-Enabled Control“ wurden erhebliche Anstrengungen unternommen, verschiedenste Methoden der Steuerungs- und Regelungstechnik, der Systemmodellierung, -identifikation und -überwachung und der Optimierung in einen Gesamtzusammenhang einzubetten, wobei insbesondere Fragen des Software-Entwurfsprozesses eine große Rolle spielen [SB03]. Die Anwendungen dieses Forschungsprogramms liegen meist in der Avionik. Dennoch sind die Ergebnisse für diesen Sonderforschungsbereich von großem Interesse.

Abgrenzung zum Sonderforschungsbereich 614

Die hier vorgestellten Forschungsprojekte beschäftigen sich mit der Entwicklung neuer regelungstechnischer Verfahren, die teilweise auch im Sonderforschungsbereich 614 Anwendung finden. Im Sonderforschungs-

bereich liegt der Schwerpunkt allerdings mehr auf der Anpassung dieser Verfahren und ihrer strukturierten Anwendung auf reale mechatronische Systeme.

4.2 Informatik

4.2.1 Agententheorie

Die Agententheorie bietet ein mächtiges Abstraktionskonzept für den Entwurf komplexer Softwaresysteme. Durch die Verwendung von Agenten, aktiven Komponenten mit einer gewissen Verhaltensautonomie, soll eine noch stärkere Kapselung erreicht und die Konstruktion intelligenter Systeme mit dynamischen Interaktionsmustern vereinfacht werden.

schwacher Agentenbegriff

starker Agentenbegriff

Es existiert bislang keine einheitliche Theorie der Agentenorientierung. Die verbreiteteste Definition des Agentenbegriffs stammt von Jennings und Wooldridge, die zwischen dem **schwachen Agentenbegriff** *(weak notion of agency)*, der Autonomie, Proaktivität, Reaktivität und soziale Interaktion voraussetzt, und dem **starken Agentenbegriff** *(strong notion of agency)*, der zusätzlich die Verwendung von Techniken der Künstlichen Intelligenz erfordert, unterscheiden (siehe [WJ95, Woo00]).

Wichtige Forschungsgegenstände der Agentenforschung sind zum einen modale Logik, Agentenkommunikationssprachen und Wissensrepräsentation, zum anderen die Erweiterung objektorientierter Entwurfs- und Modellierungstechniken und der Entwurf geeigneter Softwarearchitekturen.

Einige wesentliche Forschungsprojekte

Die Agentenorientierung ist ein grundlegendes Konzept mit Anspruch auf eine der Objektorientierung vergleichbaren allgemeinen Anwendbarkeit, wenn auch mit nicht annähernd demselben Reifegrad. Dementsprechend wird die Agentenmetapher in zahlreichen, sehr diversen Projekten als Strukturierungsmittel angewendet. Gleichzeitig existieren in diesem noch relativ jungen Forschungsgebiet zahlreiche Projekte, die grundlegende Konzepte, Architekturen oder Modellierungsmethoden der Agentenorientierung erforschen und weiterentwickeln. Beispiele für aktuelle Enwurfsmethodologien sind z.B. Gaia [WJK00] oder Tropos [GKMP04], wobei je-

de ihre spezifischen Einschränkungen besitzt. Mit Agent UML [OPB00] strebt eine Gruppe von Forschern einen Standard zur Modellierung von Agentensystemen an. Die FIPA[1] versucht, im Bereich der Agentenkommunikationssprachen einen Standard zu etablieren.

Aufgrund der Vielfältigkeit der Anwendungsgebiete und in Ermangelung einer anerkannten standardisierten grundlegenden Theorie und Methode der Agentenorientierung vermischen sich die beiden Aspekte in Projekten häufig, da erst geeignete Konzepte und Modellierungsmethoden für den jeweiligen Verwendungszweck entwickelt werden. So werden etwa im DFG-SPP 1083 der Wirtschaftsinformatik „Intelligente Softwareagenten und betriebswirtschaftliche Anwendungsszenarien“ sowohl die theoretischen und methodischen Grundlagen der Agententechnologie als auch deren betriebswirtschaftlichen Anwendungen erforscht. In vielen Forschungsprojekten wird die Agentenmetapher inzwischen aber auch lediglich als etabliertes Lösungskonzept für komplexe Koordinationsprobleme genutzt, etwa im DFG-SPP 1103 „Vernetzt-kooperative Planungsprozesse im Konstruktiven Ingenieurbau“.

Abgrenzung zum Sonderforschungsbereich 614

In der Literatur zu Multiagentensystemen lassen sich zahllose Beispiele für selbstoptimierendes Verhalten finden, z.B. besonders deutlich in den Bereichen Schwarmintelligenz oder Artificial Life. Auch beinhaltet die, in der durch die Forschung im Bereich künstliche Intelligenz geprägten Agententheorie äußerst einflussreiche, BDI-Logik (Belief-Desire-Intention) [RG91] explizit mit dem Desire-Konstrukt ein Beschreibungsmittel zur Reflektion über Ziele, die dann in operationale Ziele in Form von Intentions umgesetzt werden können. Zwar spielt hier der Anpassungsschritt häufig eine untergeordnete Rolle, formal erfüllen entsprechende Systeme aber in der Regel die Anforderungen aus der Definition der Selbstoptimierung.

Wie in der Definition begründet wurde, kann jedes selbstoptimierende System als Agent gesehen werden. Der Umkehrschluss gilt hingegen nicht: es existieren zahlreiche Multiagentensysteme, die nicht selbstoptimierend im Sinne der obigen Definition sind, da keine Reflektion über Ziele stattfindet.

[1] http://www.fipa.org

Die inhaltliche Abgrenzung zu anderen Forschungsprojekten liegt primär in der Einbettung der Agenten in mechatronische Systeme. Dies erfordert die Entwicklung von angepassten, völlig neuartigen Entwurfstechniken, die die abstrakte aber präzise Modellierung selbstoptimierenden Verhaltens unterstützen und die Möglichkeit zur Verifikation sicherheitskritischen Verhaltens erlauben.

Auf der Ebene der Architekturen ergeben sich – aufgrund des Agentencharakters selbstoptimierender Systeme nicht überraschende – Parallelen zwischen klassischen Agentenarchitekturen und dem Aufbau selbstoptimierender Systeme. Agentensysteme werden in der Regel als Zusammenspiel zwischen einem Agenten und seiner Umwelt beschrieben, in dem der Agent letztere über seine Sensoren wahrnimmt, aufgrund der gewonnenen Information Entscheidungen trifft und anschließend über seine Effektoren wieder auf seine Umgebung zurückwirkt [Woo02]. Dieser Zyklus findet sich in vergleichbarer Form auch in der Definition der Selbstoptimierung mit ihrer Abfolge aus Erfassung des Ist-Zustandes (Sensorik), Zielanpassung (Reflektion) und Verhaltensanpassung (Effektorik).

4.2.2 Künstliche Intelligenz

Künstliche Intelligenz

Künstliche Intelligenz (KI) zielt auf die Erzeugung intelligenten Verhaltens in technischen Systemen. Die Mitte der fünfziger Jahre entstandene Disziplin umfasst dabei sehr diverse Forschungsgebiete. Grob lassen sich vier Herangehensweisen unterscheiden: menschlich denken (think humanly), menschlich handeln (act humanly), rational denken (think rationally) und rational handeln (act rationally) [RN03]. Während der erste Ansatz als gescheitert anzusehen ist und die Nachbildung menschlichen Verhaltens heute vor allem im Bereich von Benutzerschnittstellen relevant ist, wurden im Bereich des **rationalen Denkens** in den vergangenen Jahrzehnten erhebliche Fortschritte gemacht. Aufbauend auf der Tradition der klassischen Logik wurde die Prädikatenlogik mit entsprechenden Syllogismen, Inferenzstrategien, Heuristiken, Notationen und Wissensrepräsentationen entwickelt. Sie führt auf die sogenannten **symbolischen Systeme** *(symbolic system)*, in denen Wissen explizit modelliert und repräsentiert wird. Diese Ansätze sind sehr erfolgreich beim Lösen komplexer Probleme in einem abstrakten Kontext (z.B. Schach), stoßen aber theoretisch hinsichtlich ihrer Berechenbarkeit und Verifizierbarkeit und praktisch beim Umgang mit komplizierten Umweltsituationen an ihre Grenzen. Im Gegensatz dazu beurteilt der Ansatz des **rationalen Handelns** die Intelligenz eines Programms im Sinne einer behavioristischen Perspektive nur nach

symbolisches System

seinem beobachtbaren Interaktionsverhalten und nicht nach seinen Entscheidungsprozessen. Welches Verhalten rational ist, wird dabei in Unkenntnis der optimalen Lösung empirisch anhand von Erfolgsquoten definiert. Diese Sichtweise führt auf die **subsymbolischen Systeme** *(subsymbolic system)*, in denen Wissen über die Umwelt nicht explizit durch Symbole kodiert ist. In diesem Kontext wurden neue empirische Verfahren vorgeschlagen, etwa „naturanaloge" Heuristiken wie neuronale Netze oder genetische Algorithmen. Diese Ansätze stehen nicht im strikten Widerspruch zu symbolischen Systemen; so wird etwa die Kombination subsymbolischer Lernverfahren zur Erzeugung einer Wissensbasis mit einem symbolisch schließenden System untersucht [BKKN03]. Daneben entwickelte sich ein Trend zur größeren Spezialisierung, generische Konzepte mit umfassendem Anspruch traten in den Hintergrund und domänenspezifisches Wissen gewann an Bedeutung. In diesem Zusammenhang gewannen auch verteilte Problemlösungsstrategien an Bedeutung, wobei häufig dezentrale Systeme aus einfachen Komponenten ohne hochentwickelte Intelligenz aufgebaut werden [Bro91].

subsymbolisches System

Eine grundlegende Debatte dreht sich um die Frage, ob prinzipiell auf der Grundlage heutiger Rechnerarchitekturen Intelligenz auf menschlichem Niveau emulierbar ist (Strong AI-Hypothese) oder nicht (Weak AI-Hypothese). Hier ist ein Paradigmenwechsel erkennbar, da viele aktuelle Projekte sich mit teilintelligenten Multiagentensystemen beschäftigen, die einfache Probleme in einer komplexen Umwelt lösen können (z.B. Fussball).

Das Ziel wissensbasierter Systeme ist es, die richtigen Informationen zur richtigen Zeit am richtigen Ort in der richtigen Qualität zu den richtigen Kosten verfügbar zu machen. Ein herausragender Aspekt wissensbasierter Systeme ist die Trennung von Fachbereichswissen und Problemlösungswissen. Dabei kann das Wissen nach [Suh93] in drei Kategorien aufgeteilt werden: Fachbereichswissen (Domänenwissen, statisches Wissen) repräsentiert das Wissen über die im Anwendungsbereich verfügbaren Wissensobjekte. Problemspezifisches Wissen (Problemlösungswissen, dynamisches Wissen) bezieht sich auf die konkret vorliegende Aufgabenstellung. Allgemeines Problemlösungswissen verknüpft das Fachwissen mit dem problemspezifischen Wissen, z.B. durch Strategien, Anweisungen und Reihenfolgewissen.

Drei Hauptfunktionalitäten wissensbasierter Systeme lassen sich unterscheiden: Wissensrepräsentation, Wissensverarbeitung und Wissensableitung. Wissen kann je nach Einsatzzweck und Aufgabenstellung unterschiedlich repräsentiert werden, u.a. regelbasiert, fallbasiert, fra-

mebasiert, constraintbasiert, mit Hilfe von semantischen Netzen, Truth-Maintenance-System (TMS)-basiert, Blackboard-basiert und auf neuronalen Netzen basiert. Eine Sortierung und Klassifikation erfolgt durch Begriffshierarchien wie Ontologien, Thesauri und Dictionaries. Die Verarbeitung vorhandenen Wissens basiert im Wesentlichen auf verschiedenen Suchmethoden zur kontextsensitiven Bereitstellung des benötigten Wissens, z.B. Methoden zur blinden Suche, heuristische Suche, und Suchverfahren aus der Spieltheorie. Die Ableitung von neuem Wissen aus bestehendem Wissen kann nach den Prinzipien der Deduktion, der Abduktion oder der Induktion erfolgen. Bei der Deduktion wird ein Beweis einer Aussage aus Axiomen durch formale Schlussfolgerungsverfahren geführt. Die Abduktion versucht aus Kausalwissen und Beobachtungen auf neues Wissen zu schließen. Bei der Induktion werden Gesetzmäßigkeit aus vielen Einzelbeobachtungen abgeleitet und daraus neues Wissen generiert.

Bei der regelbasierten Kodierung von Expertenwissen spielt die Theorie der unscharfen Mengen [Zad65] eine wichtige Rolle. Fuzzy Logic erlaubt es, unpräzise natürlichsprachliche Aussagen formal zu erfassen und aus ihnen Schlüsse zu ziehen. Im Kontext mechatronischer Systeme wird sie, besonders im asiatischen Raum, für intelligente Regler eingesetzt.

Vorausschauendes Verhalten ist eng mit der Erzeugung von Plänen verbunden. Klassische Planungsmethoden sind für den Einsatz im Kontext mechatronischer Systeme in der Regel nicht geeignet, da sie Echtzeit- und Sicherheitsanforderungen nicht berücksichtigen. Hier sind zum einen hierarchische Ansätze, die einen Aufgabengraphen in wiederverwendbare primitive Aufgaben zerlegen, und zum anderen temporallogische Planung, bei der Planungsziele mit temporallogischen Formeln als Gültigkeitskriterien annotiert werden können, von Interesse.

Einige wesentliche Forschungsprojekte

Das **Projekt Fynesse**[2] versucht, neuronale Netze mit Fuzzy-Systemen zu verbinden, die die Dynamik unbekannter Systeme ohne Vorwissen selbstständig erlernen können. Zur Erreichung eines vorgegebenen Steuerungsziels wird dazu das dynamische Verhalten der Regelungsstrecke vom System analysiert. Der Ansatz ist jedoch nicht direkt auf sicherheitskritische Systeme übertragbar, da keine Entkoppelung zwischen den verwendeten Heuristiken und dem Regelungskreis vorliegt.

[2] http://www.info.uni-karlsruhe.de/špott/Dfg/fynesse.html

Der **Sonderforschungsbereich 531** „Design und Management komplexer technischer Prozesse und Systeme mit Methoden der computational Intelligence“ befasst sich mit Methoden der Computational Intelligence (CI), wobei unter anderem Fuzzy Logic, neuronale Netze und evolutionäre Algorithmen genutzt werden. Er betreibt vornehmlich mathematisch-theoretische Grundlagenforschung zur Verwendung und besonders der Kombination dieser Verfahren.

Wissensbasierte Systeme werden in der Konstruktionstechnik zur Unterstützung der Entwicklung klassischer maschinenbaulicher Systeme eingesetzt. Dabei werden speziell die Phasen der Anforderungsmodellierung, der Funktionsmodellierung, der Prinzipmodellierung und der Gestaltmodellierung sowie die Diagnose und Simulation mechatronischer Systeme durch entsprechende Systeme unterstützt. ModCoDe und Wisent [WB02] bieten die Konzeption auf Basis von Wirkelementen und Informationsmanagement für komplexe Entwicklungsprozesse, Schemebuilder Mechatronics [NPB03] und Goldfire Innovator (ehemals TechOptimizer) [IMC] eine wissensbasierte Modellierung der Prinziplösung. Eine Entwicklungssystematik für wissensbasierte Diagnosesysteme bietet D3-Web [Pup98]; MOLTKE [KD93] ist ein fallbasiertes System für die Diagnose technischer Systeme. Beispiele für eine wissensbasierte Simulation sind The Multimodal Assembly Expert (MAX) und Concept Dynamics (CODY) der Arbeitsgruppe Wissensbasierte Systeme an der Universität Bielefeld [KJLW03] und FluidSIM der Fachgruppe Wissensbasierte Systeme an der Universität Paderborn, ein Simulator für fluidische Systeme [SCH98]. Bislang sind jedoch keine Konzepte, Vorgehenssystematiken oder Werkzeuge bekannt, die die wissensbasierte Entwicklung von Systemen unterstützen, die das Paradigma der Selbstoptimierung beinhalten – also insbesondere die wissensbasierte Modellierung intelligenten autonomen Verhaltens.

Abgrenzung zum Sonderforschungsbereich 614

Selbstoptimierung kann selbst als Form künstlicher Intelligenz angesehen werden, da sie ein möglichst rationales Verhalten des Systems erzielen soll. Allerdings verlangt Selbstoptimierung zwar die Anpassung von Zielen, ist aber insgesamt eher dem Bereich des ergebnisorientierten „rationalen Handelns“ zuzuordnen und macht somit keine Aussagen über interne Prozesse. Somit ist sie auch neutral bezüglich der Verwendung von Methoden der KI im engeren Sinne. Der Einsatz von Methoden einer „starken KI“ wie symbolischer Logik ist möglich, z.B. auf der Ebene des

kognitiven Operators der vorgeschlagenen OCM-Architektur, jedoch nicht zwingend notwendig. Eine zusammengesetzte Selbstoptimierung kann Methoden der verteilten KI verwenden, gerade bezüglich der Kooperation teilintelligenter Subsysteme, unterliegt aber ebenfalls keinen Mindestanforderungen bezüglich ihrer Intelligenz oder Komplexität.

Andererseits geht die Selbstoptimierung mechatronischer Systeme durch die durchgehende konzeptionelle Einbeziehung der mechanischen und regelungstechnischen Ebene deutlich über den Betrachtungsgegenstand der KI hinaus. Ziel des Sonderforschungsbereich 614 ist die Umsetzung selbstoptimierenden, also in der Reaktion auf eine komplexe Umwelt intelligenten, Verhaltens im Rahmen realer, sicherer, praxistauglicher mechatronischer Systeme, wobei das Treffen intelligenter Entscheidungen nur einen Aspekt des spezifischen Gesamtproblems darstellt.

4.2.3 Sonstige Forschungsprojekte

Autonomic Computing (AC) Architecture

Die Autonomic Computing Architecture Initiative [KC03] fokussiert sich auf Rechnersysteme. Dabei werden vier Aspekte des Selbstmanagements (self-management) von Rechnersystemen unterschieden. Die

Selbstkonfiguration

Selbstkonfiguration *(self-configuration)* beschreibt die automatische Konfiguration von Komponenten des Systems und folgt so genannten high-level policies. Bei der **Selbstoptimierung** *(self-optimization)* suchen Komponenten und Systeme kontinuierlich Möglichkeiten, ihre eigene Leistung und Effizienz zu steigern. Die **Selbstheilung** *(self-healing)* umfasst Aspekte, bei denen Systeme automatisch Software- und Hardware-Probleme entdecken, diagnostizieren und reparieren. Schließlich wird durch den **Selbstschutz** *(self-protection)* erreicht, dass sich Systeme automatisch gegen Angriffe von außen verteidigen. Diese Systeme nutzen ein Frühwarnsystem, um Systemfehlfunktionen frühzeitig zu erkennen und einen systemweiten Ausfall zu vermeiden. Die dargestellten Aspekte erscheinen im Rahmen des Sonderforschungsbereich 614 insbesondere für die Arbeiten in den Bereichen der Rekonfiguration und der Sicherheit wesentliche Impulse geben zu können.

Selbstoptimierung

Selbstheilung

Selbstschutz

Organic Computing (OC)

Aktuelle Projekte zum Organic Computing verfolgen ähnliche Ziele wie die Autonomic Computing Architecture, setzen dabei jedoch aus

Disziplinen wie Biologie, Verhaltensforschung, Psychologie, Informatik und Ingenieurwissenschaften kopierte naturanaloge Verfahren ein. Zu diesem Themenbereich existieren zwei unabhängige Forschungsprojekte, die von der Europäsischen Union und der DFG[3] bzw. der GI und dem VDE[4] gefördert werden, sowie ein Schwerpunktprogramm der DFG, das Anfang 2005 aufgenommen werden soll. Die Arbeiten sind wiederum für den Sonderforschungsbereich von Interesse, wobei sie noch nicht die speziellen Bedürfnisse und schärferen Zeitrestriktionen sicherheitskritischer mechatronischer Systeme berücksichtigen.

Self-Optimizing Software

Im Rahmen des DARPA Projektes zu Self-Adaptive Software wird selbstadaptive Software wie folgt definiert ([DAR97], eigene Übersetzung):

„Selbst-adaptive Software evaluiert ihr eigenes Verhalten und ändert ihr Verhalten, wenn absehbar ist, dass die Software ihren ursprünglichen Einsatzzweck nicht erreicht oder wenn eine bessere Funktionalität oder Leistung erreichbar zu sein scheint."

Hinsichtlich der Arbeiten im Sonderforschungsbereich, ist besonders die Idee hervorzuheben, Entscheidungen bezüglich des Verhaltens von Systemen vom Entwicklungszeitpunkt auf den Laufzeitpunkt zu verlagern. Dieses Paradigma wird durch mehrere Hypothesen untermauert. Es ist einfacher und oftmals effizienter, gewisse Entscheidungen so lange zu verschieben, bis ihre Notwendigkeit aus den dann vorherrschenden Bedingungen offensichtlich wird. Es ist oftmals einfacher ein Programm zu erstellen, das seine Leistungen überwacht und Fehler selbst behebt, als ein langes Programm zu schreiben, das versucht Fehler zu vermeiden. Dadurch, dass Entscheidungen zur Laufzeit gefällt werden, wenn mehr Informationen über die aktuelle Umwelt zur Verfügung stehen als zum Entwicklungszeitpunkt des Programms, wird es möglich, eine besser geeignete Vorgehensweise zum Erreichen der beabsichtigten Ziele zu wählen.

Muster zur Selbstadaption (Self-Adaptive Patterns)

In [LRS03] werden Anforderungen zum Entdecken, Charakterisieren und Katalogisieren von Muster gestellt, die den Kern einer Methode für Software-Adaptivität bilden. Das Muster **Indirekter Aufruf** *(indirect invocation)* hat die Entkopplung vom Aufruf eines Dienstes und der

indirekter Aufruf

[3] http://www.organic-computing.org
[4] http://www.gi-ev.de/download/VDE-ITG-GI-Positionspapier Organic Computing.pdf

entscheidungs-basierter Aufruf

Nützlichkeitsfunktion

Selbstüberwachung

Selbstdiagnose

Wiederherstellung

Methode, die dieser Dienst zur Verfügung stellt zum Ziel. Bei dem Muster **Entscheidungsbasierter Aufruf** *(decision theoretic invocation)* ruft ein Diensteanforderer einen abstrakten parametrisierbaren Dienst auf. Ressourcen werden gegen Kosten zur Verfügung gestellt. Der Anforderer hat eine **Nützlichkeitsfunktion** *(utility function)*, mit der er sein Kosten-Nutzen-Verhältnis nach Aufruf des Dienstes bewertet. Das **Blackboard Muster** erlaubt einer Vielzahl verschiedener Wissensressourcen auf unterschiedlichem Abstraktionsniveau kooperativ Probleme zu lösen. Des Weiteren werden noch Muster zur **Selbstüberwachung** *(self-monitoring)*, **Selbstdiagnose** *(self-diagnosis)* und **Wiederherstellung** *(self-recovery)* gefordert.

Abgrenzung zum Sonderforschungsbereich 614

Die beschriebenen Ansätze beziehen sich meist auf Systeme in der Software- bzw. Computer-Hardware Domäne. Es sind nur wenige Untersuchungen bekannt, in denen maschinenbauliche Systeme die Basis der Forschung sind [BWSG98]. Im Unterschied zu reinen softwaretechnischen Systemen, besitzen jedoch maschinenbauliche Systeme besondere Anforderungen gerade in den Bereichen der Echtzeitfähigkeit und Rekonfiguration, die von den beschriebenen Ansätzen konzeptionell noch gar nicht betrachtet werden.

Kapitel 5

Anwendungsbeispiele

Im Folgenden werden Anwendungsbeispiele für Selbstoptimierung im oben definierten Sinn vorgestellt. Dazu erfolgt zunächst eine Beschreibung der Demonstratoren des Sonderforschungsbereich 614, wobei jeweils aufgezeigt wird, in welcher Weise die Prinzipien der Selbstoptimierung zum Tragen kommen. Die vorgestellten Demonstratoren werden anschließend beispielhaft in den Rahmen der oben vorgeschlagenen Klassifikationen der Selbstoptimierung eingeordnet.

Bei den Demonstratoren handelt es sich um Prüfstande und Simulationsmodelle, die Untersuchungen an verschiedenen Teilfunktionen eines Linearmotor getriebenen Schienenfahrzeugs (Shuttles) ermöglichen. Abbildung 5.1 zeigt die vereinfachte Wirkstruktur eines Shuttles. Das Shuttle besteht aus zwei Fahrmodulen, die jeweils ein Feder-Neigmodul, eine Spurführungsmodul und ein Antriebsmodul enthalten. Diese Module realisieren die Hauptfunktionen „Antreiben und Bremsen“, „Beeinflussen der Aufbaudynamik“ und „Spur führen“. Hervorzuheben ist, dass die Funktion „Antreiben und Bremsen“ nur in Kombination der Antriebsmodule eines Shuttles und eines Statorabschnittes der Strecke umgesetzt werden können. Ein weiteres wichtiges Element ist das Energieversorgungsmodul.

Die nachfolgende Darstellung soll die Prinzipien der Selbstoptimierung illustrieren und hat nicht die Aufgabe, die Demonstratoren zu dokumentieren. Sie verzichtet daher weitgehend auf detaillierte technische Beschreibungen. Diesbezügliche Details können den entsprechenden Publikationen des Sonderforschungsbereichs entnommen werden.

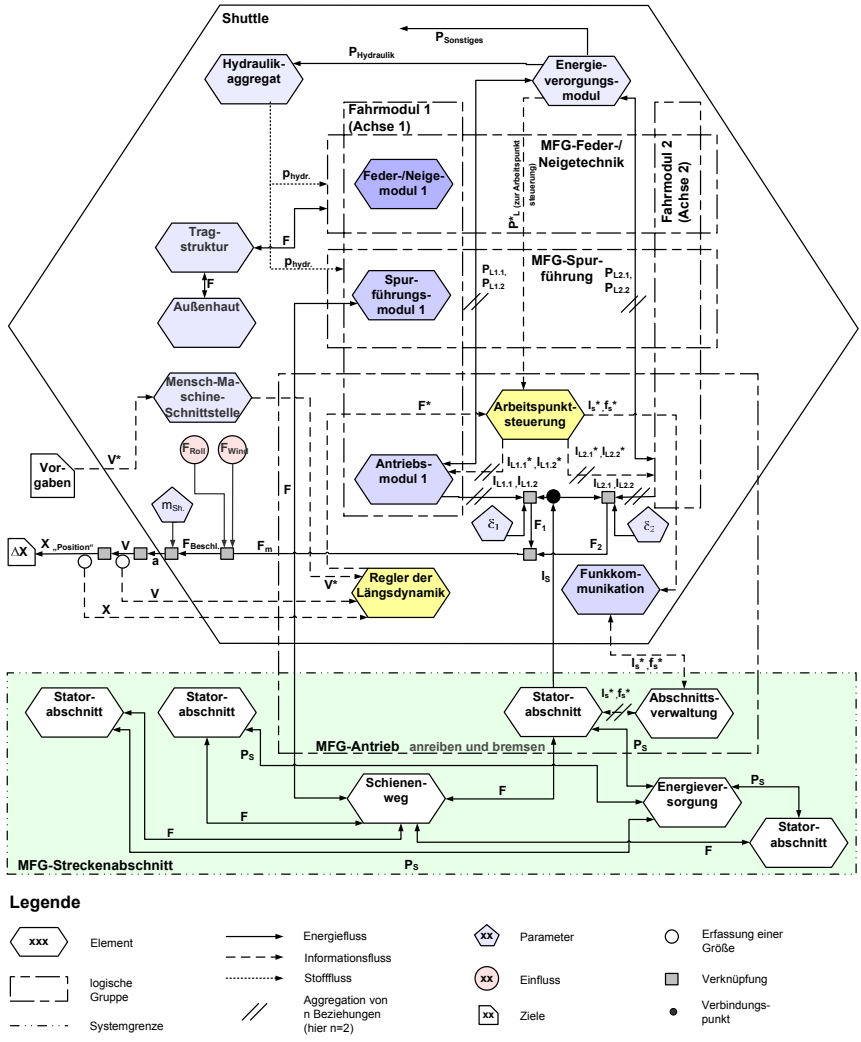


Abbildung 5.1: Positionierung der Demonstratoren im Shuttle

5.1 Antriebs-Bremsmodul

Das Antriebs-Bremsmodul (AB-Modul) stellt eine zentrale Grundfunktion für ein Fahrzeug zur Verfügung. Dabei hat das AB-Modul in dem für den Sonderforschungsbereich 614 gewählten Demonstrator (ein Schienenfahrzeug mit einem doppelt gespeisten Linearantrieb) mehrere Funktionen. Neben den Hauptfunktionen des verschleißarmen Antreibens und Bremsens besitzt das AB-Modul die Nebenfunktionen Energie zu übertragen (von der Strecke zum Fahrzeug) sowie die Regelung der Nickbewegung der Einzelachse eines jeden Fahrmoduls. An dem Linearmotorprüfstand des AB-Moduls werden die im Sonderforschungsbereich 614 entstandenen Strukturierungen für die Selbstoptimierung wie auch die praktische Anwendung von neuen Optimierungsstrategien und Algorithmen erprobt. Dabei kann auch die im Sonderforschungsbereich 614 entwickelte rekonfigurierbare Hardware zum Einsatz kommen.

Der Prüfstand besteht aus einem Wagen, der durch zwei Läuferelemente über einen mit Statorelementen ausgestatteten Schienenweg von ca. 8 m Länge bewegt werden kann. Die Läuferelemente in Verbindung mit den Statorelementen stellen den Antrieb dar. Über leistungselektronische Stellglieder, die Stromrichter, ist es möglich, gezielt Strom in den Antrieb einzuprägen. So ergeben sich die erforderlichen Schub- und Normalkräfte sowie die Energieübertragung im Antrieb. Um diese Vorgänge zu regeln, ist auf dem Fahrzeug eine Echtzeitrecheneinheit installiert. Ergänzend zu dieser Hardware steht ein PC-System zur Verfügung, auf dem neben den Optimierungen auch Wissensbasen realisiert werden. Im Folgenden werden zwei Optimierungsaufgaben vorgestellt, die jeweils unterschiedliche Aspekte der Selbstoptimierung aufzeigen sollen.

5.1.1 Optimierungsaufgaben

Am Prüfstand des AB-Moduls werden die beiden nachfolgend beschriebenen Optimierungsaufgaben behandelt. Sie sind unabhängig voneinander und zeigen unterschiedliche Aspekte der Selbstoptimierung.

Beispiel: Fahrprofile

Die Grundfunktion, das Fahren von einem Ort zu einem anderen, kann auf sehr unterschiedliche Art realisiert werden. Bei Verwendung klassischer Ansätze werden die Beschleunigungs- und Bremsvorgänge

einmal festgelegt und anschließend automatisch abgefahren. Dabei findet keine Anpassung an unterschiedliche Gegebenheiten statt. Sollen allerdings unterschiedliche Maximalbeschleunigungen, Anforderungen an Komfort und Energieeffizienz berücksichtigt werden, dann ist eine Anpassung der Beschleunigungs- und Bremsprofile unumgänglich. Eine Abwägung der unterschiedlichen Anforderungen kann mittels Optimierung erfolgen. Es ergeben sich aus der Aufgabe, Fahren von A nach B, als Ergebnis Vorgaben für Positions-, Geschwindigkeits- und Beschleunigungsverläufe. Hierbei sind die Abhängigkeiten von maximal erzielbaren Schub- und Bremskräften sowie der zulässigen maximalen Geschwindigkeit zu berücksichtigen. Das Ziel „Ankommen am Zielort“ ist inhärent und obligatorisch. Die Optimierungsziele Zeit-, Energiebedarfsreduktion sowie Komforterhöhung sind fakultativ. Insbesondere diese fakultativen, also wahlfreien, Optimierungsziele, können durch unterschiedliche Gewichtung den selbstoptimierenden Aspekten Rechnung tragen. So kann beispielsweise eine Verschlechterung des Komforts zugunsten eines geringeren Energiebedarfs zugelassen werden. Dabei liegt die Entscheidung, ob dieses Verhalten zulässig ist, bei einem übergeordneten Entscheidungsträger. Bei dieser Entscheidung können z.B. altruistische Optimierungsstrategien verfolgt werden. Aber auch die obligatorischen Ziele verdeutlichen die selbstoptimierenden Aspekte. Eine zur Verfügung stehende Energiemenge wird als obligatorisches Ziel aufgefasst. Ist diese Menge Energie für ein auszuführendes Fahrmanöver unzureichend, so wird durch den Optimierer die benötigte Energiemenge so weit wie möglich minimiert. Andere Module können durch Auslösen eines Ereignisse ggf. dazu gebracht werden, auf Energie zu verzichten. Somit werden Ziele anderer Systeme durch diese Optimierung verändert.

Beispiel: Arbeitspunktsteuerung des Linearmotors

Ausgehend von der Tatsache, dass die erzeugten Kräfte proportional von dem Produkt der Läufer- und Statorströme abhängen, ergibt sich die Frage, wie dieses Produkt aufzuteilen ist. Das Aufteilen dieses Produktes wird als Arbeitspunktsteuerung bezeichnet. Ferner wird durch die Arbeitspunktsteuerung die Frequenz von Stator- bzw. Läuferstrom vorgegeben. Dabei gilt, dass die Differenz dieser Frequenzen von der Fahrzeuggeschwindigkeit abhängt. Durch die Vorgabe der Frequenz und Ströme lässt sich die Energieübertragung zwischen Fahrzeug und Strecke maßgeblich beeinflussen. Daher ergibt sich für die Arbeitspunktsteuerung die Aufgabe, die Vorgaben aus dem Energiemanagement (die zu übertragende Leistung) sowie die Vorgaben aus der Geschwindigkeitsregelung (die zu stellende Schubkraft) zu erfüllen. Als Abhängigkeiten sind insbesondere die Verluste im Linearmotor sowie die

Normalkräfte, die für die Nickregelung bereitgestellt werden müssen, zu berücksichtigen. Ferner sind die zum Teil ortsabhängigen Motorparameter, wie z.B. der Luftspalt zwischen Läufer- und Statorelement, für die Optimierung relevant. Das Ergebnis stellt die Aufteilung der Ströme sowie die Vorgabe der Frequenz dar. Als obligatorische Optimierungsziele gelten die zu erzielende Schubkraft sowie die zu übertragende Leistung, wobei die Schubkraft aus Gründen der Sicherheit i. Allg. eine höhere Priorität besitzt. Der Wirkungsgrad des AB-Moduls sowie der Stromrichternutzungsgrad sind als fakultative Optimierungsziele anzusehen. Da die Bereitstellung der Schubkraft z.B. bei Konvoifahrten als extrem sicherheitskritisch anzusehen ist, sind für diese Optimierungsaufgaben egoistische Optimierungsstrategien von großer Bedeutung. Bei dieser Optimierungsaufgabe kommen insbesondere durch das Zusammenspiel mit dem Energiemanagement als auch der ständigen Anpassung an die aktuellen Motorparameter die Prinzipien der Selbstoptimierung zum Tragen.

5.1.2 Ablauf der Selbstoptimierung

Ausgehend von dem grundsätzlichen Ablauf der Selbstoptimierung, der in Kapitel 2.2 beschrieben ist, erfolgt hier die Konkretisierung für die beiden oben genannten Beispiele.

Beispiel: Fahrprofile

Fahrprofile werden üblicherweise vor einer Fahrt bzw. einem Fahrmanöver bestimmt. Dabei wird davon ausgegangen, dass sich die Einflüsse auf das System während des Manövers nicht ändern.

1. *Analyse der Ist-Situation*

   Bei der Analyse der Situation werden zunächst die Komfort- und Zeitvorgaben des Benutzers erfasst. Aufgrund einer Abfrage der zur Verfügung stehenden Energie und Spitzenleistung wird der Einfluss anderer Systeme bestimmt. Darüber hinaus wird der systemeigene Parameter (aktuelle) Fahrzeugmasse identifiziert. Eine Erfassung der Umweltsituation, insbesondere von wahrscheinlichem Fahrtwind sowie von Höhenprofilen, ist zwar denkbar, jedoch für die Realisierung an einem ebenen Prüfstand in einer Halle nicht sinnvoll.

2. *Bestimmung von Zielen*

   Folgende Ziele sind inhärent:

   (a) minimaler Energiebedarf,
   (b) geringer Verschleiß.

   Aus der Analyse der Einflüsse werden folgende Ziele bestimmt:

   (a) maximal zulässige Werte für Ruck und Beschleunigung,
   (b) minimal erforderliche Durchschnittsgeschwindigkeit.

3. *Anpassung des Systemverhaltens*

   Die Anpassung des Systemverhaltens besteht in diesem Beispiel aus den Schritten:

   (a) Bestimmung der Sollwertvorgaben für Geschwindigkeit, Ort
   (b) sowie der Vorgabe dieser Referenzprofile.

Beispiel: Fahrprofile

1. *Analyse der Ist-Situation*

   Da die Arbeitspunktsteuerung vollständig in das System des Shuttles eingebettet ist, erfolgen keine Vorgaben durch den Benutzer. Die Umwelt beeinflusst die Arbeitspunktsteuerung nur indirekt durch die Veränderung der Parameter des Linearmotors (Luftspalts). Einflüsse anderer Systeme wirken sich durch die Vorgabe der zum Shuttle zu übertragenden Leistung aus. Aus dem System selbst ergibt sich eine Beeinflussung durch die Veränderung des Läuferwiderstands aufgrund von Erwärmung infolge des eingeprägten elektrischen Stroms. Störungen der Nickregelung durch Anregungen, die aus der Feder-Neigetechnik des Fahrzeuges herrühren, erfordern eine hinreichend große Stellreserve für die Kraft bildenden Ströme.

2. *Bestimmung von Zielen*

   Als inhärente Ziele gelten:

   (a) der vom Energieversorgungsmodul geforderte Leistungsfluss,
   (b) die geforderte Schubkraft

(c) sowie hoher Wirkungsgrad.

Ziele, die sich aus der Analyse der Ist-Situation ergeben sind:

(a) Bereitstellung der erforderlichen Stellreserve für Schub- und Normalkräfte

(b) sowie eine geringe Läufertemperatur bzw. hohe Energieeffizienz im fahrzeugseitigen Teil des Antriebs.

3. *Anpassung des Systemverhaltens*

   Nach der Generierung der Ziele wird eine optimale Menge von Arbeitspunkten bestimmt. Durch eine Entscheidungsfunktion wird aus dieser Menge die zu verwendende Einstellung ausgewählt. Nachdem der Arbeitspunkt festgelegt ist, wird dieser den leistungselektronischen Stellgliedern auf der Strecke und im Fahrzeug vorgegeben.

5.1.3 Einordnung

Bei den beschriebenen Anwendungen der Selbstoptimierung wird davon ausgegangen, dass das AB-Modul keine für die Optimierungsaufgabe relevanten selbstoptimierenden Subsysteme enthält. Die Selbstoptimierung ist daher als individuelle Selbstoptimierung einzustufen.

Form der Optimierung

Form der Optimierung (zu Kapitel 2.3.1): Da für die Optimierung mathematische Modelle des Antriebs bzw. des Fahrzeuges genutzt werden, sind die hier vorgestellten Optimierungen als modellbasiert einzustufen. Durch den Einsatz einer ähnlichkeitsbasierten Suche in einer Wissensbasis, die insbesondere durch bereits optimierte Ergebnisse sowie Erfahrungswissen gespeist wird, kommen verhaltensbasierte Prinzipien hinzu. Somit ergibt sich insgesamt eine kombinierte Optimierung. Die Vorteile der Kombination von modellbasierter und verhaltensbasierter Optimierung werden besonders aufgrund der geringen zur Verfügung stehenden Reaktionszeiten im AB-Modul deutlich, da die Abfrage einer Wissensbasis i. Allg. schneller abläuft als eine z.B. auf Simulationen basierende Optimierung. Darüber hinaus kann das Erfahrungswissen eine wiederholte Anwendung von Optimierungsergebnissen, die aus unzureichenden Modellen entstanden sind und den Erwartungen nicht entsprochen haben, verhindern.

Zeitpunkt und Ablauf

Zeitpunkt und Ablauf (zu Kapitel 2.3.2): Die hier vorgestellten Optimierungen sollen zu einer Verbesserung des Systemverhaltens während der Laufzeit führen. Dabei wird davon ausgegangen, dass die Optimierung der Fahrprofile vor einem Beschleunigungs- bzw. Bremsmanöver durchgeführt wird. Es ergibt sich somit eine ereignisgesteuerte Optimierung. Im zweiten Beispiel ist aufgrund des ortsabhängigen Motorparameters Luftspalt eine Optimierung erforderlich, die entweder ständig (kontinuierlich) oder ereignisgesteuert abläuft. Im Falle einer signifikanten Abweichung des aktuellen Luftspalts vom zuvor berücksichtigten Luftspalt kann eine ereignisgesteuerte Optimierung ausgeführt werden. Beide genannten Optimierungsaufgaben haben direkt Einfluss auf das Verhalten und die Sicherheit des Prüfstandes. Es ist daher zwingend notwendig, dass alle Einstellungen, die aus den Optimierungsergebnissen folgen, überwacht werden. Diese Überwachung sowie die Regelungen des Linearmotors sind unter harten Echtzeitbedingungen durchzuführen. Die Optimierungen als solches können unter sehr weichen Echtzeitbedingungen erfolgen. Wenn eine Optimierung abgeschlossen ist, kann das Ergebnis an die Echtzeitumgebung übertragen und dort eingestellt werden. Das setzt allerdings voraus, dass die Bedingungen, die der Optimierung zugrunde gelegen haben, immer noch aktuell sind. Sind sie es nicht, so kann das Optimierungsergebnis in einer Wissensbasis abgelegt werden und steht so für eine spätere Anwendung zur Verfügung. Sollte eine ähnliche Situation mit vergleichbaren Bedingungen erneut auftreten, so kann durch die Abfrage einer entsprechend organisierten Wissensbasis sehr schnell eine näherungsweise optimale Lösung zur Verfügung gestellt werden. Die parallel ablaufende modellbasierte Optimierung verbessert überdies das Systemverhalten.

Sicherheitsaspekte

Sicherheitsaspekte (zu Kapitel 2.3.3): Obwohl jede Einzelachse über ein eigenes AB-Modul verfügt, sind die Aktoren des Moduls nicht redundant. Daher ist es von entscheidender Bedeutung, einen Ausfall der Aktoren und damit eines AB-Moduls zu verhindern. Sollte eine Situation eintreten, die zu einem Ausfall führen könnte, so müssen unverzüglich Gegenmaßnahmen ergriffen werden. So könnte beispielsweise eine thermische Überbeanspruchung eines Läuferelementes infolge unzulässig hoher Läuferströme zu irreversiblen Schäden und letztlich zum Ausfall des Aktors führen. In einem solchen Fall ist der Strom zunächst zu begrenzen und falls erforderlich sogar abzuschalten. Dabei muss jedoch beachtet werden, dass der Motor auch zum Bremsen genutzt wird. Somit muss eine Grundfunktionalität gegebenenfalls auch zulasten des Aktors erhalten bleiben. Regler, die eine solche Grundfunktionalität bereitstellen, müssen allerdings kein optimales Verhalten aufweisen. Diese Regler müssen lediglich schnell und zuverlässig zum Einsatz gebracht werden können. Sie

sind daher heterogen redundant zu den Reglern des selbstoptimierenden Systems zu realisieren. Diese Rückfallebene muss im Falle des AB-Moduls ein fail-operation Verhalten ermöglichen. Fail-safe Verhalten ist für das AB-Modul insbesondere bei Konvoifahrt unzureichend. Neben der internen Überwachung im AB-Modul kann eine externe Überwachung das Verhalten der beiden im Fahrzeug vorhandenen AB-Module auf Plausibilität prüfen (kontrollierte Sicherheit).

Typisierung und Struktur

Typisierung und Struktur (zu Kapitel 2.3.4): Die Ermittlung von optimalen Fahrprofilen ist in der mechatronischen Funktionsgruppe (MFG) der Antriebs-Bremsmodule angeordnet. Durch die Vorgabe der Verläufe für Beschleunigung, Geschwindigkeit und Ort wird das Verhalten des gesamten Fahrzeugs verändert. Für die Optimierungsstrategien dieser Aufgabenstellung stehen altruistische Motive im Vordergrund. Im Gegensatz dazu muss die Optimierung der Arbeitspunkte aufgrund ihrer sicherheitskritischen Eigenschaft weitestgehend egoistisch agieren. Da diese Optimierung in engem Verhältnis zum Energiemanagement steht, hat sie so indirekt Auswirkungen auf das Gesamtfahrzeug. Die Anpassungen werden allerdings nur lokal wirksam. Die Anforderungen z.B. hinsichtlich des zeitlichen Rahmens, die an diese Optimierung gestellt werden, sind durch die Nähe zur Aktorik festgelegt.

5.2 Energiemanagement

Hauptaufgabe des Energiemanagementmoduls (EM-Modul) ist die Bereitstellung der insgesamt auf dem Shuttle benötigten elektrischen Leistung. Hierzu kann es durch Sollwertvorgaben an das AB-Modul die Leistungsübertragung zum Shuttle beeinflussen. Energie, die beispielsweise im AB-Modul beim Bremsen zurückgewonnen oder zum Shuttle übertragen wird, kann das EM-Modul wahlweise in zwei unterschiedliche Energiespeicher (Batterien und Doppelschichtkondensatoren) speichern. Dabei ergibt sich die zu speichernde Leistung aus einer Leistungsbilanz, die neben der übertragenen Leistung auch die Verbraucher im Shuttle sowie die Verluste berücksichtigt. Sollten beide Energiespeicher vollständig geladen sein und somit keine zusätzliche Leistung mehr aufnehmen können, so kann das EM-Modul nicht speicherbare Leistung an einem Bremswiderstand in Wärme umsetzen. Grundsätzlich ist diese Funktionalität jedoch nicht Gegenstand der Optimierung, sondern als automatische Sicherheitsfunktion realisiert.

Im Abbildung 5.2 sind der prinzipielle Aufbau des EM-Moduls und die Richtungen der Leistungsflüsse dargestellt.

Grundsätzlich sind vier Leistungsflüsse von Bedeutung. Leistungsfluss zwischen:

1. Batterie und Zwischenkreis
2. Doppelschichtkondensatoren und Zwischenkreis
3. Zwischenkreis und Stromrichter des Linearantriebs
4. Zwischenkreis und übrigen Verbrauchern z.B. Hydraulik

Dabei sind die ersten drei Leistungsflüsse bidirektional und der letztgenannte Leistungsfluss unidirektional.

Die Leistungsflüsse zwischen Leistungszwischenkreis und Energiespeicher sind unmittelbarer Gegenstand der Optimierungsaufgabe. Der Leistungsfluss zwischen Leistungszwischenkreis und Stromrichter des Linearantriebs ist für das Energiemanagement indirekt Bestandteil der Optimierung, da hier lediglich Anforderungen für die Arbeitspunksteuerung (siehe 5.1) generiert werden, die deren OCM umsetzen soll. Daher beinhaltet das Energiemanagement eine Optimierungsaufgabe, die sowohl direkt

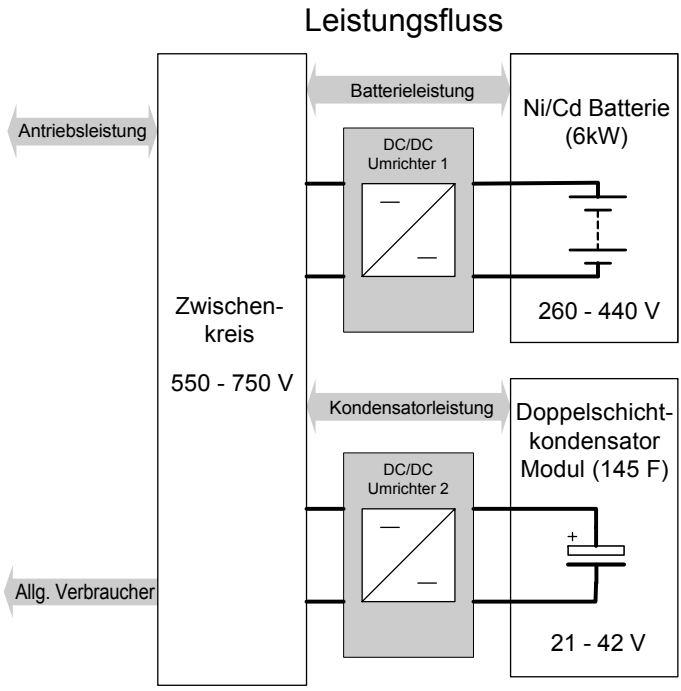


Abbildung 5.2: Struktur des Energieversorgungsmoduls

auf Aktorik (die Umrichter der Energiespeicher) wirkt, als auch die Verkopplung mit einem anderen OCM (Arbeitspunktsteuerung) berücksichtigen muss. Der unidirektionale Leistungsfluss zu den Verbrauchern wie z.B. dem Hydraulikaggregat stellt eine Störgröße für das Energiemanagement dar, die es gilt, so gut wie möglich im Voraus zu schätzen und bei der Bestimmung der optimalen Ladezustände der Energiespeicher zu berücksichtigen.

5.2.1 Optimierungsaufgabe

Die Optimierungsaufgabe besteht darin, die optimalen Ladezustände der Energiespeicher festzulegen und frühzeitig sicherzustellen. Die Gewährleistung einer aus sicherheitstechnischer Sicht stets ausreichenden Leistungsversorgung ist obligatorisches Ziel. Hohe Energieeffizienz sowie eine lange Lebensdauer der Energiespeicher sind als fakultative Ziele anzusehen. Aufgabe ist es, optimale Leistungsflüsse von und zu den Energie-

speichern sowie die Leistungsübertragung zum Shuttle festzulegen. Dazu ist eine möglichst genaue Kenntnis des zu erwartenden Leistungsbedarfs der Verbraucher und der maximal zum Shuttle übertragbaren Leistung von sehr großer Bedeutung. Hierbei spielen Kriterien wie Streckenprofil, Wirkungsgrade sowie die Lebensdauer der Energiespeicher eine entscheidende Rolle. Da diese Kriterien maßgeblich durch andere Module beeinflusst werden, ergibt sich eine starke Verkopplung mit diesen Modulen. Insbesondere resultieren durch diese starke Verknüpfung ständig neue Anforderungen an das Energiemanagement, sodass eine ständige Anpassung der Betriebsstrategie unumgänglich für einen optimalen Betrieb des Energiemanagements ist.

5.2.2 Ablauf der Selbstoptimierung

Der in Kapitel 2.2 beschriebene Ablauf der Selbstoptimierung stellt die Basis für eine effiziente Realisierung des Energiemanagements dar. Es ist erforderlich, in der Optimierung Schätzgrößen für zukünftig auf dem Shuttle benötigte Leistung zu berücksichtigen. Diese Schätzungen bedürfen einer ständigen Verbesserung, um Schätzfehler auch bei veränderlichen Einflüssen klein zu halten. Daher gilt, dass auch die Optimierung stets an die veränderlichen Einflüsse angepasst werden muss. Für das selbstoptimierende Energiemanagement werden die bekannten drei Schritte wie folgt ausgeführt.

1. *Analyse der Ist-Situation*

   Die Analyse der Situation besteht aus einer möglichst aktuellen, jedoch weit vorausschauenden Leistungsbedarfsanalyse. Basis für diese Analyse ist unter anderem das Streckenprofil, das Auswirkungen auf das für den Komfort verantwortliche Hydrauliksystem des Shuttles bzw. dessen Leistungsbedarf hat. Darüber hinaus ist das Streckenprofil maßgeblich für die maximal zum Shuttle übertragbare Leistung. Neben diesen externen Einflüssen müssen auch interne Zustände des Energieversorgungsmoduls analysiert werden. Dazu gehören speziell die Ladezustände der Speicher, die aktuell zu erwartende Lebensdauer der Energiespeicher sowie die Dringlichkeit einer stationären Wartung der Batterien.

2. *Bestimmung von Zielen*

   Auf der Grundlage der im Schritt 1 durchgeführten Analyse der Ist-Situation werden die Optimierungsziele bzw. deren Gewichtung

festgelegt. Ziele des Energiemanagements sind: Hohe Lebensdauer der Energiespeicher – insbesondere der Batterien, geringe Verluste im Energieversorgungsmodul, niedrige Ladezustände, um aus dem Antrieb zurückgewonnene Leistung aufzunehmen, hohe Ladezustände, um unvorhergesehene Leistungsanforderungen decken zu können sowie die Zwischenkreisspannung stets auf ein konstantes Niveau regeln zu können. Einige dieser Ziele sind gegenläufig, sodass eine situationsabhängige Anpassung der Ziele erforderlich ist.

3. *Anpassung des Systemverhaltens*

   Nachdem im Schritt 2 die aktuellen Anpassungsziele festgelegt sind, erfolgt eine Optimierung der Sollladezustände, die durch Variation der Leistungsflüsse angepasst werden.

5.2.3 Einordnung

Da das zu optimierende Energiemanagement keine selbstoptimierenden Subsysteme enthält, handelt es sich um individuelle Selbstoptimierung.

Form der Optimierung

Form der Optimierung (zu Kapitel 2.3.1): Die Optimierung der Ladezustände wird im Wesentlichen durch mathematische Modelle der Batterielebensdauer sowie durch Modelle für den Leistungsbedarf im Shuttle bestimmt. Daher ist die Optimierung im Rahmen des Energiemanagements als modellbasiert anzusehen. Durch den Einsatz einer Wissensbasis lassen sich die Modelle, insbesondere die der Batterielebensdauer, stetig verbessern. Dies führt jedoch nicht zu einer verhaltensbasierten Selbstoptimierung, wie sie in Kap. 2 beschrieben ist, da die Optimierung weiterhin modellbasiert erfolgt.

Zeitpunkt und Ablauf

Zeitpunkt und Ablauf (zu Kapitel 2.3.2): Bei der hier vorgestellten Optimierung wird davon ausgegangen, dass die Ladezustände der Energiespeicher kontinuierlich an die aktuellen Einflüsse angepasst werden müssen. Abhängig von den Einflüssen, wie z.B. die zu erwartende Lebensdauer der Batterie oder das zu erwartende Streckenprofil, gilt es bei der Optimierung die Ziele anzupassen.

Sicherheitsaspekte

Sicherheitsaspekte (zu Kapitel 2.3.3): Das Energieversorgungsmodul mit dem Energiemanagement stellt eine sehr zentrale Komponente im Shuttle dar. Ein Versagen des Energiemanagements z.B. in Form einer unzureichenden Energieversorgung für den Antrieb oder das Hydrauliksystem des Shuttles führt zwangsläufig zu nicht hinnehmbaren sicher-

heitskritischen Situationen. Da jedoch das Energieversorgungssystem nicht redundant vorhanden ist und es sich kaum extern überwachen lässt, ist eine interne Überwachung der Sicherheit erforderlich.

Typisierung und Struktur

Typisierung und Struktur (zu Kapitel 2.3.4): Aufgrund der zentralen Bedeutung des Energiemanagements sowie dessen Relevanz bezüglich der Sicherheit des gesamten Shuttles ist die Selbstoptimierung zunächst weitestgehend egoistisch motiviert. Jedoch lässt sich gerade durch das Zusammenspiel des Energiemanagements mit den anderen Modulen im Shuttle eine kollektive Optimierung realisieren. Durch die Kooperation mit anderen Modulen lassen sich insbesondere präzise Leistungsbedarfsanalysen erstellen. Anpassungen, die durch das Energiemanagement durchgeführt werden, werden zwar nur lokal im Energieversorgungsmodul vorgenommen, wirken sich jedoch auf das Gesamtfahrzeug aus. Die Anforderungen z.B. hinsichtlich des zeitlichen Rahmens, die an diese Optimierung gestellt werden, sind durch die Nähe zur Aktorik festgelegt.

5.3 Feder- und Neigemodul

Die beiden Feder- und Neigemodule der RailCab-Shuttles kontrollieren die Aufbaudynamik des Shuttles, also im Besonderen der Fahrgastzelle. Beim Durchfahren von Kurven sind sie darüber hinaus für eine entsprechende nach innen gerichtete Neigung verantwortlich. Ihre Hauptaufgabe liegt jedoch in der Erzielung eines hohen Fahrtkomforts, wobei als Nebenbedingung eine hohe Energieeffizienz gefordert wird. Die Feder- und Neigemodule stehen dabei im engen Dialog mit der übergeordneten Hierarchieebene des Shuttles, an die veränderte Betriebsbedingungen und Anforderungen mitgeteilt und von der veränderte Vorgaben entgegengenommen werden. Die Vertikalbewegung und Horizontalbewegung in Querrichtung eines Feder- und Neigemoduls können am entsprechenden Prüfstand untersucht werden, an dem auch bereits eine klassische Regelung ausgelegt und erprobt wurde. Der Prüfstand besteht im Wesentlichen aus drei Hydraulikzylindern zur Beeinflussung der Aufbaumasse in Vertikal- und Querrichtung, sowie drei Anregungszylindern, mit denen die Anregung durch die Schiene simuliert werden kann. Die Aufbaumasse ist über zwei Luftfedern mit geringer Dämpfung an die aktiven Hydraulikzylinder gekoppelt. Zur Energieversorgung dient ein geregelter Elektromotor mit Pumpe und Speicher. So wird für die gesamte Prüfstandshydraulik ein möglichst konstanter Versorgungsdruck gewährleistet. Die Regelung läuft auf echtzeitfähiger Hardware, die sowohl die Druckreglung, die Aufbauregelung mit unterlagerten Zylinder- und Ventilregelungen als auch die Generierung der Anregung mit den unterlagerten Zylinder- und Ventilregelungen umfasst. Zur Überwachung und zur Einflussnahme durch Benutzer steht zusätzlich ein Host-PC zur Verfügung, auf dem auch Anwendungen laufen könnten, die keinen harten Echtzeit-Bedingungen unterliegen.

5.3.1 Optimierungsaufgaben

Der Feder-Neige-Prüfstand soll dazu dienen, die innerhalb des Sonderforschungsbereich 614 erarbeiteten Verfahren und Mechanismen zur Selbstoptimierung anzuwenden und zu testen. Dazu kann einerseits entwickelte Echtzeithardware zusammen mit dem entsprechenden Echtzeitbetriebssystem zum Einsatz kommen, andererseits können auch die erarbeiteten Optimierungsstrategien zunächst auf der vorhandenen Echtzeithardware und später zusammen mit Reglerumschaltkonzepten auf der selbstentwickelten Rapid-Prototyping-Hardware implementiert werden.

Ausgangspunkt für die Selbstoptimierung ist, Informationen, die beim wiederholten Befahren desselben Streckenabschnitts gesammelt werden, für nachfolgende Shuttles nutzbar zu machen. Durch das wiederholte Befahren entstehen Änderungen im Gleisprofil, die durch die Selbstoptimierung berücksichtigt werden sollen. Die klassische Regelung versucht, die Aufbaumasse gegenüber einer fest voreingestellten Bahn zu bedämpfen (Skyhook). Diese Bahn kann durch eine parametrisierbare Funktion (wie z.B. Splines) dargestellt werden. So können die beschreibenden Parameter abhängig vom jeweiligen Streckenprofil variiert werden, um durch die so angepasste Bahn ein besseres Ergebnis zu erhalten. Die Eingriffsmöglichkeiten sind dabei auf der einen Seite die vorgegebene Bahn selbst, auf der anderen Seite der Grad der Dämpfung. Je höher die Dämpfung gewählt wird, desto besser folgt der Aufbau der vorgegebenen Bahn. Externes Ziel der Optimierung ist dabei ein möglichst hoher Komfort für die Insassen des Shuttles. Ein möglichst niedriger Energieverbrauch für die aktive Federung und niedriger Verschleiß sind inhärente Ziele des Moduls.

Am Prüfstand soll dieses Verfahren dadurch umgesetzt werden, dass ein sich veränderndes Gleisprofil als Anregung mehrmals nacheinander verwendet wird. Die Bahn, gegenüber der die Aufbaumasse bedämpft wird, erfährt infolge einer Optimierung eine Anpassung an die aktuelle Situation. Die Optimierung verwendet zur Bestimmung der angepassten Bahn ein mathematisch-physikalisches Modell des Shuttles und liefert für die Fahrt des nächsten Shuttles, das am Prüfstand durch erneutes Verwenden des entsprechenden Gleisprofils simuliert wird, eine neue Bahn, mit der auf die sich ändernden Umgebungseinflüsse reagiert wurde.

5.3.2 Ablauf der Selbstoptimierung

1. *Analyse der Ist-Situation*

   Im ersten Schritt werden die für einen bestimmten Streckenabschnitt ermittelten Zielgrößen Komfort und Energieverbrauch bestimmt, die sich aus der Anwendung der vorgegebenen Bahn mit der dazu eingestellten Dämpfung des Aufbaus ergeben haben. Der Komfort ist dabei im Wesentlichen von der bewerteten Aufbaubeschleunigung abhängig, die über am Aufbau befestigten Sensoren gemessen wird. Für die Bestimmung des Energieverbrauchs wird ein Modell der Hydraulikzylinder zusammen mit den gemessenen Längenänderungen verwendet. Weiterhin können andere Größen, wie z.B. die zur Verfügung stehende Energie, an dieser Stelle erfasst werden, die für die folgenden Schritte notwendig sind.

2. *Bestimmung von Zielen*

   Abhängig von der Analyse der Ist-Situation werden nun die Ziele angepasst. Mit Hilfe eines Suchalgorithmus' wird eine neue Bahn berechnet, die den Komfort für den betrachteten Streckenabschnitt für nachfolgende Shuttles verbessern würde. Diese Bahn dient als Zielvorgabe für das nächste Befahren dieses Streckenabschnitts. Andere zuvor erfasste Größen bestimmen, wie der Energieverbrauch zu berücksichtigen ist. Daraus ergibt sich das zu wählende Dämpfungsmaß.

3. *Anpassung des Systemverhaltens*

   Die ermittelte Bahn sowie das gewählte Dämpfungsmaß werden schließlich als Vorgabe für die implementierte Regelung verwendet. Dadurch wird das Systemverhalten an den aktuellen Verlauf der Gleise angepasst, indem neue Parameter für die Bahnvorgabe und das Dämpfungsmaß verwendet werden.

5.3.3 Einordnung

Nachfolgend wird das Feder- und Neigemodul hinsichtlich der oben eingeführten Dimensionen klassifiziert.

Form der Optimierung

Form der Optimierung (zu Kapitel 2.3.1): Die optimalen Parameter für die vorzugebende Bahn werden mit Hilfe eines mathematischen Modells berechnet. Dieser Teil ist also als modellbasiert zu bezeichnen. Für die Wahl der Dämpfung in Abhängigkeit von anderen Größen sowie für die Auswahl genau einer Vorgabebahn werden verhaltensbasierte Ansätze auf Grundlage einer Wissensbasis eingesetzt. Insgesamt ergibt sich also ein kombiniertes Optimierungsverfahren.

Zeitpunkt und Ablauf

Zeitpunkt und Ablauf (zu Kapitel 2.3.2): Die Selbstoptimierung läuft in diesem Beispiel ereignisgesteuert ab. Immer dann, wenn ein bestimmter Streckenabschnitt überfahren wurde, wird eine neue Konfiguration für das nächste Befahren des Streckenabschnitts ermittelt. Dieser Teil der Selbstoptimierung unterliegt dabei weichen Echtzeitbedingungen. Liegt bei der Durchfahrt des nächsten Shuttles noch keine neue Konfiguration vor, wird eine vorhandene Konfiguration verwendet, so dass höchstens positive Effekte ausbleiben. Fatale Konsequenzen können sich wegen des verwendeten Reglerkonzepts dadurch nicht ergeben.

Sicherheitsaspekte

Sicherheitsaspekte (zu Kapitel 2.3.3): Das Feder- und Neigemodul implementiert eine auf heterogener Redundanz basierende Lösung, bei der jederzeit auf eine Reglerkonfiguration zurückgegriffen werden kann, die einen sicheren Systemzustand garantiert. Es überwacht seinen Ausführungszustand selbsttätig intern. Allerdings werden sein Verhalten und seine Energieeffizienz auch durch das übergeordnete Shuttlesystem überwacht, das ggf. mit korrigierenden Vorgaben eingreift. Diese Mechanismen werden ebenfalls am Prüfstand implementiert.

Typisierung und Struktur

Typisierung und Struktur (zu Kapitel 2.3.4): Das Feder- und Neigemodul ist eingebettet in das Gesamtsystem Shuttle als Teil eines Multiagentensystems vom Typ KLA zu verstehen. Durch Kommunikation mit den höheren Ebenen des Systems versucht das Modul kollektiv, eine im gegebenen Gesamtkontext optimale Verhaltensstrategie zu entwickeln. Anpassungen werden lokal an den kontrollierten Reglern vorgenommen. Das Verhalten ist verantwortlich im Bezug auf das Gesamtsystem und somit altruistisch geprägt. Das Feder- und Neigemodul besteht aus einer festgelegten Kombination konkreter Agenten und ist somit auf Instanz- und Typebene statisch. Auch die von ihnen übernommenen Rollen sind statisch.

5.4 Konvoisimulation

Mit Hilfe der Konvoisimulation sollen das Verhalten und die Wechselwirkungen von mehreren Shuttles untersucht werden. Die einzelnen Shuttles werden dabei durch bereits vorhandene Modelle aus der bisherigen Arbeit abgebildet. Dazu wurde zunächst eine klassische Abstands- bzw. Positionsregelung entworfen und ausgelegt. Das so entstandene Modell von mehreren Shuttles, die miteinander kommunizieren, kann in unterschiedlichen Modellierungstiefen weiterverwendet werden, um verschiedene Anwendungsfälle abzudecken. Das kann einerseits die Simulation von wenigen relativ genau modellierten Shuttles bei der Durchfahrt durch eine Weiche sein, wobei eine optimale Bestimmung der Reihenfolge und möglichst niedriger Energieverbrauch bei hohem Komfort interessieren. Auf der anderen Seite können aber auch viele sehr einfach gehaltene Shuttlemodelle simuliert werden, um die Strategien der Selbstoptimierung auf Kommunikation und Logistik anwenden zu können. An dieser Stelle soll das Zusammenschließen von zwei Konvois an einer Weiche betrachtet werden.

5.4.1 Optimierungsaufgaben

Auch bei der Simulation der Weichendurchfahrt und dem Zusammenschließen zweier Konvois kommen die im Sonderforschungsbereich 614 erarbeiteten Verfahren zum Einsatz. Für die unterschiedlichen Aufgaben, die dabei zu bewältigen sind, werden sowohl verhaltens- als auch modellbasierte Ansätze verfolgt.

Die eigentliche Optimierung kann in mehrere Abschnitte eingeteilt werden. Zunächst müssen die beiden Konvois in ausreichender Entfernung von der Weiche erkennen, dass an der Weiche möglicherweise eine Kollision eintritt oder dass ein gemeinsamer Konvoi gebildet werden soll. Mit diesen Informationen wird eine Reihenfolge der Shuttles in dem gemeinsamen Konvoi ermittelt, die abhängig von gegebenen Anforderungen wie der Wunschgeschwindigkeit einzelner Shuttles und der daraus resultierenden Sollgeschwindigkeit des Konvois optimiert wird.

Nachdem die Reihenfolge bestimmt ist, bleibt für beide Konvois noch die Zeit bis zum tatsächlichen Einfahren in die Weiche, um ihre Position im gemeinsamen virtuellen Konvoi einzunehmen. Bei diesem Anfahren der erforderlichen Positionen sind Brems- und Beschleunigungsmanöver not-

wendig. Diese Manöver werden mit Hilfe eines modellbasierten Verfahrens hinsichtlich Komfort und Energieverbrauch optimiert.

In einer Simulation sollen verschiedene Situationen der Weichendurchfahrt untersucht und die entwickelten Lösungsansätze zur Bestimmung der Reihenfolge und zum optimalen Einnehmen der erforderlichen Positionen erprobt werden.

5.4.2 Ablauf der Selbstoptimierung

1. *Analyse der Ist-Situation*

   Zunächst werden alle erforderlichen Daten gesammelt und analysiert. Dazu gehören die Wunschgeschwindigkeiten und aktuellen Positionen der einzelnen Shuttles. Außerdem interessiert deren zur Verfügung stehende Energie. Abhängig davon, wie viele weitere Informationen zur Verfügung stehen, können auch Aussagen über ein sinnvolles Auftrennen des gebildeten Konvois zu einem späteren Zeitpunkt oder das mögliche Bilden von weiteren Konvois an einer anderen Weiche mit einbezogen werden.

2. *Bestimmung von Zielen*

   Abhängig von der Analyse der Ist-Situation werden nun die Ziele angepasst. Mit Hilfe eines Neuro-Fuzzy-Ansatzes und basierend auf Erfahrungswissen wird eine optimale Reihenfolge des virtuellen Konvois gebildet. Diese ermittelte Reihenfolge dient wiederum als eine notwendige Zielvorgabe für die Bestimmung der optimalen Fahrmanöver der einzelnen Shuttles. Weitere Ziele, wie die Minimierung des Energieverbrauchs beim Aufholen und Abbremsen und ein möglichst hohes Maß an Komfort, fließen bei dieser modellbasierten Optimierung mit ein.

3. *Anpassung des Systemverhaltens*

   Nachdem auf die beschriebene Weise eine Reihenfolge bestimmt und entsprechende Fahrmanöver für die einzelnen Shuttles ermittelt wurden, werden diese Manöver im letzten Schritt den einzelnen Fahrzeugen der Konvois übermittelt, sodass sich bei der Weiche ein neuer Konvoi zusammenschließen kann. Dieses Zusammenschließen erfolgt optimal hinsichtlich der gewählten Ziele und es ergibt sich eine Reihenfolge, die optimal an die jeweils auftretende Ausgangssituation angepasst ist.

5.4.3 Einordnung

Nachfolgend wird die Simulation der Konvoibildung an einer Weiche hinsichtlich der oben eingeführten Merkmale klassifiziert.

Form der Optimierung

Form der Optimierung (zu Kapitel 2.3.1): Die Reihenfolge der einzelnen Shuttles im sich ergebenden Konvoi wird mit Hilfe eines Neuro-Fuzzy-Ansatzes und unter Berücksichtigung von Erfahrungswissen bestimmt. Die Optimierung der Fahrmanöver zum Erreichen der vorgegebenen Position im Konvoi wird mit modellbasierten Techniken durchgeführt. Auch hier ergibt sich also ein kombiniertes Verfahren zur Selbstoptimierung.

Zeitpunkt und Ablauf

Zeitpunkt und Ablauf (zu Kapitel 2.3.2): Die Selbstoptimierung läuft hier ereignisgesteuert ab. Sobald erkannt wird, dass sich zwei Konvois an einer Weiche treffen, werden die drei Schritte der Selbstoptimierung für die konkret auftretende Situation durchgeführt. Dabei unterliegt die Selbstoptimierung harten Echtzeitanforderungen. Die Reihenfolge und die Fahrmanöver müssen rechtzeitig vorliegen, damit die Ergebnisse sinnvoll umgesetzt werden können.

Sicherheitsaspekte

Sicherheitsaspekte (zu Kapitel 2.3.3): Auch das Konvoibilden an einer Weiche basiert auf heterogener Redundanz. Sollte zu einem kritischen Zeitpunkt vor Erreichen der Weiche noch kein Ergebnis der Selbstoptimierung vorliegen, muss auf eine klassische Regelung zur Bildung eines Konvois zurückgegriffen werden, bei der nicht alle Ziele optimal erfüllt werden, die aber das sichere kollisionsfreie Einscheren erlaubt. Da alle Shuttlepositionen und Geschwindigkeiten für die Optimierung erforderlich sind, werden automatisch alle relevanten Größen erfasst und können auf diese Weise überwacht werden.

Typisierung und Struktur

Typisierung und Struktur (zu Kapitel 2.3.4): Betrachtet man als System die Weiche mit den darauf zu fahrenden Konvois, so ergibt sich ein Multiagentensystem vom Typ IGA. Ein übergeordneter Agent analysiert alle Daten der beteiligten Shuttles, um damit das Gesamtsystem zu optimieren. Es findet also isolierte Optimierung statt. Da die Ergebnisse der Optimierung Auswirkungen auf das Verhalten des Gesamtsystems, also beider Konvois haben, handelt es sich um globale Anpassungen. Dabei stehen im Wesentlichen die Ziele des gesamten Konvois im Mittelpunkt, sodass durchaus akzeptiert wird, wenn einem einzelnen Shuttle ein suboptimales Manöver vorgegeben wird, wenn dadurch das Ergebnis des Gesamtsystems verbessert wird. Die Shuttles handeln also altruistisch. Dabei muss für die Optimierung von vornherein bekannt sein, wie viele Shuttles mit welchen Ist-Zuständen an der betrachteten Situation betei-

ligt sind. Deshalb kann man hier aus Sicht des betrachteten Selbstoptimierungsprozesses von einem auf Instanzebene statischen System sprechen.

Kapitel 6

Zusammenfassung

Der Sonderforschungsbereich 614 – Selbstoptimierende Systeme des Maschinenbaus der Universität Paderborn beschäftigt sich mit der Erforschung des Paradigmas der Selbstoptimierung und der Entwicklung technischer Systeme, die dieses Paradigma umsetzen. Als Basis für eine effektive domänenübergreifende Zusammenarbeit werden dabei ein einheitliches Verständnis der grundlegenden Begriffe und eine gemeinsame Auffassung von Selbstoptimierung benötigt. Das vorliegende Buch soll dazu einen Beitrag leisten.

Laut Definition 1 findet in einem System dann Selbstoptimierung 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

Für die Selbstoptimierung ist wesentlich, dass Ziele durch das System situationsbedingt festgelegt oder angepasst werden. Ein System, in dem nur die erste und dritte, nicht aber die zweite Aktion durchgeführt werden, enthält somit keine Selbstoptimierung. Ein selbstoptimierendes System liegt zudem nur dann vor, wenn auch tatsächlich Ziele, die dem betrachteten System direkt zugeordnet werden können, einer Anpassung unterliegen.

Es wird zwischen individuell selbstoptimierenden und zusammengesetzt selbstoptimierenden Systemen unterschieden. Im ersten Fall verläuft der

wiederkehrende Prozess der Selbstoptimierung dergestalt, dass die Aktivitäten, von denen die drei notwendigen Aktionen Analyse, Zielbestimmung und Anpassung umgesetzt werden, von den Elementen eines einzelnen, isolierten Systems durchgeführt werden. Ein zusammengesetztes selbstoptimierendes System liegt dagegen vor, wenn ein selbstoptimierendes System Subsysteme mit eigenen Zielen, die am Prozess der Selbstoptimierung beteiligt sind, enthält.

Selbstoptimierende Systeme können bezüglich ihrer Optimierungsziele, der eingesetzten Optimierungsverfahren, dem Zeitpunkt der Selbstoptimierung, den Anforderungen bezogen auf Echtzeit sowie der realisierten Sicherheitskonzepte charakterisiert werden. Eine besondere Klasse selbstoptimierender Systeme stellen solche zusammengesetzten Systeme dar, in denen mehrere klar getrennte Subsysteme verteilt Selbstoptimierung betreiben. Diese Systeme werden als Multiagentensysteme, in denen mehrere Agenten gemeinsame oder divergente Optimierungsziele verfolgen, interpretiert. In derartigen Systemen spielt das Verteilungskonzept eine wesentliche Rolle, d.h. welches die angepassten Entitäten und welches die optimierenden Entitäten sind sowie welche Motivation die Agenten zum Handeln veranlasst.

Selbstoptimierende mechatronische Systeme basieren auf komplexen Reglersystemen, deren Funktionalität durch zusätzliche Informationsverarbeitung noch erweitert wird. Besondere Bedeutung gewinnt dabei die Vernetzung solcher Reglersysteme, um kollaborative und emergente Selbstoptimierung zu unterstützen. Auf lokaler Ebene müssen neben Techniken der künstlichen Intelligenz (KI) auch Techniken für die Rekonfiguration zur Laufzeit durch geeignete Entwurfsmethoden integriert werden. Eine wesentliche Herausforderung ergibt sich zusätzlich aus dem sicherheitskritischen Charakter der Systeme, der bedingt, dass der Verbund aus Software und technischem System ein vorhersagbar korrektes Verhalten zeigen muss – trotz Vernetzung, Rekonfiguration und der Integration von Techniken der künstlichen Intelligenz. Durch eine konsequente Strukturierung der regelnden Informationsverarbeitung werden derartig komplexe Systeme beherrschbar. Zur Strukturierung selbstoptimierender Systeme wird in diesem Buch die Architektur des Operator-Controller-Moduls vorgestellt. Sie erleichtert es, komplexe Rekonfigurationen zu beschreiben und ihre Korrektheit sicherzustellen.

Bei der Entwicklung selbstoptimierender Systeme müssen die für die Selbstoptimierung typischen Aktionen „Analyse der Ist-Situation“, „Bestimmung der Systemziele“ und „Anpassung des Systemverhaltens“ vorausgedacht und im System umgesetzt werden. Hierbei helfen Wirkmuster

zur Selbstoptimierung (WM_{SO}). Sie ermöglichen die Wiederverwendung erfolgreicher Lösungen für die genannten Aktionen, indem sie Schemata für den grundlegenden Aufbau und die Funktionsweise des selbstoptimierenden Systems zur Verfügung stellen.

Das Potenzial der Selbstoptimierung und die Umsetzung des Paradigmas der Selbstoptimierung wurde an ausgewählten Beispielen wie der Vorgabe von Fahrprofilen, der Arbeitspunktsteuerung eines Linearmotors sowie der Vorgabe von Dämpfungsmaßen verdeutlicht.

Literaturverzeichnis

[AIS+77] ALEXANDER, C.; ISHIKAWA, S.; SILVERSTEIN, M.; JACOBSON, M.; FIKSDAHL-KING, I.; ANGEL, S.: *A Pattern Language*. 1. Oxford University Press, 1977

[AL98] ABEL, D.; LEMMER, K.: *Theorie ereignisdiskreter Systeme*. 1. Oldenbourg Verlag, München, 1998

[BGO04] BURMESTER, Sven; GIESE, Holger; OBERSCHELP, Oliver: Hybrid UML Components for the Design of Complex Self-optimizing Mechatronic Systems. In: *Proc. of 1st International Conference on Informatics in Control, Automation and Robotics (ICINCO 2004), Setubal, Portugal*, IEEE, August 2004

[BKKN03] BORGELT, C.; KLAWONN, F.; KRUSE, R.; NAUCK, D.: *Neuro-Fuzzy-Systeme – Von den Grundlagen künstlicher Neuronaler Netze zu Kopplung mit Fuzzy-Systemen*. 3. vieweg, Wiesbaden, 2003

[Bro91] BROOKS, Rodney A.: Intelligence Without Reason. In: MYOPOULOS, John (Hrsg.); REITER, Ray (Hrsg.): *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*, Morgan Kaufmann publishers Inc., San Mateo, CA, USA, 1991. – ISBN ISBN 1–55860–160–0, S. 569–595

[BTG04] BURMESTER, Sven; TICHY, Matthias; GIESE, Holger: Modeling Reconfigurable Mechatronic Systems with Mechatronic UML. In: *Proc. of Model Driven Architecture: Foundations and Applications (MDAFA 2004), Linköping, Sweden*, 2004

[Bus02] BUSS, M.: *VDI-Fortschrittberichte: Methoden zur Regelung hybrider dynamischer Systeme*. 1. VDI-Verlag, Düsseldorf, 2002

[BWSG98] BEINEKE, S.; WERTZ, H.; SCHÜTTE, F.; GROTSTOLLEN, H.: Identification of Nonlinear Two-Mass Systems for Self-Commissioning Speed Control of Electrical Drives. In: *IECON 1998, Aachen, Deutschland*, 1998. – EU Projekt, AG Grotstollen

[CL99] CASSANDRAS, C.G.; LAORTUNE, S.L.: *Introduction to Discrete-Event Systems*. 1. Kluwer Academic Publishers, Dordrecht, 1999

[DAR97] DARPA. *Broad Agency Announcement (BAA) for Proposals in the Area of "Self-Adaptive Software"*. 1997

[DIN94] DIN 19226: *Leittechnik, Regelungstechnik und Steuerungstechnik. Allgemeine Grundlagen*. 1. 1994. – Teil 1

[DSH04] DELLNITZ, M.; SCHÜTZE, O.; HESTERMEYER, T.: Covering Pareto Sets by Multilevel Subdivision Techniques. In: *Journal of Optimization, Theory and Applications* (2004). – erscheint

[Dud00] DUDEN: *Das Fremdwörterbuch*. 6. Dudenverlag, Mannheim, 2000

[Dul00] DULLERUD, G.E.: *A course in robust control theory*. 1. Springer Verlag, Heidelberg, 2000

[Fer92] FERGUSON, I. A.: TouringMachines: Autonomous Agents with Attitudes. In: *IEEE Computer* 25 (1992), Nr. 5, S. 51–55

[Fer01] FERBER, Jacques: *Multiagentensysteme*. 1. Addison-Wesley, München u.a., 2001. – ISBN ISBN 3–8273–1679–0

[GBK+03] GIESE, Holger; BURMESTER, Sven; KLEIN, Florian; SCHILLING, Daniela; TICHY, Matthias: Multi-Agent System Design for Safety-Critical Self-Optimizing Mechatronic Systems with UML. In: *OOPSLA 2003 - Second International Workshop on Agent-Oriented Methodologies, Anaheim, CA, USA*, 2003, S. 21–32

[GBSO04] GIESE, Holger; BURMESTER, Sven; SCHÄFER, Wilhelm; OBERSCHELP, Oliver: Modular Design and Verification of Component-Based Mechatronic Systems with Online-Reconfiguration. In: *Proc. of 12th ACM SIGSOFT Foundations of Software Engineering 2004 (FSE 2004), Newport Beach, USA*, ACM, November 2004

[GFG+05] GAUSEMEIER, J.; FRANK, U.; GIESE, H.; KLEIN, F.; SCHMIDT, A.; STEFFEN, D.; TICHY, M.: A Design Methodology for Self-Optimizing Systems. In: *Automation, Assistance and Embedded Real Time Platforms for Transportation (AAET2005), Braunschweig*, 2005. – zur Publikation akzeptiert

[GFRS04] GAUSEMEIER, J.; FRANK, U.; REDENIUS, A.; STEFFEN, D.: Development of Self-Optimizing Systems. In: *MechRob 2004 (IEEE), Mechatronics and Robotics 2004, 13.-15. September 2004*, IEEE, September 2004

[GFSV04] GAUSEMEIER, J.; FRANK, U.; SCHMIDT, A.; VÖCKING, H.: Domänenübergreifende Spezifikation der Prinziplösung selbstoptimierender Systeme. In: *2. Gemeinsames Kolloquium Konstruktionstechnik, 23./24. September 2004*, 2004

[GHJV95] GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J.: *Design Patterns: Elements of Reusable Object Oriented Software*. 1. Addison-Wesley, Reading, MA, 1995

[GJ04] GRÜNE, Lars; JUNGE, Oliver: A set oriented approach to optimal feedback stabilization. In: *Systems and Control Letters* (2004). – erscheint

[GKMP04] GIORGINI, P.; KOLP, M.; MYLOPOULOS, J.; PISTORE, M.: The Tropos Methodology: an overview. In: *Methodologies And Software Engineering For Agent Systems* (2004)

[GTB+03] GIESE, H.; TICHY, M.; BURMESTER, S.; SCHÄFER, W.; FLAKE, S.: Towards the Compositional Verification of Real-Time UML Designs. In: *Proc. of the European Software Engineering Conference (ESEC), Helsinki, Finland*, ACM Press, September 2003, S. 38–47

[GW99] GRÜNE, L.; WIRTH, F.: Feedback stabilization of discrete-time homogeneous semi-linear systems. In: *Systems & Control Letters* 37 (1999), S. 19–30

[HBN01] HESTERMEYER, T.; BECKER, M.; NEUENDORF, N.: Nichtlineare ABC-Regelungen mit Operator-Controller-Struktur, abgestimmt auf Führung und Störung der Straße. In: *Driveability, Haus der Technik*, Haus der Technik, Essen, 2001

[HO03] HESTERMEYER, T.; OBERSCHELP, O.: Selbstoptimierende Fahrzeugregelung -Verhaltensbasierte Adaption. In: GAUSEMEIER, J. (Hrsg.); LÜCKEL, J. (Hrsg.); WALLASCHEK, J. (Hrsg.): *Intelligente Mechatronische Systeme*, HNI, Paderborn, 2003

[HOG04] HESTERMEYER, Thorsten; OBERSCHELP, Oliver; GIESE, Holger: Structured Information Processing For Self-optimizing Mechatronic Systems. In: *Proc. of 1st International Conference on Informatics in Control, Automation and Robotics (ICINCO 2004), Setubal, Portugal*, IEEE, August 2004

[HZ04a] HAGENMEYER, V.; ZEITZ, M.: Flachheitsbasierter Entwurf von linearen und nichtlinearen Vorsteuerungen. In: *at - Automatisierungstechnik* 52 (2004), S. 3–12

[HZ04b] HAGENMEYER, V.; ZEITZ, M.: Internal dynamics of flat nonlinear systems with respect to a non-flat output. In: *Systems & Control Letters* 52 (2004), S. 323–327

[ILM92] ISERMANN, R.; LACHMANN, K.-H.; MATKO, D.: *Adaptive Control Systems*. 1. Prentice Hall, Herfordshire, 1992

[IMC] INVENTION MACHINE CORP., USA. *Homepage zu Goldfire Innovator*

[KB94] KREISSELMEIER, G.; BIRKHÖLZER, T.: Numerical nonlinear regulator design. In: *IEEE Transaction on Automatic Control* 39 (1994), S. 33–46

[KC03] KEPHART, Jeffrey O.; CHESS, David M.: The Vision of Autonomic Computing. In: *Computer* 36 (2003), Nr. 1

[KD93] K.-D., Althoff: *Eine fallbasierte Lernkomponente als integraler Bestandteil der MOLTKE-Werkbank zur Diagnose technischer Systeme.* 1. Akademische Verlagsgesellschaft Aka GmbH, Berlin, 1993

[KE01] KENNEDY, James; EBERHARDT, Russell C.: *Swarm Intelligence*. 1. Morgan Kaufmann publishers Inc., San Mateo, CA, USA, 2001. – ISBN ISBN 9–8102–2884–8

[KJLW03] KOPP, S.; JUNG, B.; LESSMANN, N.; WACHSMUTH, I.: Max - A Multimodal Assistant in Virtual Reality Construction. In: *KI-Künstliche Intelligenz* (2003), Nr. 4

[LRS03] LADDAGA, Robert; ROBERTSON, Paul; SHROBE, Howard: Results of the Second International Workshop on Self-adaptive Software. In: LADDAGA, Robert (Hrsg.); ROBERTSON, Paul (Hrsg.); SHROBE, Howard (Hrsg.): *The Second International Workshop on Self-Adaptive Software (IWSAS 2001)* Bd. 2614, Springer-Verlag, Berlin, 2003, S. 281–290

[MA02] MEYSTEL, A.M.; ALBUS, J.S.: *Intelligent Systems - Architecture, Design and Control*. 1. John Wiley & Sons, Inc. New York, 2002 (Wiley Series on Intelligent Systems)

[MS04] MARSISKE, H. A.; SCHMIDT, B.: Hunger, Wut und Müdigkeit - Softwareagenten mit Emotionen. In: *c't magazin für computer technik* (2004), February, S. 82ff

[MSKC04] MCKINLEY, Philip K.; SADJADI, Seyed M.; KASTEN, Eric P.; CHENG, Betty H.: Composing Adaptive Software. In: *IEEE Computer* 37 (2004), July, Nr. 7

[Nau00] NAUMANN, R.: *Modellierung und Verarbeitung vernetzter intelligenter mechatronischer Systeme*, MLaP, Universität Paderborn, Diss., 2000

[NPB03] NAGY, Z.; PORTER, I.; BRADSHAW, A.: Schemebuilder: From Conceptual Design To Control Of Mechatronic Systems. In: *Proceedings of ICOM 2003*, Loughborough University, 2003

[NR98] NAUMANN, R.; RASCHE, R.: Description and Simulation of Hybrid Mechatronic Systems. In: *International Workshop on Hybrid Systems: Computation and Control, Berkeley, CA*, 1998

[Oes98] OESTEREICH, B.: *Objektorientierte Softwareentwicklung. Analyse und Design mit der United Modeling Language*. 1. R. Oldenourg Verlag, München, Wien, 1998

[OHG04] OBERSCHELP, O.; HESTERMEYER, T.; GIESE, H.: Strukturierte Informationsverarbeitung für selbstoptimierende mechatronische Systeme. In: *Proc. of the Second Paderborner Workshop Intelligente Mechatronische Systeme*, HNI, Paderborn, Germany, 2004 (HNI-Verlagsschriftenreihe 145), S. 43–56

[OHKK02] OBERSCHELP, O.; HESTERMEYER, T.; KLEINJOHANN, B.; KLEINJOHANN, L.: Design of Self-Optimizing Agent-Based Controllers. In: *CfP Workshop 2002 - Agent-Based Simulation 3, Passau*, 2002

[OPB00] ODELL, J.; PARUNAK, H.; BAUER, B.: Extending UML for Agents. In: WAGNER, G. (Hrsg.); LESPERANCE, Y. (Hrsg.); YU, E. (Hrsg.): *Proceedings of the Agent-Oriented Information Systems Workshop at the 17th National conference on Artificial Intelligence*, iCue Publishing, Austin, Texas, USA, 2000, S. 3–17

[PB97] PAHL, G.; BEITZ, W.: *Konstruktionslehre - Methoden und Anwendung*. Bd. 4. 1. Springer-Verlag, Berlin, Heidelberg, 1997

[Pup98] PUPPE, F.: Knowledge Reuse among Diagnostic Problem Solving Methods in the Shell-Kit D3. In: *International Journal of Human-Computer Studies 49* (1998), S. 627–649

[PUS00] PETERSEN, I.R.; UGRINOVSKII, V.A.; SAVKIN, A.V.: *Robust Control Design Using H-infinity Methods*. 1. Springer Verlag, Heidelberg, 2000

[RG91] RAO, Anand S.; GEORGEFF, Michael P.: Modeling Rational Agents within a BDI-Architecture. In: ALLEN, James (Hrsg.); FIKES, Richard (Hrsg.); SANDEWALL, Erik (Hrsg.): *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1991. – ISBN ISBN 1–55860–165–1, S. 473–484

[Ris00] RISING, Linda: *The Pattern Almanac 2000*. 1. Addison-Wesley, 2000 (Software pattern Series). – ISBN ISBN 0–20161–567–3

[RN03] RUSSEL, S.; NORVIG, P.: *Artificial Intelligence - A Modern Approach*. 2. Prentice Hall, Pearson Education, Inc., New Jersey, 2003 (Series in Artificial Intelligence)

[SB03] SAMAD, T.; BALAS, G.; BALAS, G. (Hrsg.): *Software-Enabled Control - Information Technology for Dynamical Systems*. 1. Wiley Interscience, 2003

[SCH98] STEIN, B.; CURATOLO, D.; HOFFMANN, M.: Simulation in FluidSIM. In: SZCZERBICKA, H. (Hrsg.): *Workshop Simulation in wissensbasierten Systemen (SIWIS 98), Bremen* Bd. 61, 1998

[Sko00] SKOGESTAD, S.: Plantwide Control: the Search for the Self-Optimizing Control Structure. In: *Journal Process Control* 10 (2000), Nr. 487

[Sto96] STOREY, Neil: *Safety-Critical Computer Systems*. 1. Addison-Wesley, Menlo Park, CA, 1996

[Suh93] SUHM, A.: *Produktmodellierung in wissensbasierten Konstruktionssystemen auf der Basis von Lösungsmustern*. 1. Shaker Verlag Aachen, 1993

[Szy99] SZYPERSKI, C.: *Component Software Beyond Object-Oriented Programming*. 1. Addison-Wesley, 1999

[Vöc03] VÖCKING, H.: *Multirate-Verfahren und Umschaltstrategien für verteilte Reglersysteme.*, MLaP, Universität Paderborn, Diplomarbeit, 2003

[VDI93] VDI: *VDI-Richtlinie 2221 - Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte*. 1. 1993

[WB02] WELP, E.G.; BLUDAU, C.: Wissensbasierte Modellierung mechatronischer Produktkonzepte mit ModCoDe und WISENT. In: *47. Internationales Wissenschaftliches Kolloquium*, Technische Universität Ilmenau, 2002

[WJ95] WOOLDRIDGE, Michael; JENNINGS, Nicholas R.: Agent Theories, Architectures, and Languages: a Survey. In: WOOLDRIDGE, Michael (Hrsg.); JENNINGS, Nicholas R. (Hrsg.): *Intelligent Agents*. Springer Verlag, 1995, S. 1–22

[WJK00] WOOLDRIDGE, M.; JENNINGS, N.; KINNY, D.: The Gaia methodology for agent-oriented analysis and design. In: *Journal of Autonomous Agents and Multi-Agent Systems* 3 (2000), Nr. 3, S. 285–312

[Woo00] WOOLDRIDGE, Michael J.: *Reasoning About Rational Agents (Intelligent Robotics and Autonomous Agents)*. 1. MIT Press, January 2000. – ISBN ISBN 0–26223–213–8

[Woo02] WOOLDRIDGE, Michael J.: *An Introduction to MultiAgent Systems*. 1. John Wiley and Sons, Ltd., Chichester, England, 2002. – ISBN ISBN 0–471–49691–X

[Zad65] ZADEH, Lotfi A.: Fuzzy Sets. In: *Information and Control* (1965), Nr. 8

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z

Glossar Deutsch – Englisch

kontinuierlich – continuously, **33**
Koordinationsmuster – coordination pattern, 55, **62**
Lösungsmuster – solution pattern, 54
lebenszyklusgesteuert – life-cycle-dependent, **34**
lokale Anpassung – local adaptation, **38**
Mechatronisches Funktionsmodul (MFM) – mechatronic function module (MFM), **23**
MFM (Mechatronisches Funktionsmodul) – MFM (mechatronic function module), 23
Modell – model, **17**
modellbasierte Optimierung – model-based optimization, **31**
Motivation – motivation, **38**
Multiagentensystem – multi-agent system, **28**, 37
Muster – pattern, **54**
nicht zeitkritisch – no real-time requirements, **34**
obligatorische Ziele – mandatory objectives, **31**
offenes System – open system, **41**
Operator-Controller-Modul – Operator-Controller-Module, 44
optimierende Entität – optimizing entity, **37**
Parameteranpassung – parameter adaptation, **21**
Proaktivität – proactivity, 27
qualitative Ziele – qualitative objectives, **31**
quantitative Ziele – quantitative objectives, **31**
Reaktivität – reactivity, 27
reflektorischer Operator – reflective operator, **45**
Rekonfiguration – reconfiguration, **21**
Rolle – role, 41
schwacher Agentenbegriff – weak notion of agency, **27**, 80
Selbstüberwachung – self-monitoring, 88
Selbstdiagnose – self-diagnosis, 88
Selbstheilung – self-healing, 86
selbstoptimierendes System – self-optimizing system, **25**
Selbstoptimierung – self-optimization, **22**, 86
Softwarekomponente – software component, **56**
spezifisch korrekte Rekonfiguration – specifically correct reconfiguration, **52**
starker Agentenbegriff – strong notion of agency, **80**
statisches System – static system, **41**
Struktur – structure, **16**
Strukturanpassung – structural adaptation, **21**
subsymbolisches System – subsymbolic system, **83**
symbolisches System – symbolic system, **82**
System – system, **15**
Systemgrenze – system border, **16**
Systemparameter – parameter, **16**
Systemumgebung – environment, **16**
Typ – type, 41
unüberwacht – unsupervised, **36**
Verhalten – behavior, **16**
Verhaltensanpassung – behavior adaptation, **21**
verhaltensbasierte Optimierung – behavior-based optimization, **31**
Vernetztes Mechatronisches System (VMS) – connected mechatronic system (CMS), **23**
VMS (Vernetztes Mechatronisches System) – CMS (connected mechatronic system), 23
weiche Echtzeit – soft real-time, **34**
weiche Ziele – soft objectives, **31**
Wiederherstellung – self-recovery, 88
zeitgesteuert – time-triggered, **33**

Glossar Englisch – Deutsch

activity – Aktivität, **17**
adaptation – Anpassung, **19**, **22**
agent – Agent, **28**
altruistic agent – altruistischer Agent, **39**
AMS (autonomous mechatronic system) – AMS (Autonomes Mechatronisches System), 23
appropriated objectives – angeeignete Ziele, **30**
autonomous mechatronic system (AMS) – Autonomes Mechatronisches System (AMS), **23**
autonomy – Autonomie, 27
behavior – Verhalten, **16**
behavior adaptation – Verhaltensanpassung, **21**
behavior-based optimization – verhaltensbasierte Optimierung, **31**
closed system – geschlossenes System, **41**
CMS (connected mechatronic system) – VMS (Vernetztes Mechatronisches System), 23
cognitive operator – kognitiver Operator, **47**
collective optimization – kollekive Optimierung, **38**
composed self-optimizing system – zusammengesetztes selbstoptimierendes System, **26**
composite methods – kombinierte Verfahren, **32**
compositional adaptation – kompositionale Anpassung, **21**
connected mechatronic system (CMS) – Vernetztes Mechatronisches System (VMS), **23**
continuously – kontinuierlich, **33**
controller – Controller, **45**
coordination pattern – Koordinationsmuster, 55, **62**
current situation – Ist-Situation, **22**
design pattern – Entwurfsmuster, **54**
determination of objectives – Zielbestimmung, 22
egoistic agent – egoistischer Agent, **38**
element – Element, **16**
emergence – Emergenz, **26**
environment – Systemumgebung, **16**
event-triggered – ereignisgesteuert, **34**
external objectives – externe Ziele, **30**
external supervision – extern überwacht, **36**
facultative objectives – fakultative Ziele, **31**
fail-operational – fail-operational, **35**
fail-safe – fail-safe, **35**
function – Funktion, **16**
generated objectives – generierte Ziele, **30**
generation – Generierung, **22**
generically correct reconfiguration – allgemein korrekte Rekonfiguration, **52**
global adaptation – globale Anpassung, **38**
hard objectives – harte Ziele, **31**
hard real-time – harte Echtzeit, **34**
heterogenous redundancy – heterogene Redundanz, **36**
homogenuous redundancy – homogene Redundanz, **36**
individually self-optimizing system – individuell selbstoptimierendes System, **25**
influence – Einfluss, **17**
inherent objectives – inhärente Ziele, **30**
inherent safety – inhärente Sicherheit, **35**
input – Eingang, **16**
instance – Instanz, 41
internal objectives – interne Ziele, **30**

structure – Struktur, **16**
subsymbolic system – subsymbolisches System, **83**
symbolic system – symbolisches System, **82**
system – System, **15**
system border – Systemgrenze, **16**
system of objectives – Zielsystem, 19, **29**
time-triggered – zeitgesteuert, **33**
type – Typ, 41
unsupervised – unüberwacht, **36**
weak notion of agency – schwacher Agentenbegriff, **27**, 80